Preface

This document is intended as a tool for review after an instructor lead class. It is not intended to be used as a guide during class. The white papers and labs assume that some hands on work around creating definitions have taken place. The primary use is to refresh one’s skills from a previously taken class or to supplement the classroom experience with optional topics.

You may zoom in on sections with content that is hard to read.

The labs referenced in this manual are available pre-built at this DevNet address:

http://devnet.Logi Analytics.com/Training_ClassMaterials

This document has been formatted for double-sided printing, hole-punching, and placement in a ring-binder.

This document was carefully prepared by the training department at Logi Analytics.
# Table of Contents

**Preface** ................................................................................................................................. 2  
**Table of Contents** ..................................................................................................................... 3  
**Glossary** ................................................................................................................................. 11  

**Logi Application Architecture Overview** ................................................................................ 19  
  Understanding Logi Applications .................................................................................................. 19  
  Introducing Elemental Development .............................................................................................. 20  
  Definition Files ............................................................................................................................. 20  
  From XML to HTML ....................................................................................................................... 22  
  Using External Files with a Logi Application ................................................................................ 23  

**Best Practices** ......................................................................................................................... 24  

**Element Naming Guidelines** .................................................................................................... 27  

**Tokens** ....................................................................................................................................... 28  
  Token Syntax .................................................................................................................................. 28  

**Token Types** .............................................................................................................................. 30  

**@Functions** .............................................................................................................................. 33  

**@Date Token** ............................................................................................................................ 35  

**Special @Request Identifiers** .................................................................................................... 36  

**Special Identifiers for Other Token Types** ............................................................................. 37  

**File Upload Tokens** .................................................................................................................. 38  

**Summary of Training Labs** ....................................................................................................... 39  

**New Application** ...................................................................................................................... 44  
  **New Application / Hello World** .............................................................................................. 44  
    Summary: .................................................................................................................................... 44  
    Objectives: .................................................................................................................................. 44  
    Report Definition.......................................................................................................................... 44  
    Report Output ............................................................................................................................. 44  
    Directions ................................................................................................................................... 45  

**Data Tables** .............................................................................................................................. 54  
  **Lab 1 – Basic Data Table with Graphical Paging & Sorting** .................................................. 54
<table>
<thead>
<tr>
<th>Data Layers</th>
<th>88</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab 1 – Calculated Columns</td>
<td>88</td>
</tr>
<tr>
<td>Objectives:</td>
<td></td>
</tr>
<tr>
<td>Report Definition</td>
<td>89</td>
</tr>
<tr>
<td>Report Output</td>
<td>89</td>
</tr>
<tr>
<td>Directions:</td>
<td>90</td>
</tr>
<tr>
<td>Lab 2 – VB Script Functions</td>
<td>98</td>
</tr>
<tr>
<td>Objectives:</td>
<td></td>
</tr>
<tr>
<td>Report Definition</td>
<td>99</td>
</tr>
<tr>
<td>Report Output</td>
<td>99</td>
</tr>
<tr>
<td>Directions:</td>
<td>99</td>
</tr>
<tr>
<td>Lab 3 – KPI</td>
<td>105</td>
</tr>
<tr>
<td>Objectives:</td>
<td></td>
</tr>
</tbody>
</table>

| Lab 2 – Numbered Paging                                                    | 71 |
| Objectives:                                                               |    |
| Report Definition                                                         | 71 |
| Report Output                                                             | 71 |
| Directions:                                                               | 72 |
| Lab 3 – Multi Column Sort                                                 | 74 |
| Objectives:                                                               |    |
| Report Definition                                                         | 74 |
| Report Output                                                             | 74 |
| Directions:                                                               | 75 |
| Lab 4 – Summary Row                                                        | 77 |
| Objectives:                                                               |    |
| Report Definition                                                         | 77 |
| Report Output                                                             | 77 |
| Directions:                                                               | 78 |
| Lab 5 – Custom Summary Row                                                | 81 |
| Objectives:                                                               |    |
| Report Definition                                                         | 81 |
| Report Output                                                             | 81 |
| Directions:                                                               | 82 |
Table of Contents

**Exports** .................................................................................................................................................. 113

Lab 1 – Excel .................................................................................................................................................. 113
  Objectives: .................................................................................................................................................. 113
  Report Definition ....................................................................................................................................... 113
  Report Output (Both Html and Excel Export) ............................................................................................. 113
  Directions: ................................................................................................................................................. 114

Lab 2 – PDF .................................................................................................................................................... 118
  Objectives: .................................................................................................................................................. 118
  Report Definition ....................................................................................................................................... 119
  Final Report Output (HTML and PDF) ........................................................................................................ 119
  Directions: ................................................................................................................................................. 120

Lab 3 – Printable Paging (Optional Lab) ....................................................................................................... 124
  Objectives: .................................................................................................................................................. 124
  Report Definition ....................................................................................................................................... 124
  Final Report Output (HTML and PDF) ........................................................................................................ 124
  Web Print Output ....................................................................................................................................... 125
  Directions: ................................................................................................................................................. 126

**User Inputs** ............................................................................................................................................. 129

Lab 1 – Select List with Submit ..................................................................................................................... 129
  Objectives: .................................................................................................................................................. 129
  Report Definition ....................................................................................................................................... 129
  Final Report Output .................................................................................................................................... 129
  Directions: ................................................................................................................................................. 130

Lab 2 – Select List Event Driven ................................................................................................................... 138
  Objectives: .................................................................................................................................................. 138
  Report Definition ....................................................................................................................................... 138
  Final Report Output .................................................................................................................................... 138
  Directions: ................................................................................................................................................. 139

Lab 3 – Multi Select ...................................................................................................................................... 141
  Objectives: .................................................................................................................................................. 141
  Report Definition ....................................................................................................................................... 141
  Final Report Output .................................................................................................................................... 141
  Directions: ................................................................................................................................................. 142

Lab 4 – Cascading Filters ............................................................................................................................... 144

---

Report Definition ........................................................................................................................................ 106
Report Output ............................................................................................................................................... 106
Directions: ............................................................................................................................................... 107
<table>
<thead>
<tr>
<th>Lab</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab 5 – Input Validation</td>
<td>151</td>
<td></td>
</tr>
<tr>
<td>Lab 1 – Grouping Alternatives</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>Lab 2 – Grouped Flat Table &amp; Aggregates</td>
<td>171</td>
<td></td>
</tr>
<tr>
<td>Lab 3 – Drilldown with Sub Data Layers</td>
<td>178</td>
<td></td>
</tr>
<tr>
<td>Charts</td>
<td>186</td>
<td></td>
</tr>
</tbody>
</table>

### Objectives

- Report Definition
- Final Report Output
- Directions

### Hierarchical Reports

- Lab 1 – Grouping Alternatives
  - Objective
  - Report Definition
  - Final Report Output
  - Directions

- Lab 2 – Grouped Flat Table & Aggregates
  - Objective
  - Report Definition
  - Final Report Output
  - Directions

- Lab 3 – Drilldown with Sub Data Layers
  - Objective
  - Final Report Output
  - Definition
  - Directions

### Charts

- Lab 1 – Pie Chart
  - Objective
  - Report Definition
  - Final Report Output
  - Directions

- Lab 2 – Exploded Wedges
  - Objective
  - Report Definition
  - Final Report Output
  - Directions

- Lab 3 – Pie Chart – All Others Wedge
Objectives: ........................................................................................................................................ 195
Report Definition............................................................................................................................. 195
Final Report Output.......................................................................................................................... 195

Lab 4 – Drill Through to Data Table ............................................................................................... 199
Objectives: ........................................................................................................................................ 199
Final Report Output.......................................................................................................................... 199
Definition .......................................................................................................................................... 200

Lab 5 – Bar Chart ............................................................................................................................ 209
Objectives: ........................................................................................................................................ 209
Report Definition............................................................................................................................. 209
Final Report Output.......................................................................................................................... 209

Lab 6 – Extra Data Column ............................................................................................................. 214
Objectives: ........................................................................................................................................ 214
Report Definition............................................................................................................................. 214
Final Report Output.......................................................................................................................... 214

Lab 7 – Extra Grid Layer .................................................................................................................. 218
Objectives: ........................................................................................................................................ 218
Report Definition............................................................................................................................. 218
Final Report Output.......................................................................................................................... 218

Processes ........................................................................................................................................... 221

Lab 1 – Email PDF ............................................................................................................................. 221
Objectives: ........................................................................................................................................ 221
Report Definition............................................................................................................................. 221
Final Report Output.......................................................................................................................... 221

Scheduling ....................................................................................................................................... 228

Lab 1 – Scheduler Tasks ...................................................................................................................... 228
Objectives: ........................................................................................................................................ 228
Report Definition............................................................................................................................. 228
Final Report Output.......................................................................................................................... 228

Lab 2 – Scheduler Builder .................................................................................................................. 233
Objectives: ................................................................................................................................................. 233
Report Definition........................................................................................................................................ 233
Final Report Output................................................................................................................................... 233
Directions:................................................................................................................................................ 234

Lab 3 – My Scheduled Jobs .......................................................................................................................... 237
Objectives:................................................................................................................................................ 237
Final Report Output................................................................................................................................... 237
Report Definition....................................................................................................................................... 238
Directions:................................................................................................................................................ 239

Templates ..................................................................................................................................................... 243

Lab 1 – Excel Template ................................................................................................................................. 243
Objectives:................................................................................................................................................ 243
Final Report Output................................................................................................................................... 243
Directions:................................................................................................................................................ 244

Crosstabs ...................................................................................................................................................... 249

Lab 1 – Basic Crosstab................................................................................................................................. 249
Objectives:................................................................................................................................................ 249
Report Definition....................................................................................................................................... 249
Final Report Output................................................................................................................................... 249
Directions:................................................................................................................................................ 250

Lab 2 – Row Summary ................................................................................................................................. 255
Objectives:................................................................................................................................................ 255
Report Definition....................................................................................................................................... 255
Final Report Output................................................................................................................................... 255
Directions:................................................................................................................................................ 255

Lab 3 – Column Summary ............................................................................................................................ 258
Objectives:................................................................................................................................................ 258
Report Definition....................................................................................................................................... 258
Final Report Output................................................................................................................................... 258
Directions:................................................................................................................................................ 259

Lab 4 – Extra Value ...................................................................................................................................... 261
Objectives:................................................................................................................................................ 261
Report Definition....................................................................................................................................... 261
Final Report Output................................................................................................................................... 261
Directions:................................................................................................................................................ 261
Analysis Grid …………………………………………………………………………………………………………..265
  Lab 1 – Analysis Grid - Orders Employees and Order Totals……………………………………………………………265
    Objectives: ..............................................................................................................................................265
    Report Definition ...................................................................................................................................265
    Final Report Output .................................................................................................................................265
    Directions: ..............................................................................................................................................266
  Lab 2 – Bookmarks (Analysis Grid) ..............................................................................................................274
    Objectives: ..............................................................................................................................................274
    Report Definition ...................................................................................................................................274
    Final Report Output .................................................................................................................................274
    Directions: ..............................................................................................................................................275
  Lab 3 – Display Bookmarks …......................................................................................................................278
    Objectives: ..............................................................................................................................................278
    Report Definition ...................................................................................................................................278
    Final Report Output .................................................................................................................................278
    Directions: ..............................................................................................................................................279

Dashboards ......................................................................................................................................................283
  Lab 1 – Basic Dashboard .............................................................................................................................283
    Objectives: ..............................................................................................................................................283
    Report Definition ...................................................................................................................................283
    Final Report Output .................................................................................................................................284
    Directions: ..............................................................................................................................................285

Security .........................................................................................................................................................291
  Lab 1 – Security Basics (Standard Security) ..............................................................................................291
    Objectives: ..............................................................................................................................................291
    Report Definition...................................................................................................................................291
    Settings Definition .................................................................................................................................291
    Report Output ......................................................................................................................................292
    Directions: ..............................................................................................................................................293
  Lab 1a – Security Basics (SecureKey) .........................................................................................................301
    Objectives: ..............................................................................................................................................301

XOLAP .........................................................................................................................................................308
  Lab 1 – Simple Dimensions .......................................................................................................................308
    Objectives: ..............................................................................................................................................308
    Definition: ..............................................................................................................................................308
    Report Output ......................................................................................................................................308
Directions:............................................................309
### Glossary

<table>
<thead>
<tr>
<th><strong>AJAX</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>An acronym for <em>Asynchronous JavaScript and XML</em>, AJAX is a web development technique for creating interactive web applications. The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change. This is meant to increase the web page's interactivity, speed, and usability.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>APPLICATION PATH</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>In a Logi application, the Application Path is the URL of the application. It's a standard HTTP URL including the web server name and virtual directory name that has been registered for your application. Examples:</td>
</tr>
<tr>
<td><a href="http://localhost/myfirstapp">http://localhost/myfirstapp</a></td>
</tr>
<tr>
<td><a href="http://194.255.255.10/helloworld">http://194.255.255.10/helloworld</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>APPLICATION TRACE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>A diagnostic HTML page that provides information about program execution. It contains columns for the event, object, data value, and time of occurrence. Clicking a debug link (which appears when turned on in an application's _settings definition) will display this page, as will fatal errors.</td>
</tr>
</tbody>
</table>

This feature is also available via bug icon in the Logi Studio Toolbar:
| **ATTRIBUTE** | A customizable, predefined property of an element, consisting of a name and value. Some attributes, such as ID, are common to all elements and some are required for application execution. |
| **ATTRIBUTE VALUE** | The data portion of an attribute; a user-defined application setting for a specific attribute name. |
| **BLOB** | Acronym for **Binary Large Object**. In a database context, refers to binary data such as images, music files, videos, or executable files that are stored as an object in a single data table column. |
| **CHILD ELEMENT** | An element that hierarchically belongs to the element above it (its parent) in the element tree in a definition. Child elements stay with their parent elements when the parent element is copied, moved, or deleted. |
| **CLOB** | Acronym for **Character Large Object**. In a database context, refers to text data such as documents, HTML pages, and logs that are stored as an object in a single data table column. |
| **CONNECTION** | A communication link to a datasource, including its tables, views and stored procedures. |
### CONNECTION STRING

A value used to make a connection to a database or datasource. May contain a variety of information such as server, database, user id, password, etc. Example:

```sql
Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=Northwind;Data Source=MYDATASERVER
```

### CSS

An acronym for **Cascading Style Sheet**, CSS is a stylesheet language used to create style classes that describe the presentation of an HTML document. Its purpose in Logi applications is to control appearance, alignment, location, and other presentation aspects in report definitions. The CSS specifications are maintained by the [World Wide Web Consortium](https://www.w3.org) (W3C).

With the advent of Themes in Logi 10, users can pick themes that have collections of predefined classes in a style sheet stored with that theme. These classes are associated with the default behavior of elements such as Data Table and Data Table Column. Therefore the report designer need not worry about the background color of a data table or data table cell, once a theme is chosen. These default behaviors can be changed based on choices made in the class attribute of these elements.

### DIVISION

An element used as a container for other elements. The Division element can be conditionally shown or hidden, causing all of the elements it contains (its child elements) to be shown or hidden. Also useful for organizing elements for improved definition readability and for copying groups of elements.

Note that divisions cannot be refreshed using an `Action.RefreshElement` due to java conflicts.

### ELEMENT
A reusable, XML-based component, usually represented by an icon in Logi Studio, used as the visual building block of a Logi application. They are added from a Element Toolbox presented in Logi Studio.

**ELEMENTAL DEVELOPMENT**

The process of creating robust and flexible web-based reporting applications with reusable XML-based components.

**FLOW POSITIONING**

The process of defining layout based upon the hierarchical structure of elements within a report definition. Analogous to *Flow Layout*.

**FORMULA ATTRIBUTE**

A category of element attributes that are able to accept a literal value or a formula. Formulas can be algebraic expressions or functions that are derived from either VBScript or JavaScript and are evaluated at runtime. Note that formulas can also exist in the Label element's Caption attribute. Simply precede the formula with an equal sign.

**HEAT MAP**

The Heat Map chart element provides a graphical representation of data consisting of a rectangular arrangement of many smaller rectangles, where the size and color of each small rectangle represent a respective data value. Heat maps are used for purposes such as DNA tracking in Molecular Biology and stock market analysis.

**IDENTIFIER**

Each @ Token has a list of *identifiers* that are between the period and tilde (~). In the case of @Function these are system defined or (“out of the box”). Some tokens have
identifiers that are user determined. These such identifiers indicate which column in a `@Data` token, a variable in an `@Session` token etc.

**LAYOUT POSITIONING**

The process of defining layout using *fixed* or *absolute* positions for an element or group of elements within a report definition. Analogous to *Grid Layout*.

**PROCESS DEFINITION**

A type of developer-created file or definition, used in Logi applications, consisting of one or more **Tasks**, which contain a hierarchical layout of elements and attributes that defines a specific server-side task to run. Elements that make up a Task are typically ones that start with the work *Procedure*.

Tasks are used to manipulate data, send email, test data conditions, and perform other processing and are used for automated and scheduled events. **Processes are not available in Logi Report**.

**QUERY STRING**

That portion of a web page URL that is used to pass data to the web server. In this example, the Query String follows (but does not include) the question mark (?):

```plaintext
&rdTime=EDT
```

**REMARKED ELEMENT**

An element (including child elements) that have been disabled or "commented out" of a report definition. By default, remarked elements appear in a green font within Studio.

**REPORT DEFINITION**
A type of developer-created file used in Logi applications, which contain a hierarchical layout of elements and attributes that presents both database content and presentation information to a web browser. Basic building block of a Logi application which has a .lgx extension.

**REQUEST PARAMETERS**

That portion of a web page URL used to identify individual data parts to be passed to the web server. In this example, the request parameters consist of a "name" and a "value" pair connected by an equals sign. Each name-value pair is separated from the next by an ampersand (&):

```
```

In a Logi report definition, these parameters are available using @Request tokens. In the above example, the final request parameter is available is `@Request.rdTime~` and its value is "EDT". These parameters are frequently referred to as Querystring Parameters.

**SHARED ELEMENT**

A container element that holds other elements, defined in one report definition but available to be referenced from other report definitions so that elements can be defined once but used many times. This is especially useful for implementing standardized headers and footers. Shared Element elements are referenced in other definitions using the Include Shared Element element.

**SHOW MODES**

A mechanism that makes it possible to hide and show elements in reports based on a data value. The value is often passed to report pages in a Request parameter named "rdShowModes".

**STYLE SHEET**
A developer-created file that contains style classes that control the presentation aspects of Logi reports. See **CSS** above.

### SUPPORT FILES

Style sheets, images, XML data files, HTML documents, VBScript files (.vbs) and JavaScript files (.js) included with a Logi report application.

### TOKEN

A user generated or system generated "placeholder" for a value which is evaluated by the Logi Server Engine at runtime. Analogous to a "macro". Often referred to as “At Sign” or @Tokens. @Tokens have a period after the token name and a tilde at the end of the token.

### TOOL TIP

An attribute for some elements that causes explanatory text to appear when the mouse is hovered over the element. This is very useful for providing immediate context help for elements like TextInput and TextInputArea.

### VALIDATION

The process of ensuring that all required elements are present in a Logi application, with values defined for all required attributes.

### VIRTUAL DIRECTORY

In the context of Microsoft Internet Information Server (IIS), a virtual directory is an **alias** for a physical directory on the web server. This alias is used in a web page URL to request a page from the physical directory. The following example contains the virtual directory "devnet":
When an application is registered in Studio, a virtual directory is created for it by interacting with IIS directly. Under Windows, the IIS Manager program can also be used to create virtual directories manually. By using a virtual directory in the URL, browsers are not required to be given any of the physical storage details.

<table>
<thead>
<tr>
<th>WEB SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Web API that can be accessed over the Internet and executed on a remote system hosting the requested services. A web service provides specific processing and/or data, over the Internet, that developers can incorporate into their own applications. Fees are sometimes charged for the use of a web service.</td>
</tr>
<tr>
<td>Web services can be SOAP based or REST based.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Extensible Markup Language (XML) is a W3C-recommended general-purpose markup language that supports a wide variety of applications. In Logi applications, the source code for definitions is an XML document. Also, the results returned from a datasource into a datalayer is cached as XML data.</td>
</tr>
</tbody>
</table>
Logi Application Architecture Overview

In this document, you'll be introduced to Logi Analytics's application architecture. A key concept, *Elemental Development*, is introduced, along with report definition files, elements and attributes. Elemental Development is the foundation of Logi applications. In this overview, you'll learn how Logi applications deliver content to the web and how to include external files such as images, scripts, and data XMLs. Topics include:

- Understanding Logi Applications
- Introducing *Elemental Development*
- Using External Files with an Logi Application

Understanding Logi Applications

While Logi products are most frequently used as reporting tools, they can also be used to create rich, web-based applications. The Devnet web site, for example, is a Logi application. So, in this document, you'll find the terms "web site" and "application" used interchangeably as well as the terms “web page” and “definition”. In addition, for the purposes of this document, the term "Logi reporting products" refers only to our managed reporting products: Logi Report, and Logi Info.

Logi applications deliver database content in a **presentable** and **accessible** manner as **web pages**. They use Internet technologies and browser software to distribute rich, interactive, and effective reports and applications. Logi reporting tools bring several Internet technologies together, allowing developers to quickly deliver results in meaningful and efficient ways. The Logi framework consists of:

- An integrated development environment (Logi Studio)
- An XML-based modeling language, coded in Report Definition files
- A runtime server (the Logi Server Engine)
Logi reporting tools separate report development from data access, and presentation.

1. Applications and reports are developed using Report Definition files, coded in the Logi elemental based modeling language and stored as XML data. While it is possible to write definitions with any text editor, Logi Studio provides an integrated development environment with lots of tools and helpful wizards that reduce development time.

2. The Logi Report Server parses these XML-based report definitions at runtime and accesses data sources as required.

3. The retrieved data and presentation details are formatted based on the definitions and output as HTML (and javascript) which is then served to your browser for viewing.

This arrangement allows the Logi Server Engine to operate as a stand-alone entity that's completely independent of systems that call upon it for reports. Separate applications, regardless of their host OS, environment, or development language, that wish to embed Logi reports, for example, need only include a standard hyperlink to do so.

Introducing Elemental Development

Elemental Development is the process of creating flexible reporting applications using XML-based objects or "elements". The advantages of Elemental Development are:

- XML elements are reusable and encapsulate specific functionality common to most web applications
- Hierarchical layout of elements makes it easy to manage the presentation layout and functionality of large web-based reports
- A drag-and-drop / point-and-click user interface shields developers from repetitive coding and shortens development time
- Elements and their properties (or "attributes") can be easily added, deleted, moved, and modified to create almost any type of report

Definition Files

Developers using Logi reporting tools create one or more definitions to build interactive web-based report applications. One report definition is equivalent to one dynamic web page. Every Logi application contains at least one report definition and developers will create multiple definitions as the application grows.

Definitions are stored as small .lgx files stored in the folder determined at application creation time. An application with all definition types created will have a folder structure like the following:
The _settings definition (Settings.lgx above) is required in every Logi application and is its main configuration file. It contains elements that define the application's virtual path, debugging capabilities, database connections, security information and more.

Report definitions contained in the _Reports folder define the structure and layout information for one dynamic web page. A report definition typically contains a Report Header element, a Body element and a Report Footer element. The main body of the report can be any combination of charts, dashboards, data tables, key performance indicators, and more.

Process definitions provide a level of automation and contain the logic needed to perform specific tasks within a report application. Process definitions perform valuable functions such as exporting a report to PDF format or emailing it to a group of recipients.

Processes are not available in Logi Report.
**Template** definitions allow developers to take advantage of forms-based reporting (Word, Excel and PDF forms). Template definitions model the "fill-able" **form fields** contained within the source template file. The server fills-in form templates with data on demand either at runtime by the user or automatically at a scheduled time.

Templates are not available in Logi Report.

**From XML to HTML...**

Every definition file created with Logi Studio is an **XML document**; each XML “tag” is represented as an element and its “attributes”. Note the Body Element in the first box below relates to the Body tag in the middle box containing XML.

The example report definition pictured above (left image) contains the Report element "HelloWorld ", which contains a **Body** element. The Body element contains a **Label** element which has a **Caption** attribute.

The XML source code for this report shows the matching XML hierarchy and data. Note the tags as elements and the tag’s attributes.
The Logi Report Server applies an **XSL transform** to the XML code, producing the HTML for presentation.

Elements encapsulate different types of functionality and presentation; attributes enable developers to customize element properties and behavior. For example, the Chart.Pie element creates a pie chart from an associated data layer. The Chart.Pie element contains attributes to specify the height, width, color, border and more.

**Using External Files with a Logi Application**

Developers can include a variety of external files, generically called *Support Files*, in a Logi application.

Logi applications can use the following types of Support Files:

- **Style Sheet** – Cascading Style Sheet (CSS) files for presentation and formatting control
- **Script** – JavaScript or VBScript files for use within process definitions
- **Image** – GIF, JPG, PNG and other types of image file formats
- **Template** – Excel and PDF form template files
- **Data XML** – XML data files for use as a source of data within report definitions
- **Data XSL** – XSL style sheets for transforming data XML files into different XML formats
- **Include HTML** – HTML to be included within the application such as client side JavaScript.
Best Practices

1. As with all software development...begin with the end in mind and KISS (Keep It Superbly Simple).

2. DevNet is your BFF (Best Friend Forever). Go to http://devnet.LogiAnalytics.com for documentation and downloadable sample applications.

3. Debugger Links are your BFF. Always turn on Debugger Links in the development environment and turn off in deployed environments (i.e. QA and Production).

4. Spelling and Case matters. Logi definitions are based on XML which is case sensitive.

5. Always fill in the ID attribute of an element, especially when required.
   a. Use Hungarian notation (see Element Naming Guidelines below) when assigning element IDs.
   b. Required element IDs must always be unique within a Report, Template, or Widget definition, or a Process Task.
   c. Do not use special characters or spaces in report definition names or element IDs. You may use the underscore character.

6. Always set the Connection ID attribute in data layer elements even though it is optional.
   a. The attribute is optional because it will use the first connection listed under the Connections element in the _Settings definition. If another connection is added above the existing connection, there are reports that may break if the Connection ID attribute value in their Data Layers elements is left blank.

7. Save Early, Save Often. You can always undo.

8. When using Logi tokens, always re-check your token syntax - @Token.Identifier~
   a. The most frequently encountered error in training is to leave out the “@” or “~”.

9. When doing SQL comparisons, remember to quote text string comparisons but not number comparisons.

10. When using Multiple Selections in Input Elements, remember to use the IN () syntax in your SQL WHERE clause and use the SingleQuote prefix token to wrap the comma-delimited values in single quotes.

11. Always set the Data Type attribute in all elements. This forces you to check that values are being handled appropriately for their intended use.
12. If an Attribute Name ends in the word “Column” there is no need to use the @Data token in that attribute value. Use the column (or field) name from the data layer without the token.

13. For performance, use SQL instead of Logi Info elements, particularly when doing summary Group By queries, or when filtering with a Where clause.

14. When editing Support Files, always “Open Externally” from within Logi Studio to ensure you are editing the correct file. Your default editor for that file type, as defined in Windows, will open the file.

15. Filters and Conditions always filter through. Your conditional script expressions should evaluate to “True” for the data or element you want to display.

16. When Exporting to Word or Excel, always use the “Native” export action elements.

17. Divisions are your BFF because they:
   a. Provide report structure by acting as containers for other elements.
   b. Allow the easy application of Security Right ID’s, Conditions, Style Classes, and Show Mode attributes for entire subsections of a report.
   c. However: Do not use an ID of a Division when specifying the Element ID attribute in an Action.RefreshElement

18. When using Templates:
   a. Always populate Excel Template Support Files with dummy data to test formatting and formulas.
   b. Always edit Template Support Files in Studio (Open Externally).
   c. Always assign the Excel Worksheet Name attribute in Pattern Block elements in Excel Template Definitions.
   d. For PDF or Word Templates, use logical field names and use the Template Support file’s field name as the Element ID in the Template Definition file.

19. Keep shared elements in separate definitions that are used only as containers for shared elements.
   a. Create a folder named Shared Elements for all of your Shared Element definitions.
   b. The Shared Element definition files should be used to organize your shared elements into logical groups to make them easier to find. E.g. Shared.Buttons, Shared.Panels, Shared.Filters, etc.

20. Use single quotes around all @ Tokens that contain text in SQL statements and in Formula and Condition attribute scripts.
21. Use query builders to generate at least the field names in the Select clause of your SQL queries to ensure standard assignment of case to field names among the entire development team. This makes it easier to share code among the development team, since the field names in query result sets are stored in XML and are therefore case sensitive.

22. When using the Studio query builder it is best to link the tables and choose all the columns before typing anything in the lower section (where the query is actually composed). Any new text may be overwritten if columns are checked.

23. Develop content meant for a Dashboard Panel in its own definition and add the content to the dashboard by including as a shared element, or as a sub-report in a dashboard panel, or, when necessary, copying and pasting into the dashboard panel.
Element Naming Guidelines

Logi Analytics recommends that you use "Hungarian notation" when creating element IDs. In this notation, an element ID starts with one or more lower-case letters which are mnemonics for the type or purpose of that element, followed by whatever name you chose. This section is an excerpt of a larger reference document under devnet in the training materials section.

The use of Hungarian notation provides numerous benefits, including ensuring unique element IDs and encouraging consistency. This can especially helpful for developers who are new to Logi reporting products and who have not worked with them long enough to recognize the element icons. For example, a Data Table element that presents customer data might have an ID of "dtCustomers"; an Image element displaying a logo might be "imgLogo".

When it comes to formulating element IDs, it’s not important that you conform to some universal standard; what is important is that you use some consistent, understandable standard, perhaps your own or your company's, regularly.

Here are some suggested guidelines and mnemonics for element ID prefixes that you may care to adopt:

<table>
<thead>
<tr>
<th>Guideline</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the first 2-3 letters of the element name</td>
<td>Division = div</td>
</tr>
<tr>
<td>Use the first few letters of each sub-part of the element name</td>
<td>AnalysisGrid = ag</td>
</tr>
<tr>
<td></td>
<td>DataTable = dt</td>
</tr>
<tr>
<td></td>
<td>DataLayer = dl</td>
</tr>
<tr>
<td></td>
<td>ExtraColumnHeader = ech</td>
</tr>
<tr>
<td></td>
<td>SubDataAggregateColumn = sdac</td>
</tr>
<tr>
<td>Use all or part of the element name without its vowels (three letters max)</td>
<td>Dashboard = dsh</td>
</tr>
<tr>
<td></td>
<td>Image = img</td>
</tr>
<tr>
<td></td>
<td>Label = lbl</td>
</tr>
<tr>
<td>Use sound-related, 3-letter combinations</td>
<td>Button = btn</td>
</tr>
<tr>
<td></td>
<td>Image = img</td>
</tr>
<tr>
<td>Apply any of the above to the second part of a two-part element type</td>
<td>Chart.Pie = pie</td>
</tr>
<tr>
<td></td>
<td>AnimatedGauge.Needle = andl</td>
</tr>
<tr>
<td>Apply any of the above to user input elements</td>
<td>Input Checkbox = chk</td>
</tr>
<tr>
<td></td>
<td>Input Hidden = hdn</td>
</tr>
<tr>
<td></td>
<td>Input Radio Buttons = rb</td>
</tr>
<tr>
<td></td>
<td>Input Select List = sel</td>
</tr>
<tr>
<td></td>
<td>Input Text = txt</td>
</tr>
<tr>
<td>Use first three letters of the final part of element name if it denotes an object type</td>
<td>Data Table Column = col</td>
</tr>
<tr>
<td></td>
<td>DifferenceColumn = dcol</td>
</tr>
<tr>
<td>Apply &quot;Camelback case&quot; when combining mnemonics (capitalize the second mnemonic)</td>
<td>Data Table ID = dtOrderDetails</td>
</tr>
<tr>
<td></td>
<td>Label ID = lblCalFreight</td>
</tr>
</tbody>
</table>
Tokens

A "token" is a **variable** of sorts that represents a value and is resolved at run time. The use of tokens in attributes or “tokenization” is a key skill used in developing more complex reporting.

Token Syntax

Tokens are extremely useful in Logi applications because of their dynamic nature. You may be familiar with variables known as "macros" in other programming languages which behave similarly. When used in report and process definitions, tokens take this general format:

```plaintext
@<TokenType>.<Identifier>~
```

They must be preceded by the "@" symbol and must end with the ~ (tilde) symbol, and they're **case-sensitive**. They can be used in many, but not all, element attributes to provide a dynamic value. Here are three examples of tokens in action:

1. In the example above, a token is used to provide part of a Label element **Caption**. At run time, the real server name will be substituted for the token. The output will look like this:

   Hello World! from localhost

   This is commonly referred to as "Tokenization". Tokens are extremely literal and are "data type neutral". For instance, in the above example, even though the server name is a string, notice that no effort was needed to **concatenate** (using + or & symbols) the token to the "My server is:" part of the Caption.

   Note: The use of quotes and the plus or ampersand symbols are still required in **Formula** and **Condition** attributes of various elements, since those attributes use script syntax (either VBScript or JavaScript).
2. Remember that tokens are case-sensitive! This is especially important in this next example:

In this case, a token is used to represent the value in the **OrderID** column for each row of data in the Datalayer. The spelling and capitalization in the column name **must exactly match** those in the column returned to the Datalayer.

3. In the final example, shown above, a token is used in a **Condition** attribute. The Condition attribute controls whether or not the divTable element (and its child elements) are visible; if the attribute is True, the divTable element is shown; if it's False, the element and its children are hidden.

In this case, the token is used in an **equation or expression** that provides the True or False value. @Request-type tokens represent data POSTed by an HTML form and values from parameters in the **query string** used to call reports. So, if the URL for this report was:

http://localhost/Logi_Training_Labs_V11(rdPage.aspx?&rdReport=DataTables.Lab1_Basic_DataTable&Mode=Table

then the divTable element would be **shown** along with its children. These URLs:
would cause the divTable element to be hidden. (If the Mode parameter in the query string is nonexistent, the token is empty and "" <> "Table" so the equation evaluates to False.)

Note that in the Condition attribute equation the token is surrounded by **double-quotes**. This is because its value is being compared to a string value. Similar syntax is required for a formula in a calculated column. If the Mode value in the query string was a number and it was being compared to a number, then the double-quotes would not be needed.

### Token Types

The following tokens types are available in Logi reporting products:

<table>
<thead>
<tr>
<th>@Token Type</th>
<th>Description</th>
<th>Logi Studio Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Application</td>
<td>Provides access from within report and process definitions to variables defined in the code of integrated .NET applications (application-scope variables).</td>
<td>Info, Report</td>
</tr>
<tr>
<td>@Chart</td>
<td>Retrieve data from a DataLayer element within a Chart element.</td>
<td>Info, Report</td>
</tr>
<tr>
<td>Format: @Chart.&lt;column name&gt;<del>&lt;br&gt;Example: @Chart.OrderDate</del></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Compare</td>
<td>Represents the True or False result of a Compare Filter evaluation. <em>(v11.0.127+) Not available in Logi Report.</em></td>
<td>Info</td>
</tr>
<tr>
<td>Format: @Compare.&lt;Compare Filter Element ID&gt;<del>&lt;br&gt;Example: @Compare.fltContainsDate</del></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Constant</td>
<td>Retrieve the value of a user-defined constant in any definition within a single report application. Constants are set in the _settings definition using the Constant element.</td>
<td>Info, Report</td>
</tr>
<tr>
<td>Format: @Constant.&lt;value of name attribute&gt;<del>&lt;br&gt;Example: @Constant.CompanyName</del></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@Cookie</td>
<td>Retrieves information saved as a cookie. Cookies can be set in Logi applications using the SaveInCookie attribute of certain Input elements or, in Logi Info, the SetCookieVars procedure.</td>
<td>Info, Report</td>
</tr>
<tr>
<td>Format: @Cookie.&lt;InputFieldID&gt;<del>&lt;br&gt;Example: @Cookie.Order</del></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Description</td>
<td>Format</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>@Data</td>
<td>Retrieve data from a DataLayer element.</td>
<td>@Data.&lt;column name&gt;~</td>
</tr>
<tr>
<td>@Date</td>
<td>Retrieve relative dates for reports and report scheduling (see Date Identifiers list below).</td>
<td>@Date.&lt;date identifier&gt;~</td>
</tr>
<tr>
<td>@FileUpload</td>
<td>Returns information in a procedure task about a file upload process (see File Upload Tokens, below).</td>
<td></td>
</tr>
<tr>
<td>@Function</td>
<td>Retrieve specific data using a predefined function (see Function Identifiers below).</td>
<td>@Function.&lt;function name&gt;~</td>
</tr>
<tr>
<td>@Heatmap</td>
<td>Retrieve data from a DataLayer element within a Heatmap element.</td>
<td>@Heatmap.&lt;column_name&gt;~</td>
</tr>
<tr>
<td>@Input</td>
<td>Used inside VBScript and JScript routines for string replacement.</td>
<td>@Input.&lt;value ID&gt;~</td>
</tr>
<tr>
<td>@Local</td>
<td>Retrieve data from the first row of a data layer below the LocalData element. Local data is accessible anywhere within the report definition.</td>
<td>@Local.&lt;value ID&gt;~</td>
</tr>
<tr>
<td>@Measure</td>
<td>Represents the value of XOLAP measures; used only in the XOLAP Formula attribute of the XOLAP Calculated Measure element. (v10.2.224+)</td>
<td>@Measure.[&lt;measure name&gt;]~</td>
</tr>
<tr>
<td>@Procedure</td>
<td>Returns information about operations within procedure tasks, including datasource interactions, file system and email operations, and associated error messages (see Special Identifiers below).</td>
<td>@Procedure.&lt;procedure ID&gt;.&lt;value ID&gt;~</td>
</tr>
<tr>
<td>Token</td>
<td>Description</td>
<td>Info, Report</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>@Request</td>
<td>Retrieve data posted by an HTML form or supplied as a query string parameter in the URL.</td>
<td>Info, Report</td>
</tr>
<tr>
<td></td>
<td>Format: @Request.&lt;request variable name&gt;~</td>
<td>Info, Report</td>
</tr>
<tr>
<td></td>
<td>Example: @Request.EmployeeID~</td>
<td>Info, Report</td>
</tr>
<tr>
<td>@Session</td>
<td>Retrieve values from ASP.NET Session variables.</td>
<td>Info, Report</td>
</tr>
<tr>
<td></td>
<td>Format: @Session.&lt;session param ID&gt;</td>
<td>Info, Report</td>
</tr>
<tr>
<td></td>
<td>Example: @Session.SessionID~</td>
<td>Info, Report</td>
</tr>
<tr>
<td>@SingleQuote</td>
<td>Prefix a token with @SingleQuote to wrap the values returned in single quotes. The @Data, @Chart and @Heatmap tokens cannot be prefixed by @SingleQuote.</td>
<td>Info, Report</td>
</tr>
<tr>
<td></td>
<td>Format: @SingleQuote.&lt;token&gt;.&lt;identifier&gt;~</td>
<td>Info, Report</td>
</tr>
<tr>
<td></td>
<td>Example: @SingleQuote.Request.MyElementID~</td>
<td>Info, Report</td>
</tr>
<tr>
<td>@Summary</td>
<td>Deprecated. No longer a valid token.</td>
<td>Info, Report</td>
</tr>
</tbody>
</table>

*Deprecated.*
@Functions

The identifiers listed in the table below are used with the @Function token. For example, to return the current date and time, use @Function.DateTime~.

<table>
<thead>
<tr>
<th>Identifiers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppPhysicalPath</td>
<td>Returns the complete physical path of the application's virtual directory on the web server. Example: C:\inetpub\wwwroot\MyLogiApp</td>
</tr>
<tr>
<td>BrowserDecimalChar</td>
<td>Returns the client browser’s decimal character.</td>
</tr>
<tr>
<td>BrowserThousandsSeparatorChar</td>
<td>Returns the client browser's thousand separator character.</td>
</tr>
<tr>
<td>BrowserMinorVersion</td>
<td>Returns the browser's minor version.</td>
</tr>
<tr>
<td>BrowserMajorVersion</td>
<td>Returns the browser's major version.</td>
</tr>
<tr>
<td>BrowserVersion</td>
<td>Returns the browser version.</td>
</tr>
<tr>
<td>Date</td>
<td>Returns the current date as a short date string. Example: 05/21/2004</td>
</tr>
<tr>
<td>DateTime</td>
<td>Returns the current date and time as a short date string. Example: 05/21/2004 10:21:37</td>
</tr>
<tr>
<td>FileUpload</td>
<td>Returns information in a procedure task about a file upload process. See</td>
</tr>
<tr>
<td>GUID</td>
<td>Returns a Globally Unique Identifier string. This function is useful for creating unique filenames. Example: a949f8c9-a83d-46fd-970b-2bfb625bd2ab</td>
</tr>
<tr>
<td>HostAddress</td>
<td>Returns the IP address of the client web browser.</td>
</tr>
<tr>
<td>LastErrorMessage</td>
<td>Returns the last error message string for any procedure executed in the current session.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PageCount</td>
<td>Returns the number of pages. This function only works with the PrintablePaging element.</td>
</tr>
<tr>
<td>PageNumber</td>
<td>Returns the current page number. This function is useful for interactive and printable paging.</td>
</tr>
<tr>
<td>QueryString</td>
<td>Returns the entire query string (everything after the &quot;?&quot;).</td>
</tr>
<tr>
<td>Referer</td>
<td>Returns the URL of the calling web page. (v8.0.31)</td>
</tr>
<tr>
<td>RowNumber</td>
<td>Displays the current row number. This function applies only to DataTable elements.</td>
</tr>
<tr>
<td>ServerName</td>
<td>Returns the name of the web server. E.g.: localhost</td>
</tr>
<tr>
<td>SessionID</td>
<td>Returns the current session ID.</td>
</tr>
<tr>
<td>TimeUtc</td>
<td>Returns the current UTC (Greenwich Mean Time) time, in 24-hour clock, as hh:mm:ss</td>
</tr>
<tr>
<td>UserCulture</td>
<td>Returns the browser's primary language string. E.g.: en-us</td>
</tr>
<tr>
<td>UserID or UserName</td>
<td>Returns the current user ID or name. (Logi Security must be in use).</td>
</tr>
<tr>
<td>UserRights</td>
<td>Returns a comma-delimited list of the current user's security rights. (Logi Security must be in use) (Info &amp; Info Lite v8.0.34+)</td>
</tr>
<tr>
<td>UserRoles</td>
<td>Returns a comma-delimited list of the current user's security roles. (Logi Security must be in use) (Info &amp; Info Lite v8.0.34+)</td>
</tr>
<tr>
<td>UniqueIdentifierValue</td>
<td>Used with Logi Intelligence Server with incremental appends: returns the last value in the data mart table for the specified unique identifier column.</td>
</tr>
</tbody>
</table>
@Date Token

The identifiers listed in the table below are used with the @Date token. For example, to return yesterday's date, enter @Date.Yesterday~.

Dates are returned by default in the yyyy-M-d format, which does not include leading zeros for single-digit day or month values. Examples: April 1st = "2009-4-1", April 25th = "2009-4-25", October 31st = "2009-10-31".

If needed, you can override this default format by specifying a new format in the Globalization element's Default Input Date Reformat attribute. The Default Input Date Reformat attribute ensures that dates entered by the user are formatted identically to dates returned from @Date identifiers. This is especially useful if you are using the @Date tokens with Validation.Date.

To work with the FiscalQuarter and FiscalYear tokens, use the Globalization element to set the first day of the fiscal year.

The Globalization element can be added in the _settings definition.

@Date.TodayUtc~ returns the UTC (Greenwich Mean Time) date.

<table>
<thead>
<tr>
<th>Identifiers</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Today</td>
<td>ThisWeekStart</td>
<td>ThisQuarterStart</td>
</tr>
<tr>
<td>TodayUtc</td>
<td>ThisWeekEnd</td>
<td>ThisQuarterEnd</td>
</tr>
<tr>
<td>Yesterday</td>
<td>NextWeekStart</td>
<td>NextQuarterStart</td>
</tr>
<tr>
<td>Tomorrow</td>
<td>NextWeekEnd</td>
<td>NextQuarterEnd</td>
</tr>
<tr>
<td>10DaysAgo</td>
<td>LastWeekStart</td>
<td>LastQuarterStart</td>
</tr>
<tr>
<td>30DaysAgo</td>
<td>LastWeekEnd</td>
<td>LastQuarterEnd</td>
</tr>
<tr>
<td>60DaysAgo</td>
<td>ThisMonthStart</td>
<td>ThisYearStart</td>
</tr>
<tr>
<td>90DaysAgo</td>
<td>ThisMonthEnd</td>
<td>ThisYearEnd</td>
</tr>
</tbody>
</table>
### Special @Request Identifiers

As mentioned earlier, @Request tokens represent data POSTed by an HTML form and values from parts of the query string used to call reports. The following special @Request token identifiers are also reserved words and have special meanings in Logi reports:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;TabsElementID&gt;</code></td>
<td>When a page containing a Tabs element is submitted, a request variable is automatically generated named for the ID of the Tabs element and assigned the value of the ID of the currently selected Tab Panel. Example: @Request.MyTab~ = &quot;Tab1&quot;</td>
</tr>
<tr>
<td>rdReport</td>
<td>Specifies a report definition.</td>
</tr>
<tr>
<td>rdProcess</td>
<td>Specifies a process definition to execute.</td>
</tr>
<tr>
<td>rdTaskID</td>
<td>Specifies a Task ID to execute.</td>
</tr>
<tr>
<td>rdPrompt</td>
<td>When set to True, a page listing all @Request tokens referenced in the definition file appears. The user can display default values for each parameter by specifying rdPromptxxxxx at the prompt, where xxxxx is the name of the request parameter.</td>
</tr>
<tr>
<td>rdShowModes</td>
<td>Specifies ShowModes defined for display in the current report.</td>
</tr>
<tr>
<td>rdDebugPdf</td>
<td>When set to True, runs the PDF report and returns the HTML rather than generating a PDF from the HTML.</td>
</tr>
<tr>
<td>rdReportFormat</td>
<td>Specify one of these values: HtmlExport, Excel, Word, PDF or CSV.</td>
</tr>
<tr>
<td>rdEmbeddedSubReport</td>
<td>Specify either True or False.</td>
</tr>
</tbody>
</table>
rdPaging  Specify either Interactive, Printable or NoPaging.

rdTaskLogFilename  Returns the name of the log file created by Task Manager for a specific job. This token is used within the Process definition that creates the job.

**Special Identifiers for Other Token Types**

<table>
<thead>
<tr>
<th>@Token Type</th>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Chart</td>
<td>rdCrosstabColumn-n~</td>
<td>Identifies a specific datalayer column in a crosstab chart, where n equals the index, beginning with 1, of the column.</td>
</tr>
<tr>
<td>@Chart</td>
<td>rdExtraColumnID~</td>
<td>In a stacked bar chart, references one of the parts of the stack (a separate data value).</td>
</tr>
<tr>
<td>@Data</td>
<td>rdCrosstabColumn~</td>
<td>Returns a crosstab column header.</td>
</tr>
<tr>
<td>@Data</td>
<td>rdCrosstabValue~</td>
<td>Returns a crosstab row value.</td>
</tr>
<tr>
<td>@Data</td>
<td>rdCrosstabValCount~</td>
<td>Returns the number of rows that were used to calculate a crosstab row value.</td>
</tr>
<tr>
<td>@Data</td>
<td>rdSalesforceTable~</td>
<td>For Salesforce.com only: retrieves table names returned from a LIST TABLES query. Requires Logi 9.</td>
</tr>
<tr>
<td>@Data</td>
<td>rdSalesforceField~</td>
<td>For Salesforce.com only: retrieves field names returned from a LIST &lt;tablename&gt; query. Requires Logi 9.</td>
</tr>
<tr>
<td>@Procedure</td>
<td>myProcedureID.ColumnName~</td>
<td>After Procedure.SQL executes, contains returned data from first row for named column. Procedure.SQL element's SQL Return Type attribute must be set to &quot;FirstRow&quot;.</td>
</tr>
<tr>
<td>@Procedure</td>
<td>myProcedureID.ErrorMessage~</td>
<td>Returns the last error message string for the procedure with the specified ID. Usage: @Procedure.MyProcedureID.ErrorMessage~</td>
</tr>
<tr>
<td>@Procedure</td>
<td>myProcedureID.MethodName~</td>
<td>After Procedure.Web Service executes, contains data returned from external web service. Web Service Method element used in procedure must have its Return Type attribute set to &quot;String&quot;.</td>
</tr>
<tr>
<td>@Procedure</td>
<td>myProcedureID.rdReturnValue~</td>
<td>After Procedure.SP executes, contains the stored procedure's return value.</td>
</tr>
<tr>
<td>@Procedure</td>
<td>myProcedureID.StoredProcedureOutputParamID~</td>
<td>After Procedure.SP executes, returns the data in an SP output parameter with the specified ID.</td>
</tr>
</tbody>
</table>
**File Upload Tokens**

Information about file upload operations is available in process tasks using several types of tokens. An InputFileUpload element is used in a report definition to receive the file and path information for the file to be uploaded and passes this information to a procedure task when the report is submitted. The task uses a SaveFileUpload procedure to save the uploaded file. Two different types of tokens are used to return the data from these two separate steps in the upload operation. Due to the nature of the upload protocol, the values returned by the tokens are provided after the upload has occurred. For more information, see the DevNet document Uploading Files.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@FileUpload.UploadFileName~</td>
<td>Returns the file name and extension entered in the InputFileUpload element, without any path information.</td>
</tr>
<tr>
<td>@FileUpload.UploadFileExtension~</td>
<td>Returns the file extension only of the file name entered in the InputFileUpload element.</td>
</tr>
<tr>
<td>@Procedure.myProcedureID.UploadFileName~</td>
<td>Returns the file name and extension entered in the InputFileUpload element, without any path information.</td>
</tr>
<tr>
<td>@Procedure.myProcedureID.UploadFileExtension~</td>
<td>Returns the file extension only of the file name entered in the InputFileUpload element.</td>
</tr>
<tr>
<td>@Procedure.myProcedureID.UploadFileContentType~</td>
<td>Returns the file's mime or content type string.</td>
</tr>
<tr>
<td>@Procedure.myProcedureID.UploadFileLength~</td>
<td>Returns the length of the uploaded file, in bytes.</td>
</tr>
</tbody>
</table>
## Summary of Training Labs

<table>
<thead>
<tr>
<th>Topic / Lab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>New Application</strong></td>
<td></td>
</tr>
<tr>
<td>New application / Hello world</td>
<td>Students will create an application on a local web server using the new application wizard. The wizard is cancelled midway through the process and prior to creating the connection to the database. This is done to facilitate a discussion of the general element in the settings definition. In this lab we will also confirm the database connection. This discussion is followed by a simple hello world example that will verify the communication with the web server.</td>
</tr>
<tr>
<td><strong>Data Tables</strong></td>
<td></td>
</tr>
<tr>
<td>Lab1 Basic DataTable</td>
<td>A basic data table is created manually with a Datalayer.Sql using a connection to Northwind. Once a single data table column is created manually, the wizard is introduced to create the remaining columns. Finish by adding the interactive paging element using the wizard as well.</td>
</tr>
<tr>
<td>Lab2 NumberedPages</td>
<td>A copy of the first lab is created followed by an exploration of the options available in the interactive paging element.</td>
</tr>
<tr>
<td>Lab3 MultiColumnSort</td>
<td>The sort options are explored by making modifications to the ShipCountry to allow sorting of the ShipCity as well.</td>
</tr>
<tr>
<td>Lab4 SummaryRow</td>
<td>A summary row is created in this lab that will allow the quick aggregation of data from a column. In this lab the freight is totaled up using a summary element in the freight column of the orders table.</td>
</tr>
<tr>
<td>Lab5 CustomSummaryRow</td>
<td>This lab is built upon Lab4 and demonstrates the customization possible in the standard summary row. Changes to text, addition of more aggregates and formatting is shown.</td>
</tr>
<tr>
<td><strong>Data Layers</strong></td>
<td></td>
</tr>
<tr>
<td>Lab1 CalculatedColumns</td>
<td>Once DataTables are covered, more attention is given to the data layer. All data layer elements are reviewed at a high level followed by the creation of calculations such as percent of total and time period columns along with a simple algebraic calculation.</td>
</tr>
<tr>
<td>Topic / Lab</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Lab2_Special_Functions</td>
<td>VB Script functions are explored here to show the use of more involved functions. A calculated column which calculates the difference between two dates is shown.</td>
</tr>
<tr>
<td>Lab3_KPI</td>
<td>This lab is built upon Lab2 and explores the use of additional style sheet classes. At this point the concept of style sheets is discussed in more detail.</td>
</tr>
<tr>
<td><strong>Exports</strong></td>
<td></td>
</tr>
<tr>
<td>Lab1_Excel</td>
<td>A basic data table is copied from the DataTables lab1 and a button is added in the footer to export the data into Excel.</td>
</tr>
<tr>
<td>Lab2_PDF</td>
<td>A copy of Lab1 is created and an additional button is added to export to PDF. The concept of ShowModes is demonstrated to allow the user to hide the buttons on the exported PDF.</td>
</tr>
<tr>
<td>Lab3_PrintablePaging</td>
<td>(Optional) - This lab demonstrates other capabilities for the formatting of headers and footers in reports that are exported.</td>
</tr>
<tr>
<td><strong>User Inputs</strong></td>
<td></td>
</tr>
<tr>
<td>Lab1_SelectList_Submit</td>
<td>A basic data table is copied from the DataTables Lab1 and a select list is created in the header that contains a list of countries. A submit button is created to refresh the list of orders based on the selected country.</td>
</tr>
<tr>
<td>Lab2_SelectList_EventDriven</td>
<td>Events are discussed in this lab. We will remove the submit button and replace it with the ability to engage some action upon the simple selection from the list.</td>
</tr>
<tr>
<td>Lab3_MultiSelect</td>
<td>This lab builds upon Lab1 and allows the user to select several countries. The use of the &quot;in&quot; clause in SQL is discussed and demonstrated along with the SingleQuote token.</td>
</tr>
<tr>
<td>Lab4_CascadingSelects</td>
<td>A second select is added to a copy of Lab2. The select list for the country will update the select list for the city. Both lists will update the list of orders.</td>
</tr>
<tr>
<td>Lab5_InputValidation</td>
<td>The validation elements that are available to user inputs are explored and demonstrated along with the inclusion of client side JavaScript (optional) to validate an email address.</td>
</tr>
<tr>
<td>Topic / Lab</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Lab1_Grouping_Alternatives</td>
<td>Here the students will see how processing of data can be achieved through the use of Logi elements or by using SQL Statements in the Data Layer</td>
</tr>
<tr>
<td>Lab2_Grouped_Flat_Table_and_Aggregates</td>
<td>This report shows how grouping and aggregates can be combined into one report.</td>
</tr>
<tr>
<td>Lab3_Drilldown_SubDataLayer</td>
<td>This is a drilldown exercise that will show how a three-level drilldown can be attained with one definition</td>
</tr>
<tr>
<td><strong>Charts</strong></td>
<td></td>
</tr>
<tr>
<td>Lab1_PieChart</td>
<td>Creation of a simple pie chart.</td>
</tr>
<tr>
<td>Lab2_Exploding_Wedge</td>
<td>Simple animated pie chart is created with an exploded wedge that is based on a business rule.</td>
</tr>
<tr>
<td>Lab3_Relevance_Filter</td>
<td>An &quot;all others&quot; wedge is created using the special data layer filter.</td>
</tr>
<tr>
<td>Lab4_Drilldown</td>
<td>A pie chart of category sales and a simple data table of related products are shown in this drilldown exercise. The user clicks on a pie slice and the products appear to the right in a simple products data table.</td>
</tr>
<tr>
<td>Lab5_BarChart</td>
<td>Simple Bar chart is created</td>
</tr>
<tr>
<td>Lab6_ExtraDataColumn</td>
<td>An additional series of data is added to a standard bar chart</td>
</tr>
<tr>
<td>Lab7_ExtraGridLayer</td>
<td>An additional series of data is added to a standard bar chart in the form of a line chart combined with the bar chart.</td>
</tr>
<tr>
<td><strong>Processes</strong></td>
<td></td>
</tr>
<tr>
<td>Lab1_Email_PDF</td>
<td>A process is created to email a user a PDF of a report that is copied from DataTables - Lab1_Basic_DataTable. This process will be used later in scheduling.</td>
</tr>
<tr>
<td><strong>Templates</strong></td>
<td></td>
</tr>
<tr>
<td>Lab1_Excel</td>
<td>A simple spreadsheet is created by the users and stored on their local &quot;web server'. This spreadsheet is then augmented with data from Northwind. A discussion of the difference between exports and templates is also facilitated.</td>
</tr>
<tr>
<td><strong>Crosstabs</strong></td>
<td></td>
</tr>
<tr>
<td>Topic / Lab</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Lab1_Basic_Crosstab</td>
<td>Basic crosstab table is constructed manually.</td>
</tr>
<tr>
<td>Lab2_Row_Summary</td>
<td>Lab1 is copied and a row summary is created.</td>
</tr>
<tr>
<td>Lab3_Column_Summary</td>
<td>Lab2 is copied and a column summary is created.</td>
</tr>
<tr>
<td>Lab4_Extra_Value</td>
<td>An additional metric to be aggregated is added to a copy of Lab3</td>
</tr>
</tbody>
</table>

### Analysis Grid

| Lab1_Orders_Employees_OrderTotals | Analysis grid is explored using a multiple table query built in Northwind. In this lab the students learn about bookmarks. A bookmark "manager" is built if time permits...otherwise a pre-built bookmark manager is given to the students and is explored in the workspace. |
| Lab2_Bookmarks | Having demonstrated the Analysis Grid and its temporary nature, bookmarks are introduced. Functionality to add a bookmark is reviewed. |
| Lab3_Bookmark_Manager | The bookmark created in the previous lab is extended to a new definition that allows the user to recall or edit an existing bookmark. |

### Dashboards

| Lab1_Basic_Dashboard | A simple dashboard is created from elements copied from charts and crosstabs. |
| Lab2_SharedElements | The pie chart in both the pie chart lab and the dashboard is replaced by a shared element. The concept of "tokenization" is explored as well. |

### Security

| Standard Security | A copy of a basic data table is used to demonstrate roles, rights and users stored in a separate security database. |

### Scheduling

<p>| Scheduler Builder | The scheduling topic is explored first through the walkthrough of a pre-build definition to create a &quot;job&quot; or scheduled task. The task created in the Processes topic will be used as the task to be &quot;scheduled&quot;. |</p>
<table>
<thead>
<tr>
<th>Topic / Lab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>My Scheduled Jobs</td>
<td>This will allow the user to see and maintain scheduled &quot;jobs&quot;.</td>
</tr>
<tr>
<td>XOLAP</td>
<td></td>
</tr>
<tr>
<td>Lab1_SimpleDimensions</td>
<td>Simple XOLAP demonstration with a single level for each dimension.</td>
</tr>
<tr>
<td>Lab2_MultipleLevels</td>
<td>A copy of Lab1 is used to show how multiple levels can be used in a drilldown fashion.</td>
</tr>
<tr>
<td>Lab3_Drilldown.Elements</td>
<td>Pre-established drilldowns are explored along with additional metrics.</td>
</tr>
</tbody>
</table>
New Application

A Logi Info Application is a special web site that makes use of a Logi Analytics engine to produce content and visualizations. Logi applications can exist on the same web application server as non-Logi applications. The .NET web application server that we use in class is IIS (Internet Information Services), but Java-based web application servers (e.g. Tomcat, WebSphere, et. al.) running on Windows or Linux servers may also be used.

New Application / Hello World

Summary:

In this lab we review the steps to create a new application for use in the remainder of the labs.

Objectives:

- Create a New Application which will be used throughout the rest of the labs
- Confirm a connection to Northwind
- Confirm the registration with IIS with a Hello World example
Directions

1. From Logi Studio Choose the File Menu and New Application.

   - The type of application is next. This decision is based on the web server that is being used. In class these labs are built for the IIS web application server, which is the .NET option. Choose .NET and click OK.

   - The New Application window appears next. This will determine the location of the application. The default Folder is C:\inetpub\wwwroot which may be changed using the Browse button. This will be the parent folder of the application folder. The Name will become the folder name of the application folder to be created. For purposes of these labs keep the folder as the default and enter a Name for the application (no special characters or spaces) and press OK.
In this manual, we use Logi_Training_Labs_V11. You can use whatever name you wish.

- Once OK is clicked the Studio creates the new web application folder under the parent folder and copies the Logi Server Engine files to the application folder.

- Click Next on the next two introductory screens to prepare a new application.

At this point IIS “registers” the web site that Logi Info Studio has just created. Some DOS Command windows may temporarily appear. The rest of the wizard will prompt for additional settings.

- Choose a Theme and click Next. The theme we are using in this manual is Professional Blue.

- Enter an Application Caption for this application. Do not use special characters. Spaces are OK. This text will appear at the top of the browser window in the title bar. Enter the caption and click Next.
Prepare a new application.

What should appear as the title for this application?

Application Caption: Training Labs

- Click **Next** when prompted for the **Application Path** to accept the default path.

The wizard assumes that you wish to immediately create your first report and prompts you to create a database connection for it. For class purposes, a discussion of the connection element is in order so we will cancel the wizard at this point and add a database connection in step 2 below.

- When the wizard prompts you to “Create a database connection now?”, click **Cancel** to stop the wizard.

2. Add a database connection element for **Northwind** and test it.

Connection elements are defined in the **_Settings** definition. The **_Settings** definition is open when the wizard is cancelled. If necessary, double-click the **_Settings** definition in the Application Panel to reopen it.

- Select the **Connections** element in the settings definition.
- In the upcoming wizard, toggle the option for “Show this when adding elements that have wizards?” off.
- Enter the **Database**, **ID**, **Server**, **User**, and **Password** attribute values for the `Connection.SqlServer` element as shown.

You will have to substitute the **User** and **Password** attribute values that pertains to your implementation of Sql Server / Northwind.

- Test the connection by clicking on the disk icon in the toolbar.
If the connection is functioning properly, the following Database Browser window will appear.

![Database Browser Window](image)

- Close the Database Browser.

**Now that a connection element is in place, reports can be created.**

3. Add a *Hello World* example report to test IIS (Internet Information Services).

- Double-click on the **Default** definition in the Application Panel.

![Application Panel](image)

- Click on the **Body** element in the Workspace Panel.
- Double-click the **Label** element in the General Elements section of the Element Toolbox.

- Enter the **Caption** and **ID** attribute values to the **Label** element as shown.
- Select the **Body** element to add a **Horizontal Rule** element. Double-click the **Horizontal Rule** element in the General Elements section of the Element Toolbox.

- Select the **Body** element again and add a second **Horizontal Rule** element from the Toolbox. Press the **F7** key to move one of the **Horizontal Rule** elements above the **Label** element.
Normally, new child elements are added to the bottom of the parent element’s children. They can be moved as needed.

Using F7 will move elements UP within the parent elements children. Using F8 will Move elements Down within the parent elements children. Toolbar Icons can also be used to reorder elements (see below).

- Click on the left disk icon (Save Selected File) in the Toolbar to save the current definition.
- Click on the right disk icon (Save All Opened Files) to save all definitions open in the Workspace Panel.

- Click on the Preview button at the bottom of the Workspace Panel to preview the report.

- It may take a few seconds but you should see the output in a browser window within the Workspace Panel.
Use the **Preview** button when you want to check for “show stoppers”. Once the definition renders in the preview window it is a good practice to run the definition in the browser. This will provide a more realistic view from a “real estate” perspective. Both the **Preview** button and the **Run in Browser** feature will save your active definition before running the report, but **will not** save any other open definitions.

4. Run the Hello World Definition in a new browser window.

   - Right-click the **Default** definition in the Application Panel, and select **Run in Browser**.

   ![Application Panel]

   - The definition’s report output will appear in a new browser window based on your current default browser.

   - You can also set this report to be the default report (home page) for the application. Right-click as in the above illustration and select **Set as Default**. You can then click on the green Play button on the Toolbar to run the application’s home page in the browser.

   ![Toolbar]

   - The **Run in Browser** option allows you to run a definition in a browser window outside of Logi Studio without having to make it the default report.
Data Tables

Data Tables are tabular reports where you define the query and the columns to be displayed. The Logi Info engine will construct the table on the output page, adding rows as necessary to display the query's resultant data set.

Lab 1 – Basic Data Table with Graphical Paging & Sorting

This lab is designed to be the basis for many labs to follow. It’s a standard tabular report that will list all the orders from the Northwind database. We add interactive pagination so that the user doesn’t have to scroll through all 830 records.

Objectives:

- Create a folder to hold the Data Table labs.
- Create a Data Table using data from the Orders table in Northwind.
- Add Sort functionality to each column.
- Add Interactive Paging controls to the table.
- Change the alignment of key columns.
Directions:

1. In the Application Panel, create a new folder called **DataTables**.
   - Right-click on the Reports folder. Select **Add**, then **New Folder** from the popup menu.
   - A folder named **newFolder** is created. Immediately type **DataTables** and press **Enter** to rename the folder.
   - You can rename the folder at any time by right-clicking the folder, selecting Rename from the popup menu, typing the new folder name, and then pressing **Enter**.

2. Create a new report definition called **Lab1_Basic_DataTable**:
   - In the Application Panel, right-click on the new **DataTables** folder. Select **Add**, then **New Definition** from the popup menu.
• A new report definition named **newReport** will appear. Rename it to **Lab1_Basic_DataTable** and press **Enter**. (If necessary, right-click the report definition, select **Rename** from the popup menu, type **Lab1_Basic_DataTable**, then press **Enter** to rename the definition.)

• Click on the **Lab1_Basic_DataTable** report definition in the Application Panel. In the Workspace Panel, the report elements will appear in a new tab labeled **Lab1_Basic_DataTable**.

3. Add a **Data Table** element:

• In the Workspace Panel, click on the **Body** element.

• In the Element Toolbox, expand the Data Tables section (if necessary) and double-click the **Data Table** element.

---

*Note that you can collapse the groups of elements by clicking on the triangles next to the group name*
A Data Table element icon will appear in the Workspace Panel immediately under the Body element of your report. If necessary, click on the Data Table element to select it.

In the Attributes Panel for this Data Table element, set the ID attribute value to dtOrders.

4. Add a DataLayer element to access the Northwind database:

   - In the Workspace Panel, click on the Data Table element dtOrders.
   - In the Relational Database DataLayers section of the Element Toolbox, double-click the DataLayer.SQL element to add it to your Data Table.
- A **DataLayer.SQL** element icon will appear in the Workspace Panel under the **dtOrders** element. If necessary, click on the new **DataLayer.SQL** element to select it.

![DataLayer.SQL icon in Workspace Panel]

- In the Attributes Panel, set the **DataLayer.SQL** element attribute values as follows:
  - Set the **ID** attribute to **dlOrders**.
  - Click the **Connection ID** attribute. Select **connNorthwind** from the select list.
  - Click the **Source** attribute. Click the ellipsis button to launch the query builder.

  ![Launch Query Builder]

  The query builder is similar to the query builder in SQL Server Management Studio. Once the ellipsis button is clicked, the query is checked for syntax.

- Once In the Query Builder, double-click the **Orders** table in the top right panel to add it to the query. Then click the checkboxes in the **Orders** table for the following fields to add them to the query.

  - **OrderID**, **CustomerID**, **OrderDate**, **ShippedDate**, **Freight**, **ShipCity**, **ShipCountry**
The Query Builder

- Double click on the **Order Subtotals** view to add it to the main workspace. Drag from the **OrderID** key field in the **Orders** table onto the **OrderID** key field in the **Order Subtotals** view. This will create an **Inner Join** in the lower query section.

- Select the checkbox for the **Subtotals** field in the **Order Subtotals** view to add it to the query.

- The following query should be generated by the Query Builder. Click the **Data** tab in the upper left corner to test the query or use the **F5** key while the cursor is in the lower section.

```sql
Select Orders.OrderID, 
    Orders.CustomerID, 
    Orders.OrderDate, 
    Orders.ShippedDate, 
    Orders.Freight, 
    Orders.ShipCity, 
    Orders.ShipCountry, 
    [Order Subtotals].Subtotal 
From Orders 
```

- Click the **OK** button in the lower right corner to exit the Query Builder. The query should appear in the **Source** attribute of the **DataLayer.SQL** element.
5. Add a Data Table Column element to the Data Table:

- In the Workspace Panel, click on the Data Table element dtOrders.

- In the Element Toolbox, expand the General Elements section (if necessary), and double-click the Data Table Column element. A Data Table Column element icon will appear in the Workspace Panel under dtOrders, immediately below the Data Layer element dlOrders. If necessary, click the Data Table Column element to select it.

- In the Attributes Panel, set the Data Table Column element attribute values as follows:
  - Set the ID attribute to colOrderID.
  - Set the Column Header attribute to Order ID.
6. Add a **Label** element to the **Data Table Column** element `colOrderID`:
   - In the Workspace Panel, click on the **Data Table Column** element `colOrderID`.
   - In the Element Toolbox, expand the General Elements folder (if necessary), and double-click the **Label** element. A **Label** element icon will appear in the Workspace Panel under the **Data Table Column** element `colOrderID`. If necessary, click the **Label** element to select it.
   - In the Attributes Panel, set the Label element attributes as follows:
     - Set the **ID** attribute value to `lblOrderID`
     - Set the **Caption** attribute value to `@Data.OrderID~`

Note that when “@” is entered into the **Caption** attribute, a list of options (IntelliSense) appears and all of the `@`-sign tokens appear. You can type “D” to use type ahead to highlight the first token starting with the letter “D”, then use the **Tab** key or **Space** bar to select the token. You can also navigate through the list using your mouse or the arrow keys on your keyboard. Double-clicking with the mouse will also select a token.

Enter a period “.” after the token and then when prompted, press the **Space** bar. A list of columns from the query in the data layer appears. Use the same process to choose the column, in this case **OrderID**, then press the Tab key or Space bar. This will complete the entry, including the ~ which follows every @Token.

---

### Preview your report now

(Click the **Preview** button at the bottom of the Workspace Panel. You should see the **OrderID** field displayed in a single column. Click the **Definition** button to return to the report definition. Note that only the **OrderID** is displayed.)
7. Use the **Add Data Columns** Wizard to add the remaining **Data Table Column**, **Label**, and **Sort** elements:
   - In the Workspace Panel, click on **Data Table** element *dtOrders*.
   - Launch the wizard by right-clicking on **Data Table** element *dtOrders*, selecting **Element Wizards** from the context menu, and then selecting the **Add Data Columns** wizard (as shown below).
• Select the following fields. Make sure the **OrderId** field is unchecked to avoid duplicating the column:

![Add Data Columns](image)

• Click **Next** once the above fields have been selected. The **Data Table Column** elements, as well as their **Label** and **Sort** child elements, will appear automatically in the Workspace Panel under **Data Table dtOrders**.

• Click **Finish** to end the **Add Data Columns** Wizard.

Notice that the **Data Table Column** element **colOrderID** is different from the ones that were added from the wizard. If you accidentally added a second **OrderId** column you can delete it by selecting it and pressing the delete key (or right mouse click, then choose delete). Note that the column headers provide sort functionality when the column header names are clicked.

8. Adjust the order of the columns to the list below:

- **colOrderID**
- **colCustomerID**
- **colOrderDate**
- **colShippedDate**
- **colFreight**
• `colShipCity`

• `colShipCountry`

• `colSubtotal`

9. Manually add a **Sort** element to the **Data Table Column** element `colOrderID`:
   
   • In the Workspace Panel, click on the top **Data Table Column** element `colOrderID`.
   
   • In the Element Toolbox, expand the General Elements section (if necessary), and double-click the **Sort** element. A **Sort** element will appear in the Workspace Panel under `colOrderID`. If necessary, click the **Sort** element to select it.

   ![Diagram](image)

   • In the Attributes Panel, set the **Sort** element attribute values as follows:
     
     o Set the **Data Column** attribute to `OrderID`.
     
     o Click the **Data Type** attribute. Select **Number** from the drop-down list.

   ![Table](image)
Check the Data Type attributes for all the Sort elements added by the Wizard to ensure that a Data Type attribute value has been set for each Sort element and that it is appropriate for the data being displayed. Change any that are not supplied or are incorrect.

-- Preview your report now. --

(Click the Preview button at the bottom of the Workspace Panel. You should now see all 8 columns queried from the Orders table. Click the Definition button to return to the report definition.)

10. Set the Format attribute of the Label element lblFreight and lblSubtotal to Currency:

- In the Workspace Panel, click on the Label element lblFreight under the Data Table Column element colFreight.
- Click the Format attribute. Select Currency from the Format drop-down list:

![Format Attribute](image)

- Repeat the process with the Label element lblSubtotal.

-- Preview your report now --

(Click the Preview button at the bottom of the Workspace Panel. You should now see the Freight column displayed in currency format. Click the Definition button to return to the report definition.)
11. The word “Subtotal” used as a column header may be misleading to the report reader. Click Data Table Column element \texttt{colSubtotal} to select it and change the Column Header attribute value from Subtotal to Order Total.
12. Add Interactive Paging to the Data Table element dtOrders:

- In the Workspace Panel, right-click on Data Table element dtOrders.
- Choose Element Wizards, then the Add Interactive Paging Controls Wizard from the context menu.
• Click **Next** to accept the default **Rows per Page** value of **20**.

![Wizard dialog for adding interactive paging controls]

• Click **Next** to accept the default **Paging Control Style** of **Graphical/Images**.

![Wizard dialog showing options for graphical/images and text paging controls]

• Click **Finish** to end the Wizard. An **Interactive Paging** element will appear as the last element in the Workspace Panel under **dtOrders**. If necessary, click the **Interactive Paging** element to select it.

• In the Attributes Panel, set the attributes for the **Interactive Paging** element as follows:
  
  o Set the **ID** attribute to **ipGraphical**.

13. Adjust alignment of data table columns.

• Click on the **Data Table Column** element **colOrderDate**.

• Set the **Class** attribute value to **ThemeAlignRight** by selecting it from the drop-down list.
You can click on multiple elements and change a common attribute such as **Class** simultaneously. Simply use the Windows standard CTRL-click to select all the elements that need to be changed.

14. Change the alignment of the **colShippedDate** and **colFreight** Data Table Columns.

- Ctrl – click the **colShippedDate** and **colFreight** columns. They should both be highlighted as shown in the screen shot below.
Set the Class attribute value to ThemeAlignRight by selecting it from the drop-down list.

Note that when selecting multiple elements, the attributes that are different among the selected columns will show (Differing Values). Be careful when changing these to avoid changing attributes by mistake.

15. Verify your report:
   - The report definition elements in the Workspace Panel should appear as shown in the Report Definition section at the beginning of this lab.
   - Click the Preview button at the bottom of the Workspace Panel. The final report should appear as shown in the Report Output section at the beginning of this lab.
   - Click the Save All Opened Files button in the Toolbar to save your report definition.

As a best practice, preview the report in the Workspace Panel to verify that the report works. Also test it in the browser by right-clicking on the report definition in the Application Panel and selecting Run in Browser.
Lab 2 – Numbered Paging

In this lab, a copy is made of Lab1_Basic_DataTable in order to investigate the variations available in the Interactive Paging element.

Objectives:

- Change the interactive paging from graphical to textual.
- Convert the interactive paging into numbered paging controls.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Report Definition" /></td>
<td><img src="image2" alt="Report Output" /></td>
</tr>
</tbody>
</table>

**Text Captions**

**Numbered Paging**

<table>
<thead>
<tr>
<th>Order Id</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>ShippedDate</th>
<th>Freight</th>
<th>ShipCity</th>
<th>ShipCountry</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>10249</td>
<td>VINET</td>
<td>7/4/1996</td>
<td>7/15/1996</td>
<td>33.38</td>
<td>Reims</td>
<td>France</td>
<td>$440.00</td>
</tr>
<tr>
<td>10249</td>
<td>TOMSP</td>
<td>7/5/1996</td>
<td>7/16/1996</td>
<td>111.61</td>
<td>Münster</td>
<td>Germany</td>
<td>$1,963.40</td>
</tr>
<tr>
<td>10250</td>
<td>HANAR</td>
<td>7/6/1996</td>
<td>7/18/1996</td>
<td>60.05</td>
<td>Rio de Janiero</td>
<td>Brazil</td>
<td>$1,052.60</td>
</tr>
<tr>
<td>10251</td>
<td>VICTE</td>
<td>7/8/1996</td>
<td>7/19/1996</td>
<td>41.54</td>
<td>Lyon</td>
<td>France</td>
<td>$630.06</td>
</tr>
<tr>
<td>10252</td>
<td>SUIPD</td>
<td>7/9/1996</td>
<td>7/11/1996</td>
<td>51.50</td>
<td>Charleroi</td>
<td>Belgium</td>
<td>$517.90</td>
</tr>
<tr>
<td>10253</td>
<td>HANAR</td>
<td>7/10/1996</td>
<td>7/12/1996</td>
<td>57.17</td>
<td>Rio de Janiero</td>
<td>Brazil</td>
<td>$1,444.00</td>
</tr>
<tr>
<td>10254</td>
<td>CHOPS</td>
<td>7/11/1996</td>
<td>7/13/1996</td>
<td>22.96</td>
<td>Bern</td>
<td>Switzerland</td>
<td>$566.62</td>
</tr>
<tr>
<td>10255</td>
<td>RICBU</td>
<td>7/12/1996</td>
<td>7/15/1996</td>
<td>148.33</td>
<td>Genève</td>
<td>Switzerland</td>
<td>$2,490.50</td>
</tr>
<tr>
<td>10256</td>
<td>WELLI</td>
<td>7/15/1996</td>
<td>7/17/1996</td>
<td>13.97</td>
<td>Resende</td>
<td>Brazil</td>
<td>$517.90</td>
</tr>
</tbody>
</table>

...
Directions:

1. Copy the report definition for `Lab1_Basic_DataTable` and rename it `Lab2_NumberedPaging`.
   - In the Application Panel’s Reports folder, right-click on the report definition `Lab1_Basic_DataTable` inside the DataTables subfolder, and select Copy from the popup menu.
   - Right-click on the DataTables subfolder, and select Paste from the popup menu.
   - A new report definition named Copy_of_Lab1_Basic_DataTable will appear. Immediately rename it to `Lab2_NumberedPaging` and press Enter. (If necessary, right-click the report definition, select Rename from the popup menu, type Lab2_NumberedPaging, then press Enter to rename the report.)

   This is the process that will be used throughout this training manual when a new lab is built upon a previous lab.

2. Change the Interactive Paging element to use the Text paging style instead of the Graphical paging style.
   - In the Workspace Panel, click the Interactive Paging element `ipGraphical` under Data Table element `dtOrders`, and change the attribute values as shown:

   ![Element - InteractivePaging](image)

<table>
<thead>
<tr>
<th>*Required Attributes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Caption Type</td>
<td>Text</td>
</tr>
<tr>
<td>Page Row Count</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optional Attributes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>Current Page Class</td>
<td></td>
</tr>
<tr>
<td>First Page Caption</td>
<td>First</td>
</tr>
<tr>
<td>Hide Previous/Next Captions</td>
<td></td>
</tr>
<tr>
<td>Hide When One Page</td>
<td>True</td>
</tr>
<tr>
<td>ID</td>
<td>ipText</td>
</tr>
<tr>
<td>Last Page Caption</td>
<td>Last</td>
</tr>
<tr>
<td>Location</td>
<td>Top</td>
</tr>
<tr>
<td>Next Page Caption</td>
<td>Next</td>
</tr>
<tr>
<td>Numbered Page Count</td>
<td></td>
</tr>
<tr>
<td>Page Number Caption</td>
<td></td>
</tr>
<tr>
<td>Page of Caption</td>
<td></td>
</tr>
<tr>
<td>Previous Page Caption</td>
<td>Previous</td>
</tr>
<tr>
<td>Previous/Next Class</td>
<td></td>
</tr>
<tr>
<td>Show Page Number</td>
<td>True</td>
</tr>
</tbody>
</table>
3. Verify your report:
   - The report definition elements in the Workspace Panel should appear as shown in **Report Definition** section at the beginning of this lab.
   - Click the **Preview** button at the bottom of the Workspace Panel then **Run in Browser**. The final report should appear as shown in the **Report Output** section (**Text Captions**) at the beginning of this lab.
   - Click the **Save All Opened Files** button in the Toolbar to save your report definition, then **Run in Browser**.
   - Verify the **First**, **Last**, **Prev** and **Next** links and page number functionality.

4. Change the **Interactive Paging** element to use the Numbered paging style instead of the Text paging style.
   - Select the **ipText** element in the workspace.
   - Change the **Show Page Number** attribute to **Numbered**.
   - Change the **Numbered Page Count** attribute to **25**.
   - Preview the report as in step 3. The final report should appear as shown in the **Report Output** section (**Numbered Paging**) at the beginning of this lab.

You can use the numbered paging functionality (**Show Page Number** attribute set to **Numbered**) together with either the Text (**Caption Type** set to **Text**) or Graphical (**Caption Type** set to **Image**) paging styles.
Lab 3 – Multi Column Sort

This lab demonstrates the options available when a user clicks on the sort link in the column header of a report. Here when the column header for the ShipCountry is clicked, the ShipCity’s sort order will also be affected. To do this we will change the attributes of the ShipCountry’s Sort element but we will not need to change the ShipCity’s. This lab will be valuable later in the Flat Table Reporting lab in Hierarchical Reporting.

Objectives:

- Reconfigure the Sort element for the ShipCountry column to sort the report by city within country when the ShipCountry column header is clicked.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram of DataTables: Lab3_MultiColumnSort" /></td>
<td><img src="image" alt="Report Output Table" /></td>
</tr>
</tbody>
</table>

Report Definition:

- Report Header
- Body
  - dOrders
    - colOrderID
      - lblOrderID
      - Sort
  - colCustomerID
    - lblCustomerID
    - Sort
  - colOrderDate
    - lblOrderDate
    - Sort
  - colShippedDate
    - lblShippedDate
    - Sort
  - colFreight
    - lblFreight
    - Sort
  - colShipCountry
    - lblShipCountry
    - Sort
  - colShipCity
    - lblShipCity
    - Sort
  - colSubtotal
    - lblSubtotal
    - Sort
  - Graphical
    - Report Footer

Report Output:

<table>
<thead>
<tr>
<th>Order Id</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>ShippedDate</th>
<th>Freight</th>
<th>ShipCountry</th>
<th>ShipCity</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>10263</td>
<td>LILAS</td>
<td>8/16/1996</td>
<td>8/23/1996</td>
<td>$54.81</td>
<td>Venezuela</td>
<td>Barquisimeto</td>
<td>$1,414.80</td>
</tr>
<tr>
<td>10296</td>
<td>LILAS</td>
<td>9/3/1996</td>
<td>9/11/1996</td>
<td>$0.12</td>
<td>Venezuela</td>
<td>Barquisimeto</td>
<td>$1,050.60</td>
</tr>
<tr>
<td>10330</td>
<td>LILAS</td>
<td>10/16/1996</td>
<td>10/28/1996</td>
<td>$12.75</td>
<td>Venezuela</td>
<td>Barquisimeto</td>
<td>$1,649.00</td>
</tr>
<tr>
<td>10357</td>
<td>LILAS</td>
<td>11/19/1996</td>
<td>12/2/1996</td>
<td>$34.88</td>
<td>Venezuela</td>
<td>Barquisimeto</td>
<td>$1,167.58</td>
</tr>
<tr>
<td>10381</td>
<td>LILAS</td>
<td>12/12/1996</td>
<td>12/13/1996</td>
<td>$7.99</td>
<td>Venezuela</td>
<td>Barquisimeto</td>
<td>$112.00</td>
</tr>
<tr>
<td>10461</td>
<td>LILAS</td>
<td>2/28/1997</td>
<td>3/5/1997</td>
<td>$148.61</td>
<td>Venezuela</td>
<td>Barquisimeto</td>
<td>$1,558.70</td>
</tr>
<tr>
<td>10499</td>
<td>LILAS</td>
<td>4/8/1997</td>
<td>4/16/1997</td>
<td>$102.02</td>
<td>Venezuela</td>
<td>Barquisimeto</td>
<td>$1,412.00</td>
</tr>
<tr>
<td>10543</td>
<td>LILAS</td>
<td>5/21/1997</td>
<td>5/23/1997</td>
<td>$49.17</td>
<td>Venezuela</td>
<td>Barquisimeto</td>
<td>$1,504.50</td>
</tr>
<tr>
<td>10760</td>
<td>LILAS</td>
<td>12/16/1997</td>
<td>12/25/1997</td>
<td>$42.13</td>
<td>Venezuela</td>
<td>Barquisimeto</td>
<td>$72.00</td>
</tr>
<tr>
<td>11065</td>
<td>LILAS</td>
<td>5/1/1998</td>
<td></td>
<td>$12.91</td>
<td>Venezuela</td>
<td>Barquisimeto</td>
<td>$189.42</td>
</tr>
<tr>
<td>11071</td>
<td>LILAS</td>
<td>5/5/1998</td>
<td></td>
<td>$0.93</td>
<td>Venezuela</td>
<td>Barquisimeto</td>
<td>$464.50</td>
</tr>
<tr>
<td>10258</td>
<td>GROSR</td>
<td>7/30/1996</td>
<td>8/2/1996</td>
<td>$56.29</td>
<td>Venezuela</td>
<td>Caracas</td>
<td>$1,101.20</td>
</tr>
<tr>
<td>10755</td>
<td>GROSR</td>
<td>12/18/1997</td>
<td>12/24/1997</td>
<td>$1.51</td>
<td>Venezuela</td>
<td>Caracas</td>
<td>$367.50</td>
</tr>
<tr>
<td>10405</td>
<td>LINOD</td>
<td>1/6/1997</td>
<td>1/22/1997</td>
<td>$54.82</td>
<td>Venezuela</td>
<td>I de Margarita</td>
<td>$400.00</td>
</tr>
<tr>
<td>10455</td>
<td>LINOD</td>
<td>3/25/1997</td>
<td>3/31/1997</td>
<td>$54.45</td>
<td>Venezuela</td>
<td>I de Margarita</td>
<td>$1,564.00</td>
</tr>
<tr>
<td>10638</td>
<td>LINOD</td>
<td>6/20/1997</td>
<td>9/1/1997</td>
<td>$158.44</td>
<td>Venezuela</td>
<td>I de Margarita</td>
<td>$2,720.05</td>
</tr>
<tr>
<td>10697</td>
<td>LINOD</td>
<td>10/8/1997</td>
<td>10/14/1997</td>
<td>$45.52</td>
<td>Venezuela</td>
<td>I de Margarita</td>
<td>$105.43</td>
</tr>
</tbody>
</table>
Directions:

1. Copy the report definition for Lab1_Basic_DataTable and rename it Lab3_MultiColumnSort.
   - In the Application Panel’s Reports folder, right-click on the report definition Lab1_Basic_DataTable inside the DataTables subfolder, and select Copy from the popup menu.
   - Right-click on the DataTables subfolder, and select Paste from the popup menu.
   - Rename the copy to Lab3_MultiColumnSort.

2. Change the column sort for the ShipCountry column. Click the Sort element under Data Table Column element colShipCountry.

   - Change the Sort element attribute values as shown (when inputting multiple comma separated values you will need to manually type in the values):

   Note that both the First Sort Sequence and Reverse Sort Sequence attributes have Ascending as the second value in the attribute. This will ensure that the ShipCity column (the second value specified in the Data Column attribute) is sorted in Ascending order, no matter what the sort order is for ShipCountry.

3. Move the ShipCountry column so it appears to the left of the ShipCity column:
   - In the Workspace Panel, click on the Data Table Column element colShipCountry.
   - Use the Up/Down arrows in the Toolbar to move colShipCountry above the Data Table Column element colShipCity.
4. Verify your report:
   - The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.
   - Click the Save All Opened Files button in the Toolbar to save your report definition.
   - Click the Preview button at the bottom of the Workspace Panel then Run In Browser. The final report should appear as shown in the Report Output section at the beginning of this lab.
   - Be sure to click on the ShipCountry column header twice to see the view in the Report Output Section.
Lab 4 – Summary Row

Totals are added in this lab which are displayed in the last row of the report.

Objectives:

- Compute the sum of all values in the Freight and Subtotal column.
- Display the total Freight Aggregation and Subtotal Aggregation in a summary row.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Report Definition Diagram" /></td>
<td><img src="image2" alt="Report Output Table" /></td>
</tr>
</tbody>
</table>

Report Definition:

```
DataTables.Labs4_SummaryRow
Style
Report Header
Body
dOrders
colOrderID
colCustomerID
colOrderDate
colShippedDate
colFreight
colShipCity
colShipCountry
colSubtotal
Report Footer
```

Report Output:

```
<table>
<thead>
<tr>
<th>Order Id</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>ShippedDate</th>
<th>Freight</th>
<th>ShipCity</th>
<th>ShipCountry</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>11065</td>
<td>QUEEN</td>
<td>5/4/1998</td>
<td></td>
<td>$81.75</td>
<td>Sao Paulo</td>
<td>Brazil</td>
<td>$2,027.08</td>
</tr>
<tr>
<td>11070</td>
<td>LEHMS</td>
<td>5/5/1998</td>
<td></td>
<td>$136.00</td>
<td>Frankfurt a.M.</td>
<td>Germany</td>
<td>$1,629.97</td>
</tr>
<tr>
<td>11071</td>
<td>LILAS</td>
<td>5/5/1998</td>
<td></td>
<td>$0.93</td>
<td>Barquisimeto</td>
<td>Venezuela</td>
<td>$484.50</td>
</tr>
<tr>
<td>11072</td>
<td>ERNISH</td>
<td>5/5/1998</td>
<td></td>
<td>$258.64</td>
<td>Graz</td>
<td>Austria</td>
<td>$5,218.00</td>
</tr>
<tr>
<td>11073</td>
<td>PERIC</td>
<td>5/5/1998</td>
<td></td>
<td>$24.95</td>
<td>Mexico D.F.</td>
<td>Mexico</td>
<td>$300.00</td>
</tr>
<tr>
<td>11074</td>
<td>SIMOB</td>
<td>5/6/1998</td>
<td></td>
<td>$18.44</td>
<td>Kopenhagen</td>
<td>Denmark</td>
<td>$232.09</td>
</tr>
<tr>
<td>11075</td>
<td>RICBU</td>
<td>5/6/1998</td>
<td></td>
<td>$6.19</td>
<td>Geneve</td>
<td>Switzerland</td>
<td>$458.10</td>
</tr>
<tr>
<td>11076</td>
<td>BONAP</td>
<td>5/6/1998</td>
<td></td>
<td>$38.28</td>
<td>Marseille</td>
<td>France</td>
<td>$702.75</td>
</tr>
<tr>
<td>11077</td>
<td>RATTC</td>
<td>1/1/1998</td>
<td></td>
<td>$5.53</td>
<td>Albuquerque</td>
<td>USA</td>
<td>$1,255.72</td>
</tr>
</tbody>
</table>

Totals: $64,942.69  $1,265,783.06
Directions:

1. Copy the report definition for Lab1_Basic_DataTable and rename it Lab4_SummaryRow.
   - In the Application Panel's Reports folder, right-click on the report definition Lab1_Basic_DataTable inside the DataTables subfolder, and select Copy from the popup menu.
   - Right-click on the DataTables subfolder, and select Paste from the popup menu.
   - Rename the copy to Lab4_SummaryRow.

2. Add a Data Column Summary element under the Data Table Column element colFreight.
   - In the Workspace Panel, click on the Data Table Column element colFreight.
   - Double-click on the Data Column Summary element in the Element Toolbox. This adds a new element under the colFreight element.

   - Set the attributes for the Summary element as follows:

     ![Element - DataColumnSummary Table]

     | Required Attributes | Data Column | Freight |
     |---------------------|-------------|--------|
     | Function            | Sum         |        |
     | ID                  | sumFreight  |        |
     | Optional Attributes | Data Type   | Number |

     Double click an attribute name for Attribute Zoom.
3. Repeat Steps for the `colSubotal` Column.

4. Add a **Summary Row** element to the data table:
   - In the Workspace Panel, click on the **Data Table** element `dtOrders`;
   - Double-click on the **Summary Row** element in the Element Toolbox. A **Summary Row** element will appear under the data table in the Workspace Panel.

   - You may have only one **Summary Row** element per **Data Table**. (You may use the **Rows** element as a child element to add multiple rows of information to the **Summary Row**.)

5. Change the attributes of the summary row as follows:

6. Verify your report:
   - Click the **Save All Opened Files** button in the Toolbar to save your report definition.
   - The report definition elements in the Workspace Panel should appear as shown in **Report Definition** section at the beginning of this lab.
   - Click the **Preview** button at the bottom of the Workspace Panel, then **Run In Browser**. The final report should appear as shown in the **Report Output** section at the beginning of this lab.
   - You will need to move to the last page of the report using the interactive pagination to check the totals.
Note that there is no specific relationship between the Summary Row and the Data Column Summary element. If you inspect the attributes of the Data Column Summary, you will not see an attribute that specifies the “companion” Summary Row. It is the same when you inspect the Summary Row...no reference to the Data Column Summary element.
Lab 5 – Custom Summary Row

In Lab 4 we found that there can be only one **Summary Row** per Data Table, and that the aggregations of our data must reside in the column that is being aggregated. Furthermore, we cannot have text captions for our aggregations that may take up space under dates and text columns in that same **Summary Row**. This lab will address these issues.

The final output will contain caption text (“Total Freight”...“Average Freight”) for both the **Freight** and **Subtotal** columns. Therefore, only four columns will be required to display these totals.

**Objectives:**

- Reconfigure the **Summary Row** from Lab 4 to remove the text in the caption attribute.
- Add **Total Freight**, **Total Subtotal** (total order sales), and caption text to the **Summary Row**.
- Add **Average Freight** and **Average Subtotal** to the summary row.

![Report Definition](image1.png)
![Report Output](image2.png)
Directions:

1. Copy the report definition for Lab4_SummaryRow and rename it Lab5_CustomSummaryRow.
   - In the Application Panel’s Reports folder, right-click on the report definition Lab4_SummaryRow inside the DataTables subfolder, and select Copy from the popup menu.
   - Right-click on the DataTables subfolder, and select Paste from the popup menu.
   - Rename the copy to Lab5_CustomSummaryRow.

2. Modify the Summary Row element srOrderSummary in the Data Table element dtOrders:
   - In the Workspace Panel, click on the Summary Row element srOrderSummary.
   - Delete the Caption attribute setting. This will leave only the ID.
   - Set the Class Setting to ThemeAlignRight. This will set any children elements to be aligned as well.

We will now add our four Column Cell elements that will contain 1) text (“captions”) for the freight, 2) the freight aggregate itself, 3) the caption text for the subtotal, and 4) the subtotal aggregate itself.

3. Add a Column Cell element under the Summary Row element srOrderSummary:
   - In the Workspace Panel, click on the Summary Row element srOrderSummary.
   - Double-click on the Column Cell element in the Element Toolbox.
• Repeat this 3 more times for a total of four **Column Cell** elements.

You can set the **Class** attribute of each column to **ThemeAlignRight**, or you may set the **Class** to **ThemeAlignRight** for the **Summary Row** element itself, as we have done above.

4. Add **Label** elements to all four of the **Column Cell** elements:

   • In the Workspace Panel, click on the Column Cell element **colFreightText**.

   • Double-click on the **Label** element in the Element Toolbox. A new **Label** element will appear in the Workspace Panel under the **Column Cell** element **colFreightText**.

   • Set the attribute values of the **Label** element as follows.
• Repeat the process by adding a Label element to colSubtotalText and set the attribute values as shown.

Note that you can add the Label element to multiple Column Cell elements by using Windows standard ctrl-clicking to select the Column Cells, then double-clicking the Label element in the toolbox. This should add the Label element as a child to all the Column Cell elements. You will still need to select them individually to set the ID and other attribute values.

5. Add Aggregations to the Data Layer to total up the Freight and Subtotal.
   • Select the Data Layer dlOrders.
   • Add an Aggregate Column Element from the Toolbox.

   • Configure the attribute values as follows:
Repeat these steps to add an additional **Aggregate Column** element to sum up the **Subtotal** column.

6. Add **Labels** to output the two **Aggregate Column** elements.
   - Select the `colFreightAgg` element under the `srOrderSummary` element.
   - Add a **Label** element from the Toolbox and enter the attribute values as follows:

   - Select the `colSubtotalAgg` element under the `srOrderSummary` element.
   - Add a **Label** element from the Toolbox and enter the attribute values as follows:
7. Verify your report:
   - Click the **Save All Opened Files** button in the Toolbar to save your report definition.
   - The report definition elements in the Workspace Panel should appear as shown in the **Report Definition** section at the beginning of this lab.
   - Click the **Preview** button at the bottom of the Workspace Panel, then **Run In Browser**. The final report should appear as shown in the **Report Output** section at the beginning of this lab.
   - You will need to move to the last page of the report to see the **Summary Row**.

Note that the columns are “bunched up” in the first four columns of the **Summary Row**. We must now consider setting the **Column Span** attribute of all four columns to specify the number of columns in the **Summary Row** that each column will “span” across.

8. Set the **Column Span** attribute values in the **Column Cell** elements under the **Summary Row**.
   - Select **colFreightText** and enter 4 in the **Column Span** attribute value.
   - Select **colSubtotalText** and enter 2 in the **Column Span** attribute value.
• Leave the **Column Span** attribute values for **colFreightAgg** and **colSubtotalText** blank since they only need to span 1 column (the default). Only the text needs to span multiple columns.

9. **Verify your report:**

• **Click the Save All Opened Files** button in the Toolbar to save your report definition.

• The report definition elements in the Workspace Panel should appear as shown in the **Report Definition** section at the beginning of this lab.

• **Click the Preview** button at the bottom of the Workspace Panel, then **Run In Browser**. The final report should appear as shown in the **Report Output** section at the beginning of this lab.

• You will need to move to the last page of the report to see the **Summary Row**.
Data Layers

In these labs we focus primarily on the DataLayer.SQL element. Most of what is learned in this topic will apply to every data layer so this portion of the training is easily re-used. We will compare and contrast the use of SQL Server vs. Logi Elements to extract our data. In subsequent topics, we will show how data manipulation such as Data Grouping, applying Filter Conditions, and Sorting can be done using either method. This allows the simple manipulation of less easily changed data sources (e.g. web services).

Lab 1 – Calculated Columns

This lab is an introduction to the use of DataLayer elements.

Objectives:

- Create a Data Table from scratch using a wizard.
- Extract and display the order date year for each order.
- Display the percent of total freight for each order.
- Create a simple calculation.
Report Definition

```
Lab1_CalculatedColumn
  .Lab1_CalculatedColumn
    .Style
    .Report Header
    .Body
      .Labels
        .Orders
          .OrdOrderID
            .Sort
          .OrdCustomerID
            .Sort
          .OrdShippedDate
            .Sort
          .OrdFreight
            .Sort
          .OrdTotalFreight
            .Sort
          .OrdFreightOverTotal
            .Sort
      .Sort
```

Report Output

<table>
<thead>
<tr>
<th>OrderID</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>ShippedDate</th>
<th>Freight</th>
<th>Pct. of Total Freight</th>
<th>ShipCity</th>
<th>ShipCountry</th>
<th>Order Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>10246</td>
<td>VNET</td>
<td>7/4/1996</td>
<td>7/19/1996</td>
<td>$32.36</td>
<td>0.00%</td>
<td>Reims</td>
<td>France</td>
<td>$440.00</td>
</tr>
<tr>
<td>10249</td>
<td>TOMB</td>
<td>7/5/1996</td>
<td>7/13/1996</td>
<td>$111.61</td>
<td>0.02%</td>
<td>Münster</td>
<td>Germany</td>
<td>$1,963.40</td>
</tr>
<tr>
<td>10250</td>
<td>HANAR</td>
<td>7/8/1996</td>
<td>7/13/1996</td>
<td>$65.83</td>
<td>0.10%</td>
<td>Rio de Janeiro</td>
<td>Brazil</td>
<td>$1,552.60</td>
</tr>
<tr>
<td>10251</td>
<td>VICTE</td>
<td>7/8/1996</td>
<td>7/15/1996</td>
<td>$41.34</td>
<td>0.06%</td>
<td>Lyon</td>
<td>France</td>
<td>$664.06</td>
</tr>
<tr>
<td>10252</td>
<td>SULPD</td>
<td>7/9/1996</td>
<td>7/11/1996</td>
<td>$101.20</td>
<td>0.02%</td>
<td>Charleroi</td>
<td>Belgium</td>
<td>$3,597.00</td>
</tr>
<tr>
<td>10253</td>
<td>PANKAR</td>
<td>7/10/1996</td>
<td>7/15/1996</td>
<td>$50.17</td>
<td>0.06%</td>
<td>Rio de Janeiro</td>
<td>Brazil</td>
<td>$1,444.00</td>
</tr>
<tr>
<td>10254</td>
<td>CHOPS</td>
<td>7/11/1996</td>
<td>7/23/1996</td>
<td>$22.96</td>
<td>0.04%</td>
<td>Bern</td>
<td>Switzerland</td>
<td>$295.62</td>
</tr>
<tr>
<td>10255</td>
<td>RONU</td>
<td>7/12/1996</td>
<td>7/15/1996</td>
<td>$18.33</td>
<td>0.02%</td>
<td>Genève</td>
<td>Switzerland</td>
<td>$2,801.10</td>
</tr>
<tr>
<td>10256</td>
<td>WELLI</td>
<td>7/15/1996</td>
<td>7/17/1996</td>
<td>$13.97</td>
<td>0.02%</td>
<td>Rosende</td>
<td>Brazil</td>
<td>$57.60</td>
</tr>
<tr>
<td>10257</td>
<td>HILAA</td>
<td>7/16/1996</td>
<td>7/22/1996</td>
<td>$81.91</td>
<td>0.13%</td>
<td>San Cristóbal</td>
<td>Venezuela</td>
<td>$1,115.50</td>
</tr>
<tr>
<td>10258</td>
<td>ERNHI</td>
<td>7/17/1996</td>
<td>7/23/1996</td>
<td>$140.51</td>
<td>0.22%</td>
<td>Graz</td>
<td>Austria</td>
<td>$1,614.88</td>
</tr>
<tr>
<td>10259</td>
<td>CENTC</td>
<td>7/18/1996</td>
<td>7/25/1995</td>
<td>$3.25</td>
<td>0.01%</td>
<td>Mexico D.F.</td>
<td>Mexico</td>
<td>$100.60</td>
</tr>
<tr>
<td>10260</td>
<td>OTIKK</td>
<td>7/19/1996</td>
<td>7/29/1995</td>
<td>$50.09</td>
<td>0.08%</td>
<td>Köln</td>
<td>Germany</td>
<td>$1,004.65</td>
</tr>
<tr>
<td>10261</td>
<td>QUEDEE</td>
<td>7/20/1996</td>
<td>7/30/1995</td>
<td>$3.05</td>
<td>0.00%</td>
<td>Rio de Janeiro</td>
<td>Brazil</td>
<td>$448.00</td>
</tr>
<tr>
<td>10262</td>
<td>RATTC</td>
<td>7/22/1995</td>
<td>7/25/1995</td>
<td>$40.20</td>
<td>0.07%</td>
<td>Albuquerque</td>
<td>USA</td>
<td>$504.00</td>
</tr>
<tr>
<td>10263</td>
<td>ERHSH</td>
<td>7/23/1995</td>
<td>7/31/1995</td>
<td>$140.06</td>
<td>0.22%</td>
<td>Gdansk</td>
<td>Poland</td>
<td>$1,823.60</td>
</tr>
<tr>
<td>10264</td>
<td>POLKDO</td>
<td>7/24/1995</td>
<td>8/23/1995</td>
<td>$3.67</td>
<td>0.01%</td>
<td>Malmö</td>
<td>Sweden</td>
<td>$885.63</td>
</tr>
<tr>
<td>10265</td>
<td>BLONP</td>
<td>7/25/1995</td>
<td>8/12/1995</td>
<td>$55.26</td>
<td>0.09%</td>
<td>Strasbourg</td>
<td>France</td>
<td>$1,176.00</td>
</tr>
<tr>
<td>10266</td>
<td>WARTH</td>
<td>7/25/1995</td>
<td>7/31/1995</td>
<td>$20.72</td>
<td>0.04%</td>
<td>Öulu</td>
<td>Finland</td>
<td>$345.56</td>
</tr>
<tr>
<td>10267</td>
<td>FRANK</td>
<td>7/29/1995</td>
<td>8/5/1995</td>
<td>$205.56</td>
<td>0.32%</td>
<td>München</td>
<td>Germany</td>
<td>$3,506.60</td>
</tr>
</tbody>
</table>
Directions:

1. In the Application Panel, create a new folder called **DataLayers**.
   - Right-click on the **Reports** folder then select **Add** and **New Folder** from the popup menu.
   - Rename the folder **DataLayers**.

2. In the **DataLayers** folder in the Application Panel, create a new definition called **Lab1_CalculatedColumn**:
   - In the Application Panel, click on the folder **DataLayers**. Right-click, then select **Add** and **New Definition**. A definition named **newReport** will appear. Rename it to **Lab1_CalculatedColumn**.

3. Add a **Data Table** element (using the wizard):
   - In the Workspace Panel, right-click on the **Body** element and choose **Element Wizards**, then the **Add a Data Table** Wizard.
Using the wizard set the following attribute values:

- Set the **Connection ID** attribute value to Northwind;
- Enter the following text when prompted for a SQL Query and click **Next**. (You may copy and paste the SQL text below.)

```
```

- You can also use the query builder at this point and paste the above SQL into the lower section of the query builder and press OK.

- When prompted for the columns to be added to the table, make sure all columns are checked.
- Choose **Yes** at the **Interactive Paging** prompt, then click **Next**.
- Accept the default value of **20** at the **Rows Per Page** prompt, then click **Next**.
- Accept **Graphical / Images** at the **Paging Control Style** prompt, then click **Next**.
- Choose **Finish** when prompted.

---

**Note:** the wizard will use default table and data layer names when using the wizard. It is advisable to rename the **Data Table** element ID to **dtOrders**, and the **DataLayer.SQL** element ID to **dlOrders**. The remaining bullets in the lab will reflect these names.

---

4. **Extract the year from the **OrderDate** column.**

- Click on the **DataLayer.SQL** element **dlOrders** in the Workspace Panel.
- In the Add and Replace Columns section of the Element Toolbox, double-click the **Time Period Column** element to add it to **dlOrders**.
• Set the attribute values as shown below.

5. Add a Data Table Column to the Data Table element *dtOrders* (using the wizard).
   - In the Workspace Panel, click the Data Table element *dtOrders*.
   - Right-click on the Data Table *dtOrders* and choose element wizards then Add data columns.
6. The wizard will display a list of all columns in the data layer. Note that the only column checked is the new column which has not yet been added to the data table.

- Click Next.
- The tpcOrderYear column will be added to the data table below the existing child elements of Data Table element dtOrders.
- Click Next. The new column is created by the wizard.
- Select the Data Table Column element coltpcOrderYear.
- Press F7 to move the Data Table Column element coltpcOrderYear up so that it is just below the colOrderDate element.
- Change the coltpcOrderYear element’s Column Header attribute value to Order Year.
- Change the coltpcOrderYear element’s Class attribute value to ThemeAlignCenter.
- Select the Sort element under coltpcOrderYear. Change the Data Type attribute value to Number.

A good best practice is to inspect all Data Table Columns created from Data Layer elements like the Time Period Column or Calculated Column. In this case, if the data type was left blank or set to Text, the sorting would have been off.
7. Add a column to the Data Layer to calculate the percent of each order’s freight against the total Freight.
   - Click on the dlOrders data layer in the Workspace Panel.
   - In the Element Toolbox, double-click the Percent of Total element icon.

   ![Element Toolbox with Percent of Total Column selected]

   - In the Attributes Panel, set the PercentOfTotalColumn element attribute values as follows:

   ![Attributes Panel with PercentOfTotalColumn element details]

8. Add a Data Table Column element to the data table to display pctTotalFreight.
   - Follow the same steps as in step 5 and select column pctTotalFreight.

9. A new column is added at the end of the “children” of dtOrders.
   - Change the Column Header attribute value of colpctTotalFreight to Pct. of Total Freight.
   - Use the F7 key to move the new column up below the colFreight element.
10. Set the attribute values for the Label element `lblpctTotalFreight`:

![Label attribute settings](image)

11. Set the **Format** attribute value to **Currency** for the Label elements `lblFreight` and `lblSubtotal`.

![Label attribute settings](image)
12. Change the alignment of the following columns (colOrderDate, colShippedDate, colFreight, colpctTotalFreight, colSubtotal).

- Hold the Ctrl key down and click each of the above columns.
- Change the Class attribute to ThemeAlignRight. This will set that attribute for all four columns. The attributes will look like this:

![Optional Attributes Table]

- Use this method to set the alignment of the colpctOrderYear element. Select ThemeAlignCenter from the Class attribute dropdown menu.

13. Change the Column Header attribute of colSubtotal to Order Total.

14. Change the Column Header attribute of colpctTotalFreight to Pct. of Total Freight.

15. Create a simple calculated column by dividing the Freight by the Order Total.

- Select the DataLayer.SQL element dlOrders.
- Double-click the Calculated Column element from the General Elements section of the Element Toolbox.
• Set the attribute values as follows for the new Calculated Column element:

You must place double quotes around any @Tokens used in any expression if they are not numeric. This includes text and date data that might be represented by the @Token. Here we were using @Tokens that represent numeric data and thus, no double quotes are needed.

• Add a column to reflect this new Data Layer element by using the Add Data Columns wizard from the table menu (as in step 5). Make sure calcFreightOverTotal is the only column from the Data Layer checked.

• Move the new column (colFreightOverOrderTotal) just above colSubTotal using the F7 key and set the Column Header attribute to Freight Over Order Total.

• Change the Format attribute value of the lblcalcFreightOverOrderTotal element to Percent.

• Change the Sort element’s Data Type attribute under colFreightOverOrderTotal to Number.

Once again we see that elements generated from a wizard must be inspected. Here the calculated column used a simple algebraic expression but the wizard isn’t designed to guess about the type of data any one expression is producing. Thus the Sort element’s Data Type attribute must be changed or else this new calculation will not sort properly.

16. Verify your report:

• Click the Save All Opened Files button in the Toolbar to save your report definition.

• The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.

• Click the Preview button at the bottom of the Workspace Panel then Run In Browser. The final report should appear as shown in the Report Output section at the beginning of this lab.

Note: the Interactive Paging element that was present before adding Data Table Columns can be moved to the end of the Data Table Columns. Though the element will function anywhere within the Data Table, it is often helpful to position the elements to lend context and a degree of “self-documentation”. Elements can be moved with the F7 or F8 keys and the blue arrows in the toolbar.
Lab 2 – VB Script Functions

In the last lab we touched on DataLayer elements that were used in calculations or in some specialized function like extracting information from a date. Here we will use functions that you may have used before in Excel. These are referred to as VBScript functions and we can utilize them in Calculated Columns.

As always, inspect any wizard generated columns that are created from Calculated Column elements. Also, the double quote rule (double quotes around any non numeric @Token reference) still applies to arguments of a VBScript function in a Calculated Column.

Finally, we will get a taste of “debugging” or providing an approach to handling issues within our reports.

Objectives:

- Compute the order shipment time using the DateDiff() function.
- Adjust the shipment time calculation for empty values using the special IIF() function.
Directions:

1. Copy the definition from **Lab1_Basic_DataTable** from the **DataTables** folder to the **DataLayers** folder and rename it **Lab2_VBScript_Functions**.
   - Select the **DataTables** folder and the **Lab1_Basic_DataTable** definition.
   - Right-click and choose **Copy**.
   - Select the **DataLayers** folder, right-click and choose **Paste**.
   - Rename the copied Lab to **Lab2_VBScript_Functions**.
2. Add a **Calculated Column** element to display ShipTime.

We will be determining **ShipTime** by computing the difference between the Order date and Shipped Dates to determine the amount of **days** from order date to the date shipped. The function to be used in this lab is the VB Script function **DateDiff**. The “d” argument in this function refers to the “days” interval. The earlier of the two days is used for the first date argument.

- Click on the **dlOrders** data layer in the Workspace Panel.
- In the Element Toolbox, double-click on the **Calculated Column** element. A new element will appear in the Workspace Panel under the **dlOrders**.

![Calculated Column Element Toolbox](image)

- Set the attribute values for the new Calculated Column as follows:

![Calculated Column Attribute Settings](image)

3. Run the Add data columns Wizard to add the **calcShipTime** column to the data table.

- Right-click on the Data Table **dtOrders** and choose **Element Wizards** then **Add Data Columns**.
When prompted to choose individual columns, select only the `calcShipTime` column and unselect the others.
• Click **Next**, then **Finish**.

4. Update the column **colcalcShipTime**.

   • Select the **colcalcShipTime** element and enter **Ship Time** in the **Column Header** attribute.
   
   • Press the **F7** key to move it under the **colShippedDate** column.
   
   • Change the **Class** attribute to **ThemeAlignRight**.

5. Click the **Sort** element under the **Data Table** column element **colcalcShipTime**.

   • In the Attributes Panel for the sort element, click the **Data Type** attribute and select **Number** from the drop down list.

---

Preview the report. Sort the report by **Ship Time** (descending) and note that there are empty values for some orders: these orders have not yet been shipped, which results in an empty **calcShipTime**. In this lab we will want to replace that empty value with text.

---

6. Solve the VBScript issue.

   • Select Debugger links from the drop down menu on the Studio toolbar:

     ![Debugger Links](image)

     • Run the report in the browser and click on the **Debug this Page** link.
     
     • Scroll down and you will see several entries such as this.
The chief complaint by our engine is that there is no argument to supply to `DateDiff` if there is no ship date. If Debugger Links is turned off (No Details in the Studio Toolbar menu), then this message is not written out to the browser and the error (warning) is ignored. However we have the ability to fix the problem.

7. Change `calcShipTime` DataLayer child element to handle the empty `ShippedDate`. Note we have an order date for every order but not a shipped date for every order.
   
   - Close any open browser.
   - Select the `calcShiptime` Calculated Column.
   - Change the Formula attribute to contain the following script (Copy and Paste is OK).
     ```
     DateDiff("d","@Data.OrderDate~",IIF("@Data.ShippedDate~"="","@Data.OrderDate ~"","@Data.ShippedDate~"))
     ```
   
   There is an `IIF()` function where the `ShippedDate` used to be. This will be the first function run and will return either the `OrderDate` (forcing `DateDiff()` to return 0 days) or the `ShippedDate` if it is filled in.

8. Preview the report. The `ShipTime` column in the report should contain a 0 where there is no shipped date.
   
   - Click on the `ShippedDate` column header to sort by `ShippedDate`. You can also sort by `ShipTime`.
   
   This is a cleaner approach since there is nothing but integers in the `ShipTime` column. Suppose now the requirements state that there should be the text “Not Shipped” when the `ShipTime` is 0. Let's utilize our new `IIF()` function to handle this problem.

9. Add a Calculated Column element to the Data Layer `dlOrders` to modify empty ship times.
   
   - Click on the `dlOrders` data layer in the Workspace Panel.
   - In the Element Toolbox, double-click the Calculated Column element. A Calculated Column element will appear in the Workspace Panel under the data layer.
   - Set the attribute values for the Calculated Column element as follows (using `calcShipTimeAdjusted` as the ID):
10. Rather than adding another Data Table Column, modify the Label element under `colcalcShipTime` to use the `calcShipTimeAdjusted` calculated column:
   - In the Workspace Panel, click on the Label element `lblcalcShipTime`.
   - In the Attributes Panel, change the Caption attribute from `@Data.calcShipTime~` to `@Data.calcShipTimeAdjusted~`:

   ![Element - Label]

   - We will leave the sort element’s Data Column attribute to `calcShipTime`. This is ok since it is a number and we can sort on that more efficiently since it has a 0 instead of text.

11. Verify your report:
   - Click the Save All Opened Files button in the Toolbar to save your report definition.
   - The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.
   - Click the Preview button at the bottom of the Workspace Panel then Run In Browser. The final report should appear as shown in the Report Output section at the beginning of this lab.
Lab 3 – KPI

Once of the most used terms in Business Intelligence is the term KPI, short for Key Performance Indicator. It can be viewed as a metric that has been applied to one or more business rule(s) with each rule having a specific visualization associated with it. A popular visualization is the stoplight KPI where Red is bad, Yellow is concerned and Green is good.

To use the Ship Time calculation in the previous lab and turn it into a KPI, we will need to learn a bit about style sheets since they provide background colors to our columns. We’ll provide a way for different background colors to be displayed based on conditions in that row of data.

Objectives:

- Modify the column Ship Time to act as a KPI or Key Performance Indicator.
Directions:

1. Copy report definition `Lab2_VBScript_Functions` and rename it `Lab3_KPI`.

   Note: Though the look and feel for our tables has been determined by the Theme chosen in the New Application Lab, we will now want to augment the theme with a style sheet which will contain “non-thematic” styles like the ones used in this lab. Each row of the Ship Time column will be evaluated against one of three business rules to be outlined later in this lab. The background color will match one of these three rules.

2. Create a style sheet to hold the three styles (Red, Yellow and Green backgrounds) for the KPI.

   - Select the red Support Files folder in the Application Panel.
   - Right-click on the Support Files folder and choose Add then New File.

     ![Add new support file](image)

     - Choose Style Sheet from the Add new support file prompt and click OK.

     ![Style Sheet](image)

     - A new file appears in the Support Files folder called newStyleSheet.css. Rename to LabStyleSheet.css and press enter.
• Delete the current text (Body and curly brackets) by highlighting the text in the Workspace tab and deleting.
• Save the file using the save icons or Ctrl-S.

3. Add the necessary classes.
• Double-click on the LabStyleSheet.css file in the Support Files folder.
• Add three classes as shown below:

```css
.backgroundGreen {
  background-color: Lime;
  text-align: right;
}

.backgroundYellow{
  background-color: Yellow;
  text-align: right;
}

.backgroundRed {
  background-color: Red;
  color: White;
  text-align: right;
}
```

• Copy and paste using this text:

```css
.backgroundGreen {
  background-color: Lime;
  text-align: right;
}

.backgroundYellow{
  background-color: Yellow;
  text-align: right;
}

.backgroundRed {
  background-color: Red;
  color: White;
  text-align: right;
}
- Close the Style sheet

![LabStyleSheet.css]

4. Associate the new style sheet with the Lab definition.

- Select the Style element in near the top of the Lab3_KPI definition.
- In the Style Sheet attribute, choose the LabStyleSheet.css style sheet from the drop-down list.

![Optional Attributes]

5. Verify one style.

- Select the colcalcShipTime column and choose backgroundGreen from the Class attribute dropdown.

![Optional Attributes]

Page 109
• Preview the report. You should see every row in the Ship Time column as having a green background.

• Reset the Class attribute back to ThemeAlignRight in colcalcShipTime. We do this so that if by chance there is not a condition met among the conditional classes, it will revert back to ThemeAlignRight.

We would like now to have the decision as to which style to use based on a business rule such as this:

<table>
<thead>
<tr>
<th>Condition</th>
<th>Background Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ship time &lt;14 days</td>
<td>Green Background</td>
</tr>
<tr>
<td>Ship Time &gt;= 14 days and &lt; 32 days</td>
<td>Yellow Background</td>
</tr>
<tr>
<td>ShipTime &gt;=32 days</td>
<td>Red Background</td>
</tr>
</tbody>
</table>

To do this we will make use of an element called a Conditional Class. This element is available as a child to any element that has a Class attribute. We will add three of these Conditional Class elements to the Data Table Column element colcalcShipTime.

6. Add three Conditional Class elements to the Data Table Column element colcalcShipTime.

• Select the Data Table Column element colShipTime.

• Add a Conditional Class element from the Element Toolbox.

• Give it an ID of condShort.

• Choose the backgroundGreen class in the Class attribute.

• Enter the Condition script that will select this Class:
• Add a second **Conditional Class** element to the **colShipTime** data table column.

• Give it an id of **condMedium**.

• Choose the **backgroundYellow** class in the class attribute.

• Enter the condition that will turn on this class:

![Conditional Class element example](image)

• Add a **Conditional Class** element to the **colShipTime** data table column. Give it an id of **condLong**.

• Choose the **backgroundRed** class in the class attribute.

• Enter the condition that will turn on this class:

![Conditional Class element example](image)

7. **Verify your report:**

• Click the Save All Opened Files button in the Toolbar to save your report definition.
- The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.

- Click the Preview button at the bottom of the Workspace Panel then Run in Browser. The final report should appear as shown in the Report Output section at the beginning of this lab.
Exports

The labs to this point have had no user interactivity outside of the pagination. Here we will provide a Button in the Report Footer that will export a data table to Excel. We will follow with an export to PDF.

Lab 1 – Excel

Objectives:

- Export the report to Microsoft Excel format.
- Apply column and data type formatting to the Excel export.
Directions:

1. Create a folder under Reports in the Application Panel and name it **Exports**.
   - Select the Reports Folder then right-click then **Add** and **New Folder**.

2. Copy the definition from **Lab1_Basic_DataTable** in the DataTables folder to the **Exports** folder.
   - Right-click **Lab1_Basic_DataTable** and choose **Copy**. Select the **Exports** Folder right-click and **Paste**.
   - Rename it to **Lab1_Excel_Export**.

3. Add **Button**, **Action** and **Target** elements to the **Report Footer** element.
   - In the Workspace Panel, click on the **Report Footer**. Double-click the **New Line** element in the Element Toolbox to add a blank line to the **Report Footer**.
   - Re-select the **Report Footer** element.
   - Double-click the **Button** element in the Element Toolbox to add a button to the **Report Footer**.
• Set the Button attribute values as follows:

![Button Attribute Values](image)

• Add an **Action.Export Native Excel** element under the button:
  
  o In the workspace panel, click on `btnExportExcel`.
  
  o Add an **Action.Export Native Excel** element.

  ![Action.Export Native Excel](image)

  o Set the ID of the action to `actExcelExport`. No other attribute values need to be changed.

  o Add a **Target.NativeExcel** element under `actExcelExport` and enter `tgtExcelExport` for the ID.

    - In the Workspace Panel, click on `actExcelExport`;
    
    - In the Element Toolbox, double-click on the **Target.NativeExcel** element. Use `tgtExcelExport` as the ID.

  ![Target.NativeExcel](image)

  o Set the attribute values of the target as follows:
Note: CurrentReport is chosen so that when the Excel Export button is clicked, the same data that fed the initial report will be the data exported. For more information refer to the information panel after clicking on the Report Definition File attribute.

4. Preview the report and click on the Export to Excel button.

- You may get the following prompt. Choose Open.

Note that the spreadsheet is not properly formatted for dates and currency. Though these problems can be fixed easily once in Excel, they are easily handled at design time with additional elements.
5. For each Data Table column in the report, add an Excel Column Format element:

- Click on the desired Data Table Column element.
- Double-click the Excel Column Format element in the Element Toolbox.

- Set the attribute values for the Excel Column Format elements for each column:
  
  o Set the Column Width to Auto.
  
  o Set the Data Type and Excel Format attribute values as appropriate for the specific column:
    
    ▪ colOrderID: set the Data Type to Number;
    
    ▪ colCustomerID: set the Data Type to Text;
    
    ▪ colOrderDate: set the Data Type to Date and set the Excel Format to Short Date;
    
    ▪ colShippedDate: set the Data Type to Date and set the Excel Format to Short Date;
    
    ▪ colFreight and colSubtotal: set the Data Type to Number and set the Excel Format to $#,##0.00;
    
    ▪ colShipCountry: set the Data Type to Text;
    
    ▪ colShipCity: set the Data Type to Text;

6. Verify your report:

- The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.

- Click the Preview button at the bottom of the Workspace Panel and then Run In Browser. The final report should appear as shown in the Report Output section at the beginning of this lab.

- Click the Save All Opened Files button in the Toolbar to save your report definition.
Lab 2 – PDF

In this lab we export to PDF which has its own peculiarities. The buttons will be exported out with the rest of the output. We will learn how to remove these buttons. We’ll also provide some specific pagination to the PDF output using the Printable Pagination element.

Objectives:

- Export the report to Adobe PDF format
- Hide the export button using Show Modes
- Set page numbers in the PDF export
Directions:

1. Copy the Report definition Lab1_Excel_Export in the Exports folder and rename the copy Lab2_PDF_Export.
   - Select Lab1_Excel_Export in the Exports folder.
   - Right-click and choose Copy.
   - Select the Exports folder and choose Paste.
   - Rename the copy to Lab2_PDF_Export.

2. Add the Export Button to the Report Footer:
   - In the Workspace Panel, click on the Report Footer;
   - Add a Space element and enter 3 for the size attribute.
   - Re-select the Report Footer element.
   - From the Element Toolbox, double-click the Button element.
   - In the Attributes Panel for the new Button, change the attribute values as follows:

3. Add an Action to the Button:
   - Click on btnExportPDF in the Workspace Panel
   - From the Element Toolbox, double-click the Action.ExportPDF element.
   - Set the ID attribute to actExportPDF.
4. Add the Target to the Action:
   - In the Workspace Panel, Click on the `actExportPDF` element.
   - From the Element Toolbox, double-click the `Target.PDF` element.

   ![Target.PDF Element](image)

   - In the Attributes Panel for the `Target.PDF`, set the attribute values as follows:

5. Run the report in the browser.
6. Click the PDF Export button to generate the PDF file. Note that the PDF Export button appears on the last page of the PDF version of the report. Close the browser to return to Logi Studio.
7. Set the Show Modes on the Report Footer to remove PDF button in exported report.
   - Click on the Report Footer element.
   - In the Attributes Panel, set the Show Modes attribute to `rdBrowser` by selecting from the drop-down list.

   ![Report Footer Element](image)
8. Add Page Header and Page Footer elements:

- Click on the topmost element in the workspace *(Report element Exports.Lab2_PDF_Export)*.
- Add a **Printable Paging** element from the Element Toolbox and press **F7** until moved to the top.

- Add a **Page Header** element to **Printable Paging** from the Element Toolbox.

- Add a **Page Footer** element to **Printable Paging** from the Element Toolbox.

- For both the **Page Header** and **Page Footer**, add a **Label**:
  - Ctrl-click on both the **Page Header** and **Page Footer** elements you just added under the **Printable Paging** element.
  - Double-click on the **Label** element in the General Elements section of the Element Toolbox to add a **Label** to both the **Page Header** and **Page Footer** elements.
• Set the attribute values to the Label under the Page Header element as follows.

![Diagram showing attribute values for Page Header]

• Set the caption attribute under the Page Footer’s label in the same way using an ID of `lblFooter`.

![Diagram showing attribute values for Page Footer]

9. Verify your report:

• Click the Save All Opened Files button in the Toolbar to save your report definition.

• The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.

• Click the Preview button at the bottom of the Workspace Panel then Run In Browser. The final report should appear as shown in the Report Output section at the beginning of this lab.

• Once the PDF has been exported, the PDF version will have the printable paging text displayed.
Lab 3 – Printable Paging (Optional Lab)

This lab will demonstrate two options for rendered output. One has page numbers on one continuous page with no pagination controls (Web Print) and one has no pagination at all and is designed for searching through text in the browser.

Objectives:

- Create a web printable and web searchable version of an interactive report (No export to Excel or PDF)

Report Definition

Final Report Output (HTML and PDF)

Data Table Columns have been collapsed.
### Web Print Output

<table>
<thead>
<tr>
<th>Order Id</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>ShippedDate</th>
<th>Freight</th>
<th>ShipCity</th>
<th>ShipCountry</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>10248</td>
<td>VINET</td>
<td>7/4/1996</td>
<td>7/15/1996</td>
<td>$32.38</td>
<td>Reims</td>
<td>France</td>
<td>$440.00</td>
</tr>
<tr>
<td>10249</td>
<td>TOMSP</td>
<td>7/5/1996</td>
<td>7/10/1996</td>
<td>$111.61</td>
<td>Münster</td>
<td>Germany</td>
<td>$1,863.40</td>
</tr>
<tr>
<td>10250</td>
<td>HANAR</td>
<td>7/8/1996</td>
<td>7/12/1996</td>
<td>$55.63</td>
<td>Rio de Janeiro</td>
<td>Brazil</td>
<td>$1,552.60</td>
</tr>
<tr>
<td>10251</td>
<td>VICTE</td>
<td>7/8/1996</td>
<td>7/15/1996</td>
<td>$41.34</td>
<td>Lyon</td>
<td>France</td>
<td>$654.06</td>
</tr>
<tr>
<td>10253</td>
<td>HANAR</td>
<td>7/10/1996</td>
<td>7/15/1996</td>
<td>$58.17</td>
<td>Rio de Janeiro</td>
<td>Brazil</td>
<td>$1,444.60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order Id</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>ShippedDate</th>
<th>Freight</th>
<th>ShipCity</th>
<th>ShipCountry</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>10277</td>
<td>MORGK</td>
<td>8/9/1996</td>
<td>8/13/1996</td>
<td>$125.77</td>
<td>Leipzig</td>
<td>Germany</td>
<td>$1,200.80</td>
</tr>
<tr>
<td>10278</td>
<td>BERGS</td>
<td>8/12/1996</td>
<td>8/16/1996</td>
<td>$92.69</td>
<td>Luleå</td>
<td>Sweden</td>
<td>$1,488.60</td>
</tr>
<tr>
<td>10279</td>
<td>LEHMS</td>
<td>8/13/1996</td>
<td>8/16/1996</td>
<td>$25.83</td>
<td>Frankfurt a.M.</td>
<td>Germany</td>
<td>$351.00</td>
</tr>
<tr>
<td>10280</td>
<td>BERGS</td>
<td>8/14/1996</td>
<td>9/12/1996</td>
<td>$8.98</td>
<td>Luleå</td>
<td>Sweden</td>
<td>$613.20</td>
</tr>
<tr>
<td>10281</td>
<td>ROMEY</td>
<td>8/14/1996</td>
<td>8/21/1996</td>
<td>$24.94</td>
<td>Madrid</td>
<td>Spain</td>
<td>$36.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order Id</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>ShippedDate</th>
<th>Freight</th>
<th>ShipCity</th>
<th>ShipCountry</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>10282</td>
<td>ROMEY</td>
<td>8/15/1996</td>
<td>8/21/1996</td>
<td>$126.99</td>
<td>Madrid</td>
<td>Spain</td>
<td>$155.40</td>
</tr>
<tr>
<td>10283</td>
<td>LILAS</td>
<td>8/16/1996</td>
<td>8/23/1996</td>
<td>$84.81</td>
<td>Barquisimeto</td>
<td>Venezuela</td>
<td>$1,414.80</td>
</tr>
<tr>
<td>10284</td>
<td>LEHMS</td>
<td>8/19/1996</td>
<td>8/27/1996</td>
<td>$76.56</td>
<td>Frankfurt a.M.</td>
<td>Germany</td>
<td>$1,170.38</td>
</tr>
<tr>
<td>10285</td>
<td>QUICK</td>
<td>8/20/1996</td>
<td>8/26/1996</td>
<td>$76.83</td>
<td>Cunevaule</td>
<td>Germany</td>
<td>$1,743.36</td>
</tr>
</tbody>
</table>
Directions:

1. Copy the report definition from Lab2_PDF and rename the copy Lab3_Web_PrintablePaging.
   - Select Lab2_PDF in the Exports folder.
   - Right-click and choose Copy.
   - Select the Exports folder and choose Paste.
   - Rename the copy to Lab3_Web_PrintablePaging.

2. Add the Web Print Version Button to the Report Footer:
   - In the Workspace Panel, click the Report Footer element.
   - Add a Button element from the General Elements folder.
   - In the Attributes Panel, set the Button element attribute values as follows:

   <table>
   <tr>
   <td style="background-color:#f2f2f2; padding:5px; width:100%;" align="center">
   <input type="button" value="Web Print" />
   </td>
   </tr>
   </table>

   - Select the Report Footer and add a Space element. Set the size attribute to 3.

   - Click on btnWebPrint in the Workspace Panel.
   - In the Attributes Panel, set the Action.Report element's ID attribute value to actShowPrintVersion.

   - In the Workspace Panel, click on the actShowPrintVersion element.
   - Add a Target.Report element from the Elements Toolbox.
   - In the Attributes Panel, set the Target.Report element attribute values as follows:
Preview the report. Click the **Web Printing** button to load the web printable version of the report and ensure that the report is rendered in one page with page numbers inserted.

5. **Add the Web Search button to the Report Footer:**
   - In the Workspace Panel, click on the **Report Footer**.
   - Add a **Button** element from the Element Toolbox.
   - In the Attributes Panel, change the attribute values as follows:
     - Select the Report Footer and add a **Space** element. Set the size attribute to 3.

6. **Add an Action to the Web Search Button:**
   - Click on **btnWebSearch** in the Workspace Panel.
   - Add an **Action.Report** element from the Element Toolbox and give it an **ID** of **actWebSearch**.
7. Add the **Target** to the Action.
   - In the Workspace Panel, Click on the **actWebSearch** element.
   - Add a **Target.Report** element from the Element Toolbox.
   - In the Attributes Panel for the **Target.Report**, set the attribute values as follows:

   ![Element - Target.Report](image)

8. Verify your report:
   - Click the **Save All Opened Files** button in the Toolbar to save your report definition.
   - The report definition elements in the Workspace Panel should appear as shown in **Report Definition** section at the beginning of this lab.
   - Click the **Preview** button at the bottom of the Workspace pane and then **Run in Browser**. The final report should appear as shown in the **Report Output** section at the beginning of this lab. There are additional sections for the **Web Printable** and **Web Search** options.
User Inputs

The User Inputs topic is designed as an introduction to capturing user input in the browser and utilizing it in queries behind visualizations. This concept is easily extended through more in-depth use of a new token that we discover called @Request. It is analogous to the Request object in web development. Using the @Request object we will replace all or part of an attribute (e.g. a source attribute in a data layer) to act as a “variable”.

Lab 1 – Select List with Submit

Objectives:

- Create an Input Select List and populate it with countries from the Northwind Orders table.
- Add a button that posts the user’s selection and reloads the report.
- Filter the data table by the country the user selected.
Directions:

1. Create a folder under Reports in the Application Panel and name it UserInputs.
   - Select the red Reports folder in the Application Panel
   - Right-click and choose Add then New Folder.
   - Rename the new folder to UserInputs.

2. Copy the definition from Lab1_Basic_DataTable in the DataTables folder.
   - Select the Lab1_Basic_DataTable lab right-click then Copy.
   - Select the UserInputs folder, right-click and choose Paste.
   - Rename to Lab1_SelectList_With_Submit.

3. Add a Input Select List element to the report header:
   - Select the Report Header element in the Workspace Panel.
   - Add an Input Select List from the Element Toolbox under the User Input folder.
   - In the Attributes Panel, change the Input Select List element’s ID attribute value to islShipCountry.

4. Add a Data Layer to the Input Select List.
   - In the Workspace Panel, click on the Input Select List element islShipCountry.
   - Add a DataLayer.SQL element from the Element Toolbox.
• Set the new DataLayer.SQL element’s attribute values as follows:

![DataLayer.SQL element](image)

5. Set the remaining attribute values of the Input Select List as follows:

![InputSelectList element](image)

Note: Most corporate database designs would dictate that separate columns in the select statement be used for the **Caption Column** and **Value Column** attributes in this element. The **Value Column** is typically associated with an ID of sorts (preferably an indexed key to speed up the SQL query), and will provide the filtering value to be used in the WHERE clause of the data table’s DataLayer.SQL element. The **Caption Column** provides a corresponding user-friendlier piece of data such as a name, abbreviation, etc. that the user will see in the select list. In the Northwind database there is no Country lookup table that would normally contain this ID and abbreviation. Therefore we must generate a unique list of countries from the **ShipCountry** column in the **Orders** table.

6. Preview the report. Confirm that the drop-down list is present and populated with distinct countries from the Orders table.

7. Add a **Button** next to the **Input Select List** to post the user’s selection and reload the report.
   • In the Workspace Panel, click on the **Report Header** element.
   • Add a **Button** element from the Element Toolbox.
• Set the button attribute values as follows:

![Button Attribute Values](image)

8. Add an **Action.Report** to the button:
   - In the Workspace Panel, click on **btnSubmit**.
   - Add an **Action.Report** element from the Element Toolbox.
   - Set the **ID** of the **Action.Report** element to **actSubmit**.

9. Add **Target. Report** to the Action:
   - In the Workspace Panel, click on **actSubmit**.
   - Add a **Target. Report** from the Element Toolbox.
   - Set the **ID** of the **Target. Report** to **tgtSubmit**.
   - Set the **Report File Definition** attribute to **UserInputs.Lab1_SelectList_With_Submit** from the drop-down list.

**Note:** Do **not** use **CurrentReport** for the **Report Definition File** attribute, since this would reuse the cached data layer result set without applying the new filtering value. Instead, specify the report definition name **UserInputs.Lab1_SelectList_With_Submit** to force the query to be rerun when the **Submit** button is clicked. That way, the data in the report will be refreshed with the filtering value applied.
10. Modify the query in the table’s Data Layer to filter orders by the selected country:
   - In the Workspace Panel, click on the `dlOrders` data layer;
   - Modify the query in the Source attribute in one of three ways:

   **Three Ways to Modify the Query**

   ![Diagram of Data Layer Configuration]

   - Change the SQL query to:
     ```sql
     Select Orders.OrderID,
             Orders.CustomerID,
             Orders.OrderDate,
             Orders.ShippedDate,
             Orders.Freight,
             Orders.ShipCity,
             Orders.ShipCountry
     From Orders
     Where Orders.ShipCountry = '@Request.is1ShipCountry~'
     ```
   - You can copy the SQL query above and paste it in the Attribute Zoom Window or the Query Builder.

11. Preview the report. Note that there are no records shown. Choose a country from the select list and click the Submit button. Run In Browser from the Application Panel.

Note that after making a selection and clicking Submit, the input select list does not remember the user’s selection. Also, after filtering by a specific country, there is no means for retrieving all the records.
12. Enable the drop-down control to remember the user’s selection.
   - In the Workspace Panel, click on the `islShipCountry` element.
   - Set the Default Value attribute to `@Request.islShipCountry~`.

13. **Preview** the report. Confirm that the **Input Select List** remembers the user’s selection after choosing a country and clicking on **Submit**.

14. Add a selection for **All Countries** to the drop-down control.
   - In the Workspace Panel, click on the `islShipCountry` element and set the attribute values as follows:

   ![Input Select List Attributes](image)

15. Modify the SQL query in the table data layer to handle the **All Countries** selection:
   - In the Workspace Panel, click on the `dlOrders` data layer.
Double-click the **Source** attribute name to open the Attribute Zoom window and add an **OR** clause to the SQL query as follows:

```sql
Select Orders.OrderID,
       Orders.CustomerID,
       Orders.OrderDate,
       Orders.ShippedDate,
       Orders.Freight,
       Orders.ShipCity,
       Orders.ShipCountry
From Orders
Where Orders.ShipCountry = '@Request.islShipCountry~'
Or '@Request.islShipCountry~' = 'All Countries'
```

You can copy the SQL query above and paste it in the **Attribute Zoom Window** or the **Query Builder**.

Preview the Report and note that the initial list of orders is empty. To provide an initial value for the Request parameter **islShipCountry**, you must add a **Default Request Parameter** and set an initial value.

16. Create a **Default Request Parameter** to set an initial value for **islShipCountry**.

   Select the topmost **Report** element **UserInputs.Lab1_SelectList_With_Submit** in the Workspace Panel.
• Add a **Default Request Parameter** element from the General Elements section of the Element Toolbox. Press **F7** until it is just under the **Report** element.

![Diagram showing the addition of a Default Request Parameter element and navigating to the Report element using F7.]

The **Attribute Panel** will change to expose an additional window for **Test Parameters**. This happens in the **Studio** any time that **Request Parameter** or **Input** elements are added to a definition. This allows the developer to set test values that will be used to override defaults when using the **Studio** editor’s **Preview** or **Run in Browser** features.

17. Enter a new request parameter named **islShipCountry**.

• Enter **islShipCountry** in the box and press Enter or Click on the Plus.

![Image showing the input of a new request parameter named islShipCountry.]

• Enter the value in the right hand section after the parameter has been added.

![Image showing the input of a value for the request parameter islShipCountry.]

The value entered for the request parameter **islShipCountry** must match both the sentinel value in the OR clause of the query in the **dlOrders** data layer, and the **Include Blank Caption** attribute value in the **Input Select List** element **islShipCountry**.
18. Verify your report:

- Click the **Save All Opened Files** button in the Toolbar to save your report definition.
- The report definition elements in the Workspace Panel should appear as shown in **Report Definition** section at the beginning of this lab.
- Right-click on **Lab1_Select_With_Submit** in the Application Panel and choose **Run In Browser**.
- The final report should appear as shown in the **Report Output** section at the beginning of this lab. Make sure you verify that there are 42 pages of records in the initial rendering. Choose a country and submit and verify orders appear from the country. Choose **All Countries**, click **Submit** and verify all 42 pages return.
In most circumstances involving a single selection input select list, there isn’t a lot of need for a Submit button. Here we see how the act of choosing a country will fire off an “event” that we can capture and act upon.

Lab 2 – Select List Event Driven

Objectives:

- Convert the drop-down list from Lab 1 to an event-driven list.
- Refresh the data table after making a country selection instead of refreshing the entire report.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Image" /></td>
</tr>
</tbody>
</table>
Directions:

1. Copy the definition from `Lab1_SelectList_With_Submit` in the `UserInputs` folder.
   - Select `Lab1_SelectList_With_Submit` lab and right-click then Copy.
   - Select the `UserInputs` folder, right-click and choose Paste.
   - Rename the copy to `Lab2_SelectList_Event_Driven`.

2. Remove the `btnSubmit` element from the report header:
   - Right-click on the `btnSubmit` icon in workspace panel under the report header, then select Delete. The button and its associated action and target (“the children”) will be removed from the header.

3. Add an Event Handler to the Input Select List.
   - In the Workspace Panel, click on `islShipCountry`.
   - Add an Event Handler element from the Element Toolbox.
   - Click on the Event Handler element in the Workspace Panel and set the attribute values as follows:
     - Set the ID to `evtShipCountry`.
     - Set the DHTML Event to `onchange`.
4. Add an **Action.RefreshElement** to the event handler:
   - In the Workspace Panel, click on **evtShipCountry**.

   ![Image of Workspace Panel with evtShipCountry highlighted]

   - Add an **Action.RefreshElement** from the Element Toolbox.

   ![Image of Element Toolbox with Action.RefreshElement highlighted]

   - Set the attribute values of the **Action.RefreshElement** element as follows:

   ![Image of Action RefreshElement attributes]

5. Select the **Data Table** element **dtOrders** and set the **Ajax Paging and Sorting** attribute to **True**.

6. Verify your report:
   - Click the **Save All Opened Files** button in the Toolbar to save your report definition.
   - The report definition elements in the Workspace Panel should appear as shown in **Report Definition** section at the beginning of this lab.
   - Right-click on **Lab2_SelectList_Event_Driven** in the Application Panel and choose **Run In Browser**.
   - The final report should appear as shown in the **Report Output** section at the beginning of this lab. Make sure you verify that there are 42 pages of records in the initial rendering. Choose a country and verify orders appear from the country. Choose **All Countries**, and verify all 42 pages return.
Lab 3 – Multi Select

This lab is the exception to the single user input not needing a **Submit** button. An event handler is not appropriate here since most any event will be triggered before the user has finished selecting countries. The **Submit** button allows the user to freely pick multiple country selections with ctrl-clicking and/or shift-clicking, and still have a separate click to run the report.

**Objectives:**

- Convert the drop-down control into a multi-select list control.
- Modify the data table SQL query to handle filtering multiple country selections.
Directions:

1. Copy the definition from \texttt{Lab1\_SelectList\_With\_Submit}.
   - In the Application Panel, select \texttt{Lab1\_SelectList\_With\_Submit}, right-click and choose \textit{Copy}.
   - Select the \texttt{UserInputs} folder, right-click and choose \textit{Paste}.
   - Rename the copy to \texttt{Lab3\_SelectList\_Multi\_Select}.

2. Modify the \textbf{Input select List} to allow the selection of multiple countries by changing the following attributes:
   - Click on \texttt{islShipCountry} on the Workspace Panel to select it. Modify the attribute values as follows:
3. Modify the SQL statement to filter for multiple countries:

- In the Workspace Panel, click on dlOrders. In the Attributes Panel, use the Attribute Zoom to edit the query in the Source attribute as follows:

    Select Orders.OrderID,
    Orders.CustomerID,
    Orders.OrderDate,
    Orders.ShippedDate,
    Orders.Freight,
    Orders.ShipCity,
    Orders.ShipCountry
    From Orders
    Where Orders.ShipCountry In
    (@SingleQuote.Request.islShipCountry~)
    Or 'All Countries' In (@SingleQuote.Request.islShipCountry~)

    The @SingleQuote token will take all the selections made in the input select list and add single quotes before and after each selection. The single quoted selections are separated by commas.

4. In the Workspace Panel, click on tgtSubmit. In the Attributes Panel, use the select list to select UserInputs.Lab3_SelectList_Multi_Select in the Report Definition File attribute.

5. Verify your report:

   - Click the Save All Opened Files button in the Toolbar to save your report definition.
   - The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.
   - Right-click on Lab3_Select_Multi_Select in the Application Panel and choose Run In Browser.
   - The final report should appear as shown in the Report Output section at the beginning of this lab. Make sure you verify that there are 42 pages of records in the initial rendering. Choose at least two countries with ctrl-click and click Submit. Choose All Countries, Ctrl-click and/or Shift-click to select at least one additional country, and click Submit. Verify that all 42 pages appear.
Lab 4 – Cascading Filters

This is one of the most popular labs. Frequently in reporting, we have a need to select from a list of something and have that selection filter the members of another list. This follows the natural hierarchy in the data. Here we will extend the selection of a country to filter a list of cities, hence the name cascading filters. Using this lab you can easily extend this to additional select lists (or other input element types) to match your data hierarchy.

Objectives:

- Create an Input Select List that displays cities filtered by the selected country.
- Modify the data table SQL query to filter by country and city.

---

**Report Definition**

- UserInputs.Lab4_Cascading.Filters
  - Default Request Parameters
  - Style
  - Report Header
    - implicitContainer
      - idShipCountry
        -.ShipCountry
        - evtShipCountry
          - actRefreshOrders_and_Cities
    - implicitCity
      - idShipCity
        - ShipCity
        - evtShipCity
          - actRefreshOrders_From_City
  - Body
    - dtOrders
      - idOrders
      - colOrderID
      - colCustomerID
      - colOrderData
      - colShippedDate
      - colFreight
      - colShipCity
      - colShipCountry
      - npGraphical
    - Report Footer

**Final Report Output**

Select a Country: All Countries
Select a City: All Cities

<table>
<thead>
<tr>
<th>OrderId</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>ShippedDate</th>
<th>Freight</th>
<th>ShipCity</th>
<th>ShipCountry</th>
</tr>
</thead>
<tbody>
<tr>
<td>10247</td>
<td>VNET</td>
<td>7/15/1996</td>
<td>7/16/1996</td>
<td>$32.38</td>
<td>Reims</td>
<td>France</td>
</tr>
<tr>
<td>10249</td>
<td>TOMSP</td>
<td>7/16/1996</td>
<td>7/16/1996</td>
<td>$11.61</td>
<td>Munster</td>
<td>Germany</td>
</tr>
<tr>
<td>10250</td>
<td>HANAR</td>
<td>7/18/1996</td>
<td>7/12/1996</td>
<td>$65.83</td>
<td>Rio de Janeiro</td>
<td>Brazil</td>
</tr>
<tr>
<td>10251</td>
<td>VICTE</td>
<td>7/17/1996</td>
<td>7/15/1996</td>
<td>$6.14</td>
<td>Lyon</td>
<td>France</td>
</tr>
<tr>
<td>10253</td>
<td>SUPRD</td>
<td>7/19/1996</td>
<td>7/16/1996</td>
<td>$58.17</td>
<td>Charleroi</td>
<td>Belgium</td>
</tr>
<tr>
<td>10254</td>
<td>HANAR</td>
<td>7/19/1996</td>
<td>7/16/1996</td>
<td>$58.17</td>
<td>Rio de Janeiro</td>
<td>Brazil</td>
</tr>
<tr>
<td>10256</td>
<td>CHOPS</td>
<td>7/20/1996</td>
<td>7/17/1996</td>
<td>$82.56</td>
<td>Berlin</td>
<td>Germany</td>
</tr>
<tr>
<td>10257</td>
<td>RICSU</td>
<td>7/21/1996</td>
<td>7/15/1996</td>
<td>$148.33</td>
<td>Genève</td>
<td>Switzerland</td>
</tr>
<tr>
<td>10260</td>
<td>HILAA</td>
<td>7/19/1996</td>
<td>7/22/1996</td>
<td>$81.91</td>
<td>San Cristobal</td>
<td>Venezuela</td>
</tr>
<tr>
<td>10261</td>
<td>EINAR</td>
<td>7/18/1996</td>
<td>7/23/1996</td>
<td>$140.51</td>
<td>Graz</td>
<td>Austria</td>
</tr>
<tr>
<td>10262</td>
<td>CENIC</td>
<td>7/19/1996</td>
<td>7/25/1996</td>
<td>$3.25</td>
<td>Mexico D.F.</td>
<td>Mexico</td>
</tr>
<tr>
<td>10265</td>
<td>CENTC</td>
<td>7/19/1996</td>
<td>7/25/1996</td>
<td>$3.25</td>
<td>Mexico D.F.</td>
<td>Mexico</td>
</tr>
<tr>
<td>10266</td>
<td>OTTI</td>
<td>7/22/1996</td>
<td>7/25/1996</td>
<td>$48.29</td>
<td>Atlanta</td>
<td>USA</td>
</tr>
<tr>
<td>10267</td>
<td>BOLM</td>
<td>7/12/1996</td>
<td>7/16/1996</td>
<td>$3.67</td>
<td>Bratislava</td>
<td>Slovakia</td>
</tr>
<tr>
<td>10268</td>
<td>BLOB</td>
<td>7/12/1996</td>
<td>7/16/1996</td>
<td>$3.67</td>
<td>Bratislava</td>
<td>Slovakia</td>
</tr>
<tr>
<td>10269</td>
<td>WARTH</td>
<td>7/16/1996</td>
<td>7/16/1996</td>
<td>$25.73</td>
<td>Oulu</td>
<td>Finland</td>
</tr>
<tr>
<td>10270</td>
<td>FRANK</td>
<td>7/29/1996</td>
<td>8/8/1996</td>
<td>$208.38</td>
<td>München</td>
<td>Germany</td>
</tr>
</tbody>
</table>

Data Table Column Elements Are Collapsed
Directions:

1. Copy the definition from Lab2_SelectList_Event_Driven.
   - In the Application Panel, select Lab2_SelectList_Event_Driven and Right-click then Copy.
   - Select the UserInputs folder, right-click and choose Paste.
   - Rename the copy to Lab4_Cascading.Filters.

2. Add an Input Grid to the Report Header element to contain and align user input elements:
   - In the Workspace Panel, select the Report Header.
   - Select the Input Grid element from the User Inputs folder in the Element Toolbox. Press F7 to move up.
   - Set the ID attribute of the Input Grid element to iptContainer.

3. Move islShipCountry with the associated Data Layer and Action into the Input Grid:
   - In the Workspace Panel, click on Input Select List element islShipCountry.
   - Right-click, then choose Cut.
4. Add an Input Select List for the ShipCity:
   - In the Workspace Panel, click on the Input Grid `iptContainer`;
   - Add a second Input Select List from the User Inputs Folder of the Element Toolbox.
   - Set the ID of the new Input Select List to `islShipCity`.

5. Add a Data Layer to `islShipCity`:
   - In the Workspace Panel, click `islShipCity`;
   - Add a DataLayer.SQL element from the Element Toolbox.

   - Click on the DataLayer.SQL element in the Workspace Panel and set the attribute values as follows:
• Cut and paste the following into the **Source** attribute value:

```
Select Distinct ShipCity From Orders Where ShipCountry = '@Request.islShipCountry~' OR '@Request.islShipCountry~' = 'All Countries'
```

6. Set the attribute values for the **islShipCity** select list as follows:

![InputSelectList](image_url)

7. Add an **Event Handler** element to the **islShipCity** input select list:

- In the Workspace Panel, click **islShipCity**.
- Add an **Event Handler** element from the Element Toolbox.
- Set the attribute values for the **Event Handler** element as follows:

![EventHandler](image_url)
8. Add an `Action.RefreshElement` element under the event handler `evtShipCity`:
   - In the Workspace Panel, click `evtSelectCity`.
   - Select an `Action.RefreshElement` element from the Element Toolbox.
   - Set the attribute values for the Action as follows:

   ![Image of Action.RefreshElement element]

9. Modify the `actRefreshOrders` action under `islShipCountry` to refresh both the cities select list and the data table:
   - Set ID to `actRefreshOrders_and_Cities`.
   - Set Element ID to `dtOrders, islShipCity`.

   ![Image of Action.RefreshElement element with updated ID]

10. Change the SQL statement for `dlOrders` (copy and paste from the SQL below):

    ```sql
    From Orders
    Where (Orders.ShipCity = '@Request.islShipCity~' Or '@Request.islShipCity~' = 'All Cities')
    And (Orders.ShipCountry = '@Request.islShipCountry~' Or '@Request.islShipCountry~' = 'All Countries')
    ```

11. Initialize the `Input Select List` element `islShipCity` to All Cities for the first run of the report.
   - In the Workspace Panel, click the Default Request Parameters element at the top of the report definition.
   - Add a new Default Request Parameter named `islShipCity` and set its value to All Cities.
Preview the report and select a country from the drop-down list. Verify that both the cities drop-down list and data table are filtered by the country selected. Select a city and verify that the data table is filtered by the country and city selected.

Now select another country and note that the table contains no data. The problem here is that the report is not updating the current city selection until a new city is selected. Consequently, the table shows no data when the selected city does not exist within the selected country.

The solution is to force `@Request.islShipCity` to contain the value ‘All Cities’ whenever a user chooses a country. We do this from an element called a Link Parameters element.

12. Force the data table to assume All Cities when a new country is selected.
   - Select `actRefreshOrders_and_Cities` in the Workspace Panel.
   - Select Link Parameters from the Element Toolbox.
   - In the Attributes Panel, create a link parameter called `islShipCity` and set the value to All Cities.
13. Verify your report:

- Click the **Save All Opened Files** button in the Toolbar to save your report definition.
- The report definition elements in the Workspace Panel should appear as shown in **Report Definition** section at the beginning of this lab.
- Right-click on **Lab4_Cascading_Filters** in the Application Panel and choose **Run in Browser**.
- The final report should appear as shown in the **Report Output** section at the beginning of this lab. Make sure you verify that there are 42 pages of records in the initial rendering.
- Choose a Country and verify the proper cities are showing and the proper Country is shown in the orders list itself.
- Choose a City and verify the orders list filters by the country and the city.
- Choose a new country, choose a city and verify that the proper orders appear.
To do simple validation we make use of child elements to the user input to be validated. These elements are simply called **Validation** elements. Once added to a user input, these validation elements can easily provide messages to the user when they have not put the correct data in a user input.

### Lab 5 – Input Validation

**Objectives:**

- Create three textbox elements that allow users to input their Last Name, SSN, birth date and email address.
- Add validation to each control to ensure valid input.
- Add a Input Text element to demonstrate the event handler using an **Action.Javascript** for Javascript based validation.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="UserInputs_Lab5_InputValidation" /></td>
<td><img src="image" alt="Message from webpage" /></td>
</tr>
<tr>
<td><img src="image" alt="UserInputs_Lab5_InputValidation" /></td>
<td><img src="image" alt="Message from webpage" /></td>
</tr>
<tr>
<td><img src="image" alt="UserInputs_Lab5_InputValidation" /></td>
<td><img src="image" alt="Message from webpage" /></td>
</tr>
</tbody>
</table>
Directions:

1. In the Application Panel, under the UserInputs folder under Reports, create a new definition called Lab5_InputValidation.
   - Right-click on the UserInputs folder;
   - Select Add then select New Definition;
   - Rename the new definition to Lab5_InputValidation.

2. Add an Input Grid to the body of the report:
   - In the Workspace Panel, click on the Body element;
   - Add an Input Grid element from the User Inputs folder in the Element Toolbox.

3. Add an Input Text element to the Input Grid:
   - In the Workspace Panel, click on the Input Grid.
   - Add an Input Text element in the element Toolbox under the User Inputs folder.
4. Add a Validation.Required element to the input text element.
   - In the Workspace Panel, click on txtLastName.
   - Add a Validation.Required element from the Element Toolbox.
   - Enter You must enter your Last Name: in the Error Message attribute of the Validation.Required element.

5. Add a Text box to hold the SSN.
   - Select the Input Grid element.
   - Add an Input.Text element from the Element Toolbox.
Set the attribute values as follows:

6. Add a Validation.Required element to the input text element.
   - In the Workspace Panel, click on txtSSN.
   - Add a Validation.Required element from the Element Toolbox.
   - In the Workspace Panel, click on the Validation.Required element. In the Attributes Panel set the error message to You must enter your SSN.

7. Add a Validation.Numeric element to the input text element.
   - In the Workspace Panel, click on txtSSN.
   - Add a Validation. Numeric element from the Element Toolbox.
   - In the Workspace Panel, click on the Validation.Numeric element. In the Attributes Panel set the error message to SSN must be numeric.

8. Add a Validation.Length element to the input text element:
   - In the Workspace Panel, click on txtSSN;
   - Add a Validation.Length element from the Element Toolbox.
• In the Workspace Panel, click on the Validation.Length element. In the Attributes Panel set the error message to **SSN must be 9 digits**. Set the Maximum Length to 9 and the Minimum Length to 9.

9. Add a Start Date element to the input Grid

• Select the **Input Grid** element

• Add an **Input.Date** element from the Element toolbox.

• Set the attribute values as follows:

![Element - InputDate](image)

• Add a **Validation.Required** element to **dateStartDate** and enter **You must enter a start date** in the Error Message attribute

• Add a **Validation.Date** element to **dateStartDate** and enter **You must enter a legal start date** in the Error Message attribute.

The **Validation.Date** element will respect whatever date format that the application defaults with. This can be changed by adding a **Globalization** element to the **_Settings** definition. There are different input date formats defined. The default is (dd-mm-yyyy).
10. Add a blank line and a **Submit** button in the body of the report:

- In the Workspace Panel, click on the **Body** element.
- Add a **New Line** element from the Element Toolbox.
- In the Workspace Panel, click on the **Body** element.
- Add a **Button** element from the Element Toolbox.
- Set the attribute values for the Button element:
  - Set the **ID** to `btnSubmit`;
  - Set the **Caption** to **Submit**.

11. Add an **Action.Report** element under **btnSubmit**:

- In the Workspace Panel, click on **btnSubmit**.
- Add an **Action.Report** element from the Element Toolbox.
- Set the **ID** of the **Action. Report** element to `actSubmitForm`;
- Set the **Validate Input** attribute to **True**:


- In the Workspace Panel, click on **actSubmitForm**.
- Add a **Target. Report** element from the Element Toolbox.
• Set the ID of the `Target.Report` element to `tgtSubmitForm`.

Since there is no data layer we need not set the `Report Definition File` attribute in the `Target.Report` element. The default target is `CurrentReport`.

13. Preview the report and verify the validations as shown in the Report Output section at the beginning of the lab.

  • Leave `SSN` empty and click on `Submit`.
  • Fill in with non numeric and click on `Submit`.
  • Fill in with 10 numbers and click on `Submit`.
  • Fill in with 9 numbers and click on `Submit`.

14. Close the browser and add another `Input Text` element to the Input Grid:

  • In the Workspace Panel, click on the `Input Grid`.
  • Add an `Input Text` element from the Element Toolbox.
  • Set the attribute values for the element as follows:
15. Add an event handler to the txtEmailAddress element.

- Select **txtEmailAddress**.
- Add an **Event Handler** element from the Element Toolbox.
- Enter **onblur** for the **DHTML Event** attribute and set the **ID** to **evtValidateEmailAddress**.

Onblur will fire when the **Input Text** element loses focus. This can happen when someone tabs away, clicks away, or sometimes when an enter key is pressed.

- Enter an **Action.Javascript** element to the **Event Handler** element. Enter **actCheckEmail** for the **ID**.
- Enter the following **JavaScript** code in the **Javascript** attribute (you can copy it from this document and paste it into the **Javascript** attribute in **Action.Javascript**):

```javascript
var emailAddr = document.forms[0].txtEmailAddress.value;
var emailReg = /^\w+([-\w.]*\w+)\@[\w][-\w]*\.(\w{2})\.(com|net|org|edu|int|mil|gov|arpa|biz|aero|name|coop|info|pro|museum)$/;
if (!emailReg.test(emailAddr))
    alert("You must type a valid email address.");
```

You may want to use the **Attribute Zoom feature** by double-clicking on the **Javascript** attribute name.
16. Run the Lab in the Browser.

- Enter legal values for **SSN** and **Start Date**.
- Enter an invalid **Email Address** and press the Tab key or click outside the text box.
Hierarchical Reports

This topic is universally popular and contains a Business Intelligence staple which is drilldown. Drilldown can be defined as a process by which a person selects a particular aggregate such as a total dollar value of all the customer’s current orders and clicks on it in order to see the next level of granularity, in this case all the orders for that customer. This allows the user to pick and choose what records need to be investigated.

To get to drilldowns, we provide some background information around the Group Filter which is a Logi equivalent to the Sql Group By statement. As always, the Sql equivalents will run faster but elements like the Group Filter are handy when Sql DataLayer elements are not available as a data source.
### Lab 1 – Grouping Alternatives

**Objective:**

- Demonstrate the grouping of data for aggregation using Logi Elements and SQL Server Functionality.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Using Logi Elements</strong></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Diagram of report definition using Logi Elements" /></td>
<td><img src="image" alt="Screenshot of final report output" /></td>
</tr>
<tr>
<td><strong>Using SQL Functionality</strong></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Diagram of report definition using SQL Functionality" /></td>
<td><img src="image" alt="Screenshot of final report output" /></td>
</tr>
</tbody>
</table>
Directions:

1. Create a folder under Reports in the Application Panel and name it `Hierarchical_Reporting`.
   - Select the red `Reports` folder in the Application Panel
   - Right-click and select `Add` and `New Folder`.
   - Rename the folder to `Hierarchical_Reporting`.

2. Create a new Definition for Grouping Alternatives.
   - Select the `Hierarchical_Reporting` folder.
   - Right-click and select `Add` and `New Definition`.
   - Rename the new definition to `Lab1_Grouping_Alternatives`.

   First we will create a table using Logi Elements. We will then copy the DataLayer and disable the original. Finally we will apply the use of SQL grouping in the copy.

3. Create a Data Table (using the wizard)
   - Right-click the `Body` Element in the Workspace Panel, and select `Elements Wizards` ➔ `Add a Data Table`. 
• Click **Next** at the starting prompt.

• Accept **connNorthwind** as the **Connection ID**.

• Enter the following SQL in the Query Window (or you can paste it in the bottom portion of the query builder (as shown below).

```
Select Orders.CustomerID, [Order Subtotals].Subtotal
From Orders Inner Join
```

• The following query is available for copy and paste.

```
Select Orders.CustomerID, [Order Subtotals].Subtotal
From Orders Inner Join
```

• Click **OK** if in the query builder.
- Click **Next** in the wizard.

- Accept both columns shown to be added to the table and click **Next**.

- Accept **Yes** for the **Interactive Paging** and click **Next**.

- Accept **Rows per Page** at 20 and click **Next**.

- Accept **Graphical / Images** and click **Next** and **Finish**.

- The following definition is built by the wizard.
- Change the DataTable and Datalayer to `dtCustomers` and `dlCustomers_Logi_Elements` respectively.

- Select the `dlCustomers_Logi_Elements` element in the Workspace Panel.

4. Add a Group Filter from the Element Toolbox to `dlCustomers_Logi_Elements`.

- Rename the new Group Filter to `grpCustomerID`.

- Update the attribute values in `grpCustomerID` as shown.

Preview the Report Now. You will see that the Customers are Grouped together but the Subtotal reflects just the first Order’s subtotal for that Customer. More work is needed to create an aggregate against the subtotal.
5. Add a **Group Aggregate Column** element to the Group Filter `grpCustomerID`.

- Select the Group Filter element `grpCustomerID`.
- Add a **Group Aggregate Column** element from the Element Toolbox.

- Fill in the attribute values for the new **Group Aggregate Column** as follows:

- Replace the **Caption** of the Label `lblSubTotal` with the new **Group Aggregate Column** (`grpAggSubtotal`).
6. Verify your report:
   - Click the Save All Opened Files button in the Toolbar to save your report definition.
   - The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.
   - Right-click on **Lab1_Grouping_Altネatives** in the Application Panel and choose **Run In Browser**.
   - The final report should appear as shown in the Report Output section at the beginning of this lab.

7. Copy the current DataLayer (**dlCustomers_Logi_Elements**)
   - Select **dlCustomers_Logi_Elements** in the Workspace Panel.
   - Right-click and choose **Copy**.
   - Select **dtCustomers** then right-click and choose **Paste**.
   - Press **F7** so that the two DataLayers are adjacent.
• Remark out the Second DataLayer which will act as the Original. Right-click on the second **dlCustomers_Logi_Elements** and choose **Remark**. The element will turn green.

• Rename the topmost DataLayer from **dlCustomers_Logi_Elements** to **dlCustomers_SQL**.

• Remove the **Group Filter** from **dlCustomers_SQL**. Select the **grpCustomerID** element. Right-click and choose **Delete** or press the **Del** key.

• Select **dlCustomers_SQL** and enter the following SQL query in the **Source** attribute.

```
Select Orders.CustomerID, Sum([Order Subtotals].Subtotal) as grpAggSubtotal
From Orders Inner Join
Order Subtotals On Orders.OrderID = [Order Subtotals].OrderID
Group by Orders.CustomerID
```

Note: the Alias assigned to the Sum aggregation in this SQL query is named **grpAggSubtotal** so that the same label can be used in the **colSubtotal** Data Table Column.

• Save the definition – Right-click on **Lab1_Grouping_Alternatives** and choose **Run In Browser**.
8. Compare the two methods for performance.

- Make sure the debugger is set to DebuggerLinks in the toolbar.

- Run in the browser

- While in the browser click on Debug this page.

- Scroll down and Pick a point in the Debug page where both grouping methods will have an entry (i.e. Done paging)

- Note the time stamp in the right most column in the page.

- Minimize (Do not close) the browser.

9. Swap the DataLayers

- Move to Logi Studio and select dlCustomers_SQL.
• Right-click and choose **Remark**

• Select **dlCustomers_Logi_Elements** then right-click and choose **Remark**. This will **Un-Remark** the DataLayer.

• Save the definition – Right-click on **Lab1_Grouping_Alternatives** and choose **Run In Browser**.

![Diagram](image.png)

• A second tab will appear next to the original run with the Logi Elements. Click on **Debug this page**.

• Check the timestamp next to **Done paging**. The value for the second tab (SQL Functionality) should be shorter than for the first tab.

This exercise is designed to reinforce one of the best practices: For best performance use SQL instead of Logi Elements. However there are many DataLayers which do not afford the luxury of modification, therefore becoming familiar with Group Filters is important.
Lab 2 – Grouped Flat Table & Aggregates

Some reports will require aggregations and details together for all records. In this lab we will be grouping on the Ship Country column in the Orders table. Within Ship Country we will group by Customer. When the Country or Customer changes within a Country there will be an aggregation displayed.

Objectives:

- Create a data table that has aggregation by multiple groups along with the details supporting the aggregation.
Hierarchical_Reporting_Lab2_Grouped_Flat_Table

Body

dbOrders

<table>
<thead>
<tr>
<th>gidShipCountry</th>
</tr>
</thead>
<tbody>
<tr>
<td>grpAggShipCountry</td>
</tr>
<tr>
<td>grpCustomerID</td>
</tr>
<tr>
<td>grpAggCustomerID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>colShipCountry</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbShipCountry</td>
</tr>
<tr>
<td>Sort</td>
</tr>
<tr>
<td>Hide Duplicates</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>colCustomerID</th>
</tr>
</thead>
<tbody>
<tr>
<td>lbCustomerID</td>
</tr>
<tr>
<td>Sort</td>
</tr>
<tr>
<td>Hide Duplicates</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>colOrderID</th>
</tr>
</thead>
<tbody>
<tr>
<td>colOrderDate</td>
</tr>
<tr>
<td>colShippedDate</td>
</tr>
<tr>
<td>colShipCity</td>
</tr>
<tr>
<td>colFreight</td>
</tr>
<tr>
<td>colSubtotal</td>
</tr>
<tr>
<td>grpGraphical</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>gsrCustomerID</th>
</tr>
</thead>
<tbody>
<tr>
<td>colCaption</td>
</tr>
<tr>
<td>lbCaption</td>
</tr>
<tr>
<td>colTotal</td>
</tr>
<tr>
<td>lbTotal</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>gsrShipCountry</th>
</tr>
</thead>
<tbody>
<tr>
<td>colCountryCaption</td>
</tr>
<tr>
<td>lbCountryCaption</td>
</tr>
<tr>
<td>colTotal</td>
</tr>
<tr>
<td>lbTotal</td>
</tr>
</tbody>
</table>

Report Footer

Some Data Table Columns Are Collapsed
Directions:

1. Add a new definition to the Hierarchical_Reporting folder.
   - Click on Lab1_Basic_DataTable in the DataTables folder.
   - Right-click and choose Copy.
   - Select the Hierarchical_Reporting folder then right-click and chose Paste.
   - Rename the copy to Lab2_Grouped_Flat_Table.

2. Add a Group filter to dlOrders.
   - Select the dlOrders element in the Workspace Panel.
   - Add a Group Filter element from the Element Toolbox.
   - Enter the attribute values as shown:

   ![Element - GroupFilter](image)

   Note: Keep Grouped Rows will set to True otherwise only one row per unique value (Country...Customer) will be shown. In a flat table we want to see all the details...therefore all the group values must be shown.

3. Add a group aggregate column to grpShipCountry.
   - Select grpShipCountry and add a Group Aggregate Column from the Element Toolbox.
   - Enter in the attribute values as shown:
4. Enter a second group filter. It will be a “child” to `grpShipCountry`.
   - Select `grpShipCountry`. Add a Group Filter from the Element Toolbox.
   - Enter the attribute values as shown:

   ![Group Filter](image)

5. Add a Group aggregate to `grpCustomerID`.
   - Select `grpCustomerID` and add a Group Aggregate Column from the Element Toolbox.
   - Enter the attribute values as shown:

   ![Group Aggregate](image)

Now that once the Group Filters are in place the Group Summary Row elements can be added after the Summary Row elements. This is the DataLayer definition to this point.
6. Add a **Group Summary Row** to **dtOrders** based on the Customer ID.

- Select **dtOrders** and add a **Group Summary Row** from the Element Toolbox. Enter **gsrCustomerID** for the ID attribute.

- Enter in **grpCustomerID** for the Group Filter ID. If the Drop down list in the Group Filter ID attribute doesn’t show your desired Group Filter ID then you must type it in exactly as in the DataLayer.

- Add a **Column Cell** element to the Group Summary Row from the Element Toolbox. Enter in **colCaption** for the ID.

- Enter in a Column Span of 6 and ThemeAlignRight for the Class attribute.

- Select **gsrCustomerID** and add a second **Column Cell** element from the Element Toolbox. Enter in **colTotal** for the ID.

- Enter a Column Span of 2 and ThemeAlignRight for the class attribute.

- Add **Label** element to **colCaption** with an ID of **lblCaption**.

- Enter in **@Data.CustomerID~ Total:** for the caption attribute of **lblCaption**.

- Add **Label** element to **colTotal** with an ID of **lblTotal**.

- Enter **@Data.grpAggCustomerID~** to the Caption. Enter Currency for the format.
7. Add a second Group Summary Row to dtOrders based on the ShipCountry.

- Select the DataTable dtOrders.
- Add a Group Summary Row element from the Element Toolbox.
- Enter gsrShipCountry for the ID attribute. Enter in grpShipCountry for the Group Filter ID attribute. If the Drop down list in the Group Filter ID attribute doesn’t show your desired Group Filter ID then you must type it in exactly as in the DataLayer.
- Enter in a Column Span of 6 and ThemeAlignRight for the Class attribute.
- Add a second Column Cell element to gsrShipCountry with an ID of colCountryTotal.
- Select colCountryCaption and add a Label element from the Element Toolbox with and ID of lblCountryCaption.
- Enter in @Data.ShipCountry~ Total: to the Caption attribute.
- Add lblCountryTotal to colCountryTotal and enter @Data.grpAggShipCountry~ in the Caption attribute.
- Enter Currency for the Format attribute of lblCountryTotal.

8. Remove the duplicates from the ShipCountry and CustomerID columns.

- Select the colShipCountry Data Table Column.
- Select the colCustomerID Data Table Column.
- Add a Hide Duplicates element from the Element Toolbox. Choose CustomerID for the Data Column.
- Right-click on Lab2_Grouped_Flat_Table in the Application Panel and choose Run In Browser.

Note: You may want to consider disabling the sort elements for this report by using right-click Remark. Clicking on sort elements at this point will break the intended groupings. Click on the Order Date column header to demonstrate.
If sorting is still required, enter several columns in to the sort element’s Data Column attribute. Be sure to always start with what you are grouping...in this case ShipCountry and CustomerID would be first two columns in any column’s sort element’s Data Column attribute. The following step is an example

9. Change the **OrderDate** column to keep the proper grouping when sorted.
   - Close the browser window if open.
   - Select the **Sort** element under the **colOrderDate** Data Table Column element.
   - Enter **ShipCountry, CustomerID, OrderDate** for the **Data Column** attribute.
   - Enter **Text, Text, Date** in the **Data Type** attribute.
   - Enter in **Ascending, Ascending, Descending** in the **Reverse Sort Sequence** attribute.

   The sorting still works within the OrderDate column but doesn’t break the groupings for ShipCountry and CustomerID.

10. Final Housekeeping
   - Change the **Class** attribute to **ThemeAlignRight** for the **colOrderDate, colShippedDate** and **colFreight** Data Table Columns.

11. Verify your report:
   - Click the Save All Opened Files button in the Toolbar to save your report definition.
   - The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.
   - Right-click on **Lab2_Grouped.Flat_Table** in the Application Panel and choose **Run In Browser**.
   - The final report should appear as shown in the Report Output section at the beginning of this lab.
This lab is a classic drilldown. Here we are presented with a list of customers with their aggregates as shown in Lab1 of this topic. A link will be created on the total for that customer and when clicked, the next level of granularity (Orders for that Customer) will appear. Each order total is a link and when clicked, we see the line items for that order.

Using this method you can extend this type of drilldown several more levels depending on real estate.

**Lab 3 – Drilldown with Sub Data Layers**

**Objectives:**

- Create a three level drilldown that uses subdata layers at each level.
Definition
Directions:

1. Copy Lab1_Grouping_Alternatives.
   - Select Lab1_Grouping_Alternatives from the Hierarchical Reports folder. Right-click and then Copy.
   - Select the Hierarchical_Reports folder. Right-click then Paste.
   - Rename the copy to Lab3_Drilldown_SubdataLayer.

2. Remove the DataLayer dlCustomers_SQL by selecting and pressing Del or right-click and choose Delete.

3. Add the Customer Name to the Data Table dtCustomers.
   - Select dlCustomers_Logi_Elements and modify the Source attribute (query builder or zoom window).
   - Update the SQL with the following (copy and paste ok).
     
     ```sql
     Select Orders.CustomerID, [Order Subtotals].Subtotal, Customers.CompanyName as CustomerName
     From Orders Inner Join
     ```
   - Add the new column in the Select statement to the Data Table. Select dtCustomers then right-click and Element Wizards then Add Data Columns.
   - Uncheck All and re-check CustomerName

![Add Data Columns](image-url)
• Click **Next** and **Finish**. Select the **colCustomerName** Data Table Column and press **F7** twice to Move Up.

• Change the Column Header in **colCustomerName** to **Customer Name**.

• Change the Column Header in **colSubTotal** to **Customer Total**.

• Change the **Class** attribute to **ThemeAlignRight**.

This amounts to the first level of the three level drilldown. Next is the MoreInfoRow and the action to control its behavior.

4. Add a **MoreInfoRow** element to **dtCustomers**.

   • Select **dtCustomers** and add a **More Info Row** element from the Element Toolbox.

   • Enter the attribute values as shown.

5. Add an **Action.ShowElement** to the label under the **colSubTotal** column (**lblSubtotal**). This will create the link to open up the second level. Be sure to add the link to the **Label** element and not the **Data Table Column** element.

6. Enter in the attribute values as shown for the Action:
For the 2\textsuperscript{nd} level we will add a SubDataTable to render the individual orders for a specific customer.

7. Add a Sub Data Table to hold the second level data.

- Select \texttt{mirOrderSummary} and add a \texttt{SubDataTable} element from the Element Toolbox.
- Enter in \texttt{sdtOrderSummary} for the ID.
- Add a \texttt{SubDataLayer} element to \texttt{sdtOrderSummary} with an ID of \texttt{sdlOrderSummary}.
- Add a conventional (child) \texttt{DataLayer.SQL} to \texttt{sdlOrderSummary}.
- Enter in \texttt{dlOrderSummary} for the ID of the child data layer and \texttt{connNorthwind} for the Connection ID.
- Enter the following sql statement to the source element of \texttt{dlOrderSummary}:

```sql
Select Orders.OrderID, Orders.CustomerID, Orders.OrderDate, 
Sum([Order Details].UnitPrice * [Order Details].Quantity * (1 - 
[Order Details].Discount)) As OrderTotal, Orders.Freight, 
Orders.ShipCountry,Orders.ShipCity, Orders.ShippedDate 
From [Order Details] Inner Join 
Orders On Orders.OrderID = [Order Details].OrderID 
Group By Orders.OrderID, Orders.CustomerID, Orders.OrderDate, 
Orders.Freight,Orders.ShipCountry, Orders.ShipCity, Orders.ShippedDate
```

Note: there must be a common column between the parent and “child” data layers. The parent in this case is the \texttt{dtCustomer Data Table}. These columns can be either in the select statement of the sql or added data layer columns. In this case the CustomerID in the “parent” will match the CustomerID in the child.

- Select the \texttt{SubDataLayer} element \texttt{sdlOrderSummary}. Add a \texttt{Subdata Layer Relation Column} from the Element Toolbox.
- Enter in \texttt{slrcCustomerID} as the ID and \texttt{CustomerID} for both the \texttt{Child Column} and \texttt{Parent Column}.

At this point standard Data Table Columns are needed under the SubDataTable element. They can be copied from Data Tables in other Labs but the Sort elements will have to be removed.
8. Add Data Table Columns to the SubDataTable.
   - Select all the Data Table Columns from the dtOrders Data Table in Lab2_Grouped_Flat_Table. Use Ctrl-click to select multiple Data Table Columns.
   - Right-click and choose Copy.
   - Select the SubDataTable element sdtOrderSummary in Lab3_Drilldown_SubDataLayers.
   - Right-click and choose Paste.
   - Delete the Sort elements and Hide Duplicate elements from all the copied Data Table Columns.

   Preview the report or Run In Browser. Notice you can now click on any row of the Customer Total Column. When you click on the total you should see orders from that customer. The Customer ID should match up. The third level can now be added.

9. Remove the Customer ID from the second level.
   - Select colCustomerID from the SubData Table element sdtOrderSummary and press Del or right-click and choose Delete.

10. Add a More Info Row to the SubData Table element.
    - Select sdtOrderSummary and add a More Info Row element from the Element Toolbox.
    - Enter the attribute values as follows.

    ![Element - MoreInfoRow](image)

    - Add a Subdata Table element from the Element Toolbox to mirOrderDetails.
    - Enter an ID of sdtOrderDetails.
• Add a Subdata Layer element under sdtOrderDetails from the Element Toolbox with an ID of sdlOrderDetails.

• Add a conventional DataLayer.SQL to sdlOrderDetails. Enter an ID of dlOrderDetails.

• Enter connNorthwind for the Connection ID.

• Add the following SQL to the Source attribute of dlOrderDetails.

```
```

Note: There must be a common column between the parent and child data layers. These columns can be either in the select statement of the SQL query or added data layer columns.

11. Link the two SubData Tables.

• Select sdlOrderDetails. Add a Subdata Layer Relation Column with an ID of slrcOrderID.

• Enter OrderID for the Child and Parent Columns.

12. Add Data Table Columns.

• Select sdtOrderDetails and manually add a Data Table Column for ProductName. Enter an ID of colProductName.

• Add a Label element to colProductName with and ID of lblProductName and enter @Data.ProductName~ in the Caption attribute.

• Repeat the two steps above for these columns:

<table>
<thead>
<tr>
<th>Data Table Column ID</th>
<th>Label ID under the Column</th>
<th>Contents of the Label’s Caption Attribute</th>
</tr>
</thead>
</table>
### Logi Training Manual V14.3.1

<table>
<thead>
<tr>
<th>colCategoryName</th>
<th>lblCategoryName</th>
<th>@Data.CategoryName~</th>
</tr>
</thead>
<tbody>
<tr>
<td>colUnitPrice</td>
<td>lblUnitPrice</td>
<td>@Data.UnitPrice~</td>
</tr>
<tr>
<td>colQuantity</td>
<td>lblQuantity</td>
<td>@Data.Quantity~</td>
</tr>
<tr>
<td>colDiscount</td>
<td>lblDiscount</td>
<td>@Data.Discount~</td>
</tr>
<tr>
<td>colOrderTotal</td>
<td>lblOrderTotal</td>
<td>@Data.TotalPrice~</td>
</tr>
</tbody>
</table>

- Enter in **ThemeAlignRight** for the **Class** attribute of all columns except `colProductName` and `colCategoryName`.

- Enter **Currency** format for `lblUnitPrice` and `lblOrderTotal`. Enter **Percent** format for `lblDiscount`.

Now that the Final Sub Data Table has been constructed, a link must be added to the second level to expose the third.

13. **Add an Action. ShowElement** element to the Second Level

   - Select the **Label** element `lblOrderTotal`. Add an **Action.ShowElement** with an **ID** of `actShowOrderDetail`.
   
   - Enter **mirOrderDetails** for the **Element ID** attribute. If it is not in the drop down menu then you will have to type it in exactly. You can cut and paste from this document.

14. **Verify your report**:

   - The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.

   - Click the Save All Opened Files button in the Toolbar to save your report definition.

   - Right-click on **Lab3_Drilldown_SubDataLayers** in the Application Panel and choose **Run In Browser**.

   - The final report should appear as shown in the Report Output section at the beginning of this lab.
Charts

Charting is another Business Intelligence mainstay. We’ll be working with Pie charts and Bar Charts primarily but the elements used for other chart types follow the same process and many of the same attributes.

The look and feel of all charts is dependent on the Theme chosen. The theme that the charts in the Report Output section of every lab is based on the Professional Blue Theme.

Lab 1 – Pie Chart

Objectives:

- Create a pie chart of Sales by Category using:
  - An animated Pie Chart
  - A Group Filter
  - A Group Aggregate Column

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Chart Diagram" /></td>
<td><img src="image" alt="Pie Chart" /></td>
</tr>
</tbody>
</table>

![Sales by Category Chart](image)
Directions:

1. In the Application Panel, under the Reports folder, create a new folder and rename it to Charts. All the chart exercises will reside here.
   - Right-click on the Reports folder. Select Add, then New Folder from the popup menu.
   - Rename the folder to Charts.

2. Create a new report definition called Lab1_PieChart:
   - In the Application Panel, right-click on the folder Charts. Select Add, then New Definition from the popup menu.
   - Rename the new definition to Lab1_PieChart.

3. Under the Body element of the report, add an Animated Chart.Pie element:
   - In the Workspace Panel, click on the Body element.
   - In the Element Toolbox, expand the Charts, Gauges and GIS Maps folder (if necessary) and double-click the Animated Chart.Pie element.
• In the Attributes Panel, set the **Animated Chart.Pie** element attribute values as follows:
  
  o Set the **ID** attribute to **pieSalesByCategory**. (We’ll set the remaining attribute values after setting up the data layer.)

4. Add a **Data Layer** element to access the **Northwind** database:

  • In the Workspace Panel, click on the **Animated Chart.Pie** element **pieSalesByCategory**.
  
  • Add a **DataLayer.SQL** element from the Element Toolbox.
  
  • In the Attributes Panel, set the **Data Layer** element attribute values as follows:
    
    o Set the **ID** attribute to **dlSalesByCategory**
    
    o Set the **Connection ID** attribute to **connNorthwind**
    
    o Select the **Source** attribute and change to the following SQL.

5. Add a Group Filter element to the Data Layer element **dlSalesByCategory**:

  • In the Workspace Panel, select the **Data Layer** element **dlSalesByCategory**.
  
  • Add a **Group Filter** element from the Element Toolbox.
In the Attributes Panel, set the attribute values for the Group Filter element as follows:

- Set the ID attribute to `grpCategoryName`.
- Click the Data Type attribute. Select Text from the drop-down list.
- Set the Group Column attribute to `CategoryName`.

6. Add a Group Aggregate Column element to the Group Filter element `grpCategoryName`:

- In the Workspace Panel, click the Group Filter element `grpCategoryName`.
- Add a Group Aggregate Column from the Element Toolbox.
- In the Attributes Panel, set the attribute values for the Group Aggregate Column element as follows:
7. Configure the pie chart attribute values:

- In the Workspace Panel, click on the **Animated Chart.Pie** element **pieSalesByCategory**:

- In the Attributes Panel, set the attribute values for the **Animated Chart.Pie** element **pieSalesByCategory** as follows:

![Pie Chart Attribute Values](image)

8. Verify your report:

- Click the Save All Opened Files button in the Toolbar to save your report definition.

- The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.

- Right-click on **Lab1_PieChart** in the Application Panel and choose **Run In Browser**.

The final report should appear as shown in the Report Output section at the beginning of this lab.
There are times when one will want to emphasize a pie slice more than the others. Often business rules will determine which slices will be emphasized. This lab we will explode the wedges that fall above a certain threshold.

**Lab 2 – Exploded Wedges**

**Objectives:**

- Create an animated pie chart of Sales by Category data
- Show Exploded Wedges for categories with large sales figures.

---

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Image of Report Definition" /></td>
<td><img src="image" alt="Image of Final Report Output" /></td>
</tr>
</tbody>
</table>

**Directions:**

1. Copy the report definition for **Lab1_PieChart** and rename it **Lab2_PieChart_ExplodedWedges**:

   - In the Application Panel’s **Reports** folder, right-click on the report definition **Lab1_PieChart** inside the **Charts** subfolder, and select **Copy** from the popup menu.
• Right-click on the Charts subfolder, and select Paste from the popup menu.
• Rename the copy to Lab2_PieChart_ExplodedWedges.

In order to utilize built-in chart features around exploded wedges, you must first create a calculation such that if the sales by category is above a certain threshold, a 1 will appear and 0 if not. This calculated column (which will be a logi element) will be referenced in the Exploded Wedges Column attribute.

2. Add a Calculated Column element to the Data Layer element dlSalesByCategory:

• In the Workspace Panel, click on the Data Layer element dlSalesByCategory.
• Add a Calculated Column element from the Element Toolbox.
• In the Attributes Panel, set the Calculated Column element attribute values as follows:
  o Set the ID attribute to IsLargeSales.
  o Set the Formula attribute to IIF(@Data.SalesPerCategory~ > 100000, “1”, “0”)

This will set the threshold to 100,000 dollars in sales.
Careful not to use commas when entering 100000 as this will throw off the IIF function.

3. Set the Exploded Wedges Column attribute of the Animated Chart.Pie element pieSalesByCategory:
   - In the Workspace Panel, click the Animated Chart.Pie element pieSalesByCategory.
   - In the Attributes Panel, set the Exploded Wedges Column attribute to IsLargeSales.

4. Verify your report:
   - Click the Save All Opened Files button in the Toolbar to save your report definition.
• The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.

• Right-click on Lab2_PieChart_ExplodedWedge in the Application Panel and choose Run In Browser.

• The final report should appear as shown in the Report Output section at the beginning of this lab.
There are times when there are too many X-Axis values to show. Using an element called a Reference Filter, we can set a threshold not unlike the exploded wedges column. If an X-Axis entry has a Y-Axis value below a certain threshold it will be placed in another column that is normally named “all others” although the name is configurable.

You can create thresholds based on percentage, a value or the top “N” number of rows.

Lab 3 – Pie Chart – All Others Wedge

Objectives:

- Create an non-animated Pie Chart.
- Add a Relevance Filter to create an ‘All Others’ wedge.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Charts_Lab3_PieChart_AllOthersWedge" alt="Diagram" /></td>
<td><img src="Charts_Lab3_PieChart_AllOthersWedge" alt="Pie Chart" /></td>
</tr>
</tbody>
</table>

- All Others: 20.00%
- Confections: 20.00%
- Beverages: 20.00%
- Dairy Products: 20.00%
- Meat/Poultry: 20.00%
Directions:

1. Add a new definition name it **Lab3_PieChart_AllOthersWedge**.
   - Right-click on the **Charts** subfolder, and select Add then New Definition from the popup menu.
   - Rename the newReport to **Lab3_PieChart_AllOthersWedge**.

2. Add a Chart.Pie from the Element Toolbox.
   - Open the **Lab1_PieChart** and copy the DataLayer **dlSalesByCategory**.
   - Close **Lab1_PieChart**.
   - Select **Lab3_PieChart_AllOthersWedge**.
   - Add a **Chart.Pie** from the Element Toolbox and give it an ID of **chtSalesByCategory**.
   - Paste the **DataLayer** from **Lab1_PieChart** under **chtSalesCategory**.
   - Enter in the attribute values as shown:
<table>
<thead>
<tr>
<th>Required Attributes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Column X-axis</td>
<td>CategorySales</td>
</tr>
<tr>
<td>Height</td>
<td>500</td>
</tr>
<tr>
<td>Width</td>
<td>500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optional Attributes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3-D Angle</td>
<td></td>
</tr>
<tr>
<td>3-Dimensional</td>
<td></td>
</tr>
<tr>
<td>Alternate Text</td>
<td></td>
</tr>
<tr>
<td>Background Color</td>
<td></td>
</tr>
<tr>
<td>Chart Texture</td>
<td></td>
</tr>
<tr>
<td>Chart Title</td>
<td></td>
</tr>
<tr>
<td>Colors</td>
<td></td>
</tr>
<tr>
<td>Doughnut Hole Radius</td>
<td></td>
</tr>
<tr>
<td>Exploded Wedges Column</td>
<td></td>
</tr>
<tr>
<td>External Layout Label Color</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>chtSalesByCategory</td>
</tr>
<tr>
<td>Label Column Data Type</td>
<td></td>
</tr>
<tr>
<td>Label Column X-axis</td>
<td>CategoryName</td>
</tr>
<tr>
<td>Label Layout</td>
<td></td>
</tr>
<tr>
<td>Label Pointer Color</td>
<td></td>
</tr>
<tr>
<td>Label Rounded Border</td>
<td></td>
</tr>
<tr>
<td>Label Shading</td>
<td></td>
</tr>
<tr>
<td>Left Border</td>
<td></td>
</tr>
<tr>
<td>Maximum Label Length</td>
<td></td>
</tr>
<tr>
<td>Outer Border Color</td>
<td></td>
</tr>
<tr>
<td>Outside Border Rounding</td>
<td></td>
</tr>
<tr>
<td>Pie Chart Type</td>
<td></td>
</tr>
<tr>
<td>Pie Color Data Column</td>
<td></td>
</tr>
<tr>
<td>Radius</td>
<td></td>
</tr>
<tr>
<td>Security Right ID</td>
<td></td>
</tr>
<tr>
<td>Show Data Values</td>
<td>True</td>
</tr>
<tr>
<td>Show Wait Icon</td>
<td></td>
</tr>
<tr>
<td>Tooltip</td>
<td></td>
</tr>
<tr>
<td>Top Border</td>
<td></td>
</tr>
<tr>
<td>Transparency</td>
<td></td>
</tr>
<tr>
<td>Wallpaper Image</td>
<td></td>
</tr>
<tr>
<td>Wedge Border Color</td>
<td></td>
</tr>
<tr>
<td>Wedge Border Thickness</td>
<td></td>
</tr>
</tbody>
</table>
3. Add a Relevance Filter element to the Data Layer element

   - In the Workspace Panel, click on the **Data Layer** element *dlSalesByCategory*.
     - Add a **Relevance Filter** from the Element Toolbox.

   ![Relevance Filter](image)

   - In the Attributes Panel, set the **Relevance Filter** element attribute values as follows:

   ![Relevance Filter Attributes](image)

5. Verify your report:

   - Click the Save All Opened Files button in the Toolbar to save your report definition.
   - The report definition elements in the Workspace Panel should appear as shown in Report Definition section at the beginning of this lab.
   - Right-click on **Lab3_PieChart_AllOthersWedge** in the Application Panel and choose Run In Browser.
   - The final report should appear as shown in the Report Output section at the beginning of this lab.
Lab 4 – Drill Through to Data Table

This lab has evolved over time from a two definition lab to a one definition lab. The chart and its companion drilldown will be in the same definition side by side. The drilldown will be from a Category aggregation chart to a data table of products for the category selected. This can be greatly extended to show Chart to Chart among other visualizations.

Objectives:

- Create a pie chart of Sales by Category data
- Drill-Through to related Product Data when clicking on a pie slice.

---

Final Report Output

![Diagram of pie chart and data table]

<table>
<thead>
<tr>
<th>Products for Confections</th>
<th>CompanyName</th>
<th>Quantity/Period</th>
<th>UnitPrice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parkina</td>
<td>Parkina Ltd.</td>
<td>32 - 500 g boxes</td>
<td>17.4500</td>
</tr>
<tr>
<td>Teatime Chocolate Biscuits</td>
<td>Specialty Biscuits, Ltd.</td>
<td>10 boxes x 12 pieces</td>
<td>9.2000</td>
</tr>
<tr>
<td>Sir Rodney's Homeade</td>
<td>Specialty Biscuits, Ltd.</td>
<td>300 g tins</td>
<td>01.0000</td>
</tr>
<tr>
<td>Sir Rodney's Stones</td>
<td>Specialty Biscuits, Ltd.</td>
<td>24 bags x 4 pieces</td>
<td>10.0000</td>
</tr>
<tr>
<td>NACHA Nut-Rouge-Creme</td>
<td>Hei Süsswaren GmbH &amp; Co. KG</td>
<td>20 - 450 g glasses</td>
<td>14.0000</td>
</tr>
<tr>
<td>GmbhFabianBakery</td>
<td>Hei Süsswaren GmbH &amp; Co. KG</td>
<td>100 - 250 g bags</td>
<td>31.2300</td>
</tr>
<tr>
<td>Sembol Schokolade</td>
<td>Hei Süsswaren GmbH &amp; Co. KG</td>
<td>100 - 100 g pieces</td>
<td>43.8000</td>
</tr>
<tr>
<td>Zweiem Fisch</td>
<td>Zweiem Speriplanek</td>
<td>10 - 4 oz tins</td>
<td>5.9000</td>
</tr>
<tr>
<td>Chocolate</td>
<td>Zweiem Speriplanek</td>
<td>10 slugs</td>
<td>12.7500</td>
</tr>
<tr>
<td>Manduks</td>
<td>Kanri Oy</td>
<td>24 - 10 g pkgs</td>
<td>20.0000</td>
</tr>
<tr>
<td>ValhORIES sukses</td>
<td>Kanri Oy</td>
<td>12 - 100 g bars</td>
<td>10.2500</td>
</tr>
<tr>
<td>Tiret au sucrose</td>
<td>Tiret d'elabes</td>
<td>45 pieces</td>
<td>49.3000</td>
</tr>
<tr>
<td>Scottish Longbread</td>
<td>Specialty Biscuits, Ltd.</td>
<td>10 boxes x 8 pieces</td>
<td>12.0500</td>
</tr>
</tbody>
</table>
Definition
Directions:

1. Copy the report definition for `Lab3_PieChart_AllOthersWedge` and rename it `Lab4_PieChart_DrillThrough`:
   - In the Application Panel’s Reports folder, right-click on the report definition `Lab2_PieChart_ExplodedWedges` inside the Charts subfolder, and select Copy from the popup menu.
   - Right-click on the Charts subfolder, and select Paste from the popup menu.
   - Rename the copy to `Lab4A_PieChart_DrillThrough`:

Normally when you put two visualizations in the same definition they will be arranged vertically when rendered. Here we wish to put the chart on the left portion of the page and the data table on the right portion of the page. We will achieve this with a `Rows` element that will allow the construction of an HTML table with rows and columns created at design time.

2. Create a Rows Element.
   - Select the `Body` element and add a `Rows` element from the Element Toolbox.
   - Add a `Row` element from the Element Toolbox.
• Add a **Column Cell** element to the **Row** element and give it an **ID** of **colChart**.

![Column Cell](image)

• Add a second **Column Cell** element to the **Row** element and give it an **ID** of **colTable**.

• The definition should look like the following at this point.

![Diagram](image)

3. Cut the chart and paste it in **colChart**.

   • Select **chtSalesByCategory** and right-click then choose cut.
   
   • Select **colChart** then right-click **Paste**.

4. Create a **DataTable** to drilldown to.

   • Select the **colTable** element.
   
   • Right-click and select **Element Wizards** then **Add a Data Table**.
Follow the prompts as follows:

- Click **Next** on the initial prompt then choose **SQL** for the **DataLayer Type**.

- Accept **connNorthwind** as the **Connection ID**.

- Copy the following text and paste it in the SQL Query portion of the wizard:

  ```sql
  Select Products.ProductName,
          Suppliers.CompanyName,
          Products.QuantityPerUnit,
          Products.UnitPrice
  From Products
  Inner Join Suppliers On Suppliers.SupplierID = Products.SupplierID
  ```

- Click **Next** then Accept all the columns shown.
- Accept all the default prompts by clicking **Next** for the **Interactive Paging**.

5. Preview the Report.
   - The chart and table should appear side by side.

6. Make the chart clickable.
   - Close the browser or click on the definition button if you were previewing.
   - Select **chtSalesByCategory**.
   - Add an **Action.Report** element from the Element Toolbox. Give it an **ID** of **actChooseCategory**.
   - Add a **Target.Report** from the Element Toolbox. Give it an **ID** of **tgtChooseCategory**.
   - Choose **Chart.Lab4_PieChart_Drillthrough** in the **Report Definition File** attribute.

Since the chart is not a User Input we cannot assume that the click we make on a pie slice will be added to @Request so we must add a LinkParameters element to define what will be sent to the DataTable holding our products.

7. Add link parameters to the action
   - Select **actChooseCategory**.
   - Add a **LinkParameters** element.
   - Enter in this Parameter and press enter:
     - Enter the **CategoryName** in the same fashion.
     - Enter @Chart.CategoryID~ for the **CategoryID** Link Parameter.
     - Enter @Chart.CategoryName~ for the **CategoryName** LinkParameter.
   - The Link Parameter element should look like this:
This will take the CategoryID and CategoryName from the DataLayer supporting the chart and pass it into the respective @Request tokens. We use @Chart since we are not dealing with a DataTable but a chart and we are also clicking on the chart itself.

8. Change the DataTable of Products to recognize the Chart Selection.
   - Select the DataTable under colTable.
   - Change the ID to dtProducts.
   - Select the DataLayer under dtProducts.
   - Change the ID to dlProducts.
   - Change the Source attribute to reflect the following query:

     ```sql
     Select Products.ProductName, Suppliers.CompanyName, Products.QuantityPerUnit, Products.UnitPrice
     From Products
     Inner Join Suppliers On Suppliers.SupplierID = Products.SupplierID
     Where Products.CategoryID = '@Request.CategoryID~'
     ```

   - Click on a Slice and the list of products should change to reflect the click. You can judge by the nature of the choice whether each product belongs in that category.
   - Close the browser or click on Definition.
   - Select dtProducts and enter **Products for @Request.CategoryName~** in the Table Caption. This will reinforce the choice if the report is printed (otherwise the consumer of the report may not know).
This makes for an unprofessional looking rendering. Let’s show a label that will give directions to the user on the initial rendering. Then when the user picks a slice in the pie chart the label goes away in favor of the Products table. To do this we will use two Division elements which are containers for elements that can be shown or not based on a condition.

10. Add Two divisions to colTable.

- Select the colTable element.

- Add a Division element from the toolbox and give it an ID of divTable.

- Add a second Division element and give it an ID of divInstructions.

- Select the DataTable dtProducts and choose right-click cut.

- Select divTable and choose right-click paste.

- Select divInstructions and add a Label element from the toolbox. Give it an id of lblInstructions.

- Type Click on a Pie Slice to view the corresponding Product List in the Caption of lblInstructions.

- Choose ThemeTextLarge in the Class attribute of lblInstructions.

11. Add conditions to the Divisions

- Select divTable

- Enter @Request.CategoryID > 0 in the Condition attribute.
- Enter @Request.CategoryID = 0 in the Condition attribute.
- Since Category ID will have no value on the initial rendering, we must provide the initial value in the form of a Default Request Parameters element.

12. Handle the initial value of Category ID

- Select the top element of the report and add a Default Request Parameter element from the toolbox.

- Enter in CategoryID in the Parameter section and press enter.
- Enter 0 in for the value of that parameter.
- Press F7 enough times so that the Default Request Parameter moves to the top of the definition.

13. Verify your reports:

- Click on the **Save All Opened Files** button in the toolbar to save all changes to both reports.

- Open the data table report in the Workspace Panel. The data table report definition elements in the Workspace Panel should appear as in the Report Output for the Drillthrough Table at the beginning of this lab.

- Open the pie chart report in the Workspace Panel. The pie chart report definition elements in the Workspace Panel should appear as shown in the Report Output Pie Chart at the beginning of this lab.
The Bar Chart from an attribute standpoint, is very similar to the Pie Chart. While the concept of a Y-Axis is sometimes hard to assimilate, it is much more at home with a Bar Chart. The chart is actually known as a Chart.XY element. This element has several flavors including Spline, Line and Area.

Lab 5 – Bar Chart

Objectives:

- Create a bar chart of Sales by Category data.
- Formatted X-axis labels, Y-axis labels, and tooltips.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Charts.Lab5_BarChart]</td>
<td>![Bar Chart Output]</td>
</tr>
</tbody>
</table>

- Clarity
- Report Header
- Body
  - barSalesByCategory
  - dSalesByCategory
    - grpCategoryName
      - SalesPerCategory
- Label Font
- Format Data
- Report Footer
Directions:

1. Create a new report definition called **Lab5_BarChart**.
   - In the Application Panel, right-click on the **Charts** folder. Select **Add**, then **New Definition** from the popup menu.
   - Rename the new definition to **Lab5_BarChart**.

2. Under the **Body** element of the report, add a **Chart.XY** element.
   - In the Workspace Panel, click on the **Body** element.
   - In the Element Toolbox, expand the **Charts, Gauges and GIS Maps** folder (if necessary) and double-click the **Chart.XY** element.
   - In the Attributes Panel, set the **Chart.XY** element attribute values as follows.
     - Set the **ID** attribute to **barSalesByCategory**. (The remaining attribute values will be assigned after setting up the Data Layer.)
3. Copy the **Data Layer** element `dlSalesByCategory` from the pie chart in report definition **Lab1_PieChart**:

   - In the Application Panel, double-click report definition **Lab1_PieChart** to open it in the Workspace Panel.
   - Right-click the **Data Layer** element `dlSalesByCategory` and select **Copy** from the popup menu.
   - At the top of the Workspace Panel, click the **Lab5_BarChart** tab to make it the active report.
   - Right-click the **Chart.XY** element `barSalesByCategory` and select **Paste** from the popup menu.
• The Data Layer element `dlSalesByCategory` will appear under the Chart.XY element `barSalesByCategory`.

![Diagram](image)

4. **Add a Label Font element under the Chart.XY element barSalesByCategory to control the X-axis labels.**

  • In the Workspace Panel, select the Chart.XY element `barSalesByCategory`.
  
  • Add a Label Font element from the Element Toolbox.

  ![Diagram](image)

  • In the Attributes Panel, set the Label Font element attribute values as follows:
    • Set the Font Angle attribute to 45 degrees.

5. **Add a Format Data element under the Chart.XY element barSalesByCategory to control the Y-axis format.**

  • In the Workspace Panel, select the Chart.XY element `barSalesByCategory`.
  
  • Add a Format Data element from the Element Toolbox.

  ![Diagram](image)

  • In the Attributes Panel, set the Label Font element attribute values as follows:
    • Set the Decimal Points attribute to 0.
    • Set the Chart Text Format attribute to `${value}`.
6. Verify your report:

- Click on the Save All Opened Files button in the toolbar to save all changes to both reports.
- Open the Lab5_BarChart definition in the Workspace Panel and Run In Browser. The definition should appear as in the definition section at the beginning of this Lab.

7. Choose the Lab5_BarChart definition in the Application Panel. Right-click and choose Run In Browser. The Report Output should match the Output shown at the beginning of this Lab.
Lab 6 – Extra Data Column

Objectives:

- Modify Charts Lab 5 to create a clustered bar chart of Sales by Category and Projected Sales
- Extra Data Column element.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Chart Diagram" /></td>
<td><img src="image" alt="Bar Chart" /></td>
</tr>
</tbody>
</table>

- **Chart Diagram**: Shows the structure of the report definition with elements like `chart`, `style`, `report header`, and `body`. It includes data elements such as `SalesPerCategory`, `ProjectedSalesPerCategory`, and `Report Footer`.

- **Bar Chart**: Represents the final report output with sales data categorized by category and projected sales, showing a comparison of sales across different product categories.
Directions:

1. Copy the report definition for Lab5_BarChart and rename it Lab6_BarChart_ExtraDataColumn:
   - In the Application Panel’s Reports folder, right-click on the report definition Lab5_BarChart inside the Charts subfolder, and select Copy from the popup menu.
   - Right-click on the Charts subfolder, and select Paste from the popup menu.
   - Rename the copy to Lab6_BarChart_ExtraDataColumn.

2. Add a Calculated Column element to the Data Layer element dlSalesByCategory:
   - In the Workspace Panel, click on the Data Layer element dlSalesByCategory.
   - Add a Calculated Column element from the Element Toolbox.
• In the Attributes Panel, set the **Calculated Column** element attribute values as follows:

![Calculated Column Attributes Panel](image)

3. Add an **Extra Data Column** element to the **Chart.XY** element barSalesByCountry.

• In the Workspace Panel, click on the **Chart.XY** element **barSalesByCountry**.

• Add an **Extra Data Column** element from the Element Toolbox.

![Element Toolbox](image)

• In the Attributes Panel, set the **Extra Data Column** element attribute values as follows:

![Extra Data Column Attributes Panel](image)
4. Modify the attribute values for the Chart.XY element barSalesByCategory:
   - In the Workspace Panel, click on the Chart.XY element barSalesByCategory.
   - In the Attributes Panel, set the attribute values for the Chart.XY element barSalesByCategory as follows:

   ![Attribute Values](image)

5. Verify your report:
   - Click on the Save All Opened Files button in the toolbar to save all changes to both reports.
   - Open the Lab6_BarChart_ExtraDataColumn definition in the Workspace Panel. The definition should appear as in the definition section at the beginning of this Lab.
   - Choose the Lab6_BarChart_ExtraDataColumn in the Application Panel. Right-click and choose Run In Browser. The Report Output should match the Output shown at the beginning of this Lab.
This lab is very similar to Lab 6 only the second value is a different chart type. Further explorization of this topic would uncover secondary data axis and data scale elements.

Lab 7 – Extra Grid Layer

Objectives:

- Modify Charts Lab 6 to create a combination bar/line chart of Sales by Category data
- Extra Grid Layer element.
Directions:

1. Copy the report definition for `Lab6_BarChart_ExtraDataColumn` and rename it `Lab7_Barchart_ExtraGridLayer`:
   - In the Application Panel’s `Reports` folder, right-click on the report definition `Lab6_BarChart_ExtraDataColumn` inside the `Charts` subfolder, and select `Copy` from the popup menu.
   - Right-click on the `Charts` subfolder, and select `Paste` from the popup menu.
   - Rename the new definition to `Lab7_Barchart_ExtraGridLayer`.
   - Delete the `Calculated Column` element `ProjSalesPerCategory`:
     - In the Workspace Panel, right-click on the `Calculated Column` element `ProjSalesPerCategory` and select `Delete` from the popup menu.

2. Delete the `Extra Data Column` element `edcProjSalesPerCategory`:
   - In the Workspace Panel, right-click on the `Extra Data Column` element `edcProjSalesPerCategory` select `Delete` from the popup menu.

3. Add an `Extra Grid Layer` element to the `Chart.XY` element `barSalesByCategory`.
   - In the Workspace Panel, click on the `Chart.XY` element `barSalesByCategory`.
   - Add an `Extra Grid Layer` element from the Element Toolbox. This is the definition to this point:

   ![Image of Workspace Panel showing Extra Grid Layer element]

   An `Extra Grid Layer` element icon will appear in the Workspace Panel under the `Chart.XY` element `barSalesByCategory`. If necessary, click on the new `Extra Grid Layer` element to select it.
• In the Attributes Panel, set the Extra Grid Layer element attribute values as follows.

![Attributes Panel]

4. Modify the attribute values for the Chart.XY element barSalesByCategory:

• In the Workspace Panel, select the Chart.XY element barSalesByCategory.

• In the Attributes Panel, set the attribute values for the Chart.XY element barSalesByCategory as follows:
  - Set the Transparency attribute to 10.

5. Verify your report:

• Click on the Save All Opened Files button in the toolbar to save all changes to both reports.

• Open the Lab7_BarChart_EXTRAGridLayer definition in the Workspace Panel. The definition should appear as in the definition section at the beginning of this Lab.

• Choose the Lab7_BarChart_EXTRAGridLayer in the Application Panel. Right-click and choose Run In Browser. The Report Output should match the Output shown at the beginning of this Lab.
Processes

Processes extend Logi Info’s capabilities well beyond a reporting tool. Using special Process definitions containing special procedural Task elements, we can construct a very programmatic functionality while retaining the ease of elemental development. A Task element is made up of Procedure elements that have the ability to do branching, error trapping and looping among other programming constructs.

Lab 1 – Email PDF

Objectives:

- Create a single task within a single Process definition to send an email with a PDF attachment.
- Add a link to the same PDF inside the body of the email.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="emailpdf.png" alt="EmailPDF" /></td>
<td>No Report Output. An email should be received containing a PDF of the report used in the export.</td>
</tr>
</tbody>
</table>

- [EmailPDF](emailpdf.png)
  - `tskEmailPDF`
    - `procSetPDFName`
      - `OS Session Parameters`
    - `procExportPDF`
      - `tgtExportPDF`
    - `procSendPDF`
      - `attachOrdersReport`
    - `Response.Report`
      - `DataTables.Lab1_DataTable`
Directions:

1. Add a new Process definition to the Processes folder.
   - Select the red Processes Folder in the Application Panel.
   - Right-click on the folder and choose Add then New Definition.
   - Rename the newProcess to Email PDF.

2. Add a Task element
   - Add a Task element from the Element Toolbox.
   - Enter an ID of tskEmailPDF for the Task.

3. At this point you enter Procedure elements that provide encapsulated functionality around a specific purpose.
   - Some of the things procedures can do:

<table>
<thead>
<tr>
<th>Run Stored Procedures</th>
<th>Run SQL Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide Looping Functions</td>
<td>Export to CSV, Excel, Word, PDF or XML</td>
</tr>
<tr>
<td>Create Emails with Attachments</td>
<td>Execute Web Service Calls</td>
</tr>
<tr>
<td>Run Plug-Ins</td>
<td>Send Tweets and Text Messages</td>
</tr>
</tbody>
</table>
   | Create folders and Copy Files | Error Handling and Control Branching

   - These functions are usually run behind the scenes and are generally attached to a button or link in a report or other definition (i.e. criterion page).
   - These functions can also be used in the scheduling server provided with Logi Info.

4. Planning the Task.
   - The task at hand is to email a specific recipient a PDF of a report. Lab1_Basic_DataTable is sufficient as a source for the report. The report is not as important as the method around the construction of a task.
   - Planning a task is no different than working out the logic of a program or function in your favorite language. This is a list of things we would like to get done in this task.
o Derive a file name for the exported PDF.

o Export Lab1_Basic_DataTable with the derived file name and in the newly created folder.

o Create an email

o Attach the exported file to the email and send

- You will need to create a folder manually to hold the PDF. This folder is called ExportedFiles and goes under the root of your application.

- While there you should create the other folders that will be needed...Dashboards and UserBookmarks.

- You may need to address security concerns about your new folders. Full control needs to be given to IIS in order to write data to these folders. IIS will be doing the writing of the files.

- We will also need to add a connection to our application to enable emailing.

5. Add the Procedures

- Derive a file name for the exported PDF.

  - Make sure that tskEmailPDF is selected.

  - Add a Procedure.SetSessionVars element from the Element Toolbox

  - Give the new procedure an ID of procSetPDFName.

  - Add a Session Parameters element from the Element Toolbox. The attribute window resembles the same window used when creating Default Request Parameters or Link Parameters.
• Enter in PDF_Filename as shown and press Enter or click on the Green Plus Sign.

The Contents of this variable must be completely unique. To do this a Logi Function called @Function.GUID~ will be used to generate this file name. A GUID is a Global Unique IDentifier and is unique across all computers worldwide.

• Enter the contents of the new parameter PDF_Filename as shown below.

• Export Lab1_Basic_DataTable with the derived file name and in the newly created folder.

  • Re-Select tskEmailPDF and add a Procedure.ExportPDF element from the Element Toolbox.

  • Enter in an ID of procExportPDF.

  • Enter the Filename attribute as shown below. Notice the Session variable defined above is used here along with the folder name.

• Add a Target.PFD Element to procExportPDF from the Element Toolbox.

  • Add an ID of tgtExportPDF and set the Report Definition File attribute to DataTables.Lab1_Basic_DataTable.

• Create a connection element for the SMTP Server.

  • Emails are sent through the use of a SMTP server located in every organization. Generally it is a server name or an IP address.

  • Open the _Settings Definition in the Application Panel by double-clicking.
• Select the **Connections** Element and add a **Connection.SMTP**.

![Diagram of Connections Element]

• Update the attribute values as shown. If you don’t know your SMTP server name then contact your systems administrator.

![Attribute Values Diagram]

  o Save and close the **_Settings** definition.

• Create an Email.
  
  • Re-Select tskEmailPDF and add a **Procedure.Send Mail** element from the Element Toolbox.
  
  • Enter in an ID of **procSendPDF** for the new procedure.
  
  • Enter the other attribute values as shown:
The From Email Address attribute can be a properly formatted email address but need not be authentic. The Reply to Email Address attribute should be if entered.

- Attach the exported file to the email and send.
  - Select procSendPDF.
  - Add an Attachment element from the Element Toolbox with and ID of attchOrdersReport.
- Enter in the attribute values as follows.

6. Each Task must have some kind of Response element.

- Select tskEmailPDF and add a Response.Report element from the Element Toolbox.
- Add a Target.Report element from the Element Toolbox. Enter in DataTables.Lab1_Basic_DataTable for the Report Definition File attribute.
7. Verify your definition:

- Click on the **Save All Opened Files** button in the toolbar to save all changes to both reports.
- Since this is a Process file, testing it is not like a preview as it is in the standard report definitions. In the lower right corner is a new window that has a select list of the tasks in this process file:

![Run Task Window](image)

- At the bottom of the workspace panel is a Run button. If there are several tasks in the same process file you will need to select the task from the dropdown list as shown.

![Workspace Panel](image)

- Click on the Run Button. IF there are no errors you should see the report listed in the Target.Report element in the preview window.

Now that we have a task that has been created, we can schedule it. We refer to a task that has been scheduled as a Job. In this topic we will construct a set of tasks that are designed to maintain these jobs.

Next we will create two definitions in a new folder under Reports (therefore they will be rendered). The first will allow us to create new jobs and the second the ability to maintain them. The tasks that we will construct in the first lab are run from buttons in the two definitions.
Scheduling

Now that we have a task that has been created, we can schedule it. We refer to a task that has been scheduled as a Job. In this topic we will construct a set of tasks that are designed to maintain these jobs.

Next we will create two definitions in a new folder under Reports (therefore they will be rendered). The first will allow us to create new jobs and the second the ability to maintain them. The tasks that we will construct in the first lab are run from buttons in the two definitions.

Lab1 – Scheduler Tasks

Objectives:

- Build the necessary tasks in a new process file to manage the adding, updating and removal of jobs.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Diagram.png" alt="Diagram" /></td>
<td>No Output for this definition. All tasks in a process file.</td>
</tr>
</tbody>
</table>
Directions:

1. Add a new process file to the application.
   
   • Click on the red Process folder then right-click and choose Add -> New Definition.
   
   • Name the new process file **SchedulerTasks**

2. Add a Task to create new jobs.

   • Select the uppermost element in the definition

   • Add a **Task** element from the Element Toolbox. Enter **tskCreateJob** as the ID.

   • Add **Procedure.SchedulerCreateTask** procedure from the Element Toolbox. Look for the purple Task Scheduler folder.

   • Enter the attribute values as follows:

   ![Element - Procedure.SchedulerCreateTask]

     • Substitute your URL with the URL shown above.

     • Add a **Target.Process** element to prcCreateJob from the Element Toolbox.

   ![Element - Target.Process]
• Add a Response.Report element to tskCreateJob with and ID of rspMyScheduledJobCreate.

• Add a Target.Report element to rspMyScheduledJobCreate. Enter an ID of tgtMyScheduleTasksCreate and a Report Definition File of Schedule.MyScheduledJobs.

Note that we are hard coding the Task ID in the Target.Process (tgtEmailTasks). The task ID can be tokenized with an @Request token and fed from the button or definition that calls tskCreateJob.

• Here is the resulting Task:

3. Add a Task to run an existing job.

• Add a Task element from the Element Toolbox. Enter tskCreateJob as the ID.

• Add Procedure.SchedulerRunTask procedure from the Element Toolbox. Look for the purple Task Scheduler folder.

• Enter the attribute values as follows:

• Add a Response.Report element to tskRunJob with and ID of rspMyScheduledJobRun.

• Add a Target.Report element to rspMyScheduledJobRun. Enter an ID of tgtMyScheduleTasksCreate and a Report Definition File of Schedule.MyScheduledJobs.
Here is the resulting Task:

4. Add a Task to delete a job.

   - Add a Task element from the Element Toolbox. Enter tskCreateJob as the ID.
   - Enter the attribute values as follows:

   - Add a Response.Report element to tskDeleteJob with and ID of rspMyScheduledJobDelete.

   - Here is the resulting Task:

5. Add a Task to update a job.

   - Add a Task element from the Element Toolbox. Enter tskCreateJob as the ID.
• Enter the attribute values as follows:

```plaintext
*Required Attributes
Connection ID          connScheduler
ID                    procCreateJob
ScheduleXml           @Request.ScheduleUI~
```

```plaintext
Optional Attributes
Scheduler Application ID          LogiTraining
Scheduler Custom Column 1          LogiTraining
Scheduler Custom Column 2          LogiTraining
Scheduler Process Url               http://localhost/LogiTraining
Scheduler RunAs                   @Function.UserName~
Scheduler Task Disabled            False
Scheduler Task Name                @Request.txtJobName~
```

• Substitute your URL with the URL shown above.

• Add a **Target.Process** element to **prcUpdateJob** from the Element Toolbox.

• Enter the attribute values as follows:

```plaintext
Optional Attributes
ID                    tgtEmailTasks
Process Definition File MyProcesses
Task ID                tskEmailReportPDF
```

• Add a **Response.Report** element to **tskUpdateJob** with and ID of **rspMyScheduledJobsUpdate**.

• Add a **Target.Report** element to **rspMyScheduledJobsUpdate**. Enter an ID of **tgtMyScheduledJobsUpdate** and a **Report Definition File** of **Schedule.MyScheduledJobs**.

There is no way to test these tasks until we first have definitions that can be rendered to launch them. The next two labs are designed around building these two definitions. The first is Scheduler Builder.
In this lab we will build a definition that is not quite a report though it has user inputs. The job of this definition is provide enough information to the task we build earlier called tskCreateJob. We will re-use this same Scheduler Builder definition to update a job using tskUpdateJob. Deleting and running jobs are covered by the third lab’s definition (MyScheduledJobs) which has buttons for these actions.

Lab 2 – Scheduler Builder

Objectives:

- Create a connection to the Logi Info Scheduler
- Build upon the Process Lab that exports / sends a PDF of an existing report definition.
- Create a definition to create a job in the Logi Scheduler
Directions:

1. Create a connection to the Logi Info Scheduler
   - Double-click on the _Settings definition.
   - Select the Connections element
   - Add a Connection.Scheduler from the Element Toolbox. Enter an Id of connScheduler.
   - Enter the attribute values as follows:

2. Create a folder under the purple reports folder called Scheduling.
   - Select the Reports folder right-click then Add and New Folder.
3. Add a new definition called SchedulerBuilder to the Scheduling folder.
   - Select the Body Element.
4. Add a **Schedule** Element with and ID **ScheduleUI** to the body.

- Add the following attribute values:

  ![Schedule Attributes](image)

- Add a new line and an **Input.Text** element to the Body

- Input text has an ID of **txtJobName** and a new line element to follow.

**Note:** once a task is scheduled it will be referred to as a “job”

- Enter a division element with an ID of **divCreateSchedule** to act as a container for the Create Schedule scenario (a separate division will be created for the Update Schedule scenario).

- Enter a division element with an ID **divUpdateSchedule** to act as a container for the Update Schedule scenario.

- Under **divCreateSchedule** add a new line element and a button element with and ID **btnCreateSchedule**.

- Add an **Action.Process** element under **btnCreateSchedule** with and ID **actCreateScheduler**.

- Enter **SchedulerTasks** as the process definition file

- Enter **tstkCreateJob** as the Task ID.

- Enter the following in the condition of the **divCreateSchedule** division element (including the quotes): “**@Request.NewJob**” = “**True**”
Note: The process.action is referring to a process definition file and task that may not yet exist. These ids (i.e. tskCreateJob) will be constructed in a separate process definition file that will contain all the tasks for Creating, Editing, Deleting and Running Jobs.

- Under divUpdateSchedule add a new line element and a button with and ID btnUpdateSchedule.
- Enter SchedulerTasks as the process definition file
- Enter tskUpdateJob as the Task ID.
- Enter the following in the condition of the divCreateSchedule division element:
  
  "@Request.NewJob~" = "False"

- Enter a Default Request Parameters element to the top element in the definition. This will provide a default value for the NewJob request parameter.
- Enter in a new parameter in the Default Request Parameters element called NewJob. Enter True in the attribute value.
Lab 3 – My Scheduled Jobs

Objectives:

- Build a user interface for the user to update, delete or run a job stored.
- Create a definition to create a job in the Logi Scheduler

Final Report Output

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Task Created</th>
<th>Task Modified</th>
<th>Task Last Run</th>
<th>Task Next Run</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>My exported PDF</td>
<td>08/12/2010 01:20</td>
<td>08/12/2010 01:20</td>
<td>08/12/2010 01:39</td>
<td>08/12/2010 01:39</td>
<td>At 1:39 PM on 08/12/2010 Run Job Delete Job Update Job</td>
</tr>
<tr>
<td>intermittent pdf export</td>
<td>08/12/2010 01:30</td>
<td>08/12/2010 01:30</td>
<td>08/24/2010 08:35</td>
<td>08:35 AM every 3 days, starting 08/12/2010</td>
<td>Run Job Delete Job Update Job</td>
</tr>
</tbody>
</table>
Directions:

1. Under the Scheduling folder create a new definition called **MyScheduledJobs**.

2. Under the report header add a button to create a new job if needed
   - Add a **Button** element to the report header with and ID **btnCreateScheduledJob**.
   - Add an **Action.Report** to **btnCreateScheduledJob** with and ID **actScheduledJob**.
   - Add a link parameters element to **actScheduledJob**.
     - Add a parameter called **NewJob** with a parameter value of **True**.
   - Add a **Target.Report** element for the action with an ID **tgtScheduledJob**.
   - Specify SchedulerBuilder as the report definition to target.

3. Add a standard data table to the body with an ID **dtScheduledJobs**.
   - Add a **DataLayer.Scheduler** with and id **dlScheduledJobs**.
   - Enter **connScheduler** as the connection ID.
   - Enter **LogiTraining** as the Scheduler Application ID.
Note: the application id is an arbitrary string that is used to differentiate jobs in the scheduler’s database if necessary. This can be tokenized.

4. Use the add data columns wizard to add the following columns to the data table.
   - Select dtScheduledJobs right-click then **Element Wizards** and **Add Data Table**.
     - Task Name
     - Task Created
     - Task Modified
     - Last Run
     - Next Run
     - Schedule Description

5. Add a data table column with and ID colRunJob to hold the buttons that will run an individual job on demand.
   - Add a button element to **colRunJob** with and ID **btnRunJob**.
   - Add an Action.Process element to **btnRunJob** with an ID of **actRunJob**.
     - Enter **SchedulerTasks** as the process definition file.
     - Enter **tskRunJob** as the Task ID.
     - Add a Link Parameters element to pass in the TaskID from the job in the current row.
       - Enter **JobID** as the parameter name and **@Data.TaskID~** as the value.

6. Add a data table column with and ID **colDeleteJob** to hold the buttons that will run an individual job on demand.
   - Add a button element to **colDeleteJob** with and ID **btnDeleteJob**.
   - Add an **Action.Process** element to **btnDeleteJob** with an ID of **actDeleteJob**.
     - Enter **SchedulerTasks** as the process definition file.
     - Enter **tskDeleteJob** as the Task ID.
7. Add a data table column with an ID colUpdateJob to hold the buttons that will run an individual job on demand.

   • Add a button element to colUpdateJob with an ID btnUpdateJob.
   • Add an Action.Report element to btnUpdateJob with an ID of actUpdateJob.
   • Add a Link Parameter element to pass in the TaskID from the job in the current row.
   • Enter JobID as the parameter name and @Data.TaskID~ as the value.
   • Enter JobName as the parameter name and @Data.TaskName~ as the value.
   • Enter NewJob as the parameter name and False as the value.
   • Add a Target.Report to actUpdateJob with an ID tgtSchedulerBuilder.
   • Specify a report definition file: SchedulerBuilder.


   • Select the Scheduler Builder definition in the Application Panel and Run In Browser.
   • Pick a Schedule, start time, run on etc. You can pick Once and select a time in the near future.
   • Enter in a Job Name.
   • Click on Create Scheduled Job.

---

You will see that when you update a job the timing is not coming through. .NET does not allow the movement of XML through the Request object therefore the @Request object. Instead we will consume the @Request.JobID~ token to load the job in a Scheduler.DataLayer. We will leverage the LocalData element to do this.
9. Add a Local Data element to SchedulerBuilder.

- Open SchedulerBuilder if not already open
- Select the top most element in the definition.
- Add a LocalData element from the toolbox.
- Press F7 until it is at the top of the definition
- Add a DataLayer.Scheduler and set the attribute values as follows:

- Finally select the ScheduleUI element.
- In the ScheduleXml attribute enter @Local.ScheduleXml

Local Data returns only one row of data which is perfect for this circumstance. Only one Job needs to be accessed. You can refer to any column in the DataLayer under the Local Data element using @Local.ColumnName


- Select the MyScheduledJobs definition in the Application Panel and Run In Browser.
- Choose an existing Job and click it’s Update Button.
- The proper time from that job should appear.
**Templates**

Unlike Exporting, which produces a new document every time it's invoked, Templates allow an existing document to be “exported” with fresh data each time it’s invoked. Here we will populate a sheet inside an Excel workbook with data from a Datalayer.Sql element. Thus in addition to a definition to handle the job, we need an existing document. We will create both in this lab.

**Lab 1 – Excel Template**

**Objectives:**

- Create an Excel Template and insert aggregate data from Northwind.

<table>
<thead>
<tr>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
</tbody>
</table>
Directions:

1. Building the Excel Spreadsheet:
   - From the start menu, run Microsoft Excel (Start->All Programs->Microsoft Office->Microsoft Excel). You are creating a file that will be named ExcelTemplate.xls.
   - Select Sheet1.
   - In cell A1 enter the text Category Name. Set the cell format to Bold (drag the cursor over the text and press control-b to make the text bold).
   - In cell B1 enter the text Category Sales, setting the format to Bold.
   - In cell A5 enter the text Total Sales, also in Bold.
   - Set the format for cells A2:A4 to Text;
   - Set the format for Cell B2:B4 to Currency.
   - In cell B5 enter a formula =SUM(B2:B4)

2. The worksheet should look like this:
• Save the workbook in My Documents and Name it ExcelTemplate.xls

3. Add the Template to Logi Info Support Files:

• In the Application Panel, right-click the support Files folder, select Add then Existing File

![Add Existing File](image)

• Browse for the file with a .xls extension (or Templates Files and click the Open button. If you used Excel 2007 you will need to hand enter the extension as follows and press enter.

![Browse for File](image)

• Choose ExcelTemplate.xls (or ExcelTemplate.xl, as the case may be) and the file will appear in the Support Files folder.

![Support Files](image)

**Note:** if you need to make changes to the Excel file, be sure to open the file externally:

• In the Application Panel, right-click ExcelTemplate.xls under Support file and select Open Externally:

![Open Externally](image)
4. Building the Logi Info Template definition

   • Create a new Template definition:
     - In the Application Panel, click the Templates folder, select Add New Definition.
     - Rename the definition Lab1_ExcelTemplate.

   Notice that there are no elements such as Style, Header, Footer or Body. This is due to the fact that the Template Definition is not to be rendered into HTML. Rather, Excel will be launched to product the output.

5. In the Workspace Panel, click on the Lab1_ExcelTemplate and then add an Excel Template element from the Element Toolbox.

   • Set the attribute values as shown.

6. Add a Pattern Block element under ex1CategorySales:

   • Set the attribute values as follows.

7. Add a DataLayer.SQL under the pbCategorySales from the Element Toolbox.

   • Set the Connection ID to connNorthwind.
• Set the **Source** attribute to the following SQL (OK to cut and paste):

```sql
Select Categories.CategoryName,
Sum([Order Details].UnitPrice * [Order Details].Quantity * (1-[Order Details].Discount)) as CategoryTotal
From [Order Details]
Inner Join Categories On Categories.CategoryID = Products.CategoryID
Group by Categories.CategoryName
```

8. Add a **Pattern Block Cell** element under the **Pattern Block** element **pbCategorySales**:

• Set the attribute values as follows.

9. Add a second **Pattern Block Cell** under the **Pattern Block** element **pbCategorySales**.

• Set the attribute values as follows.

10. Test the template definition

• Click the **Preview** button at the bottom of the Workspace Panel.
- Click the Open button in response to the message “Do you want to open or save ?”

![File Download]

- The spreadsheet file containing your report will be displayed:

```
<table>
<thead>
<tr>
<th></th>
<th>Category Name</th>
<th>Category Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Beverages</td>
<td>$267,868.18</td>
</tr>
<tr>
<td>2</td>
<td>Condiments</td>
<td>$106,047.08</td>
</tr>
<tr>
<td>3</td>
<td>Confections</td>
<td>$167,357.22</td>
</tr>
<tr>
<td>4</td>
<td>Dairy Products</td>
<td>$234,507.29</td>
</tr>
<tr>
<td>5</td>
<td>Grains/Cereals</td>
<td>$95,744.59</td>
</tr>
<tr>
<td>6</td>
<td>Meat/Poultry</td>
<td>$163,022.36</td>
</tr>
<tr>
<td>7</td>
<td>Produce</td>
<td>$99,984.58</td>
</tr>
<tr>
<td>8</td>
<td>Seafood</td>
<td>$131,261.74</td>
</tr>
<tr>
<td></td>
<td>Total Sales</td>
<td>$1,265,793.04</td>
</tr>
</tbody>
</table>
```
Crosstabs

Crosstabs differ from traditional tabular reporting. In most tabular reports, you know up front which columns and how many are going to be displayed. In a crosstab we won’t know until we run the report how many columns we will get. This lab will aggregate sales by Employee and by Order Year. The Order Years will be the columns of the report. If a new order year starts, one more column is added. In order to construct a crosstab, a CrosstabTable element is used and augmented with a DataLayer that has a CrosstabFilter as a child.

One can also think of a Crosstab as a “static Pivot Table.

Lab 1 – Basic Crosstab

Objectives:

• Create a crosstab table showing the Sales by Employee for the years 1996, 1997 and 1998

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="CrossTabs.Lab1_Basic_CrosTab" /></td>
<td><img src="image" alt="Employee Sales by Year" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Davole, Nancy</td>
<td>$35,764.51</td>
<td>$93,148.09</td>
<td>$58,364.78</td>
</tr>
<tr>
<td>Fuller, Andrew</td>
<td>$21,757.06</td>
<td>$70,644.14</td>
<td>$70,568.58</td>
</tr>
<tr>
<td>Leverling, Janet</td>
<td>$18,223.96</td>
<td>$108,026.14</td>
<td>$76,562.74</td>
</tr>
<tr>
<td>Peacock, Margot</td>
<td>$49,945.12</td>
<td>$128,809.79</td>
<td>$47,006.70</td>
</tr>
<tr>
<td>Buchanan, Steven</td>
<td>$15,383.92</td>
<td>$30,716.47</td>
<td>$18,691.90</td>
</tr>
<tr>
<td>Suyama, Michael</td>
<td>$15,642.61</td>
<td>$43,126.37</td>
<td>$12,758.66</td>
</tr>
<tr>
<td>King, Robert</td>
<td>$15,232.16</td>
<td>$50,471.19</td>
<td>$43,915.69</td>
</tr>
<tr>
<td>Callahan, Laura</td>
<td>$22,240.12</td>
<td>$56,032.62</td>
<td>$45,589.85</td>
</tr>
<tr>
<td>Dodsworth, Anne</td>
<td>$9,094.52</td>
<td>$26,316.39</td>
<td>$40,245.16</td>
</tr>
</tbody>
</table>
Directions:

1. Create a New Folder for **Crosstabs**.
   - In the Application Panel click on the folder Reports then right-click and **Add** -> **New Folder**
   - Rename the new folder **Crosstabs**.

2. Create a new definition for the report, called **Lab1_Basic_Crosstab**:
   - Click on the folder **Crosstabs**. Right-click **Add** then **New Definition**.
   - Rename the new definition to **Lab1_Basic_Crosstab**.

3. In the body of the report, add a **Crosstab Table** element:
   - In the Workspace Panel, click on **Body**
   - Add a **Crosstab Table** element from the Element Toolbox under the DataTables folder.
• In the Attributes Panel set the attribute values as follows.

4. Add a **Data Layer** element to access the Northwind database:

   • In the Workspace Panel, click on the Crosstab table element `ctTotalSalesByEmployee`.
   
   • Add a **DataLayer.SQL** element from the Element Toolbox.
   
   • Set the attribute values as follows:

```
[108x274]Select YEAR(Orders.OrderDate) As OrderYear, [Order Subtotals].Subtotal, Orders.EmployeeID, Employees.LastName + ', ' + Employees.FirstName As EmployeeName
From Orders, [Order Subtotals], Employees
Order By Orders.EmployeeID
```

5. Add a **Crosstab Filter** element

   • In the workspace Panel, select `dlSalesByCategory`.
• Add a **Crosstab Filter** element from the Element Toolbox

![Crosstab Filter](image)

• Set the attribute values of the Crosstab Filter as follows:

![Element - CrosstabFilter](image)

6. Add a Crosstab Table Label Column under `ctTotalSalesByEmployee`:

![General Elements](image)

• In the workspace Panel, select `ctTotalSalesByEmployee`.

• In the Data Element Toolbox, double-click the **Crosstab Table Label Column** element.

![ctTotalSalesByEmployee](image)

• Set the attribute values of the **Crosstab Label Column** as follows:

![Element - CrosstabTableLabelColumn](image)
7. Add a Label under `ctlcEmployeeName`:
   - In the workspace Panel, select `ctlcEmployeeName`.
   - Add a Label Element from the Element Toolbox.
   - Set the attribute values of the Label as follows:

![Label Element](image)

8. Add a **Crosstab Table Value Column** element under `ctTotalSalesByEmployee`:
   - In the workspace Panel, select `ctTotalSalesByEmployee`.
   - Add a **Crosstab Table Value Column** element from the Element Toolbox.
   - Set the attribute values of the Crosstab Table Value Column element as follows:

![Crosstab Table Value Column](image)

**Note:** the Column Header element contains a reference to the data layer via the @Data token. Since the data determines the number of columns the Column Header must contain data that will be dynamic as more columns are added by the engine. This column `rdCrosstabColumn` is a special item in the DataLayer that allows the developer to display the column values coming out of the crosstab filter.
9. Add a Label under `ctvcEmployeeSales`.
   - In the workspace Panel, select `ctvcEmployeeSales`.
   - Add a Label element from the Element Toolbox.
   - Set the attribute values of the Label as follows.

```
*Required Attributes
Caption: %Data.rdCrosstabValue~

Optional Attributes
Class
Error Result
Format: Currency
ID: lbEmployeeSales
Security Right ID
Tooltip
```

10. Verify your report:
   - Click on the Save All Opened Files button in the toolbar to save all changes to both reports.
   - Open the Lab1_Basic_Crosstab definition in the Workspace Panel. The definition should appear as in the definition section at the beginning of this Lab.
   - Choose the Lab1_Basic_Crosstab in the Application Panel. Right-click and choose Run In Browser. The Report Output should match the Output shown at the beginning of this Lab.
Lab 2 – Row Summary

Objectives:

- Build on Lab1 with a summary for each Employee (Row).

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Report Definition Diagram" /></td>
<td><img src="image" alt="Final Report Output Table" /></td>
</tr>
</tbody>
</table>

Directions:

11. Copy Lab1_Basic_Crosstab.
   - Select Lab1_Basic_Crosstab from the Crosstabs report folder in the Application Panel.
   - Rename the copy Lab2_RowSummary.

12. Add a Crosstab Row Summary Column element under the Crosstab Filter ctTotalSalesByEmployee:
• Select `ctTotalSalesByEmployee`.

• Add a **Crosstab Row Summary Column** element from the Element Toolbox.

• Set the **ID** to `csrctotalSalesByEmployee`.

• Set the function to **Sum**.

13. Add a **Crosstab Table Label Column** to the Crosstab Table `ctTotalSalesByEmployee`.

• Select the Crosstab Table `ctTotalSalesByEmployee`.

• Add a **Crosstab Table Label Column** to the Crosstab Table with an **ID** of `ctlcTotalSalesByEmployee`.

• Add a Label element to `ctlcTotalSalesByEmployee` with an **ID** of `lblTotalSalesByEmployee`.

• Enter in the attribute values as shown.
14. Verify your report:

- Click on the **Save All Opened Files** button in the toolbar to save all changes to both reports.

- Open the **Lab2_RowSummary** definition in the Workspace Panel. The definition should appear as in the definition section at the beginning of this Lab.

- Choose the **Lab2_RowSummary** in the Application Panel. Right-click and choose Run In Browser. The Report Output should match the Output shown at the beginning of this Lab.
Lab 3 – Column Summary

Objectives:

- Create column totals for each Order Year.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Report Definition" /></td>
<td><img src="image" alt="Final Report Output" /></td>
</tr>
</tbody>
</table>
Directions:

1. Copy `Lab2_RowSummary` the Crosstabs report folder. Rename the copy `Lab3_ColumnSummary`.

2. Add a `Summary` element under the Crosstab Table Value Column `ctvcEmployeeSales`:
   - Select the Crosstab Table Value Column `ctvcEmployeeSales` and add a `Data Column Summary` element from the Element Toolbox.
   - Enter the attribute values as follows:

   ![Data Column Summary Element](image)

3. Add a `Summary Row` element under the Data Table `ctTotalSalesByEmployee`.
   - Select the Crosstab Table `ctTotalSalesByEmployee`.
   - Add a `Summary Row` element from the Element Toolbox.
   - Set the attribute values as follows:

   ![Summary Row Element](image)

4. Total up the Row Summaries.
   - Select the `Crosstab Table Label Column` element `ctTotalSalesByEmployee`.
   - Add a `Data Column Summary` element from the Element Toolbox and give it an ID of `sumGrandTotal`.
• Enter the attribute values as follows:

![Attribute Values Table]

Summary elements appear automatically in a Summary Row with no customization.

5. Verify your report:

• Click on the **Save All Opened Files** button in the toolbar to save all changes to both reports.

• Open the **Lab3_ColumnSummary** definition in the Workspace Panel. The definition should appear as in the definition section at the beginning of this Lab.

• Choose the **Lab3_ColumnSummary** in the Application Panel. Right-click and choose Run In Browser. The Report Output should match the Output shown at the beginning of this Lab.
Lab 4 – Extra Value

Objectives:

- Show the Average Sales per Employee for each Order Year.

### Report Definition

#### Final Report Output

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirollo, Nancy</td>
<td>$35,764.51</td>
<td>$1,175.56</td>
<td>$60,148.09</td>
<td>$1,693.80</td>
<td>$55,364.78</td>
<td>$1,498.53</td>
<td>$187,277.38</td>
<td>$1,521.90</td>
<td></td>
</tr>
<tr>
<td>Fuller, Andrew</td>
<td>$21,757.06</td>
<td>$1,135.02</td>
<td>$70,444.14</td>
<td>$1,718.15</td>
<td>$70,569.58</td>
<td>$1,969.24</td>
<td>$182,760.78</td>
<td>$1,679.49</td>
<td></td>
</tr>
<tr>
<td>Levering, Janet</td>
<td>$18,223.90</td>
<td>$1,412.44</td>
<td>$106,626.14</td>
<td>$1,521.49</td>
<td>$76,562.74</td>
<td>$2,184.01</td>
<td>$206,912.64</td>
<td>$1,516.25</td>
<td></td>
</tr>
<tr>
<td>Peacock, Margaret</td>
<td>$49,945.12</td>
<td>$1,611.13</td>
<td>$126,009.75</td>
<td>$1,550.24</td>
<td>$47,003.75</td>
<td>$1,205.55</td>
<td>$225,730.70</td>
<td>$1,469.97</td>
<td></td>
</tr>
<tr>
<td>Buchanan, Steven</td>
<td>$16,353.52</td>
<td>$1,677.27</td>
<td>$30,719.47</td>
<td>$1,706.47</td>
<td>$19,691.96</td>
<td>$1,514.76</td>
<td>$68,752.29</td>
<td>$1,620.02</td>
<td></td>
</tr>
<tr>
<td>Suyama, Michael</td>
<td>$16,642.61</td>
<td>$1,109.51</td>
<td>$42,126.27</td>
<td>$1,306.06</td>
<td>$12,756.86</td>
<td>$750.51</td>
<td>$72,257.84</td>
<td>$1,955.63</td>
<td></td>
</tr>
<tr>
<td>King, Robert</td>
<td>$15,232.16</td>
<td>$1,364.74</td>
<td>$66,471.19</td>
<td>$1,679.76</td>
<td>$43,915.89</td>
<td>$1,969.18</td>
<td>$191,810.24</td>
<td>$1,886.59</td>
<td></td>
</tr>
<tr>
<td>Callahan, Laura</td>
<td>$22,240.42</td>
<td>$1,170.83</td>
<td>$58,033.62</td>
<td>$1,677.84</td>
<td>$45,868.95</td>
<td>$1,687.78</td>
<td>$123,042.09</td>
<td>$1,265.65</td>
<td></td>
</tr>
<tr>
<td>Dodsworth, Anne</td>
<td>$9,804.52</td>
<td>$1,978.60</td>
<td>$26,310.59</td>
<td>$1,534.76</td>
<td>$40,245.16</td>
<td>$2,235.64</td>
<td>$76,450.07</td>
<td>$1,996.99</td>
<td></td>
</tr>
<tr>
<td>Totals</td>
<td>$208,683.88</td>
<td>$1,648.21</td>
<td>$517,658.70</td>
<td>$1,816.44</td>
<td>$444,884.46</td>
<td>$1,651.33</td>
<td>$2,359,885.63</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Directions:**

1. **Copy** **Lab3_ColumnSummary**.
   - Select **Lab3_ColumnSummary** in the Application Panel.
   - Right-click then **Copy**
   - Select the **Crosstab** folder, right-click then **Paste**
   - Rename the copy **Lab4_ExtraValue**.
Since a **Crosstab Filter** only contains one Value Column from the Data Layer, an **Extra Crosstab Value Column** is added to the Crosstab Filter as an element. As with any Data Layer children elements, they must have a companion **Label** element to be rendered.

2. **Add an Extra Crosstab Value Column.**
   - Select the **Crosstab Filter** element `cfTotalSalesByEmployee`
   - Add an **Extra Crosstab Value Column** from the Element Toolbox.

   ![Extra Crosstab Value Column](image)

   - Enter the attribute values as follows:

   ![Attribute Values](image)

3. **Add a Crosstab Row Summary Column** element to handle the average by employee.
   - Select the **Crosstab Filter** element `ctTotalSalesByEmployee`.
   - Add a **Crosstab Row Summary Column** from the Element Toolbox.
   - Set the attribute values as follows:

   ![Row Summary Column](image)

Though the Extra Crosstab Value Column for average sales by order year / employee could have its own column – in this example it will share the same column (on a second row) as the total sales by order year /employee.
4. Add the employee average for each order year.

- Select the **CrosstabTable** element **ctTotalSalesByEmployee**
- Add a second Crosstab Table Value Column element

```
4. Add the employee average for each order year.

- Select the **CrosstabTable** element **ctTotalSalesByEmployee**
- Add a second Crosstab Table Value Column element

```

- Set the attribute values as follows:

```
- Set the attribute values as follows:

```

5. Update the Employee Row Totals column with the Average Sales by Employee.

- Select the Crosstab Table Label Column **ctlcTotalSalesByEmployee**.
- Add a **New Line** element from the Element Toolbox.
• Re-Select `ctlTotalSalesByEmployee` and add 2 `Label` elements after the `New Line` element. Use the F8 key to move the `New Line` element down between the 2 `Label` elements.

• Set their attribute values as follows:
  o Set the ID of the first new Label to `lblTotalSalesByEmployee`
  o Set the Caption to `@Data.csctTotalSalesByEmployee~`
  o Set the Format to Currency:

![Label Properties](image)

  o Set the ID of the next Label to `lblEmployeeRowAverage`;
  o Set the Caption to `@Data.csceEmployeeAverage~`
  o Set the Format to Currency:

![Label Properties](image)

6. Verify your report:

• Click on the `Save All Opened Files` button in the toolbar to save all changes to both reports.

• Open the `Lab4_ExtraValue` definition in the Workspace Panel. The definition should appear as in the definition section at the beginning of this Lab.

• Choose the `Lab4_ExtraValue` in the Application Panel. Right-click and choose Run In Browser. The Report Output should match the Output shown at the beginning of this Lab.
Analysis Grid

Analysis Grids allow "Ad Hoc" like analysis to be done using Logi Info. The user will run this definition and have the ability to customize it through creating filters, eliminating or adding columns as well as adding charting and crosstabs. Since this customization is lost when the browser is closed, supplemental labs will show how to retain the customized analysis grid using bookmarks.

Lab 1 – Analysis Grid - Orders Employees and Order Totals

Objectives:

- Create an Analysis Grid with data from the Orders Table in Northwind.
Directions:

1. In the Application Panel, under the Reports folder, create a new definition and rename it to AnalysisGrid.
   - Select the top Reports folder in the Application Panel.
   - Right-click, then choose Add New Definition and rename to AnalysisGrid_Lab.

2. In the body of the report, add a Analysis Grid element (Using the Wizard):
   - In the Workspace Panel, click on Body.
   - Right-click the Body element, select the Element Wizards, and choose Add Analysis Grid.
   - Select Next – Select connNorthwind on the next step then click Next.
   - Enter in the following SQL in the next window.

   ```sql
   ```
   - Click Next - accept all the checked columns on the next step. Click Next and Finish.
   - In the Attributes Panel, rename the Analysis Grid element’s ID attribute value from AnalysisGrid1 to agOrders.
   - In the Attributes Panel, set the Datalayer.SQL element’s ID attribute value to dlOrders.

The Analysis Grid is best viewed and customized in the full browser window. In the Application Panel right-click and choose Run in Browser.

- Once in the browser changes can be made to the Analysis Grid. These changes however are lost once the browser is closed. The next lab will provide a guide to creating a “bookmark” which will take a snapshot of the Analysis Grid format (columns, filtering, grouping etc.) for use later with new data.
• The initial screen is as follows:

![Image of initial screen]

• The column is added to the end of the list of columns.

• Click on the **Formula** button in the **Analysis Grid** Toolbar.

• This screen controls the arrangement and visibility of the columns in the report.

| Move the column with the Orange Triangles. | Click Ok to set the order |
The **Sort** button allows the initial sort of the report when opened initially (only applies to bookmark Analysis Grids).
The **Filter** button allows the user to limit the rows of the report based on a point and click interface.

Once a filter has been built, the records are filtered immediately. Clicking on the filter will fill in the column, comparison and value from that filter.

Multiple filters are possible and the logic can be altered by changing the **And** to **Or** and by moving the filter rows with the orange triangles.

Remove the filters for the remainder of the labs so that all the records in the orders table are available.

The **Group** button allows grouping and aggregation by groups.

Add one or several groups by selecting a column to group on and click on **Add**.
• Add additional columns by selecting a second column and clicking Add again.

![Image of Row Grouping interface]

• This grouping results in this report.

• Aggregates are added to these groups with the Aggregate button.

• Choose a column (including ones added with the Formula button) and an aggregate function and click Add.

![Image of Aggregate interface]

• The aggregates use the same formatting as the displayed column that is being aggregated.
To clean up the report set the formatting and alignment in the Analysis Grid Columns.

- Set the class attribute for `colOrderDate`, `colShippedDate`, `colSubtotal` and `colFreight` Columns to `ThemeAlignRight`. Set the `Format` attribute to Currency for `colSubtotal` and `colFreight`.

- The Chart button allows the addition of a Chart to the Table.

- Select the Chart Type and provide a Label (X – Axis) a Data Column (Y-Axis), add an aggregation and click Add.
Add a Crosstab to the report by clicking on the Crosstab button.

- Enter a **Header Values** column that will control the Column Headings in the Crosstab. Choose OrderYear.

- Enter the **Label Values** column that will be the row label in the left hand column. Choose LastName

- Enter the **Aggregate Values** column. This is the value that is aggregated in both column and row directions. Choose Subtotal.
The resulting Crosstab is as follows:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LastName</td>
<td>1996</td>
<td>1997</td>
<td>1998</td>
</tr>
<tr>
<td>Buchanan</td>
<td>$16,363.92</td>
<td>$30,716.47</td>
<td>$19,691.90</td>
</tr>
<tr>
<td>Suyama</td>
<td>$16,642.61</td>
<td>$43,126.37</td>
<td>$14,144.16</td>
</tr>
<tr>
<td>Peacock</td>
<td>$49,945.12</td>
<td>$128,809.79</td>
<td>$54,155.94</td>
</tr>
<tr>
<td>Leverling</td>
<td>$18,223.96</td>
<td>$108,028.14</td>
<td>$76,662.74</td>
</tr>
<tr>
<td>Dodsworth</td>
<td>$9,894.52</td>
<td>$26,310.39</td>
<td>$41,103.16</td>
</tr>
<tr>
<td>Davollo</td>
<td>$35,764.51</td>
<td>$93,148.09</td>
<td>$63,195.00</td>
</tr>
<tr>
<td>Callahan</td>
<td>$22,240.12</td>
<td>$56,032.62</td>
<td>$46,569.55</td>
</tr>
<tr>
<td>Fuller</td>
<td>$21,757.06</td>
<td>$70,444.14</td>
<td>$74,336.55</td>
</tr>
<tr>
<td>King</td>
<td>$16,202.16</td>
<td>$60,471.19</td>
<td>$48,864.88</td>
</tr>
</tbody>
</table>

- The Paging button controls whether or not paging is shown.

![Paging Button](image)

- Note that on the Crosstab and Table section you can export out the data to Excel, CSV or PDF.

- You can control all the aspects of the Columns and the buttons at design time with either the Analysis Grid element itself or the respective columns.

- Once the browser is closed you will lose any of the customization set during that session. The next lab will demonstrate the addition of bookmarks which allows the retention and labeling of the customization to the Grid.
Lab 2 – Bookmarks (Analysis Grid)

Note: this lab can be used with OLAP Grids, Dimension Grids and Analysis Grids using these same steps. A bookmark is an XML file that captures the structure of the Analysis Grid but not the data.

Objectives:

- Build upon existing OLAP Grid, Dimension Grid or Analysis Grid definition. Copy AnalysisGrid_Lab and into the root of the Reports folder and rename AnalysisGrid_Lab_With_Bookmark.

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Table" /></td>
</tr>
</tbody>
</table>
Directions:

1. Insure that there is \@Function.AppPhysicalPath\"\UserBookmarks\" in the Bookmark Location attribute that belongs to the General Element in the _Settings definition.

   - Add the folder to the root folder of the application on the server.

2. Add a button to the Report footer to save a bookmark
   - Add a button named btnSaveBookmark to the report footer with a caption Save Bookmark.
   - Add the Action.AddBookmark action from the Bookmarks folder in the Element Toolbox. Enter in the ID actAddBookMark.
3. Add the manage (or display) bookmarks button:

- Select the Report Footer element.
- Add a space element from the Element Toolbox and re-select the Report Footer element.
- Add button with id btnDisplayBookmarks (optionally insert a space element first).
- Add a target with id tgtDisplayBookmarks
- Enter Display_Bookmarks in the definition name attribute.

- Optionally, you may also leave blank and fill in after completing the display bookmarks lab.

4. Verify the bookmark:

- The manage bookmark lab is to follow. It is in a separate lab so that it can be taught after this lab or after a bookmark lab that involves analysis grids, OLAP grids or Dimension Grids.
- Run Analysis_Lab_With_Bookmarks in the Browser. Make some kind of change (i.e. group on something).
• Click on the **Save Bookmark** button and enter a description then click **OK**.

![Save Bookmark Button](image)

• The status bar (if enabled) in your browser should say the bookmark was saved (IE Only).

• Refer to these bookmarks when initially demonstrating the manage bookmarks definition.
Lab 3 – Display Bookmarks

Objectives:

- Create a new definition to hold the Display Bookmarks elements.
- Create a data table using data from the bookmarks data layer
- Add elements to edit and delete bookmarks
- Add a link to launch a saved or bookmarked report
- Change AnalysisGrid_Lab to link to the Bookmark Manager

<table>
<thead>
<tr>
<th>Report Definition</th>
<th>Final Report Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Report Definition Diagram]</td>
<td>![Final Report Output Diagram]</td>
</tr>
</tbody>
</table>
Directions:

1. In the Bookmarks folder, create a new report definition called **Display_Bookmarks**.
   - In the Application Panel, right-click on the Reports folder and add a new definition then rename to **Display_Bookmarks**

2. Under the **Body** element of the report, add a **Data Table** element:
   - Select the **Body** element.
   - Add a **Data Table** element from the Element Toolbox.
   - In the Attributes Panel, set the **Data Table** element attribute values as follows:
     - Set the **ID** attribute to **dtBookMarks**.
     - Set the **Caption** to **Saved Bookmarks**.

3. Add a **Data Layer** element to access the bookmarks collection:
   - Click on the **Data Table** element **dtBookMarks**.
   - Add the **DataLayer.Bookmarks** element from the Element Toolbox.
     - A **Data Layer** element icon will appear in the Workspace Panel under the **dtBookMarks** element. If necessary, click on the new **Data Layer** element to select it.
     - In the Attributes Panel, set the **Data Layer** element attribute values as follows:
       - Set the **ID** attribute to **dlBookMarks**.
       - Add the name of the collection **MyBookmarks**.

   - This **Bookmark Collection** is hard coded in this example but can easily be made flexible in more advanced implementations by using a **@Request** token or **@Function** token.
• Note the **Bookmark Collection** entry was defined in the previous lab.

4. Add a Data Table Column elements to the Data Table:

   • In the Workspace Panel, click on the **Data Table** element `dtBookMarks`
   
   • Right mouse click `dtBookMarks` and choose element wizards then **Add data columns**.
• Choose the following columns from the wizard

![Add Data Columns]

5. (Click the Preview button at the bottom of the Workspace Panel. You should now see the three columns displayed with the bookmark information provided in a previous lab.

6. Add a link to the description label:

• The label for the description column in the data table `dtBookMarks` is the element that will contain the link to the saved bookmark.

• Select the description label element (`lblDescription`)

• Add an `Action.RunBookMark` to the label and give it an id of `actRunBookMark`

• For the `BookmarkID` attribute, enter `@Data.BookmarkID~`

• Add a target element to `actRunBookmark` with an id of `tgtRunBookMark`

• Choose New Window for the FrameID in `tgtRunBookMark`

7. Manually Add a data table column named `colAction` with a column heading of “Action” column to hold buttons for editing and deleting.

• Add a data table column to `dtBookMarks` with an id of `colAction`
• Add a button to the column with an id of **btnDelete**
  
  o Enter “**Delete Bookmark**” for the caption of the button
  
  o Add an **Action.RemoveBookmark** element to the button
  
  o In the Bookmark Collection attribute enter **MyBookmarks**
  
  o The bookmark ID attribute is supplied by the BookmarkID column in the bookmarks data layer. Enter @Data.BookmarkID~
  
  o Enter an id of **actRemoveBookmark**
  
  o Enter a confirmation message.

• Add a button to the column with an id of **btnEdit** (may also add a space element first)
  
  o Enter “**Edit Bookmark**” for the caption of the **button**
  
  o Add an **Action.EditBookmark** element to the button and the attribute values as shown above.

8. Verify your report:

• Run the report in Browser and the bookmark saved in the previous lab should appear. Click on the Description and the saved report (**Analysis Grid**) should appear.

• Check the Edit Bookmark and verify that the description appears in the window and the Ok button works.

• Click on the **Delete** button if the row doesn’t go away then press F5.
Dashboards

Lab 1 – Basic Dashboard

Objectives:

- Construct a simple dashboard with three panels.

Report Definition
Final Report Output

Sales by Category - Pie Chart

Employee Sales by Order Year

<table>
<thead>
<tr>
<th>Employee</th>
<th>1996</th>
<th>1997</th>
<th>1999</th>
</tr>
</thead>
<tbody>
<tr>
<td>Davolio, Nancy</td>
<td>$36,794.51</td>
<td>$33,140.89</td>
<td>$58,364.78</td>
</tr>
<tr>
<td>Fuller, Andrew</td>
<td>$21,757.96</td>
<td>$70,444.14</td>
<td>$78,656.58</td>
</tr>
<tr>
<td>Levering, Janet</td>
<td>$103,223.96</td>
<td>$185,826.16</td>
<td>$196,562.74</td>
</tr>
<tr>
<td>Peaced, Margaret</td>
<td>$49,945.12</td>
<td>$132,809.79</td>
<td>$47,005.79</td>
</tr>
<tr>
<td>Buchman, Steven</td>
<td>$18,383.92</td>
<td>$31,716.47</td>
<td>$19,691.90</td>
</tr>
<tr>
<td>Sizane, Michael</td>
<td>$16,642.61</td>
<td>$43,126.37</td>
<td>$12,758.99</td>
</tr>
<tr>
<td>King, Robert</td>
<td>$15,232.16</td>
<td>$50,471.19</td>
<td>$43,915.89</td>
</tr>
<tr>
<td>Calahan, Laura</td>
<td>$22,240.12</td>
<td>$55,832.82</td>
<td>$49,568.95</td>
</tr>
<tr>
<td>Dobsonsh, Anne</td>
<td>$9,994.52</td>
<td>$30,110.39</td>
<td>$49,245.16</td>
</tr>
</tbody>
</table>

Sales by Category - Pie Bar Chart
Directions:

1. Prepare the Application
   - Create a folder called Dashboards at the root of the physical folder location of the application.

2. Add a New Definition to the Report Folder
   - Select the Report Folder in the Application Panel then right-click and Add -> New Definition
   - Rename the definition to Dashboard

3. Add the Dashboard Element
   - Select the Body Element and add a Dashboard element from the Element Toolbox.
- Enter in the attribute values as shown.

Note that the Save file is hard coded. This brings up an important decision about the management of the dashboard. Should the Dashboard by shared by multiple people or does everyone get their own. If Shared you must set the Dashboard Adjustable attribute to False. If everyone has their own then in place of the word Dave in the Save File attribute one would put `@Function.Username~`. This assumes that security will be turned on.

4. Add First Dashboard Panel

- Select the Dashboard element and add a Panel element from the Element Toolbox.

- Enter in the attribute values as shown.

- Add a Panel Content Element
• At this point add whatever content you wish to appear in the Dashboard Panel. You can copy and paste from other definitions or add to the panels individually.

• Copy the Pie Chart from Lab1_Pie Chart and paste under the Panel Content element
  
  o Open Lab1_PieChart and select the Pie Chart Element. Right-click then Copy.
  
  o Select the Panel Content element in the Dashboard definition. Right-click then Paste.

5. Add Second Dashboard Panel

• Select the Dashboard element and add a Panel element from the Element Toolbox

• Enter in the attribute values as shown.

• Add a Panel Content element

• At this point add whatever content you wish to appear in the Dashboard Panel. You can copy and paste from other definitions or add to the panels individually.

• Copy the Crosstab table from Lab5_BarChart and paste it under the Panel Content element.
  
  o Open Lab5_BarChart and select the Bar Chart Element. Right-click then Copy.
  
  o Select the Panel Content element in the Dashboard definition. Right-click then Paste.
6. Add Third Dashboard Panel.

- Select the Dashboard element and add a Panel element from the Element Toolbox

- Enter in the attribute values as shown.

- Add a Panel Content Element

- At this point add whatever content you wish to appear in the Dashboard Panel. You can copy and paste from other definitions or add to the panels individually.

- Copy the Crosstab table from Lab1_Crosstab and paste it under the Panel Content element.
  
  o Open Lab1_Crosstab and select the Crosstab element ctTotalSalesByEmployee element. Right-click then Copy.
  
  o Select the Panel Content element in the Dashboard definition. Right-click then Paste.

7. Run the Dashboard in the browser

- Right-click on the Dashboard definition in the Application Panel. Right-click Run In Browser.

- The first time that the dashboard is run there will be no file in place. Logi will create the file after the choices are made on this screen.
8. **Optional** - Add a default request parameter to handle the height and width

Adjust the height and width in each of the individual panels. It is advisable to have a standard height and width for each panel. One common method is to add a **Default Request Parameters** element that defines two request parameters, **height** and **width**. You would then use them in the **Height** and **Width** attributes of the **AnimatedChart.Pie**, **Chart.XY** (bar chart), and **Crosstab Table** elements.
• Add a **Default Request Parameter** element to the topmost element **Dashboard**.

• Enter a height and width parameter

![Parameters](image)

• Enter in request parameters instead of hard coded values for the height and width of the pie chart.

![Required Attributes](image)

• Enter in request parameters instead of hard coded values for the height and width of the bar chart.

![Required Attributes](image)

You can now control the height and width from one spot. This concept of “tokenization” can be extended to non dashboard definitions / elements as well. The height and width can also be defined as constants. In this case instead of @Request you would use @Constant.

You can control the Dashboard contents at any time by clicking on **Change Dashboard**.
Security

Lab 1 - Security Basics (Standard Security)

Objectives:

- Demonstrate Logi Security by invoking **Page Level** – **Element Level** and **Record Level** Security to a Definition
### Report Output

#### Logon

Username: 
Password: 
[Logon]

---

**User: Bob**  
Roles: USER_GROUP_A  
Rights: ViewDataTable

<table>
<thead>
<tr>
<th>Order Id</th>
<th>CustomerID</th>
<th>OrderDate</th>
<th>ShippedDate</th>
<th>ShipCity</th>
<th>ShipCountry</th>
</tr>
</thead>
<tbody>
<tr>
<td>19248</td>
<td>VINET</td>
<td>7/4/1996</td>
<td>7/16/1996</td>
<td>Roins</td>
<td>France</td>
</tr>
<tr>
<td>19249</td>
<td>TOMSP</td>
<td>7/5/1996</td>
<td>7/10/1996</td>
<td>Münsler</td>
<td>Germany</td>
</tr>
<tr>
<td>19250</td>
<td>HANAR</td>
<td>7/6/1996</td>
<td>7/12/1996</td>
<td>Rio de Janeiro</td>
<td>Brazil</td>
</tr>
<tr>
<td>19251</td>
<td>VICTE</td>
<td>7/6/1996</td>
<td>7/15/1996</td>
<td>Lyon</td>
<td>France</td>
</tr>
<tr>
<td>19252</td>
<td>SUPRD</td>
<td>7/6/1996</td>
<td>7/11/1996</td>
<td>Charleroi</td>
<td>Belgium</td>
</tr>
<tr>
<td>19253</td>
<td>HANAR</td>
<td>7/10/1996</td>
<td>7/16/1996</td>
<td>Rio de Janeiro</td>
<td>Brazil</td>
</tr>
<tr>
<td>19255</td>
<td>RICSU</td>
<td>7/12/1996</td>
<td>7/15/1996</td>
<td>Genève</td>
<td>Switzerland</td>
</tr>
<tr>
<td>19256</td>
<td>WELLI</td>
<td>7/15/1996</td>
<td>7/17/1996</td>
<td>Rosendo</td>
<td>Brazil</td>
</tr>
<tr>
<td>19257</td>
<td>HILAA</td>
<td>7/16/1996</td>
<td>7/22/1996</td>
<td>San Cristóbal</td>
<td>Venezuela</td>
</tr>
<tr>
<td>19258</td>
<td>ERNSH</td>
<td>7/17/1996</td>
<td>7/23/1996</td>
<td>Graz</td>
<td>Austria</td>
</tr>
<tr>
<td>19259</td>
<td>CENTC</td>
<td>7/19/1996</td>
<td>7/25/1996</td>
<td>México D.F.</td>
<td>Mexico</td>
</tr>
<tr>
<td>19260</td>
<td>OTTIK</td>
<td>7/19/1996</td>
<td>7/28/1996</td>
<td>Köln</td>
<td>Germany</td>
</tr>
<tr>
<td>19261</td>
<td>QUEDE</td>
<td>7/19/1996</td>
<td>7/30/1996</td>
<td>Rio de Janeiro</td>
<td>Brazil</td>
</tr>
<tr>
<td>19262</td>
<td>ERNSH</td>
<td>7/23/1996</td>
<td>7/31/1996</td>
<td>Graz</td>
<td>Austria</td>
</tr>
<tr>
<td>19265</td>
<td>BLONP</td>
<td>7/25/1996</td>
<td>9/12/1996</td>
<td>Strasbourg</td>
<td>France</td>
</tr>
<tr>
<td>19266</td>
<td>WARTH</td>
<td>7/26/1996</td>
<td>9/31/1996</td>
<td>Oulu</td>
<td>Finland</td>
</tr>
<tr>
<td>19267</td>
<td>FRANK</td>
<td>7/29/1996</td>
<td>9/9/1996</td>
<td>München</td>
<td>Germany</td>
</tr>
<tr>
<td>19268</td>
<td>GROSR</td>
<td>7/30/1996</td>
<td>9/2/1996</td>
<td>Caracas</td>
<td>Venezuela</td>
</tr>
</tbody>
</table>
This Lab demonstrates various features around security such as rights, roles and users. In general, users belong to roles and these roles have associated rights. Each of the four security methods deals with roles and rights in a slightly different manner. We will leverage Standard Security in this Lab. Here roles and rights are determined through a set of queries.

Though some implementations of security can omit roles, rights are required in any security method. Rights can be applied at three different levels:

- **Page** – Can the user see this report or not?
- **Element** – Can this user see the salary or not?
- **Record** – Can the user see orders from this customer or not?

To expedite the process, a special Sql Server database called the Lab Security Database is used for pre-stored users roles and rights. Before continuing this lab, this database must be attached through a standard connection to Sql Server. Before this connection is created however, the MDF files provided must be attached to an instance of Sql Server. For simplicity it is recommended that the same instance that contains Northwind be used to attach the security database.

The Lab Security Database can be downloaded from Devnet:


It is a zip file that is extracted in the Program Files directory that currently holds your existing database files (i.e. C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data)

**Directions:**

1. Add the Lab Security Database Database
   - Open the _Settings Definition File
   - Select the Connections element then add a Connection.SqlServer element from the Element Toolbox.
   - Set the attribute values as follows (you may have to change the server name and credentials)
Once the database is connected – security elements can be added to the _Settings definition. Then the security will be leveraged against an actual report definition.

2. Add Security Elements to the _Settings definition.

   - Open the _Settings definition.
     
     Select the top most element General element.

   - Add a Security Element from the Element Toolbox.

   - Update the Authentication Source to Standard.

Set the Cache Roles and Rights as Session.
Note: The Security Enabled attribute blank by default, but make sure to set to “True” before testing.

An authentication element is then added to allow various data sources to be used for authentication. A username and Password must be contained in this data layer. The following data layers are available as children to the Authentication element:

3. Add an Authentication Element

- Select the Security element.
- Add and Authentication element from the Element Toolbox with an ID of Standard_Security_Authentication.
- Add a DataLayer.SQL element from the Element Toolbox and enter dlAuthentication as the ID.
- Choose connSecurity as the Connection ID.
- Enter the following SQL in the Source attribute (cut and paste ok)
  ```sql
  Select lgxUser.USER_NAME, lgxUser.USER_ID
  From lgxUser
  Where lgxUser.USER_NAME = '@Request.rdUsername~'
  And lgxUser.PASSWORD = '@Request.rdPassword~'
  ```

Note: If any records are returned by this query then the user is authorized and the application loads. For XML files, Static Data Layers and Google Spreadsheets, condition filters will be needed to check against “@Request.rdUsername~” and “@Request.rdPassword~”.

User Roles elements work like the Authentication element in that they use various datalayers as children to determine the roles in the application:

4. Add a User Roles element

- Select the Security element.
- Add a UserRoles element from the Element Toolbox. Enter an ID of Standard_Security_Roles.
• Add a DataLayer.Sql with an id of dlRoles

• Enter the following SQL in the Source attribute (cut and paste ok)

```sql
Select Distinct lgxRole.ROLE
From lgxUserRole Inner Join
    lgxUser On lgxUserRole.User_ID = lgxUser.User_ID Inner Join
    lgxRole On lgxUserRole.Role_ID = lgxRole.ROLE_ID
Where lgxUser.User_Name = '@Function.Username~'
```

This will return one or several roles “records” for the supplied username. The system will also accept several roles in one column (separated by commas) in one record returned by a similar query. If no records are returned, then the user will only see reports that have limited or no security applied.

At this point a user has been assigned one or several roles. Roles have one or several rights. The next step is to find the unique set of rights associated with these roles.

5. Add **User Rights** elements.

• Select the Security element.

• Add a **User Rights** element from the Element Toolbox.

Once a **User Rights** element is created. Several options are available to determine rights:

<table>
<thead>
<tr>
<th>Right from Roles:</th>
<th>One and only one right exists for each role. One <strong>Rights from Roles</strong> element is added with no attributes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rights from DataLayer:</td>
<td>Rights are pulled directly from a datalayer using &quot;[@Request.UserName~&quot; or some other @ token (i.e. @Session.UserID~ or @Session.UserName~) to return a single column of one or many rights. Only one “Rights from DataLayer” element is needed.</td>
</tr>
<tr>
<td>Rights from Role:</td>
<td>Each right has an element. Each right element has a datalayer that can return a right based on a query. This query will use a static piece of text in its where clause (or condition filter) that relates directly to a role. This text is usually the name of the role. If a user has that role then they have that</td>
</tr>
</tbody>
</table>
Here Rights from Role will be utilized.

- Add a RightFromRole element to the User Rights element with an ID of ViewDataTable
- Add a Datalayer.Sql element from the Element Toolbox with an ID of ViewDataTable.
  - Add the following SQL to the Source attribute
    
    ```sql
    Select Distinct lgxRole.ROLE
    From lgxRole Inner Join lgxRoleRights On lgxRole.ROLE_ID = lgxRoleRights.ROLE_ID
    Inner Join lgxRights On lgxRights.RIGHT_ID = lgxRoleRights.RIGHT_ID
    Where RIGHT_NAME = 'ViewDataTable'
    ```

- Add a second RightFromRole element to the User Rights element with an ID of ViewFreight
- Add a Datalayer.Sql element from the Element Toolbox with an ID of ViewFreight.
  - Add the following SQL to the Source attribute
    
    ```sql
    Select Distinct lgxRole.ROLE
    From lgxRole Inner Join lgxRoleRights On lgxRole.ROLE_ID = lgxRoleRights.ROLE_ID
    Inner Join lgxRights On lgxRights.RIGHT_ID = lgxRoleRights.RIGHT_ID
    Where RIGHT_NAME = 'ViewFreight'
    ```

- Add a third RightFromRole element to the User Rights element with an ID of ViewUSAOrders
- Add a Datalayer.Sql element from the Element Toolbox with an ID of ViewUSAOrders.
  - Add the following SQL to the Source attribute
    
    ```sql
    Select Distinct lgxRole.ROLE
    From lgxRole Inner Join lgxRoleRights On lgxRole.ROLE_ID = lgxRoleRights.ROLE_ID
    Inner Join lgxRights On lgxRights.RIGHT_ID = lgxRoleRights.RIGHT_ID
    Where RIGHT_NAME = 'ViewUSAOrders'
    ```
Now that **User Rights** elements are established, review the rights and role capabilities. Note that the relationship to the users and their roles was pre-established in the **Lab Security Database**.

<table>
<thead>
<tr>
<th>User</th>
<th>Password</th>
<th>Role</th>
<th>Rights given to this role.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Password</td>
<td>User_Group_A</td>
<td>ViewDataTable</td>
</tr>
<tr>
<td>Jim</td>
<td>Password</td>
<td>User_Group_B</td>
<td>ViewDataTable,ViewFreight</td>
</tr>
<tr>
<td>Chris</td>
<td>Password</td>
<td>User_Group_C</td>
<td>ViewDataTable,ViewUSAOrders</td>
</tr>
</tbody>
</table>

- Bob will be able to view non-USA orders. The Freight column will be hidden.
- Jim will be able to view non-USA orders. The Freight column will be visible.
- Chris will be able to view All records but not the Freight column.

This completes the configuration changes required in the settings definition. At this point the definitions that are to be secured are changed. Definitions that don’t have any security rights assigned to them in their respective definitions will be unaffected and can be loaded by anyone who is authenticated.

6. Now augment the a copy of **Lab1_Basic_DataTable**

   - Select the Reports Folder in the Application Panel.
   - Right-click then Add then New Folder
   - Rename the folder to **Security**
   - Select **Lab1_Basic_DataTable** definition from the DataTables folder
   - Right-click then **Copy**
   - Select the Security folder the right-click **Paste**.
   - Rename the definition to **Lab1_Security_Basics**
7. Add the security information to the Report Header

- Select the Report Header in Lab1_Security_Basics
- Add a Label element from the Element Toolbox.
  - Enter @Function.UserName~ for the Caption.
  - Enter lblUserName for the ID.
- Add a New Line element then a Label element from the Element Toolbox.
  - Enter @Function.UserRole~ for the Caption.
  - Enter lblUserRoles for the ID.
- Add a final New Line element then a Label element from the Element Toolbox.
  - Enter @Function.UserRights~ for the Caption.
  - Enter lblUserRights for the ID.

8. Apply the Security Rights IDs

- Select the topmost element in Lab1_Security_Basics
- Enter ViewDataTable for the Security Report Right ID attribute.
- Select colFreight and enter ViewFreight for the Security Right ID attribute.
- Add a Security Filter element to the DataLayer dlOrders with an ID of sfUSA_Orders_Only.
- Enter in the following for the Condition attribute (including the quotes) “@Data.ShipCountry~” = “USA”
- Set the Security Filter Right ID to ViewUSAOrders
- Add a second Security filter to the Data Layer dlOrders with an ID of sfNonUSA_Orders to handle the other users.
- Enter in the following for the Condition attribute “@Data.ShipCountry~” <> “USA”
- Set the Security Filter Right ID to ViewDataTable
Note: Security filters limit the rows to a condition. Assuming that all security filters in place return some records, only users with rights associated with a security filter will see any records. For example, if a user has rights to a data table but not to any security filter he/she will not see any records even if there are records that don’t match any condition of any existing security filters.

9. Turn Security On

- Select the _Settings definition in the Application Panel
- Select the Security element and set Security Enabled to True.
- Set Lab1_Security_Basics as the default definition – right-click the definition in the Application Panel and choose “Set as Default”

10. Test the Security

- Run the application (Green start button or F5)
- Log in as Chris – You should see the data table but not USA orders or the Freight column.
- Close the browser and re-run the application
- Log in as Jim – You should see the data table and the Freight column but not the USA orders
- Close the browser and re-run the application
- Log in as Bob – You should see the data table and the USA orders but not the Freight column.
Lab 1a – Security Basics (SecureKey)

Objectives:

- Demonstrate SecureKey authentication in the same application using the same Lab1_Security_Basics definition.
- Introduce a separate C# application to be used as a Single Sign On (SSO) surrogate.

This adaptation of the Lab1_Security_Basics demonstrates the use of SecureKey Authentication. Using Securekey we make use of a separate “surrogate” single sign on (SSO) application written in C#. This C# application will communicate with this application and conduct a dialog wherein the rights and roles can be defined and initialized directly from the SSO.

The application is downloaded from the same Training Materials Site that this PDF and the pre-constructed Labs are located. Follow the instructions in the word document provided.


The same rights that were used in the standard security part of this lab will be used in this SecureKey demonstration for continuity. An additional Security element will be added to the settings.

In any SSO implementation, the identity, rights and roles are determined prior to any Logi Report being rendered by the SSO application itself. The username, rights and roles can then be sent to the Logi Server during this dialog. In this example, the SSO will be sending only rights.

Make sure that the Security Enabled Attribute is set to True when you are ready to test the SSO application. Once this attribute is set for SecureKey you will not be able to access the application except through the SSO application.
1. Make the appropriate changes to the Settings Definition.

   - Open the Settings Definition and select the top settings element.
   - Right click on the existing security element and choose Remark (see screen below). This will disable the existing standard security authentication method. If you want to switch back then repeat the process to un-remark.

   ![Diagram of Settings Definition]

   - Add a Security element from the Element Toolbox.

   ![Diagram of Element Toolbox]

   - Enter in the following attribute values:

     ![Attribute Table]

     - Required Attributes
       - Authentication Source
       - SecureKey
     - Optional Attributes
       - Access Denied Page
       - Authentication Client Addresses: 127.0.0.1
       - Cache Rules and Rights
       - Development Client Addresses
       - Development Username
       - Login Fail Page
       - Login Page
       - NT Authentication Domain
       - Restart Session
       - Scheduler Username
       - Security Enabled: True
• Set the security enabled attribute just before testing.

• Open The Single Sign On app that was downloaded from the Class Materials site in Visual Studio 8. Installation instructions are found on Devnet:


• Play the application and pick a combination of rights from the drop down menu then submit. The SSO application is designed to communicate directly to this Logi application. Note that there may be a break point in the SSO. You can press F10 to proceed.

The rights used in the SSO and Training Lab Definitions are listed as follows:

<table>
<thead>
<tr>
<th>Rights</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewDataTable</td>
<td>This will allow the user to see the data table in the Lab1_Security_Basics definition. Non USA orders will be shown. The Freight column is not visible.</td>
</tr>
<tr>
<td>ViewUSAOrders</td>
<td>This will allow the user to see the data table in the Lab1_Security_Basics definition. USA orders will be shown. The Freight column is not visible.</td>
</tr>
<tr>
<td>ViewDataTable, ViewFreight</td>
<td>This will allow the user to see the data table in the Lab1_Security_Basics definition. Non USA will be shown. The Freight column should be visible.</td>
</tr>
<tr>
<td>ViewUSAOrders, ViewFreight</td>
<td>This will allow the user to see the data table in the Lab1_Security_Basics definition. USA orders will be shown. The Freight column should be visible.</td>
</tr>
</tbody>
</table>
There is a built in break point so that you may step through the SSO. Press F10 to advance to the next step:

```csharp
protected void btLogIn_Click(object sender, EventArgs e)
{
    string strRedirect = "http://localhost/Logi_Training_Labs_V11/rdTemplate/rdGetSecureKey.aspx";
    string strGetKeyURL = "/rdTemplate/rdGetSecureKey.aspx?UserName=" + txtUserName.Text + "&Rights=
    string strFinalURL = strRedirect + strGetKeyURL;
    HttpRequest webRequest = (HttpRequest)WebRequest.Create(strFinalURL);
    HttpResponse webResponse;
    string strRedirectURL = null;
    try
    {
        webResponse = (HttpResponse)webRequest.GetResponse();
        StreamReader myStream;
        StreamReader ar = new StreamReader(myStream);
        string key = ar.ReadToEnd();
    }
}
```

Note that there are two passes. The first pass constructs a URL that will be sent to the web server in order to get a secure “key”. The url in this case is

http://localhost/Logi_Training_Labs_V11/rdTemplate/rdGetSecureKey.aspx?UserName=Dave&Rights=ViewUSAOrders,ViewFreight&Roles=&ClientBrowserAddress=127.0.0.1

In this case the roles were omitted for simplicity. The roles can be populated as well in this manner.
The results from a successful first pass is a string that is completely unique to this user's username, roles, rights and machine IP address. This key cannot be re-used in another browser on another machine. Once the browser is closed this key is no longer valid since it is also tied to the browser section.
The key is used to construct a second URL in a second pass that will contain the key and the report that is to be run from the application:

```
protected void btnLogin_Click(object sender, EventArgs e)
{
    string strRedirect = "http://localhost/Logi_Training_Labs_V11";
    string sGetKeyURL = "/rdTemplate/rdGetSecureKey.aspx?UserName=" + txtUserName.Text + "&Rights=" +
    string sFinalURL = strRedirect + sGetKeyURL;
    HttpWebRequest webRequest = (HttpWebRequest)HttpWebRequest.Create(sFinalURL);
    HttpWebResponse webResponse;
    string strRedirectURL = null;
    try
    {
        webResponse = (HttpWebResponse)webRequest.GetResponse();
        Stream myStream;
        myStream = webResponse.GetResponseStream();
        StreamReader sr = new StreamReader(myStream);
        string sKey = sr.ReadToEnd();
    }
}
```

The key is used to construct a second URL in a **second** pass that will contain the key and the report that is to be run from the application:

This second pass is made to launch the actual report to be rendered. This pass includes only the key and the desired report:

```
http://localhost/Logi_Training_Labs_V11/rdpage.aspx?rdSecureKey=a9dcb79b6ef742a6a9d7726dd09159d9
&rdReport=Security.Lab1_Security_Basics
```

In this way the users name, roles, rights etc. are hidden from the url.

- Continue to use F10 to step through (or press F5 to continue without stepping through).
Once a string containing the URL is constructed the SSO simply "redirects" to the second pass’s URL.

Close the SSO between each successive iteration. You should run the SSO for each group of rights.

The SecureKey approach does not directly require the use of additional roles and rights elements in the SecureKey element. If rights are used only (no roles), the rights can be directly typed into the Security Right ID attribute values of the appropriate elements. No rights need be defined in the Security element in settings.
XOLAP

Lab 1 – Simple Dimensions

Objectives:

- Create a query against Northwind to return un-aggregated data for one measure and three dimensions.
- Create a dimension grid with the three dimensions and single measure.

### Definition:

#### Report Output
Directions:

1. Create a new folder under the Reports folder called XOLAP.
2. Add a new definition called Lab1_Basic_Dimension_Grid.
3. Add a Dimension Grid to the body with an ID of dgOrderAnalysis.
4. Add a XOLapCube element to dgOrderAnalysis with and ID of XOlapCube_Order_Analysis.
5. Add a data layer named dlOrderAnalysis to XOlapCube_Order_Analysis.
6. Create a query against Northwind that has the following tables and columns. Do not aggregate. This query will be suitable for the next 2 labs so you will not have to change it from copy to copy.

```sql
FROM Employees INNER JOIN Orders ON Employees.EmployeeID = Orders.EmployeeID
INNER JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID
INNER JOIN Products ON Products.ProductID = [Order Details].ProductID
INNER JOIN Categories ON Categories.CategoryID = Products.CategoryID
INNER JOIN Customers ON Customers.CustomerID = Orders.CustomerID
```

2. Add a Time Period Column to the data layer dlOrderAnalysis called OrderYear. Set the Data Column to OrderDate and choose Year for the time period.

3. Add A XOLAP Dimensions element to the XOlapCube_Order_Analysis element.
• Add three dimensions in the form of XOLAP Dimension Elements.
  o dimTime – dimension name: Time
  o dimEmployees – dimension name: Employees
  o dimCategories – dimension name: Categories

• Each dimension must have at least one level. Add a level to the three dimensions.
  o dimTime
    • xOrderYear
      • Data Column – OrderYear
      • Level Name – lvlYear
  ii. dimEmployees
    • xLName
      • Data Column – LastName
      • Level Name – lvlLastName

    • xCategory
      • Data Column – CategoryName
      • Level Name – lvlCategory

4. Add a XOLAP Measures Element to XOlapCube_Order_Analysis

b. Add a XOLAP Measure Element with and ID of sumOrderTotal

  o Data Column – Subtotal
  o Measure Name – Order Totals
  o Format – Currency
  o Function - Sum