



rego*U*niversity 2019

SAN DIEGO

# Administration | Advanced

Your Guides: David Zywiec and Jair Flores

# Introductions

- Take 5 Minutes
- Turn to a Person Near You
- Introduce Yourself
- Business Cards



# Agenda

---

- Objects, Attributes and Views
- UI Themes and Menu Customization
- Introduction to SQL
- Lookups, Queries and Portlets
- Introduction to Workflow (Processes)
- XOG (XML Open Gateway)

# Objects, Attributes and Views

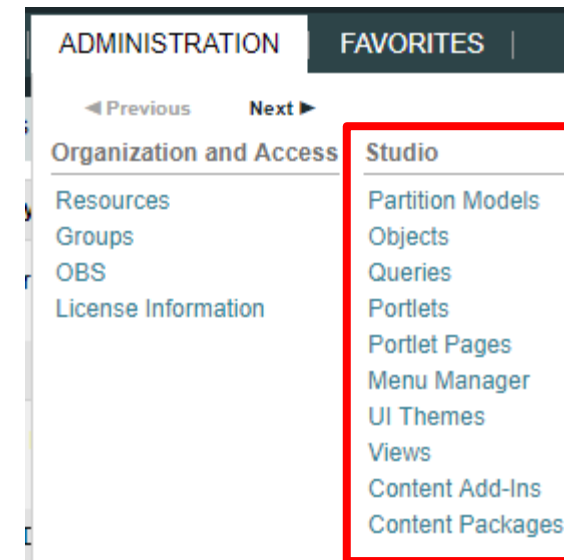


Let Rego be your guide.



# Clarity Studio

- Studio is the interface used to create new or modify existing components and customize the UI to meet the needs of your organization.
- A Clarity Studio license is required to use this functionality and users must have a variety of related security rights (Administration-Studio, Create/Edit Objects, Portlets, Pages, etc).
- Studio is accessed via the Administration menu



# Objects

- Objects are the major functional components of CA PPM
- Objects define the attributes (fields), subpages (links), page layout, and views that make up your configured instance of CA PPM
- In addition to the stock objects delivered with the system, you can create custom objects. Custom objects are essentially tables inside the database that begin with “ODF\_CA”
- Use the default objects or create custom objects and sub-objects to manage information for specific business needs
- Once you create an object, add attributes, links, and actions and set up the views

# Objects (continued)

- Each object has four distinct pieces you can configure
  - Properties
  - Attributes
  - Links
  - Views
- Things to remember
  - You can only delete Custom Attributes
  - Adding more than 100 custom attributes to a single custom object may impact performance
  - A hierarchy with a maximum of three levels of objects can be created, and allow child objects to inherit properties and access rights from parent objects

# Objects: Types

---

- Stock Objects
  - Primary Standard Objects
    - Project
    - Task
    - Team
    - Resource
    - Company
    - Application
- Custom Objects
  - Master Objects
  - Sub-Objects



# Objects: Investment Object

- The Investment object is a special component that provides the ability to define attributes one time and share them across select OOTB objects
- These objects “inherit” attributes from the Investment object:
  - Project, Idea, Other Work, Application, Asset, Product, Service
- Streamlines the creation process and ensures consistency across objects
- You may re-label attributes on shared objects if needed (ID remains the same)
- Attributes defined at the investment level are available to the stock objects noted above but are not required
- You must make updates to Investment attributes **at the Investment level**

# Attributes

- Attributes are the fields on an object that store information
- The attributes of each object are available on the Attribute screen within the object
- Many attributes are delivered out-of-the-box, but you can create an unlimited amount of additional attributes using Studio
- Once created, you can organize and place attributes on views and portlets and use for reporting
  - Example: “Start Date” is an attribute of the project object
- Details on the various data types can be found in the Studio Developer Guide

# Attribute Data Types

- When creating attributes the following data types are available:
  - String (2000-character maximum)
  - Large String
  - Number
  - Calculated
  - Money (includes currency code)
  - Boolean (checkbox)
  - Date
  - Lookup (related lookup needs to be available - create prior to creating attribute)
  - Multi-Valued lookup (same note above applies)
  - Attachment
  - Time-varying
  - URL (Links to actual data)

# Calculated Attributes

- A Calculated attribute displays a dynamic, read-only value
- Values are calculated from other existing attribute values
- Values are calculated every time the user accesses or refreshes the page
- These values **are not stored** in the database
- Calculated attributes can only read the following data types :
  - **Number**: Use this data type to calculate a number value like a sum or average
  - **String**: Use this data type to concatenate two or more text values
    - Example: concatenate the value of the attribute “created\_by” and the constant “2007” to produce a result of “ssmith 2007”
  - **Date**: Use this data type when you need to calculate dates using basic arithmetic or to provide the current date
- NOTE – You cannot delete source attributes used in a calculated attribute!

# Attributes: Auto-Numbering

- Clarity provides the ability to create your own numbering/naming scheme for object instances (PRJnnnnnn, APPnnnnnn, etc)
- The scheme can be numeric or a mixture of characters and numbers
- Two out-of-the-box attributes that are commonly auto-numbered are “Name” and “ID”
- Configuration is done via the Auto-numbering tab on the attribute detail

The screenshot displays the 'Auto-numbering' configuration window in Clarity. At the top, there are two tabs: 'Properties' and 'Auto-numbering', with 'Auto-numbering' being the active tab. Below the tabs, the text 'Object: AI Logger | Attribute: ID - Attribute Auto-numbering' is visible. The main area is divided into two sections: 'General' and 'Schemes'. In the 'General' section, there is a checkbox labeled 'Auto-numbered' which is currently unchecked. Below this checkbox are three buttons: 'Save', 'Save And Return', and 'Return'. The 'Schemes' section is currently empty, showing only a header bar with the text 'Runtime Next Number' on the right side.

# Attributes: Exercise

- Create a new custom sub-object to the project object
  - Administration -> Objects
  - Click New and fill out the required fields

**Object Name:** Meaningful name summarizing the object function

**Object ID:** Use a standard naming convention; many start with a customer ID + “\_” (special characters and spaces should not be used in IDs)

## Master or Subobject

- Choose **Master** if object is to be standalone
- Choose **Subobject** if object is to be accessed via another object (1 to many relationship)
  - Choose the Master object by using the browse button

Create Object Definition

Object Name: Training Object

Object ID: rego\_training

Content Source: Customer

Description: This object is used for training purposes

Master or Subobject: ☒ Master ☐ Subobject

Partition Model: [Browse]

Master Object: [Browse]

Event Enabled: ☐

Copy Enabled: ☐

Export Enabled: ☐

View All Enabled: ☐

Save Save And Return Return

Legend: [Red Square] = Required [Green Arrow] = Enter Once [Green Star] = Unique

- Select the following checkboxes if they apply
  - **Event Enabled:** Specifies that the process engine is notified of object instances that are created or updated. (If a process needs to get driven off the object)
  - **Copy Enabled:** Specifies that copies can be made of the object instances.
  - **Export Enabled:** Specifies that object instances can be exported to XML.
  - **View All Enabled:** Specifies that the object instances can have a view containing all properties, sub-object lists, and page portlets that can be personalized on a single page.



# Attributes: Exercise (cont)

- Notice the default fields included in the newly created object

The screenshot shows a web interface with a tabbed menu at the top containing 'Properties', 'Attributes' (selected), 'Linking', and 'A'. Below the tabs, the text 'Object: Training Object - Attributes' is displayed. The main area contains a form with the following elements:

- 'Attribute Name' with an empty text input field.
- 'Attribute Display' with a dropdown menu showing 'All Attributes'.
- Three buttons: 'Filter' (blue), 'Show All' (blue), and 'Clear' (blue).
- A table with a checkbox in the first column and two columns labeled 'Attribute' and 'Descript'.

<input type="checkbox"/>	Attribute	Descript
<input type="checkbox"/>	Created By	
<input type="checkbox"/>	Created Date	
<input type="checkbox"/>	ID	
<input type="checkbox"/>	Last Updated Date	
<input type="checkbox"/>	Name	
<input type="checkbox"/>	Page Layout	Page Layout
<input type="checkbox"/>	Partition	Code that identifies the
<input type="checkbox"/>	Updated By	

**Created By and Created Date:** Keep track of who and when the record was created

**Last Updated By and Updated By:** Keep track of the last person who updated the record and when

**\*\* Note:** Outside of custom objects there are OOTB jobs / processes that will scew the results of the last updated by and date fields, as the application often makes updates to the record

**Page Layout:** Each object defaults to a standard layout with tabs such as Properties, Processes, and Audit. This can be customized by adding a new custom page layout. (Details later on)

**Name and ID:** Used to identify the record; Name can be repeated multiple times while the ID has to be unique. Auto-numbering is often used to force that uniqueness and standardization.

# Attributes: Exercise (cont)

- Add 3 or 4 attributes of different varieties to your new custom object
  - Click New and fill out required fields as well as Data Type

Object: Training Object | Attribute: - Object Attribute

\* Attribute Name

\* → \* Attribute ID   
( ID must be alphanumeric, underscore is permitted. It must not be a SQL or Clarity reserved word. )

Description

→ Data Type

\* → Lookup

Default   
( Click Save to update this field after selecting a new lookup. )

Populate Null Values with the Default ☐

Value Required ☐

Presence Required ☐

Read-Only ☐  
( In order to make an attribute read-only a default must be selected )

\* = Required   → = Enter Once   \* = Unique

**Attribute Name:** Name of attribute (display name can be changed later in fields if need be)

**Attribute ID:** unique ID for attribute (spaces and special characters not allowed)

**Description:** Meaningful explanation for attribute usage

**Data Type:** Type of attribute

**Lookup:** Only shows up for “Lookup” and “Mutli Valued Lookup” types

**Default:** Default value to populate attribute with ( if applicable)

# Attributes: Exercise (cont)

- Select the following checkboxes if they apply:
  - **Populate Null Values with the Default:** If an attribute is added after instances have already been created, this will populate the existing records with the default value
  - **Value Required:** Specifies whether a value is required for the attribute.
  - **Presence Required:** Specifies that the attribute always appears in the Edit Properties view.
  - **Read-Only:** Specifies that a user cannot make changes to the value in the attribute

Populate Null Values with the Default ☐

Value Required ☐

Presence Required ☐

Read-Only ☐

( In order to make an attribute read-only a default must be selected )

\* = Required   → = Enter Once   \* = Unique

# Attributes: Exercise (cont)

- Auto-number the ID attribute
  - Select the auto-numbering tab
  - Check the “Auto-numbered” box and click “Save”
  - Select Scheme -> Edit

Properties Auto-numbering

Object: Training Object | Attribute: ID - Attribute Auto-numbering

General

Auto-numbered ☒

Save Save And Return Return

Schemes

Partition: System ▼

Partition	Runtime Next Number	Scheme
System	00000001	[Edit]

# Attributes: Exercise (cont)

- The default segment type is numeric but this can be modified to include text characters as well

Object: Training Object | Attribute: ID | Partition: System - Auto-numbering Scheme

General Information

Scheme Name

System Default

Partition

System

Next Number

00000001

Status

Active

Segments

<input type="checkbox"/>	Type of Segment	Text Value	Counter Starting Number	Counter Length	Auto-extended
<input type="checkbox"/>	Numeric Counter		00000001	8	✓

New

✓ Delete

Reorder

Save

Save And Return

Return

= Required

- Select “New” and set Type of Segment = “Text” and type the text value into “Value” field.
- Click Save and Return

Segment Properties

Type of Segment

Text

Value

TRN

Save And Return

Return

# Attributes: Exercise (cont)

Object: Training Object | Attribute: ID | Partition: System - Auto-numbering Scheme

**General Information**

\* Scheme Name: System Default

Partition: System

Next Number: 00000001TRN

Status: Active

Note the new numbering scheme

**Segments**

<input type="checkbox"/>	Type of Segment	Text Value	Counter Starting Number	Counter Length	Auto-extended
<input type="checkbox"/>	Numeric Counter		00000001	8	✓
<input type="checkbox"/>	Text	TRN		3	

New Delete Reorder Save Save And Return Return

- The new scheme defaults into the order it was entered. To reorder to have the text in front select “Reorder” and move the segments accordingly.
- Select “Save and Return”

Reorder Auto-numbering Scheme Segments

Scheme Segments: Text(Next Value: TRN)  
Numeric Counter(Next Value: 00000001)

Save Save And Return Return



# Views

- Object Views control what attributes are displayed, and what they look like on the screen
- There are four types of views:
  - **Create:** What the user sees when creating a new record
  - **Edit:** What the user sees for an existing record (e.g. opening a project)
  - **List:** What the user sees when viewing a list of records (e.g. Home->Resources)
  - **Filter:** The contents and layout of a List view filter section
- Users can be given the ability to configure their own List and Filter views for a specific object by checking “Allow Configuration” in the List options:

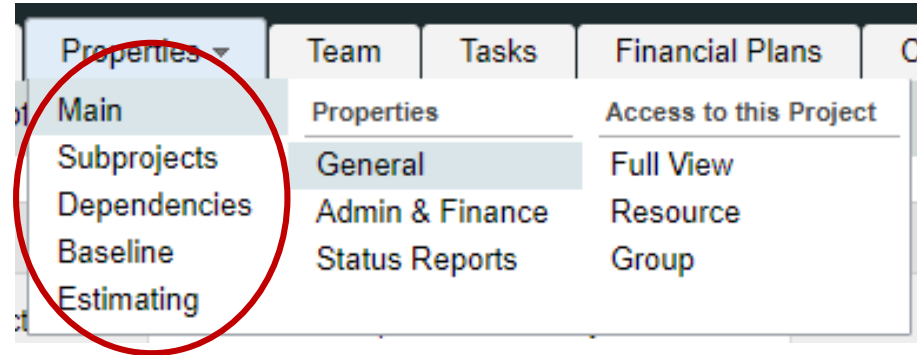
Category	
Properties	[Layout: Create] [Layout: Edit] [Actions Menu]
List Column	[Layout] <b>[Options]</b> [Aggregation] [Actions Menu]
List Filter	[Layout]

# Views: Subpages and Sections

- Within each View separate subpages and sections can be created to group attributes together in meaningful ways
  - Subpages are accessed via a link in the drop-down menu and are generally used to drive different functions
  - A section is an area on a page or subpage that divides the page into logical groupings
  - Note: A subpage on a master custom object can also be configured as a tab instead of a link
- The same attribute can be placed on multiple sub-pages within the same object
  - This is useful to allow editing of the field on a restricted page but display it as read-only on other pages (secured subpage)
  - Another option is to use a Calculated field to make a field read-only on specific screens

# Views: Subpages and Sections

- Subpages are separate screens accessed via a dropdown menu



- Sections provide the ability to break up a busy screen into smaller pieces

A screenshot of a project management form titled 'Project: 2017 Software Implementation Project - Properties - Main - General'. The form is divided into sections. The 'General' section is highlighted with a red circle and contains the following fields: 'Project Name' (2017 Software Implementation Project), 'Project ID' (PR000009), 'Project Type' (Application Change), 'Status' (Approved), 'Charge Code' ([--Select--]), 'Progress' (Started), 'Project Manager' (Bonham, Jessica), 'Stage' (Initiation), 'Goal' (Infrastructure Improvement), and 'Priority' (25). The 'Description' section is also highlighted with a red circle and contains a text area labeled 'Description'. The form has a header bar with 'Open in Scheduler' and 'Scenario: [--Select--]' buttons.

# Views: Subpages and Sections

Object: Status Report | Partition: System | View: General | Mode: Edit - *Property Layout*

<input type="checkbox"/>	Page > Subpage			Level
-	[Edit Status Report Properties]		≡	Page
+	General		≡	Subpage
<div>Create Subpages</div> <div>Return</div>				

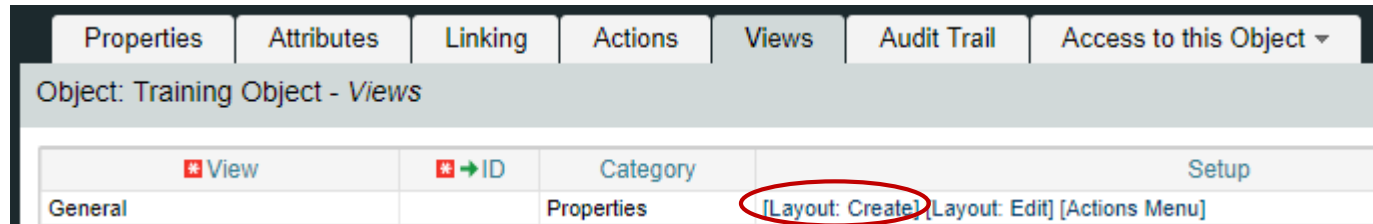
Top	<input type="checkbox"/>	Subpage > Section			Level
	-	[General]		≡	Subpage
	<input type="checkbox"/>	+	Status Report	≡	Section
	<input type="checkbox"/>	+	Overall	≡	Section
	<input type="checkbox"/>	+	Schedule	≡	Section
	<input type="checkbox"/>	+	Scope	≡	Section
	<input type="checkbox"/>	+	Cost and Effort	≡	Section
Create Sections    ✓ Delete    Return					

# Views: Dynamic Pages

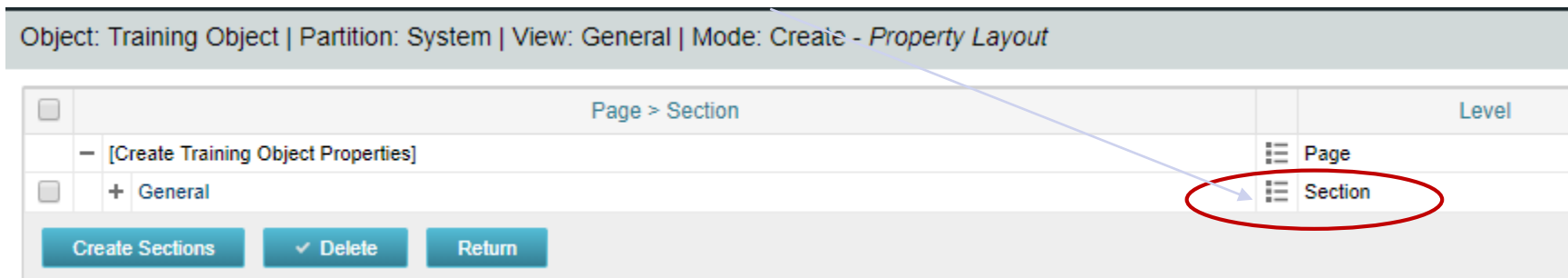
- You can show or hide views based on display conditions or securing the page. The differences in these two are defined below.
- **Display Conditions**
  - Defines a set of conditions that determine when a subpage appears. The expression builder is used to create these conditions and utilizes attributes available to the object or security groups
- **Secured Subpages**
  - Check the “secured subpage” box inside the page properties to create rights to either view or edit the values on this page
  - Secured subpages are not available on the Task object

# Views: Exercise

- Using the new object do the following:
- Modify the create view to add 1 or 2 attributes to General section of General subpage
  - Navigate to Administration -> Objects -> Views -> Layout:Create



- Select the properties icon to the left of the Section to alter the layout





# Views: Exercise (cont)

- Move an attribute to the 2nd column using the arrows underneath the columns below

Object: Training Object | Partition: System | View: General | Mode: Create - Section Properties

**Layout**

Required fields are marked with an asterisk (\*)  
Hidden fields are marked with a tilde (~)

Available	Selected (Left Column)	Selected (Right Column)
Created Date Last Updated By Last Updated Date Partition~	Name* ID* Page Layout*	Created By

Add Field →      ← Move Field      ← Move Field

**Title**

Parent Create Training Object Properties

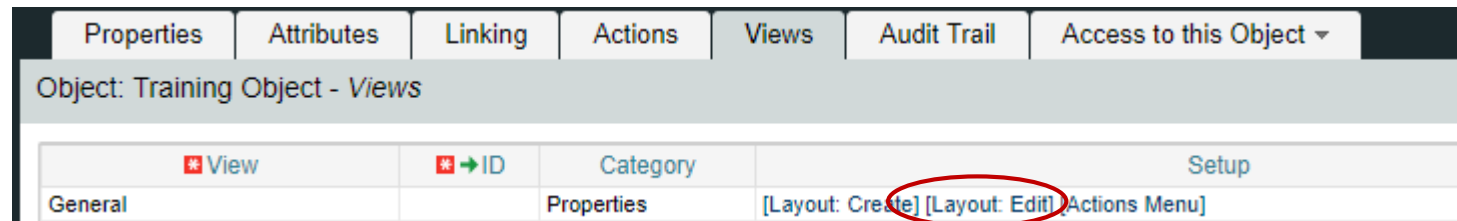
\* Section Name General

Save Save And Return Return

Note: The Name, ID, and Page Layout will show up by default as they are required. They can be hidden, if desired. (In later slides)

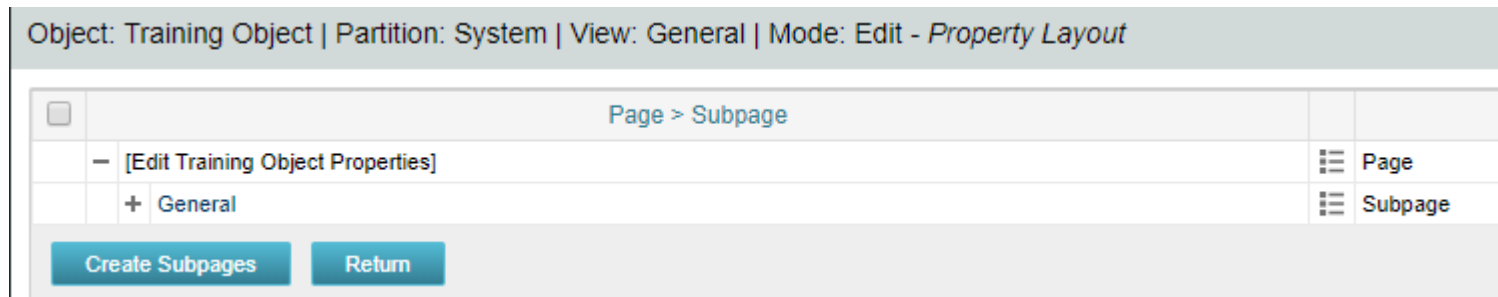
# Views: Exercise (cont)

- Create a separate subpage within the Edit View
  - Navigate to Administration -> Objects -> Views -> Layout:Edit



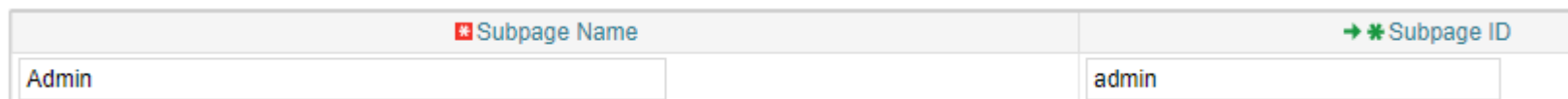
Properties	Attributes	Linking	Actions	Views	Audit Trail	Access to this Object ▾
Object: Training Object - Views						
View	ID	Category	Setup			
General		Properties	[Layout: Create] [Layout: Edit] [Actions Menu]			

- Select “Create Subpages” and fill out subpage “Name” and “ID”. Click Save and Return.



Object: Training Object | Partition: System | View: General | Mode: Edit - Property Layout

<input type="checkbox"/>	Page > Subpage		
<input type="checkbox"/>	[Edit Training Object Properties]	≡	Page
<input checked="" type="checkbox"/>	General	≡	Subpage



Subpage Name	Subpage ID
Admin	admin

# Views: Exercise (cont)

- Add a section within the subpage by clicking on the subpage link first

Page > Subpage

– [Edit Training Object Properties]

+ General

Admin

Create Subpages Delete Return

- Select “Create Subpages” and fill out subpage “Name” and “ID”
- Select “Create Sections” and type the name of each section.
  - You can add up to 5 at time
  - Save and Return

Top

Subpage > Section

[Admin]

Create Sections Return

Parent /Edit Training Object Properties/Admin

\* Section Names Admin

Save And Return Return

\* = Required

# Views: Exercise (cont)

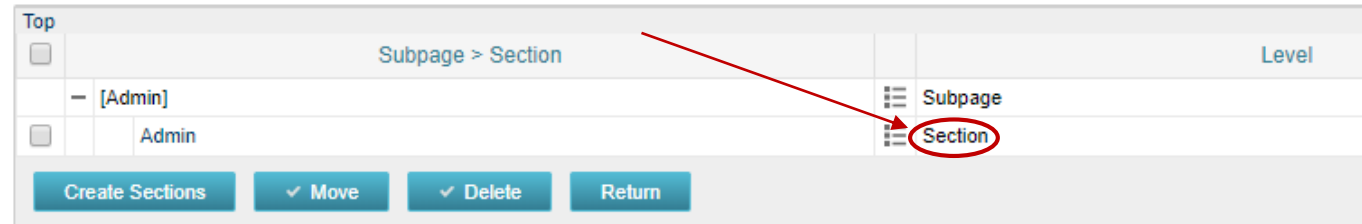
- Add an attribute to the new subpage and section by clicking on the section properties link, selecting the field(s) from the Available column and moving to the Left or Right Column(s)

Object: Training Object | Partition: System | View: General | Mode: Edit - *Property Layout*

Top

<input type="checkbox"/>	Subpage > Section		Level
-	[Admin]		
<input type="checkbox"/>	Admin		

Create Sections   ✓ Move   ✓ Delete   Return



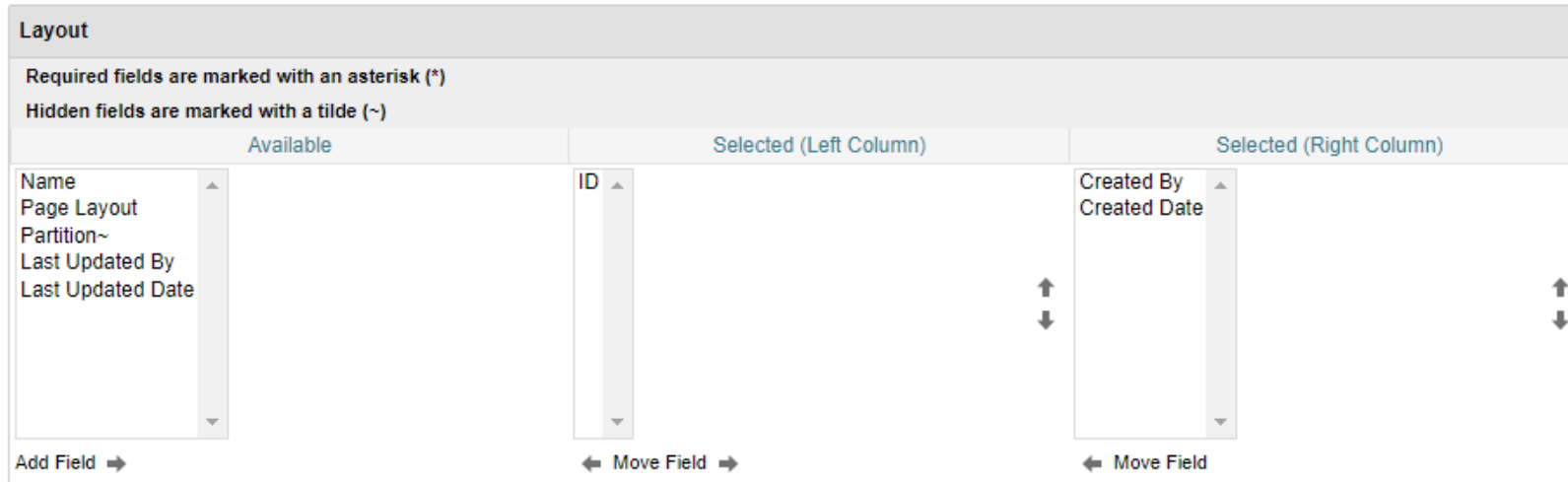
Object: Training Object | Partition: System | View: General | Mode: Edit - *Section Properties*

Layout

Required fields are marked with an asterisk (\*)  
Hidden fields are marked with a tilde (~)

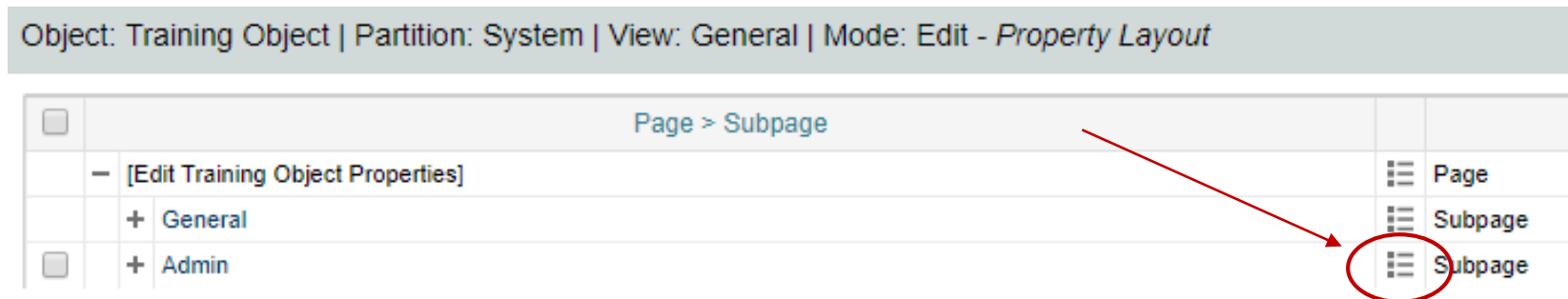
Available	Selected (Left Column)	Selected (Right Column)
Name Page Layout Partition~ Last Updated By Last Updated Date	ID	Created By Created Date

Add Field ➡   ← Move Field   ← Move Field



# Views: Exercise (cont)

- Create a display condition on the new sub-page
  - Navigate to Administration -> Objects -> Views -> Layout:Edit-> Admin Subpage and select the properties icon

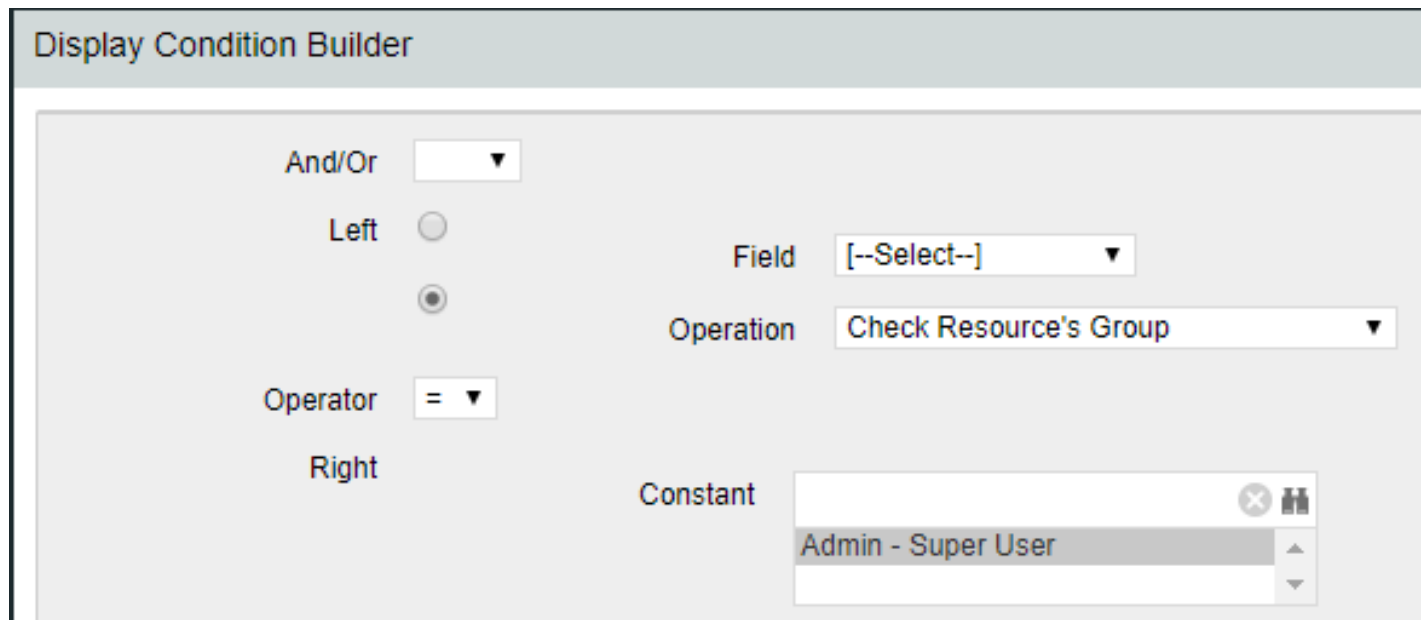


- Select “Define Display Conditions” within the “Display Conditions” section



# Views: Exercise (cont)

- Select the radio button next to “Operation” and in the “Operation” drop down select “Check Resource’s Group”
- Use the browse within the “Constant” box to choose the group for which the page should only display



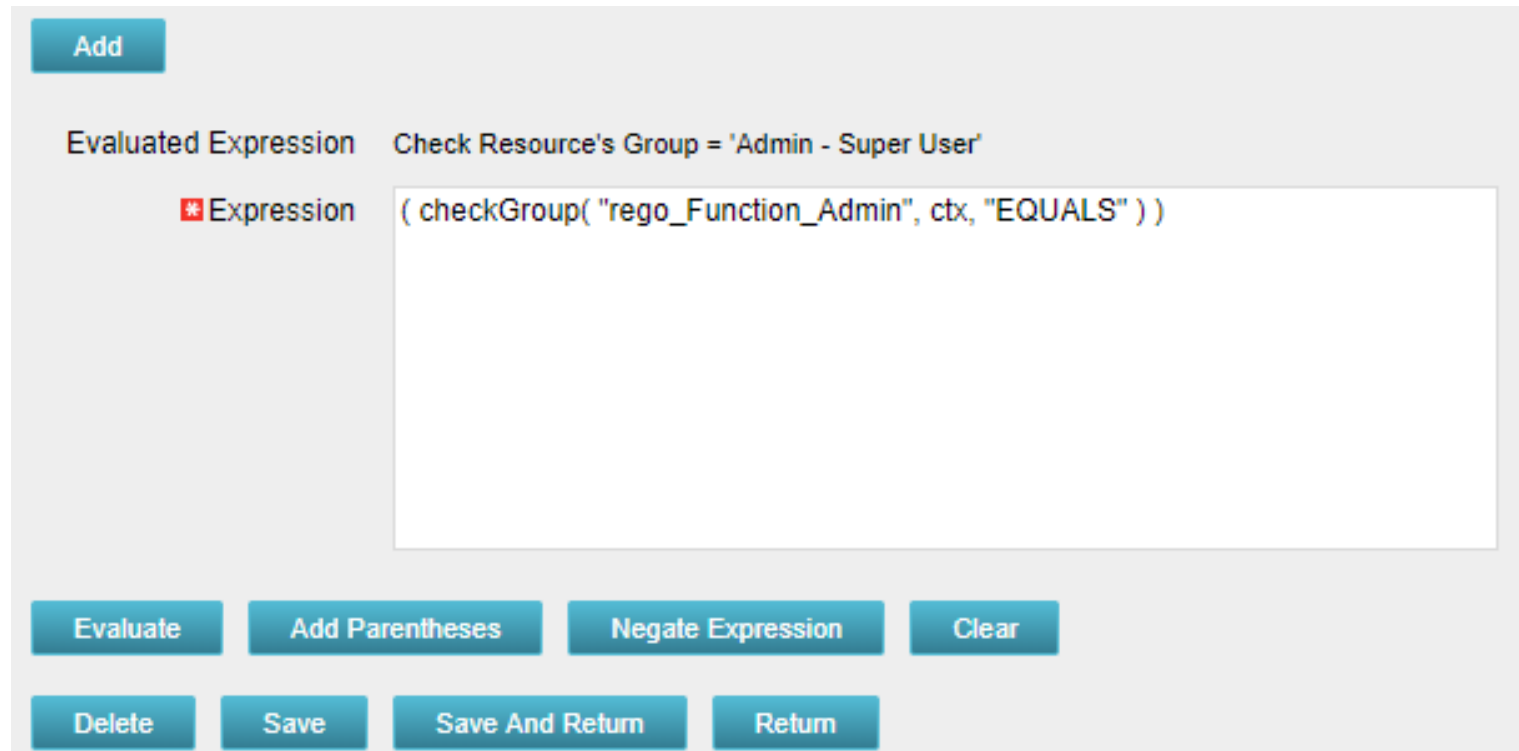
The screenshot shows the 'Display Condition Builder' window. It contains the following elements:

- And/Or:** A dropdown menu.
- Left:** A radio button.
- Operator:** A dropdown menu showing '='.
- Right:** A radio button.
- Field:** A dropdown menu showing '[--Select--]'.
- Operation:** A dropdown menu showing 'Check Resource's Group'.
- Constant:** A text input field with a browse button (represented by a list icon) to its right. The browse button is open, showing a list with 'Admin - Super User' selected.



# Views: Exercise (cont)

- Select the “Add” button to build the expression and click “Save and Return”



The screenshot shows a web interface for building expressions. At the top left is a blue button labeled "Add". Below it, the text "Evaluated Expression" is followed by "Check Resource's Group = 'Admin - Super User'". To the left of a large text input field is a red asterisk icon and the label "Expression". The input field contains the text: `( checkGroup( "rego_Function_Admin", ctx, "EQUALS" ) )`. At the bottom of the interface is a row of buttons: "Evaluate", "Add Parentheses", "Negate Expression", and "Clear". Below this row is another row of buttons: "Delete", "Save", "Save And Return", and "Return".

# Fields

- Fields are representations of the available attributes which are placed onto screens and list views
- Field names can be different than the attribute (relabeled)
- They provide additional options and allow flexibility in view configuration:
  - Browse vs Pull-Down (for lookups)
  - Hints and Tooltips
  - Size of the field on the screen and text boxes
  - Default value
  - Required (even if NOT required)
  - List alignment
  - List column width
  - List word wrapping

Properties	Attributes	Linking	Actions	Views	Audit Trail	Access to this Object ▾
Object: Project - Views						
View	Category	Setup				
Project Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Program Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Project List	List Column	[Layout]	[Options]	[Aggregation]	[Actions Menu]	[Fields]

# Fields: Adding to a Screen

- Fields can be added or removed on the Create and Edit screens

Properties	Attributes	Linking	Actions	Views	Audit Trail	Access to this Object ▾
Object: Project - Views						
View	Category	Setup				
Project Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Program Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Project List	List Column	[Layout]	[Options]	[Aggregation]	[Actions Menu]	[Fields]

Page > Section			Level
<input type="checkbox"/>	<input type="checkbox"/> [Create Asset Properties]		<input type="checkbox"/> Page
<input type="checkbox"/>	<input type="checkbox"/> + General		<input type="checkbox"/> Section
<input type="checkbox"/>	<input type="checkbox"/> + Detail		<input type="checkbox"/> Section
<input type="checkbox"/>	<input type="checkbox"/> + Attachements		<input type="checkbox"/> Section
<input type="checkbox"/>	<input type="checkbox"/> + OBS		<input type="checkbox"/> Section
Create Sections			Delete Return

Page > Subpage			Level
<input type="checkbox"/>	<input type="checkbox"/> [Edit Asset Properties]		<input type="checkbox"/> Page
<input type="checkbox"/>	<input type="checkbox"/> + Asset Summary		<input type="checkbox"/> Subpage
<input type="checkbox"/>	<input type="checkbox"/> + Schedule & Performance		<input type="checkbox"/> Subpage
<input type="checkbox"/>	<input type="checkbox"/> + Alignment & Risk		<input type="checkbox"/> Subpage
<input type="checkbox"/>	<input type="checkbox"/> + Financial Summary		<input type="checkbox"/> Subpage
<input type="checkbox"/>	<input type="checkbox"/> + Compliance		<input type="checkbox"/> Subpage
<input type="checkbox"/>	<input type="checkbox"/> + Settings		<input type="checkbox"/> Subpage
Create Subpages			Delete Return

# Fields: Adding to a List

- Fields can also be added or removed on the List view

Properties	Attributes	Linking	Actions	Views	Audit Trail	Access to this Object ▾
Object: Project - Views						
View	Category	Setup				
Project Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Program Properties	Properties	[Layout: Create]	[Layout: Edit]	[Actions Menu]	[Fields]	
Project List	List Column	[Layout]	Options	[Aggregation]	[Actions Menu]	[Fields]

Column Layout

Available Columns

Actuals for Labor Resources  
Actuals Sum for Labor Resources  
Alignment  
Architectural Fit  
Billing Currency Code  
Blueprint  
Blueprint Active ID  
Board Plan  
BTM Integration  
Budgeted Benefit

Selected Columns

ID  
Compliance  
Value Metrics  
Gray Bar  
Related Idea  
Approved Flag  
Status  
Progress  
Start  
Finish

# Fields: List Options

- Consider the following List view options to improve the user experience:
  - Rows per Page
  - Allow Configuration
  - Filter results

Properties	Attributes	Linking	Actions	Views	Audit Trail	Access to this Object ▾
Object: Project - Views						
View		Category	Setup			
Project Properties		Properties	[Layout: Create] [Layout: Edit] [Actions Menu] [Fields]			
Program Properties		Properties	[Layout: Create] [Layout: Edit] [Actions Menu] [Fields]			
Project List		List Column	[Layout] [Options] [Aggregation] [Actions Menu] [Fields]			

**Display Options**

Secondary Value Display

☒ Mouseover only

☐ Mouseover and redline text  
( Used when any list column field displays a secondary value )

☐ Show Null Secondary Values

Filter

☒ Automatically show results

☐ Do not show results until I filter

Rows per Page

50 ▾

Highlight Row by Attribute

Approved Flag ▾  
( A row will be highlighted when this attribute is not zero )

Display Currency Code in Column

☐  
( Applies when only a single currency is active )

Allow Configuration

☒

Allow Label Configuration

☒

Attribute Value Protection

☒ Use display conditions and secured subpages to protect attribute values on this list

☐ Use only secured subpages to protect attribute values on this list

☐ Display all attribute values on this list

# Fields: Exercise

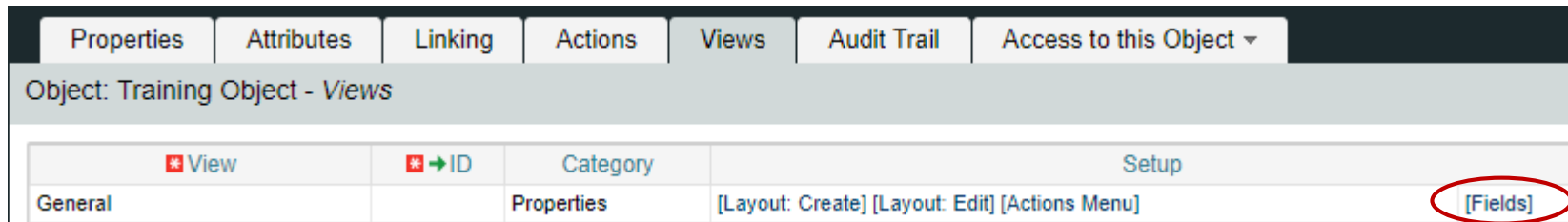
- Using the new object do the following:
  - Modify field “ID” to make read-only
    - Navigate to Administration -> Objects -> <Object Name> -> Attributes
    - Select the ID field and open up. Set a default value. Click Save and Return.

The screenshot shows a web-based configuration interface. At the top, there are two tabs: 'Properties' and 'Auto-numbering', with 'Auto-numbering' being the active tab. Below the tabs, a header bar reads 'Object: Training Object | Attribute: ID - Object Attribute'. The main content area displays configuration options for the 'ID' attribute. It includes a red asterisk icon next to 'Attribute Name' with the value 'ID'. Below that, a red asterisk icon, a green arrow, and another red asterisk icon are next to 'Attribute ID' with the value 'code'. There is a 'Description' label without a value. Below that, a green arrow and the text 'Data Type' are next to the value 'String'. At the bottom, the label 'Default Value' is next to a text input field containing 'TRN00000'.

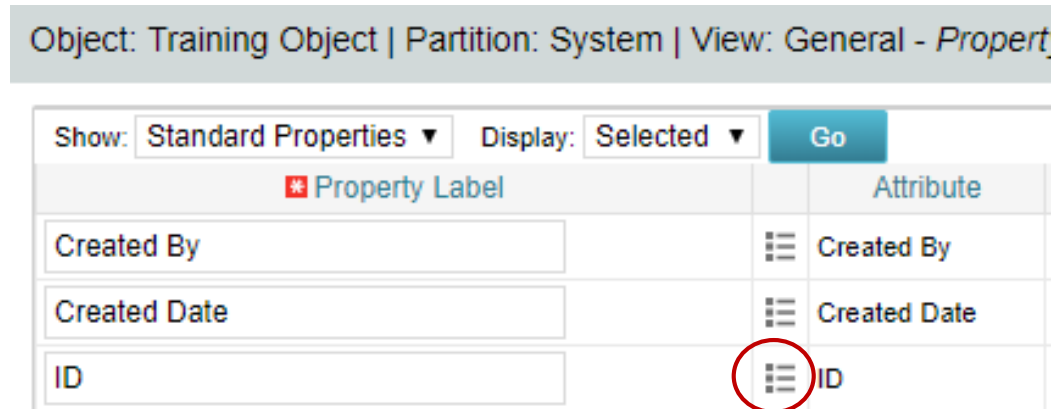
* Attribute Name	ID
* → * Attribute ID	code
Description	
→ Data Type	String
Default Value	TRN00000

# Fields: Exercise (cont)

- Select “Views” tab
- On the General -> Properties line click the “Fields” link




- Click on the properties icon next to the “ID” field



# Fields: Exercise (cont)

- Select the “Hidden” checkbox

Object: Training Object | Partition: System | View: General - Property Field

Attribute	code
Data Type	String
Property Label	ID 
Display Type	Text Entry ▼
Hint	<input type="text"/>
Hint Position	Below ▼
Tooltip	<input type="text"/>
Attribute Default	TRN00000
Override Default Value	<input type="text"/>
Width	20 <input type="text"/>
Value Required	<input checked="" type="checkbox"/>
Enter Once	<input type="checkbox"/>
Hidden	<input type="checkbox"/>

( In order to make a property field hidden a default must be selected. )



# Fields: Exercise (cont)

- Make one of the new attributes “Required” within the field properties
  - Navigate to Administration -> Objects -> <Object Name> -> Attributes
  - Select a field (newly created) and open up properties.
  - Select the “Value Required” checkbox and click “Save”.

Object: Training Object | Attribute: Description - *Object Attribute*

Attribute Name	Description
Attribute ID	v_desc
Description	
Data Type	String
Default Value	
Maximum Size	2000
	( The maximum size is 2000. For 3 byte Unicode the actual maximum size is 1333. )
Populate Null Values with the Default	<input type="checkbox"/>
Value Required	<input checked="" type="checkbox"/>

# Fields: Exercise (cont)

- Add a hint for entering one of the values and place it below the field
  - Navigate to Administration -> Objects -> <Object Name> -> Views
  - From the General -> Properties line click on “Fields”
  - Choose a field that you want to add some verbiage to help the user when entering information.
  - Add verbiage to the “Hint” text box and choose a position

Object: Training Object | Partition: System | View: General - Property Field

Attribute	name
Data Type	String
* Property Label	<input type="text" value="Name"/>
Display Type	Text Entry ▼
Hint	<input type="text" value="Enter the name"/>
Hint Position	Below ▼

# Actions

- Object actions are individual operations that can be selected to be done from either the list or properties view within an object instance
  - Examples of actions are the ability to run a report (only Business Objects), initiate a process instance, copy an object instance, etc.
- Each object has some default actions available to them
- To utilize an action the action menu needs to be configured
  - Within an object this is located within the “Views” tab and by clicking on “Actions Menu” for the specific view being configured
- Actions and the associated menus can be renamed as needed

# UI Themes and Menu Customization



Let Rego be your guide.

# User Interface (UI) Themes

- What is a UI theme?
  - Determines the look and feel of CA PPM
  - Can be adjusted or extended
  - Can be different per partition
  - Created using cascading style sheets (CSS)
- Create your own theme:
  1. Copy CSS from stock theme
  2. Modify
    - Main page background color (3366AA)
    - Logo
      - Convert to base64 string – free tools on web
      - Replace string on CCS for The logo in the upper left hand corner...
    - Remove a button from a custom object view

# UI Themes Demonstration / Exercise

- Modify the UI Theme to change the background color
  - Navigate to Administration -> Studio -> UI Themes
  - Pick an existing UI theme to copy from
    - Open and copy CSS
  - Select “New” and give your UI theme a name and ID
  - Paste the CSS into the CSS field
  - Change the following snippet to the hexadecimal # representing the color you wish to change it to

```
/* The main page background color */
.ppm_page_bg {
  background: #3366AA;
  background: -webkit-gradient(radial,50% 10, 1, 50% 100, 600, from(#052E5F), to(#3366AA ));
  background: -moz-radial-gradient(50% 10% 0deg,ellipse contain, #052E5F, #3366AA);
}
```

# UI Themes Demonstration / Exercise (cont)

- Apply new UI Themes
  - Navigate to Administration -> Studio -> Partition Models
  - Click the name of the partition you want to view
  - Select the “Partitions” tab
  - Click on the properties icon next to the name of the partition to which you want to set a UI theme
  - In the UI Theme field select the new UI Theme created

# Menus

- Menus are utilized to assist in simplifying the organization and accessibility of information for users, based on their organizational role
- Two Main Menu Areas
  - Home (Application )
  - Administration
- Consists of:
  - Links to a Page or Action
  - Sections
- Menus are setup without regard to security
  - User will only see what they have access to
- Changes to the menu require a refresh or logout of the existing session



# Menus Demonstration / Exercise

- Modify the Home menu by adding a new section called “Rego U”
  - Navigate to Administration -> Studio -> Menu Manager
  - Select Application Menu
  - Click on the “Add” button and select “Section”
  - Name the section “Rego U-<your initials>” with an ID = “regou\_<your initials>”
  - Click Save and Return

# Menus Demonstration / Exercise (cont)

- Move a link from the current section to the new “Rego U” section
  - Navigate to Administration -> Studio -> Menu Manager
  - Select Application Menu
  - Click on the checkbox next to the link you want to move
  - Select “Move”
  - Select the radio button next to the new section created
  - Click “Save and Return”

# Introduction to SQL



Let Rego be your guide.

# What Is SQL?

---

- SQL stands for Structured Query Language
- SQL is an ANSI (American National Standards Institute) standard
- SQL is semantically easy to understand and learn
- SQL lets you access and manipulate databases and is great for performing the types of analysis and aggregations normally done in Excel
- SQL allows you to traverse much larger datasets and multiple tables at the same time

# What Is a Database?

- A database is an organized collection of data
- Tables are part of what makes up a database and are similar to the layout of spreadsheets
- Tables are more formalized inside a database with each column having a unique identifier as its heading
- Within databases, tables are organized in schemas
- Schemas are defined by usernames, whereas all of the tables related to that schema will be loaded under it
  - E.g. CA PPM uses NIKU as the default schema in which all of the related tables reside

# Using SQL in Clarity

- Some common uses of SQL are:
  - To extract ad-hoc data
  - As a basis for NSQL in portlet writing
  - To get data within a process for data manipulation
- Clarity Data Model
  - Knowing the CA PPM data model is half the battle to grabbing the data you need
  - Three main areas where data is stored:
    - Core Tables (Real Time): Investment, Resource, Timesheet
    - Time Slice Tables: Houses summarize data by daily, weekly, monthly, etc.
    - DataMart Tables: Provides summary and rollup data

# Basic SQL Syntax

- SQL is NOT case sensitive; SELECT is the same as select
- Some database systems (Oracle) require a semicolon at the end of each SQL statement
- There are two required ingredients in any SQL query: a SELECT statement and a FROM statement—and they have to be in that order
  - SELECT indicates which columns you'd like to view, and FROM identifies the table that they live in
- Column names should be separated by commas in the query
- If you want to select every column in a table, you can use \* instead of the column names

# Basic SQL Syntax (cont)

- The following statements will extract all rows from a table based on the columns selected:
  - `SELECT column_name,column_name  
FROM table_name;`
  - `SELECT * FROM table_name;`



# Basic SQL Syntax (cont)

- To further limit the results returned from a query use the WHERE clause
  - SELECT column\_name,column\_name  
FROM table\_name  
WHERE column\_name operator value;
- Operators in the WHERE clause

Operator	Description
=	Equal
<>	Not equal. <b>Note:</b> In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

# Query Exercise #1

- Write a select query to pull all the resources from the system and their associated user name

- a. Start with a select from the resource table

```
select r.full_name  
from srm_resources r
```

- b. Add a join to the users table

```
select r.full_name  
from srm_resources r  
join cmn_sec_users u ON u.id = r.user_id
```

- c. Add the user name column from the users table

```
select r.full_name, u.user_name  
from srm_resources r  
join cmn_sec_users u ON u.id = r.user_id
```

# SQL Aliases

- Table Aliases

- Improve readability of SQL queries
  - Use meaningful table aliases
- Allow queries to be easily created/modified
- Improve performance by eliminating the need for the database to search the tables for the reference column.

- E.g.

```
SELECT column_name1, column_name2  
FROM srm_resources res
```

- In the above example “res” is the alias and will be used to reference the table “srm\_resources” throughout the rest of the query

# SQL Column Names

- Database columns can be renamed in a SELECT statement as such:
  - *SELECT user\_name as user FROM cmn\_sec\_users*
    - In the above example the column will be displayed in the results as “user”
    - Note that the keyword “as” is implied and therefore not required
- You can also display a column in a more descriptive way with upper/lowercase and spaces. E.g.
  - *SELECT user\_name as “User Name”, email as “Email Address” FROM cmn\_sec\_users*
    - The double quotes ARE REQUIRED when you want to include spaces and lowercase text
    - The “as” keyword is optional here as well

# SQL Joins

- Joins

- Clauses used to combine rows from two or more tables, based on common fields between them.
- Types
  - INNER JOIN: Returns all rows when there is at least one match in BOTH tables
  - LEFT JOIN: Return all rows from the left table, and the matched rows from the right table
  - RIGHT JOIN: Return all rows from the right table, and the matched rows from the left table
  - FULL JOIN: Return all rows when there is a match in ONE of the tables

# More SQL Concepts

- **DISTINCT Statement**

- In a table, a column may contain many duplicate values; and sometimes you only want to list the different (distinct) values
- The SELECT DISTINCT statement is used to return only distinct (different) values.
- Syntax:
  - *SELECT DISTINCT column\_name, column\_name FROM table\_name;*

- **ORDER BY Statement**

- The ORDER BY keyword is used to sort the result-set
- Syntax:
  - *SELECT column\_name, column\_name  
FROM table\_name  
ORDER BY column\_name ASC/DESC, column\_name ASC/DESC;*

# SQL Functions

- Functions are used to perform processing on string and numeric data
- Types
  - Aggregate
    - Return a single value, calculated from values in a column
      - Examples: AVG (Average), COUNT, MAX, MIN, SUM
    - GROUP BY statements are required when using aggregate functions
  - Scalar
    - Return a single value, based on the input value
      - Examples: ROUND, UCASE (Uppercase), LCASE (Lowercase), FORMAT

# SQL Subqueries

- Subqueries are nested queries (a query within a query)
- They must be enclosed in parentheses
- Subqueries can be included within SELECT, INSERT, UPDATE or DELETE statements
- Example:
  - *SELECT full\_name FROM srm\_resources  
WHERE user\_id IN (SELECT manager\_id FROM srm\_resources)*



# Query Exercise #2

- Write a query to pull all active projects starting in 2018 with a count of their tasks.
- Use aliases for table names and rename columns to be meaningful
  - a. Start with a select from the investment table to pull all investments (projects, ideas, other, etc.)

```
select inv.name, inv.code  
from inv_investments inv
```

- b. Add a “WHERE” clause to restrict the result set to active projects only and those that begin in 2018

```
select inv.name, inv.code  
from inv_investments inv  
where inv.odf_object_code = 'project'  
and inv.is_active = 1  
and inv.schedule_start >= to_char('2018-01-01','yyyy-mm-dd')
```

# Query Exercise #2 (cont)

- c. Add a join to the investments table to get the tasks

\*\* Note the join will be a left join as we want all projects and just a count of tasks where they exist on the project

```
select inv.name, inv.code, count(t.prid)
from inv_investments inv
left join prtask t ON t.prprojectid = inv.id
where inv.odf_object_code = 'project'
and inv.is_active = 1
and inv.schedule_start >= to_char('2015-01-01','yyyy-mm-dd')
group by inv.name, inv.code
```

# Query Exercise #2 (cont)

d. Add column names to make the results meaningful

```
select inv.name as "Project Name", inv.code as "Project ID", count(t.prid) as "Task Count"  
  from inv_investments inv  
 left join prtask t ON t.prprojectid = inv.id  
 where inv.odf_object_code = 'project'  
 and inv.is_active = 1  
 and inv.schedule_start >= to_char('2015-01-01','yyyy-mm-dd')  
 group by inv.name, inv.code
```

# What Else Can SQL Do?

---

- SQL can insert records into a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables and views in a database
- SQL can create stored procedures in a database
- SQL can set permissions on tables, procedures, and views

# Lookups, Queries and Portlets



Let Rego be your guide.

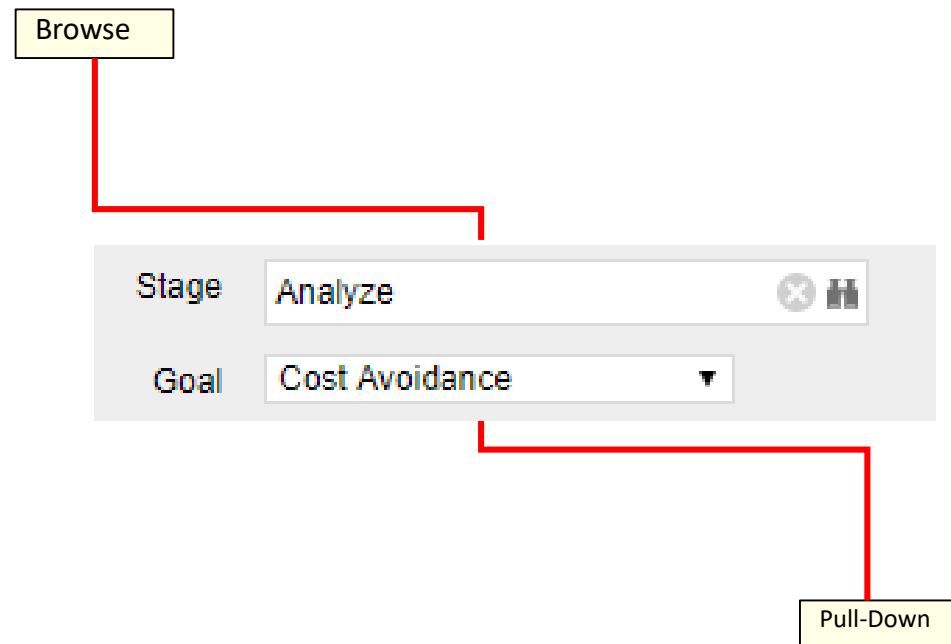
# Lookups

- What is a Lookup?
  - When attached to a field, a lookup allows users to select pre-defined values from a pull-down or browse list.
  - The lookup field's choices can be static values entered by an administrator, or dynamic values returned from a database query
  - Lookups values can be displayed as text or icons
  - 3 Source Types
    - Static List
    - Static Dependent List
    - Dynamic Query

# Lookups – Static List

- Static List Lookup

- Use this type of lookup when working with a standard set of values. Static list lookups are often used as pull-down/browse lists for object fields, portlets/reports, and custom forms.



# Exercise #1: Static Lookup

- Create a static list lookup containing the values Yes/No that we can use later during this training.
  - Navigate to Administration → Data Administration → Lookups
  - Create a new static list lookup using the below details
    - Name: Yes and No
    - Source: Static List
    - Values (corresponding id): No (0), Yes (1)



# Lookups (cont)

---

- Static Dependent Lookup
- Use this type of lookup to create a hierarchy of lookups and values.
- Items that appear on the second and subsequent lists depend upon choices previously made by the user.
- For example, if the user selects “USA” from a country browse list, then a state list may appear from which the user can select an appropriate state.

# Lookups (cont)

- Dynamic Query Lookup

- Use this type of lookup to capture data from the Clarity database in real time to populate the drop-down or browse lists.
- These lookups provide the most up-to-date values possible and are often used inside browse windows.
- Eliminates the need to maintain a list since the values are dynamically pulled from tables within the database (e.g. list of all resources)
- Dynamic queries are written in NSQL – more on this in a couple of slides

# Exercise #2: Dynamic Query Lookup

- Create a Dynamic NSQL Query to pull basic information for all investments in the system.
  - Navigate to Administration → Data Administration → Lookups
  - Create a new Query using the below details
    - Name: All Investments
    - Content Source: Dynamic Query
    - Query:

```
SELECT
@SELECT:code:Investment_ID@,
@SELECT:name:Investment_name@
FROM inv_investments
WHERE @FILTER@
```
  - Display Attribute: Investment\_name
  - Hidden Key: Investment\_ID

# Queries

- A Query is created and managed in Studio (Administration->Studio->Queries)
- Queries extract data from the system for use in portlets
- Similar to Dynamic Lookups, Queries are also written in NSQL
- NSQL is a version of SQL specific Clarity; it is used to designate query segments as metric values, dimensions, dimension properties, or parameters
- NSQL can only **select** data from a database, it cannot update data

# Queries (cont)

- The **SELECT** statement retrieves column data from tables
  - NSQL Queries must start with SELECT however for each column a @SELECT@ tag must be used.
- A dimension is a grouping of similar data elements from one or more tables
  - Defining Dimensions
    - <Dimension> is a user-defined name for the dimension
    - <Table.Field> is the table or alias name retrieved in the FROM statement
    - <label> is the name you want to appear in the column list in clarity

## *SELECT*

*@SELECT:DIM:USER\_DEF:IMPLIED:<Dimension>:<Table.Field>:<label>@*

- DIM: Indicates the line is the primary key for the dimension
- There can only be one DIM to each dimension.

*@SELECT:DIM\_PROP:USER\_DEF:IMPLIED:<Dimension>:<Table.Field>:<label>@*

- DIM\_PROP: Indicates columns for the dimension
- There can be many DIM\_PROPS defined to one dimension.

# Queries (cont)

- The **FROM** clause is a standard SQL statement which defines which table to gather data from  
*FROM <Table>*
- The **WHERE** statement filters data returned by a query to be used on portlets
  - The @FILTER@ statement is required and allows the system to filter the values defined with the @SELECT@ tag  
*WHERE <Condition>*  
*AND @FILTER@*
- The **GROUP BY** clause is typically used to combine database records with identical values in a specified field into a single record, usually for the purposes of calculating some sort of aggregate function
- The syntax for the **HAVING** statement is @HAVING\_FILTER@ which can be used when a query uses metrics
  - The Developer guide states this is required but it is **NOT**.

# Exercise #3: NSQL Query

- Create a new Query (nsql) to pull basic investment information
  - a. Navigate to Administration → Studio → Queries
  - b. Create a new Query using the below NSQL statement to extract the data

- a. Name: Investment Details Query

- b. Query:

```
SELECT  
@SELECT:DIM:USER_DEF:IMPLIED:INVESTMENT:code:id@,  
@SELECT:DIM_PROP:USER_DEF:IMPLIED:INVESTMENT:name:name@,  
@SELECT:DIM_PROP:USER_DEF:IMPLIED:INVESTMENT:schedule_start:start_date@,  
@SELECT:DIM_PROP:USER_DEF:IMPLIED:INVESTMENT:schedule_finish:finish_date@,  
@SELECT:DIM_PROP:USER_DEF:IMPLIED:INVESTMENT:is_active:is_active@
```

```
FROM inv_investments
```

```
WHERE @filter@  
HAVING @HAVING_FILTER@
```

- c. Associate the two of the attributes with the two lookups created in the earlier exercises.

# Portlets

- What is a Portlet?

- Portlets are snapshots of Clarity data and can consist of grids, graphs, or snippets of HTML
- Portlets do not replace Clarity reports but are often preferred due to ease of use and access
- Portlets obtain information and business intelligence from Clarity, other databases within the enterprise, and external sources available in HTML (for example, business news and network status information)
- Users can populate Portlets with graphs, tables, workflows, best practices, documents, and forms and have the information update in real-time without running a report



# Portlet Types

- **Chart Portlet** – A graphical view of CA PPM data (for example, pie and line charts)
- **Grid Portlet** – A list or table of data you can filter in real time. Can be sourced by Objects or Queries.
- **HTML Portlet** – Displays information on a CA PPM page from internal or external web sites formatted as HTML
- **Filter Portlet** – Applies a common filter to all Portlets on a page
- **Interactive Portlet** – Displays visually rich, real-time CA PPM data using imported Xcelsius visualizations

# Portlet Data Sources

## Query-based (NSQL)

- Portlets created using queries in Clarity to define the data that can be used.
- Pros
  - Logic
  - Parameters
  - Security
  - Customizable
  - Matrices
- Cons
  - Not Dynamic
  - Development Time
  - No In-Line Editing

## Object-based

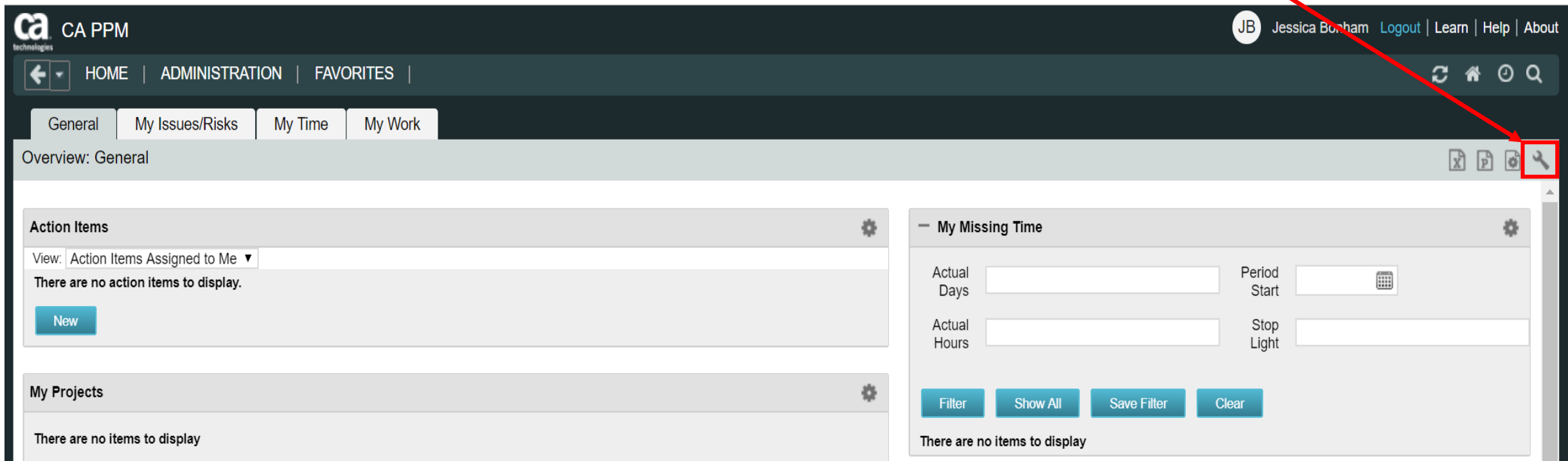
- Portlets created using an Object instead of a query to define the data set gathered.
- Pros
  - Customizable
  - Security
  - In-Line Editing
  - Dynamic
  - Time Scaled Values
- Cons
  - Multiple Objects Difficulty
  - No Custom Logic

# Exercise #4: Basic Grid Portlet

- Create a portlet that displays basic information for all investments in the system.
- Navigate to Administration -> Studio -> Portlets
- Create a new Grid Portlet using the below details
  - Name: Investment Details
  - Data Provider: Investments Details Query
  - List Layout:
    - ID, Name, Start Date, Finish Date
  - Filter Layout:
    - ID, Is Active?

# Exercise #4: Basic Grid Portlet (cont)

- Use the Manage My Tabs link on the toolbar at the far right to add the portlet to your home page:



The screenshot displays the CA PPM user interface. At the top, the header includes the CA PPM logo, user information (JB Jessica Bonham), and navigation links (Logout, Learn, Help, About). Below the header is a navigation bar with tabs for General, My Issues/Risks, My Time, and My Work. The main content area shows the Overview: General section. On the right side of the toolbar, there is a red box highlighting the Manage My Tabs link (represented by a wrench icon), with a red arrow pointing to it from the text above.

CA PPM technologies

JB Jessica Bonham Logout | Learn | Help | About

HOME | ADMINISTRATION | FAVORITES

General | My Issues/Risks | My Time | My Work

Overview: General

Action Items

View: Action Items Assigned to Me

There are no action items to display.

New

My Projects

There are no items to display

My Missing Time

Actual Days

Period Start

Actual Hours

Stop Light

Filter Show All Save Filter Clear

There are no items to display

# Introduction to Workflow (Processes)



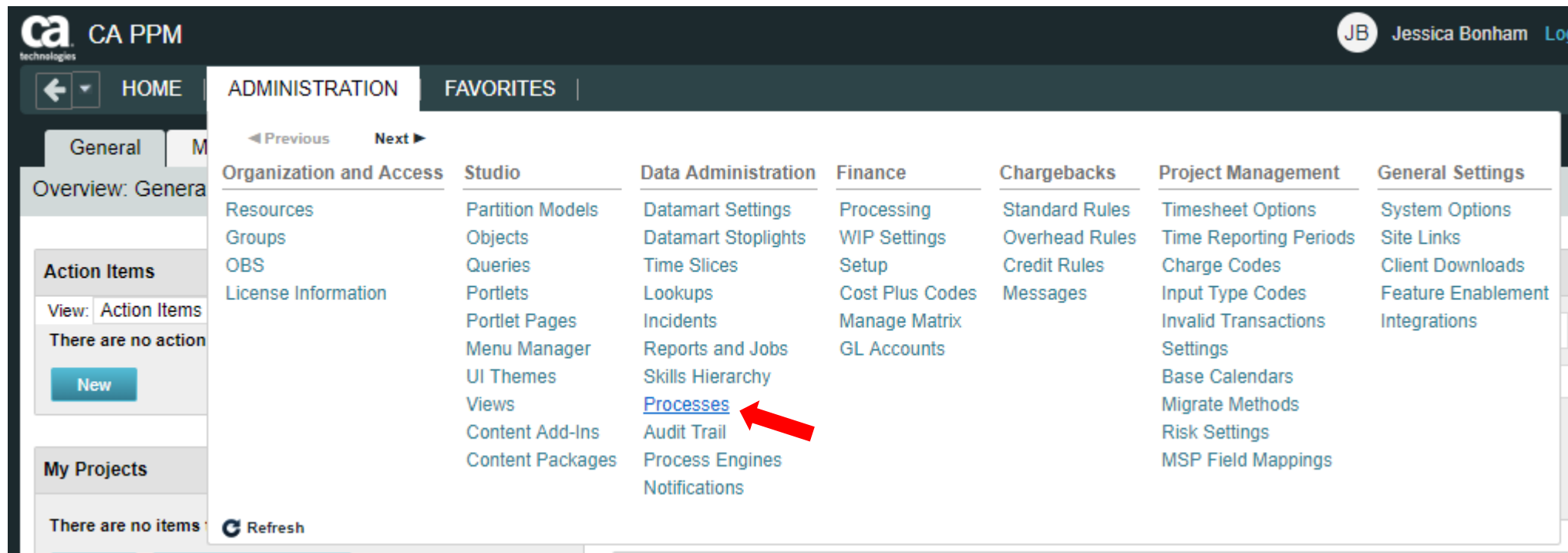
Let Rego be your guide.

# Processes: Overview

- What is a process?
- Processes automate repetitive steps that you would otherwise perform manually through the user interface
  - To accurately reproduce a user action, the process impersonates the process initiator to perform the process steps
  - A process includes a Start step, a Finish step, and if desired any number of steps in between
  - Each step performs one or more actions that move the process toward completion
  - Processes use pre and post conditions to connect the steps
- CA PPM provides stock processes that you can use to
  - Approve documents
  - Approve timesheets
  - Approve ideas

# Processes: Overview (cont)

- Process definitions are accessed via the Administration menu:



# Processes: Design Basics

---

- Creating a process involves the following:
  - Define the Process properties
  - Determine if it will be associated with a Clarity object
  - Set Start option
  - Determine/Define the number of steps required
  - Create Actions
  - Create pre and post-conditions if needed
  - Validate the process
  - Activate the process



# Processes: Design Basics (cont)

- Define Process Properties

- Set the name and ID of the process
- Processes are always created in “Draft” mode (default)

The screenshot shows a web-based form titled "Process Definition: Properties" with a tabbed interface. The "Properties" tab is selected. The form is divided into a "General" section and a "Status" section. In the "General" section, there are three required fields: "Process Name" (containing "MM Idea Approval"), "Process ID" (containing "mm\_idea\_approval"), and "Content Source" (a dropdown menu set to "Customer"). Below these is a "Description" text area. In the "Status" section, the "Status" is set to "Not Validated". Under "Mode", there are three radio buttons: "Active", "Draft" (which is selected), and "On Hold". A note below the "On Hold" option states: "( To change the mode, first validate the process. )". At the bottom of the form are four buttons: "Save and Continue", "Save", "Save And Return", and "Return". A legend at the very bottom indicates that a red asterisk (\*) means "Required", a green arrow (→) means "Enter Once", and a green asterisk (\*) means "Unique".

Process Definition: Properties

General

✳ Process Name MM Idea Approval

✳→✳ Process ID mm\_idea\_approval

✳ Content Source Customer

Description

Status Not Validated

Mode

☐ Active

☒ Draft

☐ On Hold

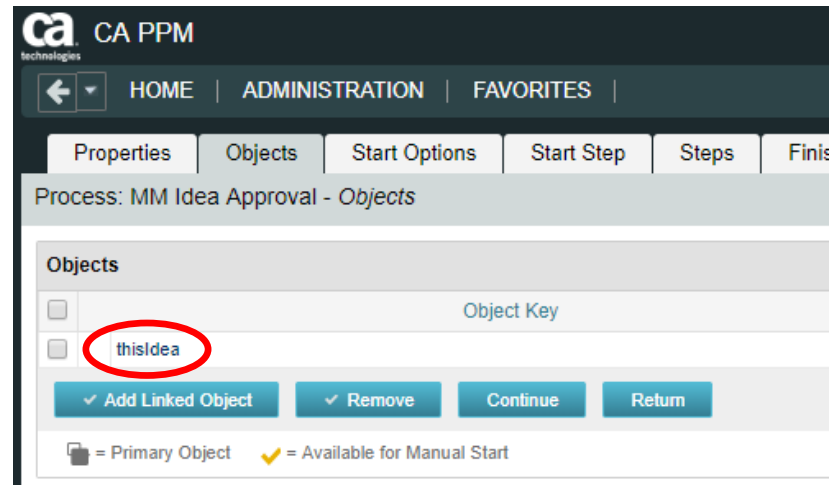
( To change the mode, first validate the process. )

Save and Continue Save Save And Return Return

✳ = Required → = Enter Once ✳ = Unique

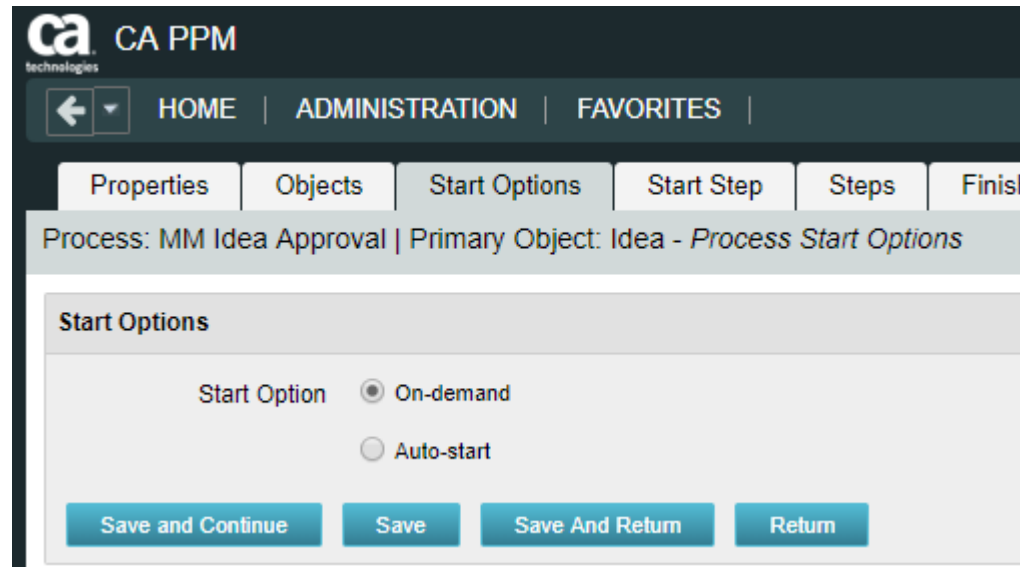
# Processes: Design Basics (cont)

- Associate the process with a Clarity object if:
  - The process should be triggered by a user update (or create)
  - The process will include system actions, manual actions or sub-processes
- In this example the process is associated with Ideas:



# Processes: Design Basics (cont)

- Set Start option
  - On Demand: process will be kicked off manually or by another process calling that process
    - Note: Only a process that is NOT associated with an object can be scheduled by a job
  - Auto-Start: process will be kicked off when a user updates or creates an object instance



The screenshot displays the CA PPM web application interface. At the top, the CA PPM logo is visible. Below it, a navigation bar includes links for HOME, ADMINISTRATION, and FAVORITES. A secondary navigation bar contains tabs for Properties, Objects, Start Options (which is currently selected), Start Step, Steps, and Finish. The main content area shows the breadcrumb 'Process: MM Idea Approval | Primary Object: Idea - Process Start Options'. The 'Start Options' section contains a 'Start Option' label with two radio button choices: 'On-demand' (which is selected) and 'Auto-start'. At the bottom of this section, there are four buttons: 'Save and Continue', 'Save', 'Save And Return', and 'Return'.

# Processes: Design Basics (cont)

- Define Process Steps

- Steps comprise the groups of actions that take place inside a process to accomplish a set of activities
- Process logic is the Pre-Condition or Post-Condition of each step
  - When defining a pre-condition to a step, you can use attributes from objects added to the process
  - Pre-conditions will determine whether the step should begin
  - After defining the pre-conditions that trigger a step, you must define post-conditions that connect this step to the next step or the Finish step
  - Post-conditions will determine where and whether the step will move
  - Examples of pre and post-conditions include:
    - Checking the status of action items
    - Checking between object attributes values (except for MVL attributes)
    - Waiting for a sub-process to complete before joining the master

# Processes: Design Basics (cont)

- Splits / Joins determine the direction of the process flow
  - A Split branches into multiple directions
  - A Join brings multiple branches back together

## Split Types

- Serial
- Parallel
- Decision Point
- Multi choice

## Join Types

- Rendezvous (AND)
- Merge (XOR)
- Wait and Merge
- Multi-thread
- First in Line

# Processes: Design Basics (cont)

## Example of Post-conditions and Split Type:

Process: Assign Incidents | Step: Acquire Incident - Step Details

**General**

Step Name: Acquire Incident

Step ID: AcquireIncident

Status: Validated

Description:

Group: [--Select--]

Raise a Warning After: Days

Milestone: ☐

**Referring Steps**

Assign IT Worker.Incident Not Resolved

**Pre-conditions**

Join Type: None

There is no precondition to display.

**Post-conditions**

Split Type: Decision Point (XOR)

Post-conditions will be checked in the order they are listed below.

If ...	Then Go To	Description
( Action Item.AcquireIncident.AcquireIncident Number of assignees with Status Done >= 1 )	Work On Incident	Incident Acquired
( Action Item.AcquireIncident.AcquireIncident Number of assignees with Status Rejected >= 1 )	Incident Escalated	Incident Not Acquired

**Notifications**

Send Notification: ☐ When step is started  
☐ When step is completed  
☐ When step is in error

Send Notification To:

Notify Owner: ☐

**Escalation**

There is no escalation rule setup to display.

**Actions**

Name	Description
Send acquire incident action item	Please review and acquire incident.

# Processes: Design Basics (cont)

## Example of Pre-condition Join Type:

Process: Assign Incidents | Step: Incident Escalated - Step Details

**General**

Step Name: Incident Escalated

Step ID: IncidentEscalated

Status: Validated

Description:

Group: [--Select--]

Raise a Warning After: Days

Milestone: ☐

**Referring Steps**

Resolution Verification, Acquire Incident

**Pre-conditions**

Join Type: Merge (XOR)

There is no precondition to display.

**Post-conditions**

Split Type: Serial

Post-conditions will be checked in the order they are listed below.

If ...	Then Go To	Description
Incident Not Resolved		

**Notifications**

Send Notification: ☐ When step is started  
☐ When step is completed  
☐ When step is in error

Send Notification To:

Notify Owner: ☐

**Escalation**

There is no escalation rule setup to display.

**Actions**

Name	Description
Escalate Incident	Set incident status to Escalated.

# Processes: Design Basics (cont)

- A step **Action** can be one of the following:
  - Manual Action
    - Example(s): Actions Items
    - With manual actions you can include some object data using variables
  - System Action
    - Example(s): Lock/Unlock Attributes, Set Attribute Values, and certain System Operations (copy a financial plan from a template)
  - Run a Job
    - Example(s): Post Timesheets
  - Custom Script (run a GEL script)
    - Example(s): Notification Scripts
  - Subprocess
    - Sub processes are invoked as embedded processes within the context of the current process



# Processes: Design Basics (cont)

## Example of step Action:

Process: Assign Incidents | Step: Assign IT Worker - Step Details

**General**

Step Name: Assign IT Worker

Step ID: AssignITWorker

Status: Validated

Description:

Group: [--Select--]

Raise a Warning After: Days

Milestone:

**Referring Steps**

Start:

**Pre-conditions**

Join Type: None

There is no precondition to display.

**Post-conditions**

Split Type: Serial

Post-conditions will be checked in the order they are listed below.

If ...	Then Go To	Description
	Acquire Incident	Incident Assigned

**Notifications**

Send Notification:

- ☐ When step is started
- ☐ When step is completed
- ☐ When step is in error

Send Notification To:

Notify Owner:

**Escalation**

There is no escalation rule setup to display.

**Actions**



Name	Description
Run assign incident job	Run assign incident job.

# Processes: Design Basics (cont)

- Validate and Activate Process

- Use the process validations page to monitor the latest validation statuses and errors at the step and process level
- Validation rules are used to validate steps and conditions and the structure of a process

Process: Assign Incidents - Process Validation

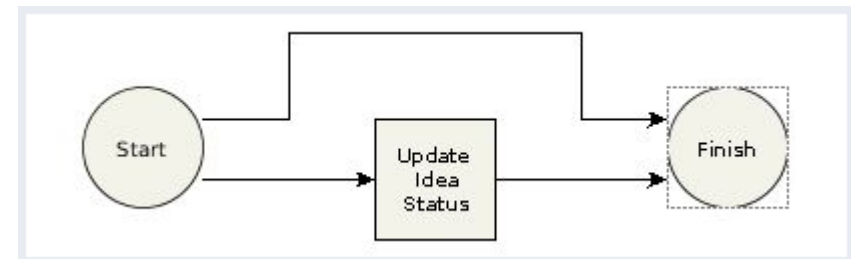
	Validation Object	Status
<input type="checkbox"/>	Record Effort	Validated
<input type="checkbox"/>	Resolution Verification	Validated
<input type="checkbox"/>	Incident Not Resolved	Validated
<input type="checkbox"/>	Incident Escalated	Validated
<input type="checkbox"/>	Process	Validated
<input type="checkbox"/>	<div><div> BPM-0664: The process contains cycle.</div><div>Cycle steps: Acquire Incident Work On Incident Record Effort Resolution Verification Incident Escalated Incident Not Resolved</div><div>Entry steps: Acquire Incident</div><div>Exit steps: Resolution Verification</div></div>	
<input type="checkbox"/>	<div><div> BPM-0664: The process contains cycle.</div><div>Cycle steps: Acquire Incident Incident Escalated Incident Not Resolved</div><div>Entry steps: Acquire Incident</div><div>Exit steps: Acquire Incident</div></div>	

Validated = Validated    Errors Encountered = Errors Encountered    Re-validation Required = Re-validation Required    Not Validated = Not Validated

Validate   Activate Process   Validate All and Activate   Continue   Return

# Processes: Exercise

1. Create a new process using your initials as an identifier
2. Link the process to the idea object
3. Make the process auto-start on update, with status set to “Submitted for Approval” and prior status not set to “Submitted for Approval”
4. Create one step in addition to the Start and Finish steps
5. Add an action to the new step that sets the idea status to Approved
6. Add a second action to the new step that sends an action item / notification to the idea owner that the idea has been approved
7. Once all steps are created, create links between the steps
8. Ensure the conditions point to the correct step
9. Validate and Activate the process
10. Now TRY IT OUT!! 😊



# XOG

*rego*University 2019  
SAN DIEGO

Let Rego be your guide.

# XOG: Overview

---

- Introduction
- Requirements
- Installation and Setup
- XOG Files and Procedure
  - Configuration – (Query, Portlet, Workflow, etc)
  - Web version (13.1 and below)
  - Data
- Best Practices
- Other XOG Methods and XML Creation

# XOG: Introduction

---

- CA PPM has a Web Services interface called XML Open Gateway (XOG)
- What XOG Does
  - Export data and configuration
  - Import data and configuration
- When to Use XOG
  - Move configuration or data from one environment to another
  - Handle data imports via batch processing

# XOG: Requirements

- Operating System
  - Microsoft Windows XP Pro or later (32 or 64 bit)
  - Mac OS X 10.4 or later
  - Linux
- Java Runtime Environment (JRE)
  - Oracle Java 7 Runtime Environment version 1.7.0\_25
  - Oracle Java 8 Runtime Environment version 1.8.0
- CA PPM
- XOG client version that matching the CA PPM
- CA PPM user with XOG Global Rights
  - Administration – XOG
  - Administration – Access
  - XOG rights for individual objects (for example, Resource, Project, OBS)

# XOG: Installation

- Download compatible JRE from <http://www.java.com/en/>
  - Installation folder: C:\jre8\
- Set the environment variables:
  - **JAVA\_HOME**=C:\jre8
  - **PATH**=;%JAVA\_HOME%\bin
- Test for Java using a command prompt
  - java -version

Command Prompt

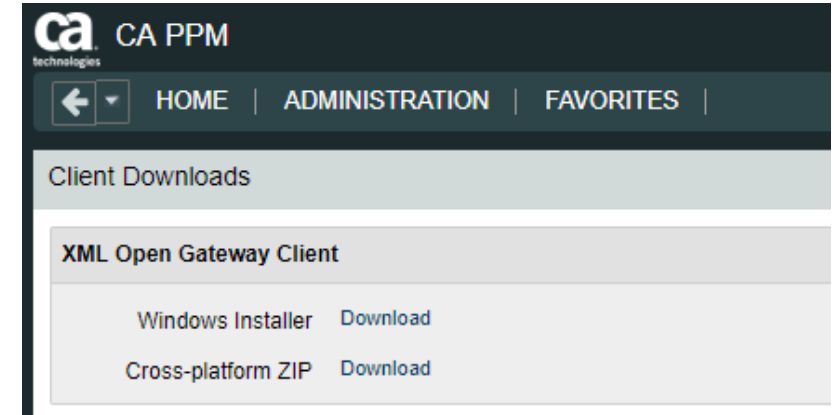
```
C:\>java -version
java version "1.8.0_172"
Java(TM) SE Runtime Environment (build 1.8.0_172-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.172-b11, mixed mode)

C:\>
```



# XOG: Installation (cont)

- Download the XOG client from the Admin tool
  - Download the cross-platform client
  - Extract the ZIP file to C:\xog\
- Test for XOG client using a command prompt
  1. Access C:\xog\bin\ folder
  2. Type xog and press enter
  3. Type exit to quit



```
C:\>cd xog/bin
C:\XOG\bin>xog
-----
PPM XML Open Gateway ( version: 15.4.0.270 )
-----
Usage: xog

login [user/passwd@host:port] Create new active session.
logout                        Close active session.
call <file>                   Invoke given XOG request.
verbose <on|off>              Turn on verbose error logging.
output <console|path>         Output to console or file.
exit                          Logout and exit the shell.
?                              Display this help information.

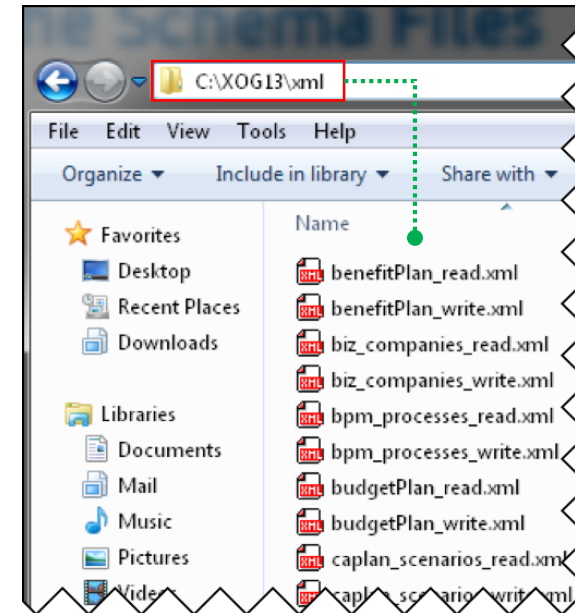
Examples:
  call xml/cmn_users_read.xml
>
```

# XOG: Verify Connectivity

- Use XOG client shell commands to test the connection between the XOG client and CA PPM server as follows:
  1. Open a command prompt
  2. Type **cd C:\xog\bin\** and press *enter*
  3. Type **xog (xog -sslenabled true** if your connection is using HTTPS) and press *enter*
  4. Type **login** *<username>/<mypassword>@<myserver>:<port>* and press *enter*  
For example, if your username was rodmi03, your password was Niku2000, and your CA PPM instance was https://cppm1234-dev.ondemand.ca.com/niku on port 443, you would type:  
**login** *rodmi03/Niku2000@cppm1234-dev.ondemand.ca.com:443*
  5. Verify login succeeded

# XOG: XML Files

- The XML files are valid examples of read and write requests that can run using the XOG client
- There are XML files for each CA PPM object you can manipulate with XOG (for example, Resource, Project, Group)
- XML files come in pairs
  - Read (Export)
  - Write (Import)
- Access the XML files from the XOG client installation folder (C:\xog13\xml)



# XOG: XML Read Files

- Use the XML Read files to export a specific item from CA PPM
- Each Read XML file contains the following structure:
  - **Header:** Supported CA PPM version, Operation (Read), and the object (Resource, Project, etc.)
  - **Arguments:** The type of information associated to the object to be included in the export (for example, include tasks and team members for projects)
  - **Query filters:** Limit the export data to (for example, Export only Project-A23 and Project-B89)

# XOG: XML Read Files

The Query Filter section supports criteria values to limit the scope of the export and accepts EQUALS, OR, BETWEEN, AFTER, BEFORE

```
<Header version="6.0.11" action="read" objectType="project" externalSource="NIKU">
  <args name="include_tasks" value="false"/>
  <args name="include_resources" value="false"/>
</Header>
<Query>
  <Filter name="projectId" criteria="EQUALS">prj123</Filter>
  <Filter name="projectId" criteria="OR">prj123,prj244</Filter>
  <Filter name="start" criteria="BETWEEN">2007-01-07,2009-01-15</Filter>
</Query>
```

# XOG: XML Read Files

- Use the % character as a wildcard

```
<Filter name="projectID" criteria="EQUALS">prj1%</Filter>
```

- Alternatively, you can filter objects based on custom attributes created using Clarity Studio

```
<FilterByCustomInfo name="attribute_id" criteria="EQUALS">prj1</FilterByCustomInfo>
```

- The regular criteria values apply to the Query Filter By Custom section (EQUALS, OR, BETWEEN, AFTER, BEFORE)

# XOG: Properties File

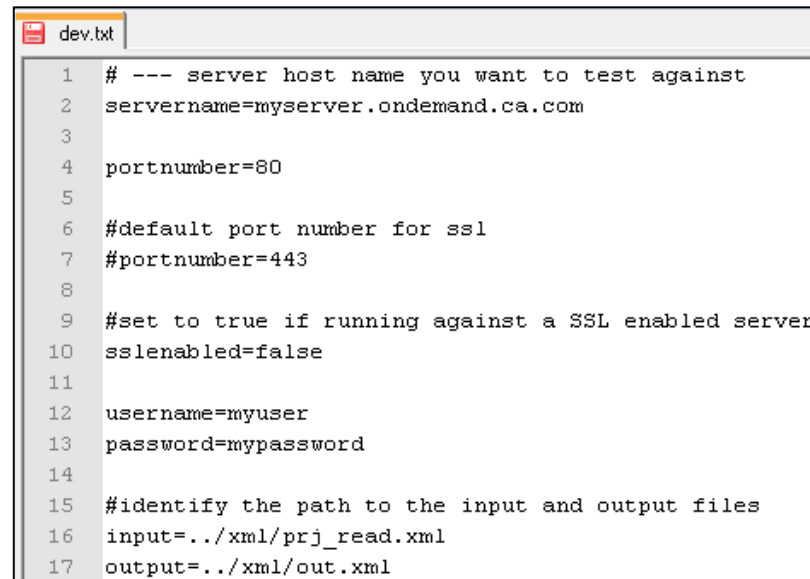
- You can submit a XOG request using XOG client shell commands:

**xog -servername <host> -portnumber <port>**  
**-username <username> -password <password>**  
**-input <input filepath> -output <output filepath>**

- You can create a .properties file to store the parameters for common XOG requests
  - Use the example .properties file provided with the XOG Client (**C:\xog13\bin\test.properties**)
  - Store the new .properties file in the bin directory
  - Name the file whatever you want ( for example, dev.txt, test.txt, prod.txt)
  - Use a simple text editor like MS Notepad

# XOG: Properties File

- The following properties are required to make a XOG request
  - **servername**=myserver
  - **portnumber**=80 | 443
  - **sslenabled**=false | true
  - **username**=myuser
  - **password**=mypassword
  - **input**=../xml/prj\_read.xml
  - **output**=../xml/out.xml



```
1 # --- server host name you want to test against
2 servername=myserver.ondemand.ca.com
3
4 portnumber=80
5
6 #default port number for ssl
7 #portnumber=443
8
9 #set to true if running against a SSL enabled server
10 sslenabled=false
11
12 username=myuser
13 password=mypassword
14
15 #identify the path to the input and output files
16 input=../xml/prj_read.xml
17 output=../xml/out.xml
```



# XOG: Exercise #1 – Export Data

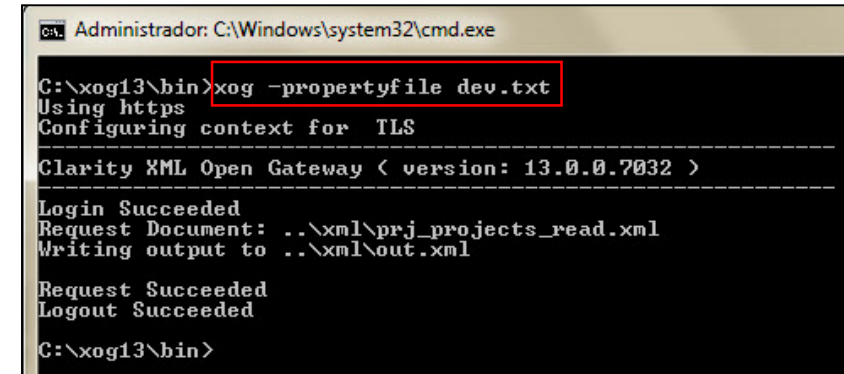
- Submit a XOG Read request using the XOG client as follows
  - Create a .properties file with the default values for the XOG parameters
  - Create an input XML file with the necessary header information, arguments, and query filters
  - Navigate to the “bin” folder under the XOG client installation folder by typing cd C:\xog13\bin\ and pressing enter
  - Type **xog -propertyfile** *<properties.txt>*
  - Verify the operation succeeded and check the output file

# XOG: Exercise #1 – Export Data

## Properties File

- # --- server host name you want to test against
- servername=myserver.ondemand.ca.com
- #portnumber=80
- #default port number for ssl
- portnumber=443
- #set to true if running against a SSL enabled server
- sslenabled=true
- username=myuser
- password=mypassword
- #identify the path to the input and output files
- input=../xml/prj\_projects\_read.xml
- output=../xml/out.xml

## XOG Commands



```
Administrator: C:\Windows\system32\cmd.exe
C:\xog13\bin>xog -propertyfile dev.txt
Using https
Configuring context for TLS
-----
Clarity XML Open Gateway < version: 13.0.0.7032 >
-----
Login Succeeded
Request Document: ../xml/prj_projects_read.xml
Writing output to ../xml/out.xml

Request Succeeded
Logout Succeeded
C:\xog13\bin>
```

# XOG: Exercise #1 – Export Data

## Input File

```
<?xml version="1.0" encoding="UTF-8"?>
<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../xsd/nikuxog_read.xsd">
  <Header version="6.0.11" action="read" objectType="project" externalSource="NIKU">
    <!-- you change the order by simply swap 1 and 2 number in the name attribute -->
    <args name="order_by_1" value="name"/>
    <args name="order_by_2" value="projectID"/>
    <args name="include_tasks" value="true"/>
    <args name="include_dependencies" value="false"/>
    <args name="include_subprojects" value="false"/>
    <args name="include_resources" value="false"/>
    <args name="include_baselines" value="false"/>
    <args name="include_allocations" value="false"/>
    <args name="include_estimates" value="false"/>
    <args name="include_actuals" value="false"/>
    <args name="include_custom" value="false"/>
    <args name="include_burdening" value="false"/>
  </Header>
  <Query>
    <Filter name="projectID" criteria="EQUALS">csk.%</Filter>
  </Query>
</NikuDataBus>
```

# XOG: Exercise #1 – Export Data

## Output File

```
<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.w3.org/2001/XMLSchema-instance" >
  <Header action="write" externalSource="NIKU" objectType="project" version="13.0.0.7032"/>
  <Projects>
    <Project active="true" approved="false" approvedForBilling="1" assgnPool="0" billingCurrencyCode="USD"
equipmentExchangeRateType="AVERAGE" evCalcMethod="0" expenseExchangeRateType="AVERAGE" finish="2012-04-26T03:00:01" materialExchangeRateType="AVERAGE" name="Application Change Template" openForTimeEntry="true"
csk.appChange="true" requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-04-09T08:00:00" useSystemDefinedTotalCostOfCapital="true">
    <Project active="true" approved="false" approvedForBilling="1" assgnPool="0" billingCurrencyCode="USD"
expenseExchangeRateType="AVERAGE" financialStatus="O" finish="2007-04-26T10:40:00" format="11" fullPageLayoutCode="da" materialExchangeRateType="AVERAGE" name="Application COTS Template" openForTimeEntry="true" pageLayoutCode="da"
requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-04-09T08:00:00" useSystemDefinedTotalCostOfCapital="true">
    <Project active="true" approved="false" approvedForBilling="1" assgnPool="0" billingCurrencyCode="USD"
expenseExchangeRateType="AVERAGE" financialStatus="O" finish="2007-06-25T17:00:00" format="11" fullPageLayoutCode="da" materialExchangeRateType="AVERAGE" name="IT Infrastructure Deployment Template" openForTimeEntry="true" pageLayoutCode="da"
requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-05-01T08:00:00" useSystemDefinedTotalCostOfCapital="true">
    <Project active="true" approved="false" approvedForBilling="1" asOf="2007-03-15T00:00:00" assgnPool="0" billingCurrencyCode="USD"
equipmentExchangeRateType="AVERAGE" evCalcMethod="0" expenseExchangeRateType="AVERAGE" finish="2007-03-12T08:00:00" materialExchangeRateType="AVERAGE" name="New IT Project" openForTimeEntry="true" pageLayoutCode="da"
requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-03-12T08:00:00" useSystemDefinedTotalCostOfCapital="true">
  </Projects>
  <XOGOutput>
    <Object type="project"/>
    <Status state="SUCCESS"/>
    <Statistics failureRecords="0" insertedRecords="0" totalNumberOfRecords="4" updatedRecords="0"/>
    <Records/>
  </XOGOutput>
</NikuDataBus>
```

# XOG: XML Write Files

- Use the XML Write files to import a specific object into CA PPM
- Each Write XML file contains the following structure:
  - **Header:** Supported CA PPM version, Operation (Write), object type (Resource, Project, etc.)
  - **Body:** Data to import
- You can create XML write files manually or by modifying the XML Write file examples provided with the XOG client or by using the output of an XML read request
- The output file from a Read response becomes the input file when moving data from one system to another

# XOG: XSD Files

- The XSD files are the XML Schema Definition files
- Each XSD file is used to validate the structure and content of the XML write file
- XSD files contain constraints such as required fields, field lengths, accepted values, etc.
- A valid XML editor can be used to validate an XML Write file against it's schema definition prior to loading to CA PPM
  - If the file is not valid an error will be thrown with the validation error in the output XML file during the write

# XOG: Exercise #2 – Import Data

- Submit a XOG Write request using the XOG client as follows
  - Create a .properties file with the default values for the XOG parameters
  - Create an input XML file with the necessary header and import data
  - Navigate to the XOG client installation “bin” folder by typing `cd C:\xog13\bin\` and pressing enter
  - Type **xog -propertyfile** *<properties.txt>* and press enter
  - Verify the operation succeeded and check the output file

# XOG: Exercise #2 – Import Data

## Properties File

- # --- server host name you want to test against
- servername=myserver.ondemand.ca.com
- #portnumber=80
- #default port number for ssl
- portnumber=443
- #set to true if running against a SSL enabled server
- sslenabled=true
- username=myuser
- password=mypassword
- #identify the path to the input and output files
- input=../xml/groups\_write\_allRights.xml
- output=../xml/out.xml

## Input File

- `<NikuDataBus  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:noNamespaceSchemaLocation="../xsd/nikuxog_group.xsd">`
- `<Header action="write" externalSource="NIKU"  
objectType="group" version="7.5"/>`
- `<groups>`
  - `<group code="rodmi03.AllRights" isActive="true">`
    - `<nls languageCode="en" name="All Rights"/>`
    - `<members>`
      - `<resource userName="admin"/>`
    - `</members>`
    - `<rights>`
      - `<GlobalRights/>`
      - `<InstanceRights/>`
      - `<InstanceOBSRights/>`
    - `</rights>`
  - `</group>`
- `</groups>`
- `</NikuDataBus>`



# XOG: Exercise #2 – Import Data

## XOG Commands

```
Administrator: C:\Windows\system32\cmd.exe

c:\XOG\XOG130\bin>xog -propertyfile dev.txt
Using https
Configuring context for TLS

-----
Clarity XML Open Gateway < version: 13.0.0.7032 >
-----

Login Succeeded
Request Document: ..\xml\groups_write_allRights.xml
Writing output to ..\xml\out.xml

Request Succeeded
Logout Succeeded
```

## Output File

- `<XOGOutput xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../xsd/status.xsd">`
- `<Object type="group"/>`
- `<Status elapsedTime="1.079 seconds" state="SUCCESS"/>`
- `<Statistics failureRecords="0" insertedRecords="1" totalNumberOfRecords="1" updatedRecords="0"/>`
- `<Records/>`
- `</XOGOutput>`

# XOG: Common Errors

Error	Description
Login Failed	Verify username and password are correct by accessing CA PPM with the same credentials.
No valid input file specified	Verify the file and directory indicated in the input parameter are valid.
Unexpected end of file from server	Check to see if the connection uses SSL (CA PPM URL begins with https://). If the connection uses SSL, set the sslenabled parameter to true.
Failed to retrieve response document java.io.FileNotFoundException:	Verify the directory indicated in the output parameter exists.
HTTP Error: Status-Code 504: Gateway Timeout	The XOG Client cannot connect to the Clarity server. Verify the connection port and test connectivity.
[Fatal Error] :5:47: The entity name must immediately follow the '&' in the entity reference.	Make sure you have escaped all special characters.
[Fatal Error] :6:14:	Verify there are no syntax errors in the XML file.

# XOG: Best Practices

- Use an XML Editor that supports color syntax highlighting, UNICODE, verification, and validation of XML files
  - Altova XMLSpy (License)
  - Notepad ++ (Open Source)
  - XML Pad (Freeware)
  - XML Copy Editor (Open Source)
- Use the XML files from the installation folder of the XOG client as a baseline to create your own XML files
- Verify XML file syntax is correct and validate the XML files against the object schema before running XOG
- Make sure the CA PPM server and XOG client versions match

# XOG: Creating XML

---

- In addition to manually creating XML for loading to CA PPM other methods can be utilized to generate XML
  - GEL scripting (both locally and in a process)
  - Perl scripting
  - Mail Merge
  - Excel formulas
  - 3rd party tools such as Kettle, etc.

# XOG: Special Characters

- XML predefines the following five entity references for special characters that need to be escaped (to prevent them from being considered part of the markup)

Name	Character	Escaped
Ampersand	&	&amp;
Left angle bracket	<	&lt;
Right angle bracket	>	&gt;
Straight quotation mark	"	&quot;
Apostrophe	'	&apos;&apos;

- Use CDATA ( `<![CDATA[ ..... ]]>` ) to escape special characters like SQL code

# XOG: Special Characters

- Escaping special characters example:

```
<Action code="setInitiator" synchronized="true" type="BPM_SAT_CUSTOM">
  <nls languageCode="en" name="Set Process &amp; Initiator"/>
  <customScript languageCode="gel">
    <scriptText>
      <gel:script xmlns:core="jelly:core"
xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary" xmlns:sql="jelly:sql"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <gel:setDataSource dbId="Niku" var="dataSource"/>
        <sql:update dataSource="{dataSource}"><![CDATA[
          SELECT
            gg_totcost_currency GTC,
            CODE,
          FROM
            ODF_CA_GG_ECOACTIVITY
          WHERE
            id=${gel_objectInstanceId}]]></sql:update>
        </gel:script>
      </scriptText>
    </customScript>
  </Action>
```

# Questions?



*rego*University 2019  
SAN DIEGO

Let Rego be your guide.

# Thank You For Attending regoUniversity

## Instructions for PMI credits

- Access your account at pmi.org
- Click on **Certifications**
- Click on **Maintain My Certification**
- Click on **Visit CCR's** button under the **Report PDU's**
- Click on **Report PDU's**
- Click on **Course or Training**
- Class Name = **regoUniversity**
- Course Number = **Session Number**
- Date Started = **Today's Date**
- Date Completed = **Today's Date**
- Hours Completed = **1 PDU per hour of class time**
- Training classes = **Technical**
- Click on **I agree** and **Submit**



Let us know how we can improve!  
Don't forget to fill out the class survey.



### Phone

888.813.0444



### Email

[info@regouniversity.com](mailto:info@regouniversity.com)



### Website

[www.regouniversity.com](http://www.regouniversity.com)