Creating logical SAP BusinessObjects Web Intelligence reports the Vulcan way

Alan Mayer
Solid Ground Technologies
In This Session

- Learn how to create reports that are easier to understand and maintain
- Understand the need for more advanced techniques
- Discover how to program using multi-layered logic
- Fine-tune the control of your calculations using contexts
- Interface your reports to other reports and applications
- Standardize the development of future reports
What We’ll Cover

- Introduction
- Annotated Report Standards
- Calculation Contexts
- Layered Variables
- Interface Points
- Wrap up
Life as a Vulcan

- The one species where logic means everything
- Emotions are tightly reigned in
- Allows data to drive their decisions rather than belief
- Couldn’t choose a better role model as an architect or technician
Life as We Practice It

- Pretending to be logical
- Reliving the same problems by trying the same solution
A Vulcanized Approach to Web Intelligence

- Let’s apply the data-driven logical Vulcan viewpoint to our efforts
- We’ll visit some of the most common trouble spots
  - Fully annotating reports
  - Future-proofing calculations by fully defining their contexts
  - Creating flawless variables that are maintainable
  - Launching your reports from within and outside SAP BusinessObjects
What We’ll Cover

• Introduction
• Annotated Report Standards
• Calculation Contexts
• Layered Variables
• Interface Points
• Wrap up
From the Book of Vulcan

- Discipline goes beyond action.
- Structure your thoughts by giving each a name and purpose for being.
Naming Report Elements

- Everything you create should have a name

**Queries**

**Blocks**

**Cells**

**Variables**
The Reason for Names

- **Placement**
  - Helps position blocks and cells relative to one another

- **Sorting**
  - Allows variables sorted by query rather than alphabetically
Naming Variables

• Consider the following rules for variable names
  ♦ ALWAYS add a prefix to a report variable’s name
• Different prefixes give you more control
  ♦ v_ generic prefix for any variable name
  ♦ c_ add to constants
  ♦ p_ add to prompts
Reasons for Variable Names

- Easier maintenance
- Can manipulate the report without refreshing
- Examples:
  - v_PaymentDue
  - p_Year
  - c_BucketSize

Names are ESSENTIAL for developing layered variables (coming up!)
Document Descriptions

- Add descriptions and keywords wherever possible

Good descriptions will allow BI Platform Search to locate this document quicker.
Title Pages

- Add a report page as an introduction
  - A cell formatted to wrap text contains your document description
  - An empty two-column table contains your prompts
  - Position the prompt table relative to the description cell

```
This is a sample document created for the presentation "Web Intelligence reports the Vulcan way". It demonstrates the importance of annotating report elements when building a report. The same concepts can be used to retrofit an existing report.
```

<table>
<thead>
<tr>
<th>Prompt</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>FY2006</td>
</tr>
<tr>
<td>Month</td>
<td>Aug</td>
</tr>
</tbody>
</table>

\[
p_{\text{Year}} = \text{UserResponse("Year")} \]

\[
p_{\text{Month}} = \text{UserResponse("Month")}
\]
Change Logs

- Document who modified the report and when
- This could be added to your title page
  - Include another 3 to 4 column table
    - Include document version, date, author, description

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>12-1-2013</td>
<td>Alan Mayer</td>
<td>Created initial document</td>
</tr>
<tr>
<td>1.1</td>
<td>12-3-2013</td>
<td>Alan Mayer</td>
<td>Revised format based on Vulcan guidelines</td>
</tr>
</tbody>
</table>
Prompts

- Embed prompts as repeated headers
  - Create a variable that will contain prompt values
  - Repeat that cell on every page
    - Format Cell > Layout > Repeat on every page

```
= "Resort Revenue for " + [p_Year] + " " + [p_Month]
```
Demonstration - Annotations
What We’ll Cover

• Introduction
• Annotated Report Standards
• Calculation Contexts
• Layered Variables
• Interface Points
• Wrap up
From the Book of Vulcan

- Precise logical control allows us to rise above chaos
What is a Calculation Context?

- Represents the environment in which a formula is calculated
- Similar to the English definition
  - Circumstances around an event, saying or idea
- Example:
  - Spock:
    - “I would advise no action at this time”
  - Spock in context:
    - “I would advise no action at this time with the 10 Klingon ships that surround us”
Why are Calculation Contexts Important?

- If you do not define the context of your formulas, BusinessObjects will

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Month</th>
<th>Sales revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>Q1</td>
<td>1</td>
<td>$1,003,541</td>
</tr>
<tr>
<td>2004</td>
<td>Q1</td>
<td>2</td>
<td>$630,073</td>
</tr>
<tr>
<td>2004</td>
<td>Q1</td>
<td>3</td>
<td>$1,027,085</td>
</tr>
<tr>
<td>2004</td>
<td>Q2</td>
<td>4</td>
<td>$895,260</td>
</tr>
<tr>
<td>2004</td>
<td>Q2</td>
<td>5</td>
<td>$865,615</td>
</tr>
<tr>
<td>2004</td>
<td>Q2</td>
<td>6</td>
<td>$517,819</td>
</tr>
<tr>
<td>2004</td>
<td>Q3</td>
<td>7</td>
<td>$525,904</td>
</tr>
<tr>
<td>2004</td>
<td>Q3</td>
<td>8</td>
<td>$173,756</td>
</tr>
<tr>
<td>2004</td>
<td>Q3</td>
<td>9</td>
<td>$668,181</td>
</tr>
<tr>
<td>2004</td>
<td>Q4</td>
<td>10</td>
<td>$655,206</td>
</tr>
<tr>
<td>2004</td>
<td>Q4</td>
<td>11</td>
<td>$484,024</td>
</tr>
<tr>
<td>2004</td>
<td>Q4</td>
<td>12</td>
<td>$649,350</td>
</tr>
</tbody>
</table>

Sum: $8,095,814

$36,387,203

Calculation engine differences between versions make this painfully clear!
Calculation Context Syntax

- Examine the following formula:

  \[
  \text{sum([Sales revenue] In Body) In Report}
  \]

  - Input Context
  - Output Context

- Every formula has an Input and Output Context
- Each of these contexts will be fully explored in the slides that follow
Calculation Context Keywords

1. Report
   - Everything on a report page

2. Block
   - Everything in a table or chart

3. Body
   - Everything on a row
   - CurrentPage
     - All content on the current report page
Calculation Context Keywords, cont’d

4 Section
  • Everything within a report section

5 Break
  • Everything inside a report break

<table>
<thead>
<tr>
<th>Quarter</th>
<th>State</th>
<th>Sales revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>California</td>
<td>$519,220</td>
</tr>
<tr>
<td></td>
<td>Colorado</td>
<td>$131,797</td>
</tr>
<tr>
<td></td>
<td>DC</td>
<td>$208,324</td>
</tr>
<tr>
<td></td>
<td>Florida</td>
<td>$137,530</td>
</tr>
<tr>
<td></td>
<td>Illinois</td>
<td>$256,454</td>
</tr>
<tr>
<td></td>
<td>Massachuse</td>
<td>$92,596</td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td>$555,983</td>
</tr>
<tr>
<td></td>
<td>Texas</td>
<td>$758,796</td>
</tr>
<tr>
<td>Q1</td>
<td>Sum:</td>
<td>$2,660,700</td>
</tr>
</tbody>
</table>
Input Context

- **Input Context follows these rules:**
  - It’s always found INSIDE the parens of the aggregating function
  - It tells WHAT to calculate the sum of
  - In other words, it determines the level of aggregation
- **The BODY keyword is used to include all dimensions on the row**
  - Individual dimensions could be listed in parens instead
    - **Example:** In ([Year];[Quarter])

- **Example:** `sum([Sales revenue] In Body) In Report`
Using ForEach( )

- The ForEach operator *adds* dimensions to the default context
  - The following are the same for the block below:
    - `Sum([Sales Revenue] In Body)`
    - `Sum([Sales Revenue] In ([Year];[Quarter])`
    - `Sum([Sales Revenue] ForEach ([Quarter]) )`

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Sales revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>Q1</td>
<td>$2,660,700</td>
</tr>
<tr>
<td>2004</td>
<td>Q2</td>
<td>$2,278,693</td>
</tr>
<tr>
<td>2004</td>
<td>Q3</td>
<td>$1,367,841</td>
</tr>
<tr>
<td>2004</td>
<td>Q4</td>
<td>$1,788,580</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$8,095,814</td>
</tr>
</tbody>
</table>

*Tip*

ForEach is often used for contexts with a long list of dimensions. IN would work as well but all dimensions would have to be specified.
Using ForAll()

• The ForAll operator *removes* dimensions from the default context
  • The following are the same for the block below:
    ▶ Sum([Sales Revenue] In ([Year])
    ▶ Sum([Sales Revenue] ForAll ([Quarter]))

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Sales revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>Q1</td>
<td>$2,660,700</td>
</tr>
<tr>
<td>2004</td>
<td>Q2</td>
<td>$2,278,693</td>
</tr>
<tr>
<td>2004</td>
<td>Q3</td>
<td>$1,367,841</td>
</tr>
<tr>
<td>2004</td>
<td>Q4</td>
<td>$1,788,580</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$8,095,814</td>
</tr>
</tbody>
</table>

ForAll works but is not used as often. When in doubt, the IN operator will always work!
Visualizing Input Contexts

- Draw an imaginary table with all dimensions specified by the input context
- Add your formula as the last column to that table
- The value of that formula per row is WHAT you are aggregating!

\[ \text{sum(<Sales revenue> In ([Year]; [Quarter]))} \]

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Sales revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>Q1</td>
<td>$2,660,700</td>
</tr>
<tr>
<td>2004</td>
<td>Q2</td>
<td>$2,278,693</td>
</tr>
<tr>
<td>2004</td>
<td>Q3</td>
<td>$1,367,841</td>
</tr>
<tr>
<td>2004</td>
<td>Q4</td>
<td>$1,788,580</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$8,095,814</td>
</tr>
</tbody>
</table>
Output Context follows these rules:

- It’s always found OUTSIDE the parens of the aggregating function
- It tells HOW MANY rows to consider

In the above example, consider all rows in the report
- Individual dimensions could be listed in parens instead

Example: In ([Year])
Output Contexts using Where( )

- The Where( ) operator allows a calculated range of rows
  - Add a simple filter:
    - $\text{Sum}([\text{Sales Revenue}]) \text{ Where } ([\text{Month}] \leq 6)$
    - $\text{Sum}([\text{Sales Revenue}]) \text{ Where } ([\text{Month}] \leq 6 \text{ and } [\text{Year}] = \text{"FY2004"})$

<table>
<thead>
<tr>
<th>Year</th>
<th>Month</th>
<th>Monthly Revenue</th>
<th>Half Year Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>1</td>
<td>$1,003,541</td>
<td>4,939,392.9</td>
</tr>
<tr>
<td>2004</td>
<td>2</td>
<td>$630,073</td>
<td>4,939,392.9</td>
</tr>
<tr>
<td>2004</td>
<td>3</td>
<td>$1,027,085</td>
<td>4,939,392.9</td>
</tr>
</tbody>
</table>

More complicated conditions can be calculated using variables within the Where() operator

Tip:
Visualizing Output Contexts

- Revisit that imaginary table draw from the input context
- Output contexts determine how many rows to aggregate over

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Month</th>
<th>Sales revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>Q1</td>
<td>1</td>
<td>$1,003,541</td>
</tr>
<tr>
<td>2004</td>
<td>Q1</td>
<td>2</td>
<td>$630,073</td>
</tr>
<tr>
<td>2004</td>
<td>Q1</td>
<td>3</td>
<td>$1,027,085</td>
</tr>
<tr>
<td>2004</td>
<td>Q2</td>
<td>4</td>
<td>$895,260</td>
</tr>
<tr>
<td>2004</td>
<td>Q2</td>
<td>5</td>
<td>$865,615</td>
</tr>
<tr>
<td>2004</td>
<td>Q2</td>
<td>6</td>
<td>$517,819</td>
</tr>
<tr>
<td>2004</td>
<td>Q3</td>
<td>7</td>
<td>$525,904</td>
</tr>
<tr>
<td>2004</td>
<td>Q3</td>
<td>8</td>
<td>$173,756</td>
</tr>
<tr>
<td>2004</td>
<td>Q3</td>
<td>9</td>
<td>$668,181</td>
</tr>
<tr>
<td>2004</td>
<td>Q4</td>
<td>10</td>
<td>$655,206</td>
</tr>
<tr>
<td>2004</td>
<td>Q4</td>
<td>11</td>
<td>$484,024</td>
</tr>
<tr>
<td>2004</td>
<td>Q4</td>
<td>12</td>
<td>$649,350</td>
</tr>
</tbody>
</table>

Sum revenue by:
- Row
- Quarter
- Year

Key Feature

Very common way to represent subtotals
Visualizing Input and Output Contexts

- Using both completely cover the entire report

**INPUT Context**
WHAT to aggregate

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Month</th>
<th>Sales revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>Q1</td>
<td>1</td>
<td>$1,003,541</td>
</tr>
<tr>
<td>2004</td>
<td>Q1</td>
<td>2</td>
<td>$630,073</td>
</tr>
<tr>
<td>2004</td>
<td>Q1</td>
<td>3</td>
<td>$1,027,085</td>
</tr>
<tr>
<td>2004</td>
<td>Q2</td>
<td>4</td>
<td>$895,260</td>
</tr>
<tr>
<td>2004</td>
<td>Q2</td>
<td>5</td>
<td>$865,615</td>
</tr>
<tr>
<td>2004</td>
<td>Q2</td>
<td>6</td>
<td>$517,819</td>
</tr>
<tr>
<td>2004</td>
<td>Q3</td>
<td>7</td>
<td>$525,904</td>
</tr>
<tr>
<td>2004</td>
<td>Q3</td>
<td>8</td>
<td>$173,756</td>
</tr>
<tr>
<td>2004</td>
<td>Q3</td>
<td>9</td>
<td>$668,181</td>
</tr>
<tr>
<td>2004</td>
<td>Q4</td>
<td>10</td>
<td>$655,206</td>
</tr>
<tr>
<td>2004</td>
<td>Q4</td>
<td>11</td>
<td>$484,024</td>
</tr>
<tr>
<td>2004</td>
<td>Q4</td>
<td>12</td>
<td>$649,350</td>
</tr>
</tbody>
</table>

**OUTPUT Context**
Over HOW MANY ROWS

Key Feature
This one visualization will serve as a trusted compass whenever contexts are concerned.
Demonstration – Calculation Contexts
What We’ll Cover

• Introduction
• Annotated Report Standards
• Calculation Contexts
  • Layered Variables
• Interface Points
• Wrap up
From the Book of Vulcan

- Irrational fear is the hallmark of an undisciplined intellect
- Construct your understanding layer by layer in order to achieve the desired result.
Layered Variables

- Concept that works well for complex logic
- Break up the formula into simpler pieces ...
- ... then use those pieces to build the final variable
- Let’s look at an example using sample data
  - Island Resorts Marketing
  - Group invoices based on date ranges
  - Very similar to aging
Creating Constants

- Programming constants makes the report more maintainable
  - Base date is set to some date
    - Usually the current date for most companies
  - Buckets defined by number of days within that range
  - Each bucket begins where last bucket ends

<table>
<thead>
<tr>
<th>Variable</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_BaseDate</td>
<td>=ToDate(“2006-01-01”,”yyy-MM-dd”)</td>
</tr>
<tr>
<td>c_Bucket1</td>
<td>=30</td>
</tr>
<tr>
<td>c_Bucket2</td>
<td>=30</td>
</tr>
<tr>
<td>c_Bucket3</td>
<td>=30</td>
</tr>
</tbody>
</table>

Tip: Constants could be programmed with prompts for even more interactivity.
Define Bucket Start and End Dates (Layer 1)

- Dates for Bucket 1 are shown below
- Buckets 2 – 3 and the Remainder bucket would be defined the same way
- Note how these formulas use the constants previously defined

<table>
<thead>
<tr>
<th>Variable</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>v_Bucket1_Begin</td>
<td>=c_BaseDate</td>
</tr>
<tr>
<td>v_Bucket1_End</td>
<td>=RelativeDate([v_Bucket1_Begin];[c_Bucket1] – 1)</td>
</tr>
</tbody>
</table>

The Remainder bucket picks up all invoices after Bucket3 through the end of the year
Define Indicators (Layer 2)

- Indicators determine what bucket an invoice belongs in
- The Indicator for Bucket 1 is shown below

<table>
<thead>
<tr>
<th>Variable</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>v_Bucket1_Indicator</td>
<td>=If [Invoice Date] Between([v_Bucket1_Begin];[v_Bucket1_End]) Then 1 Else 0</td>
</tr>
</tbody>
</table>

The formula almost reads like a sentence. Easier to edit and maintain.
Create the Buckets (Layer 3)

- Create the Bucket Totals as the last step
- The formula uses Indicators to make the decision easier

<table>
<thead>
<tr>
<th>Variable</th>
<th>Formula</th>
</tr>
</thead>
</table>
| v_Bucket1_Total  | =Sum(
|                 |     If [v_Bucket1_Indicator] = 1
|                 |     Then [Revenue]
|                 |     Else 0)                                          |
Alternatives to layered logic

- **Complicated logic can be pushed back to the universe**
  - Best practice where possible
  - Many reports can benefit from the same central definitions
  - Can push back even further to HANA models / views

- **Trading database performance for centralized definitions**
  - This may not be acceptable for certain projects
Demonstration – Layered Variables
What We’ll Cover

- Introduction
- Annotated Report Standards
- Calculation Contexts
- Layered Variables
- Interface Points
- Wrap up
From the Book of Vulcan

- Seek multiple ways to the same end
- Reuse that which is valuable, for its value is measured by that use
Invoking Web Intelligence Reports

• We’ll investigate three methods
  • From another BusinessObjects report
  • From within a BI Workspace
  • From another application

FYI

BI Workspaces can consume portions of Webi reports
Creating Document Links

- Documents can be linked to other via prompts
  - Links are created in the source document
  - Those links answer prompts from a second document
Examine the Source Document

- Decide which area should be dynamic
- Cells around that area become the “context”

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>e-Fashion Austin</td>
<td>$314,430</td>
<td>$273,608</td>
<td>$294,798</td>
<td>$252,644</td>
</tr>
<tr>
<td>e-Fashion Boston Newbury</td>
<td>$220,301</td>
<td>$220,528</td>
<td>$237,464</td>
<td>$208,877</td>
</tr>
</tbody>
</table>

**Context:**
- Year = 2006
- Quarter = Q2
- Store = e-Fashion Austin
Create the Destination Document

- Add prompts based on the context previously described
- Any additional detail can be added
  - Format can be graphical or tabular
Link Source to Destination

- **Right-click on the selected area in the Source document**
  - Linking > Add document link

1. **Locate the document to link to**
2. **Refresh that document**
3. **Map prompts from that document to objects in this one**
4. **Choose a new or existing window to display**
Demonstration – Document Links
Working With BI Workspaces

1. Start the Editor
2. Choose your content as modules
3. Add those modules here
4. Link those modules together
Adding Content

- Modules control content
  - Many options to choose from
  - We’ll focus on Public reports
Arranging Content

Documents are dragged to the workspace from the Module Library.
Selecting Report Pieces

Piece 1

![Table showing sales data for e-Fashion Austin and e-Fashion Boston Newbury in Q2 2006.](image)

Resize each module and select only the pieces you need.

Piece 2

![Table showing sales revenue, quantity sold, and discount for different years and weeks.](image)

Resize each module and select only the pieces you need.
Link the Pieces

- Use Content Linking to define how one module will drive the others

1. Create a connection between modules by dragging

2. Map parameters (prompts) between modules

<table>
<thead>
<tr>
<th>Source Parameter</th>
<th>Target Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store name</td>
<td>Click here to select the Target parameter.</td>
</tr>
<tr>
<td>Type  STRING</td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>Click here to select the Target parameter.</td>
</tr>
<tr>
<td>Type  STRING</td>
<td></td>
</tr>
<tr>
<td>Quarter</td>
<td>Click here to select the Target parameter.</td>
</tr>
<tr>
<td>Type  STRING</td>
<td></td>
</tr>
</tbody>
</table>
Demonstration – BI Workspaces
Launching Reports From Other Programs

- **What about integrating Webi content from existing programs?**
  - Other than BI launch pad, Infoview, …
- **Common request for custom portals**
  - An HTML application developed by another company
  - The initial user interface is handled by that application
  - All prompt values for the report are gathered
  - Webi reports are invoked through OpenDocument commands

---

**Heads-Up**

A BusinessObjects session must be previously established before attempting this!
Creating the OpenDocument Command

- The OpenDocument command looks like a hyperlink
  - What appears below is a sample
  - Argument values in red

iDocID=AajYahfR9Z9Gh_BSbOiqNZM
&sIDType=CUID
&sType=wid
&sRefresh=Y
&lsMYear=2006
Document Arguments

- **iDocID** – Uniquely identifies the document
  - Example: iDocD=AajYahfR9Z9Gh_BSbOiqNZM
- **sDocName** – Document name (may not be unique)
  - Example: sDocName=2006+Sales
- **sIDType** – The type of ID supplied (CUID, InfoObjectID)
  - Example: sIDType=CUID
- **sType** – The type of document (wid = Webi)
  - Example: sType=wid
- **sInstance** – Opens the latest instance (User, Last, Param)
  - Example: sInstance=Last
- **sRefresh** – Refresh the document before viewing (Y,N)
  - Example: sRefresh=Y
Prompt Arguments

- **lsC – Selects a universe context**
  - Example: lsC=Rentals
- **lsS<prompt> - Single value for a prompt**
  - Example: lsSYear=2006
- **lsM<prompt> - One or more values for a prompt**
  - Example: lsMCountries=USA;Germany
- **lsR<prompt> - Range of values for a prompt**
  - Example: lsRDates=[Date(2013,07,01)..Date(2013,07,15)]
Last Words of Advice

- **Use the plus sign to cover spaces in the URL**
  - Example: `sDocName=2006+Sales`

- **Remove any trailing spaces in a prompt**
  - Example: Prompt is “Choose a year: ”
  - Change to:
    - `lsSChoose+a+year:`

- **The length of the URL may be limited by the browser**
  - For Internet Explorer: 2083 chars
Demonstration – Invoking from Another Application
What We’ll Cover

- Introduction
- Annotated Report Standards
- Calculation Contexts
- Layered Variables
- Interface Points
- Wrap up
7 Key Points to Take Home

- Several Web Intelligence techniques can be hard to master
- Think about them logically vs. emotionally
- Annotating report elements helps start that process by providing standard “handles” for more advanced techniques
- Good naming conventions also standardize your report development
- Develop fully-formed calculation contexts to future-proof reports
- Creating variables in layers allows much more complex logic
- Reuse entire reports or portions using interface techniques
Where to Find More Information


• http://help.sap.com
  • Follow Analytics ➔ Business Intelligence ➔ Web Intelligence
Your Turn!

Questions?

Alan Mayer
214-295-6250 (office)
214-755-5771 (mobile)
alan.mayer@solidgrounded.com
Twitter: @solidgrounded

Please remember to complete your session evaluation
Disclaimer

SAP, R/3, mySAP, mySAP.com, SAP NetWeaver®, Duet™, PartnerEdge, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Wellesley Information Services is neither owned nor controlled by SAP.