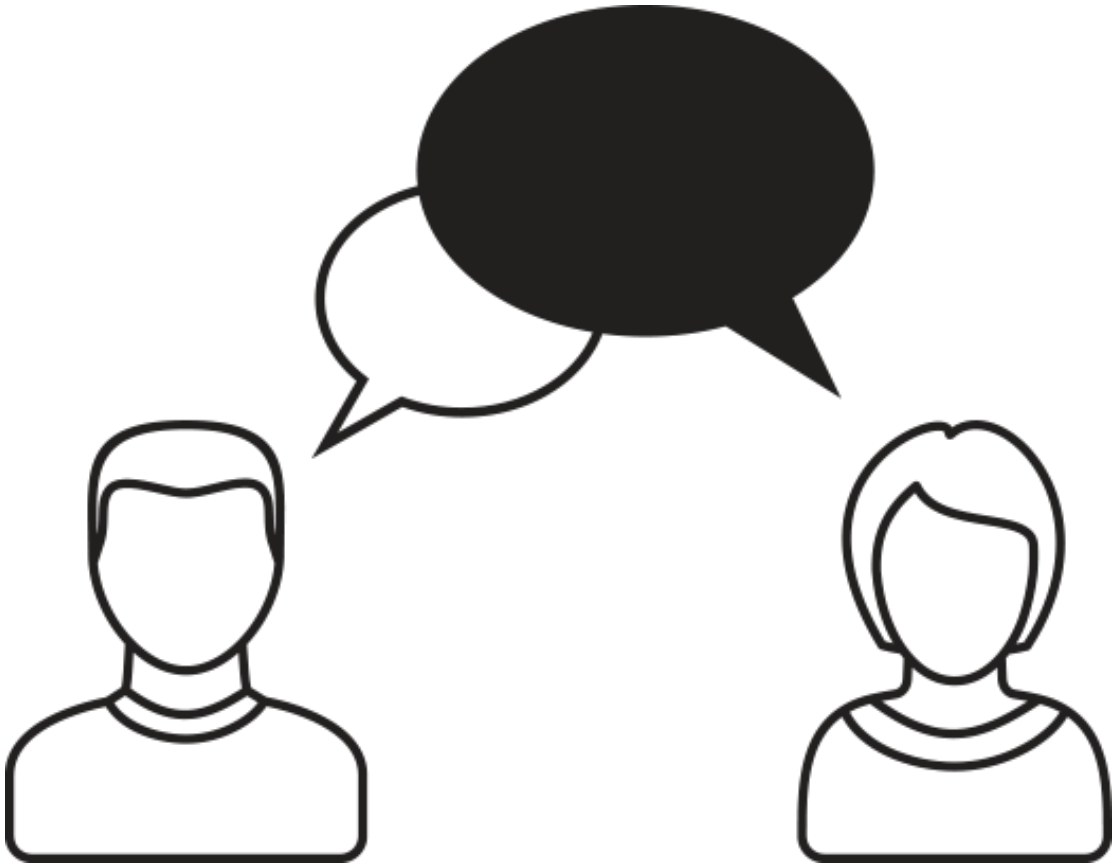

regoUniversity 2017
Administration | Advanced
Your Guides: Anthony Alcala, Marlon McKenzie, Jenn Rinella

# Introductions

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



# Agenda

- Objects and Fields
- Basic SQL
- Object Portlets
- Basic Processes
- XOG

# Objects and Fields

regoUniversity 2017

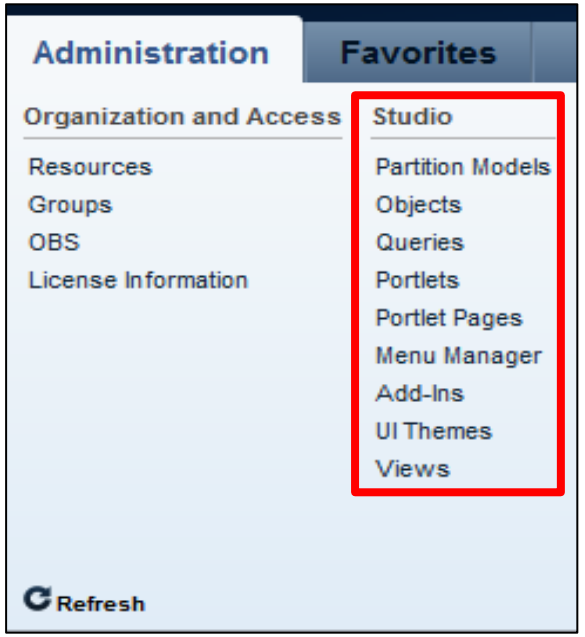
# Objects And Fields: Course Outline

- CA PPM Studio Overview
- Objects
- Attributes and Fields
- Views
- Action Menu

# CA PPM Studio

CA PPM Studio is the interface used to create and deploy portals, dashboards, menus, and objects that can be configured or customized to match organization needs

- An organization must have a CA PPM Studio license to use this functionality
- The user must have “Administration-Studio” access assigned as well as rights to create/edit Objects, Portlets, and Pages



# 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

- 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

# The Investment Object

- Allows you to define object attributes used across multiple objects (Project, Idea, Application, Asset, Product, Service, Other Work)
- Streamlines the creation process and ensures consistency across objects
- You may re-label attributes on shared objects if needed (Attribute 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

# Objects Types

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

# Exercise #1: Create an Object

- Create a new custom sub-object to the project object
  a. Administration -> Objects


  b. Click New and fill out the required fields (see next slide)

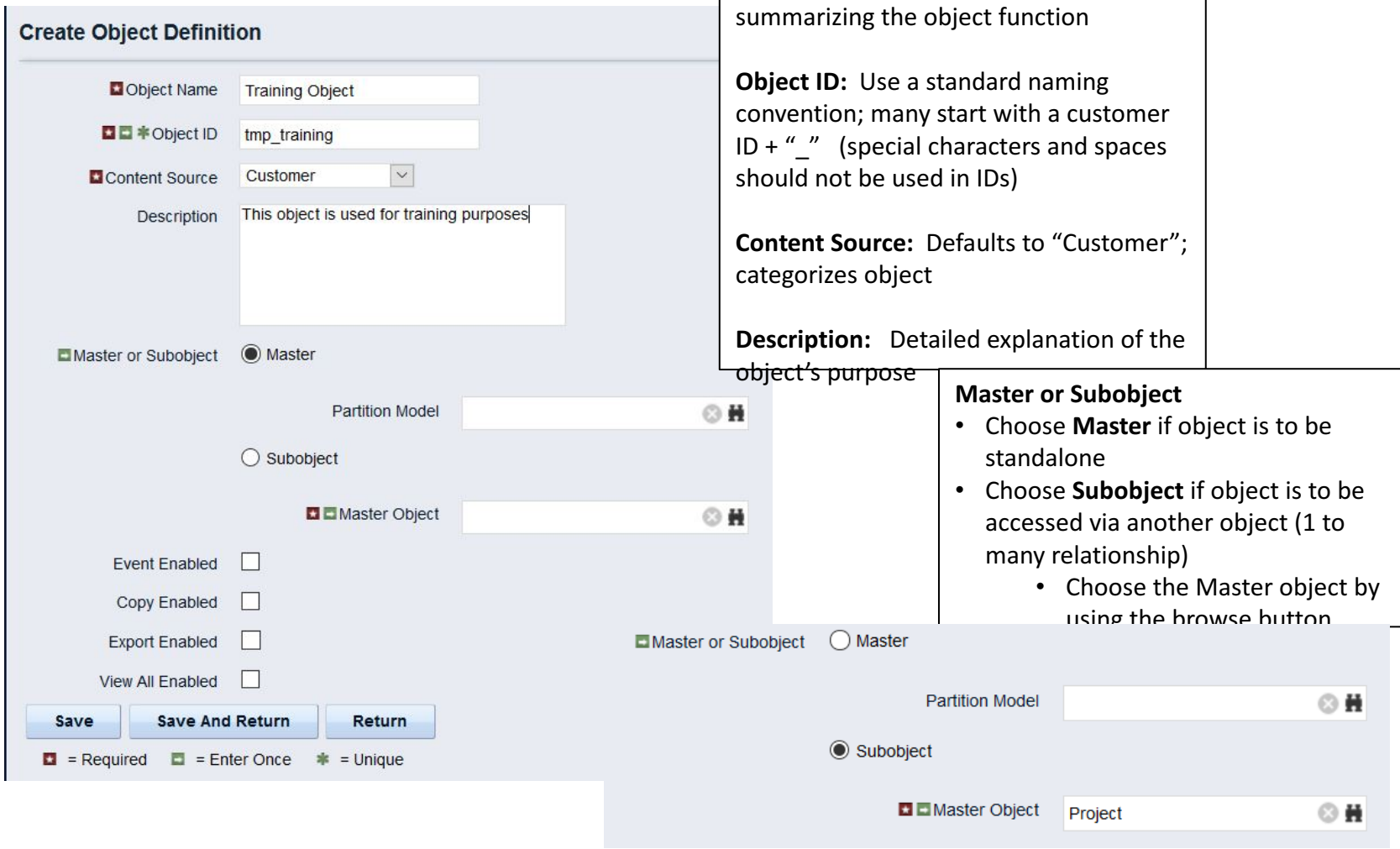
  c. Select the following checkboxes if they apply
     - **Event Enabled:** Specifies that the process engine is notified of object instances that are created or updated. (If a process needs to get driven off the object)
     - **Copy Enabled:** Specifies that copies can be made of the object instances.
     - **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.

- Notice the default fields included in the newly created object

# Exercise #1: Create An Object

a. 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.

➢ Notice the default fields included in the newly created object




**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)

**Content Source:** Defaults to "Customer"; categorizes object

**Description:** Detailed explanation of the object's purpose

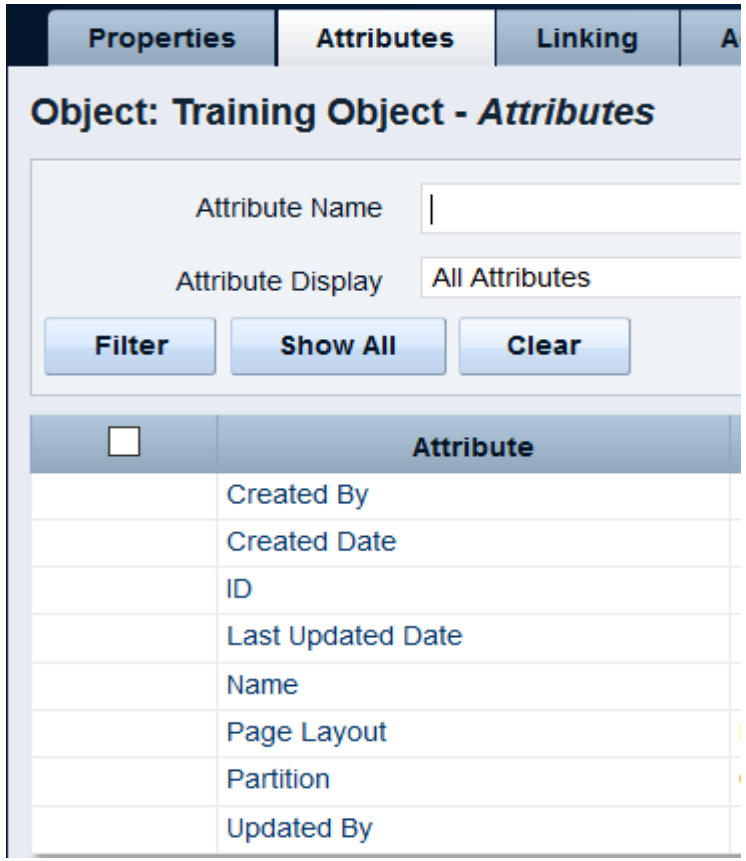
**Master or Subobject**
- Choose **Master** if object is to be standalone
- Choose **Subobject** if object is to be accessed via another object (1 to many relationship)
  - Choose the Master object by using the browse button

# Exercise #1: Create an Object

- 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.

# Exercise #1: Create an Object

- Notice the default fields included in the newly created object




**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 skew 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.

# Objects: Attributes

- Attributes are the fields on any 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 CA PPM 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

# Objects: Attribute Data Types

- When creating a new attribute, the procedure used depends on the data type selected
- The following data types are available for creating attributes
    - String (2000 character maximum)
    - Large String
    - Number
    - Formula
    - Money (includes currency code)
    - Boolean (checkbox)
    - Date
    - Lookup (related lookup needs to be available / created prior to creating attribute)
    - Multi-Valued lookup (related lookup needs to be available / created prior to creating attribute)
    - Attachment
    - Time-varying
    - URL (Links to actual data)
    - Virtual fields (Not actual data)

** More in depth descriptions of how to create each field is in the Studio Developer Guide (pg. 25)

# Objects: Calculated Fields

- A Calculated Field is an attribute that displays a dynamically-calculated read-only value
- Values are calculated from existing system values
- Values are calculated when accessing the page
- Values are not stored in the database
- Calculated fields can be only one of the following data types
  - **Number**: Use this data type when a calculated attribute requires a number value like a sum or average
  - **String**: Use this data type when a calculated attribute requires the concatenation of two or more values
    - Example: a concatenation of the value of the attribute “created_by” and the constant “2007” would 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

# Objects: Calculated Fields

- You cannot use the following attribute types with calculated attributes
  - Formula
  - Time-varying
  - Attachment
  - Long String
  - Multi-Value Lookups
  - Virtual

**Note:** You cannot delete source fields for a calculated attribute

# Objects: Auto Numbering

- Often businesses want a meaningful unique identifier within object instances;  Auto-numbering using text and numeric values will accomplish this
  - Within an object select an attribute, usually the ID and/or the Name
  - Select the auto-numbering tab
  - Select Scheme -> Edit
  - The default segment type is numeric but this can be modified to include text characters as well
  - Set the counter length

# Objects: Lookups

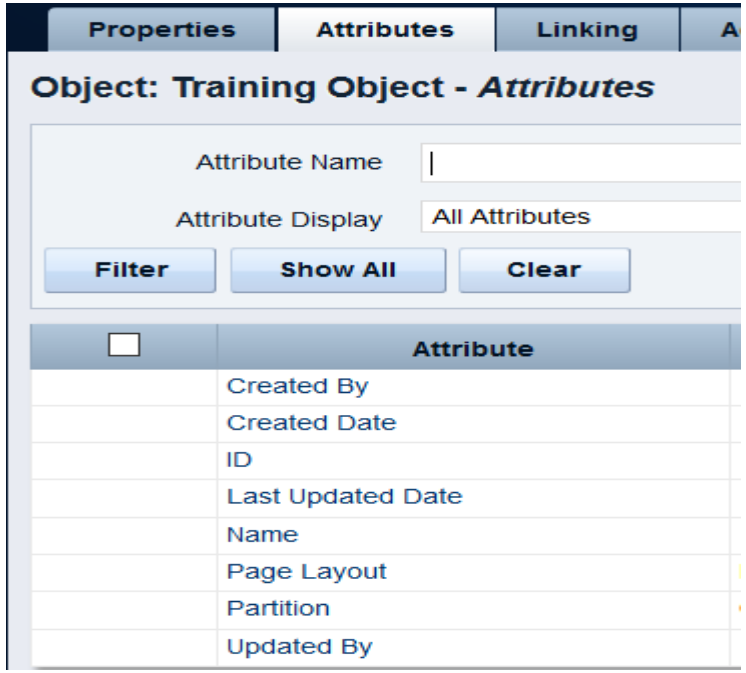
- A lookup is a field the user can select from a drop-down or pull down a list of pre-defined choices
- Lookup field choices can be static values entered by an administrator, or dynamic values returned from a database query
- Lookups can display as a value or an icon

| Lookup Type | Description |
|---|---|
| Static List | Use this type of lookup when working with a standard set of values. Static list lookups are often used as pull-down lists for fields, reports, and custom forms. |
| Static Dependent Lists | 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. |
| Dynamic Queries | Use this type of lookup to capture data from the CA PPM database in real time to populate the drop-down or browse lists. (Using NSQL) These lookups provide the most up-to-date values possible and are often used inside browse windows. |

**Note:** You can nest a static lookup inside a dynamic query lookup. You cannot nest a static dependent lookup inside a dynamic query lookup.

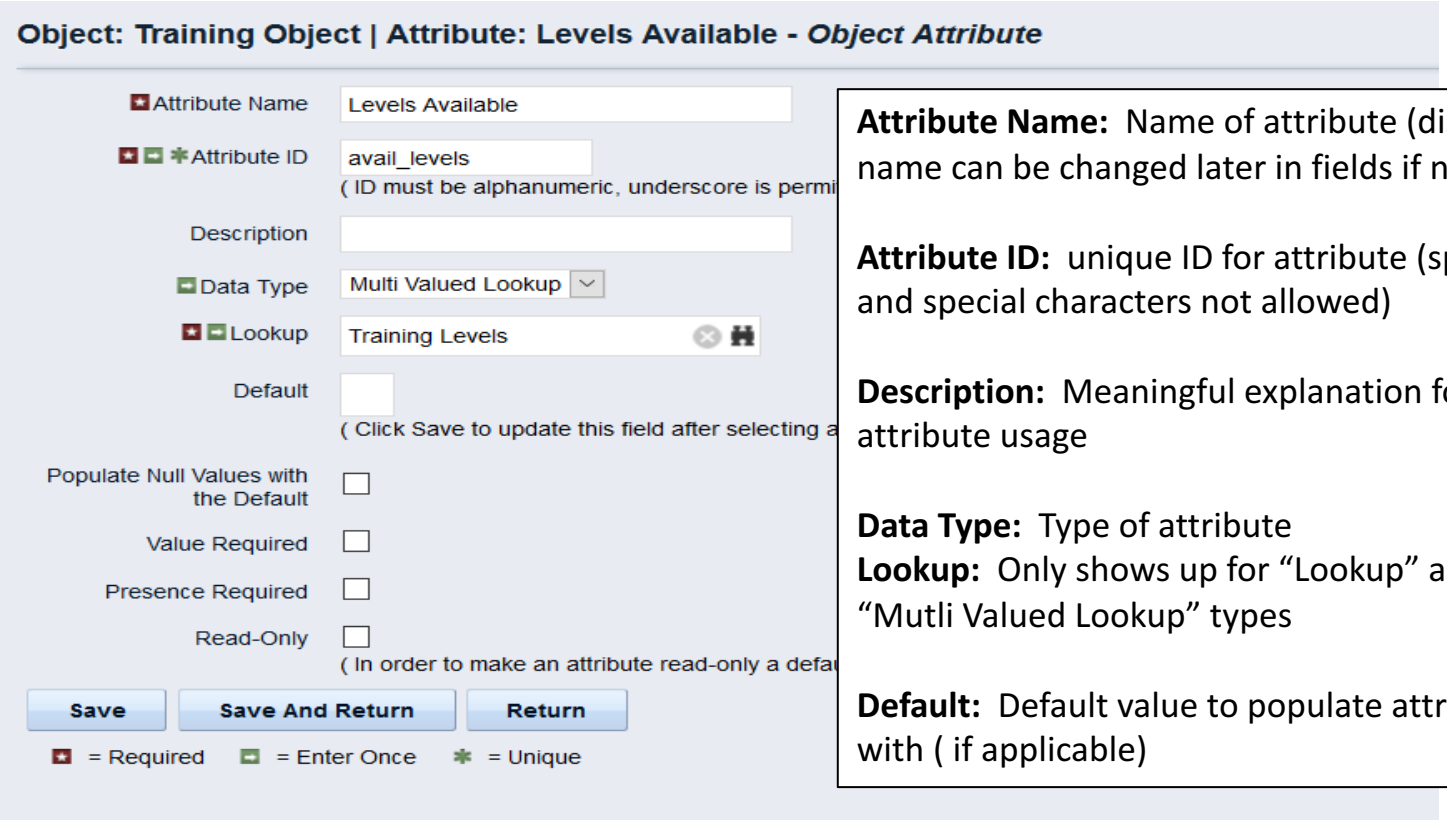
# Exercise #2: Objects And Attributes

- Add at least 3 or 4 attributes of different varieties to your new custom sub-object
  a. Administration -> Objects -> <*New Object*> -> Attributes



# Exercise #2: Objects And Attributes

b. Click New and fill out required fields as well as Data Type of new field




**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)

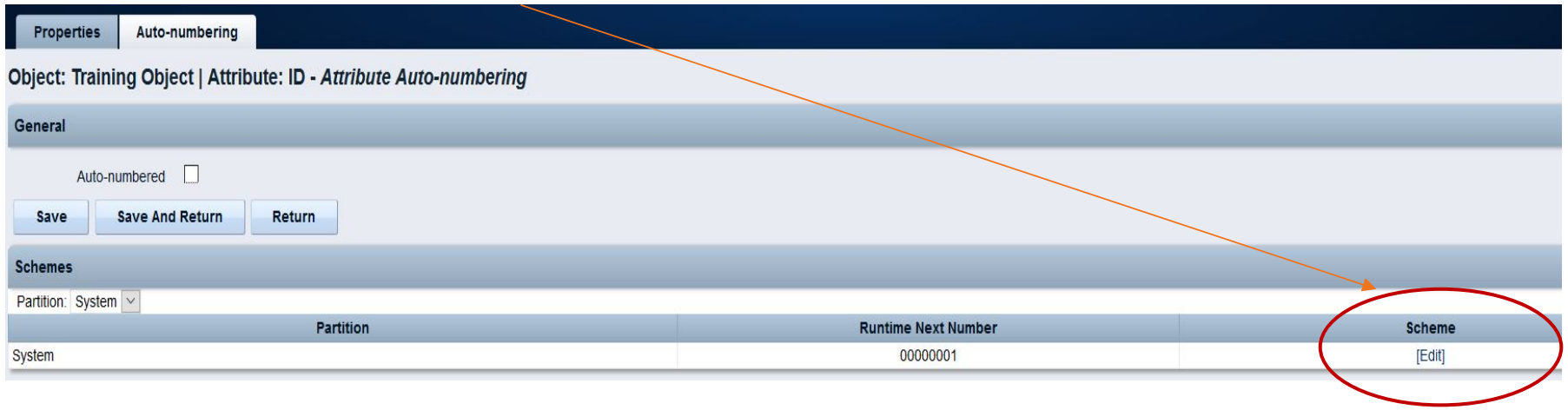
# Exercise #2: Objects And Attributes


- Select the following checkboxes if they apply:
  - **Populate Null Values with the Default:** If an attribute is created later on and instances have already been created this will populate the new attribute values with the default value set
  - **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

# Exercise #2: Objects And Attributes

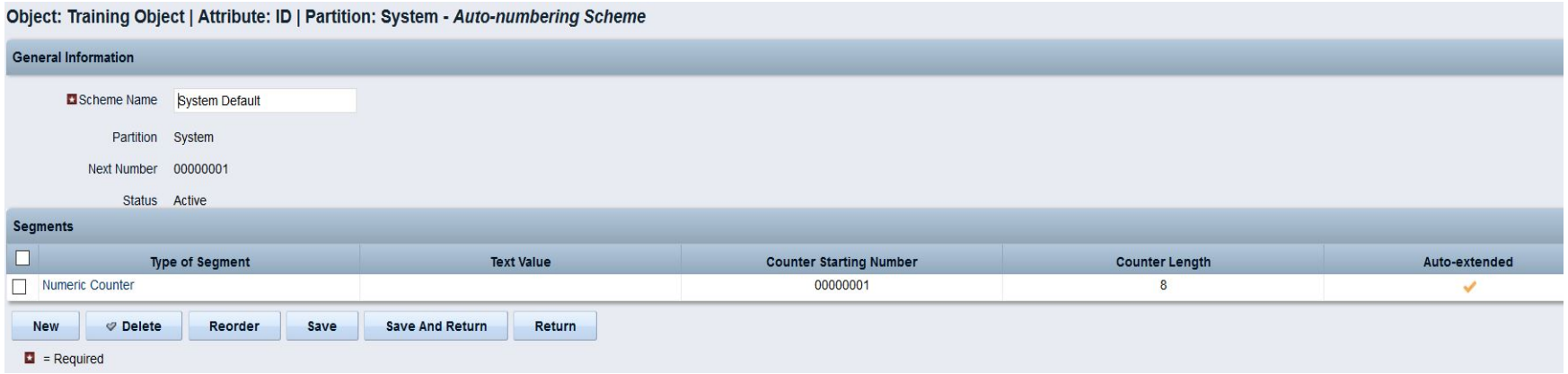
## Auto-number the ID attribute

a. Within an object select an attribute, usually the ID and/or the Name
b. Select the auto-numbering tab
   - Select the “Auto-numbered” checkbox and click “Save”
   - Select Scheme -> Edit

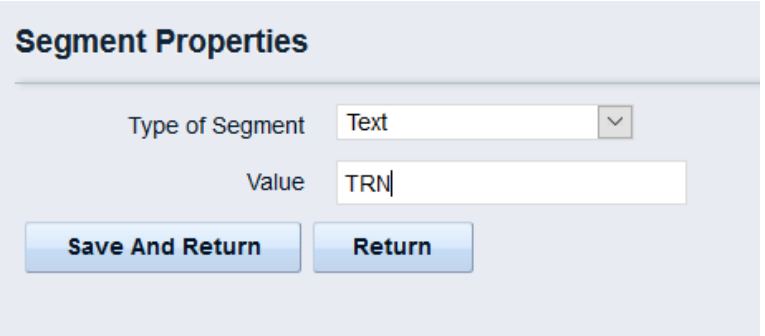


# Exercise #2: Objects And Attributes

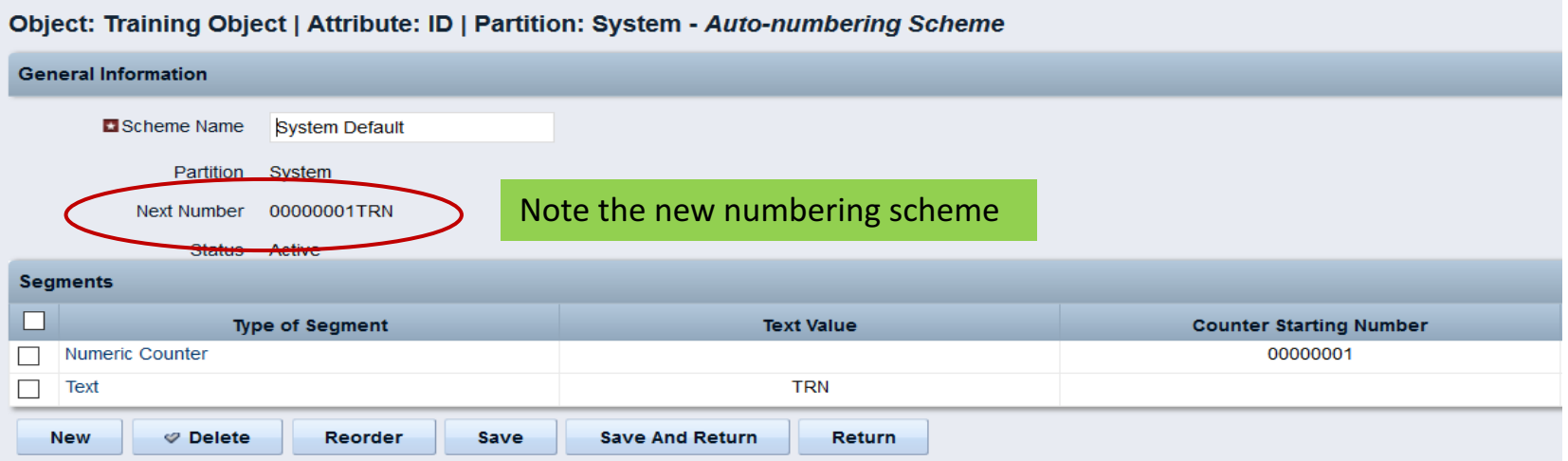
- The default segment type is numeric but this can be modified to include text characters as well




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



# Exercise #2: Objects and Attributes

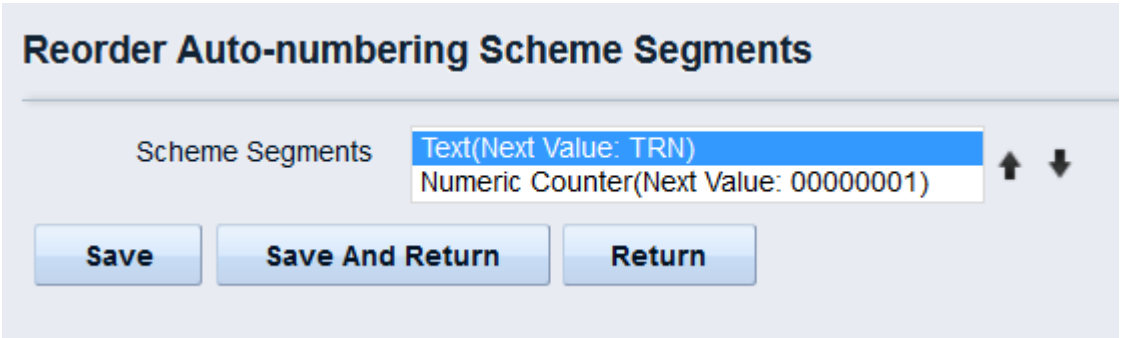



- The new scheme defaults into the order it was entered. To reorder to have the text in front select “Reorder” and move the segments accordingly.

  Select

  “Save and Return”

  when finished



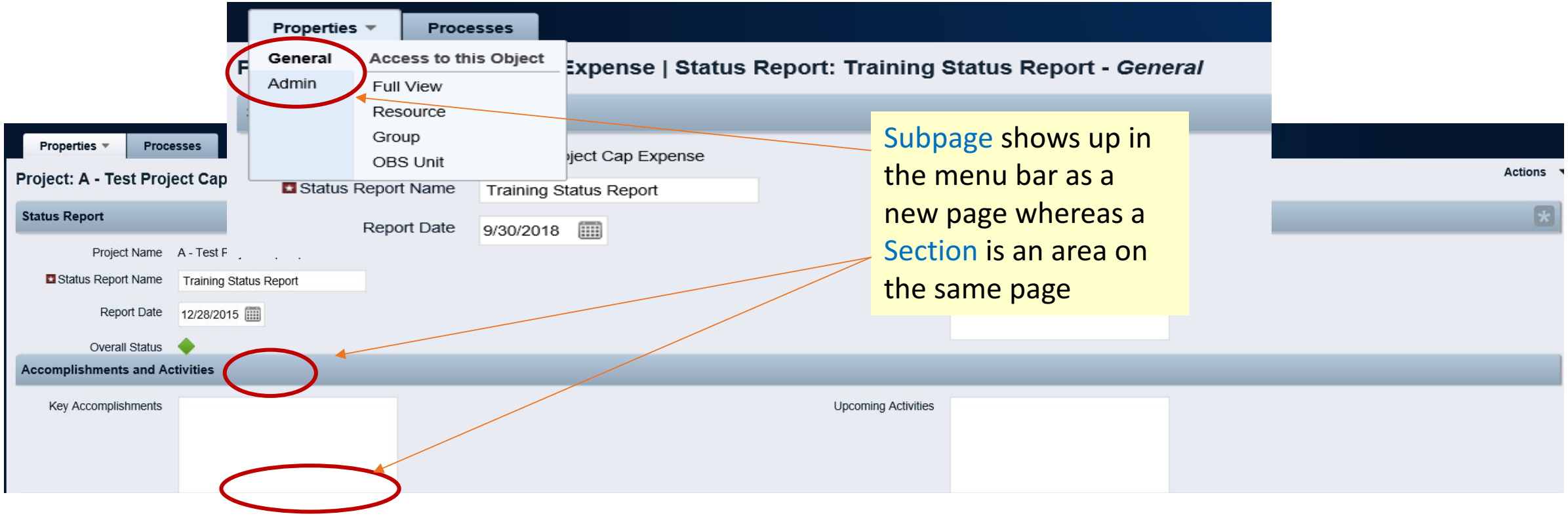
# Objects: Views

- Object views control what attributes users see / update
- There are four types of views:
  - **Create:** What the user sees when creating a new entry in the object
  - **Edit:** What the user sees when accessing a record in an existing object (for example, accessing a project)
  - **List:** What the user sees when first entering the object (for example, clicking the Projects link on the Home menu)
  - **Filter:** What the user sees within the list view, giving them the capability to further refine results

# Objects: Views

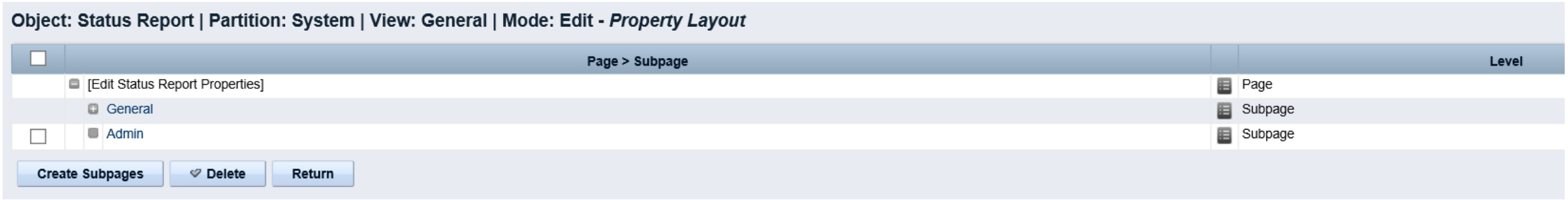
- Within each view separate sub-pages and sections can be created to group attributes together in meaningful ways
  - A sub-page is accessed via a link in the drop down menu and are generally used to drive different functions
  - A section is an area on a page or sub-page that divides the page up into logical groupings
  - Note:  A sub-page 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 normally done so that certain pages can be used for editing but the value will still display on another page

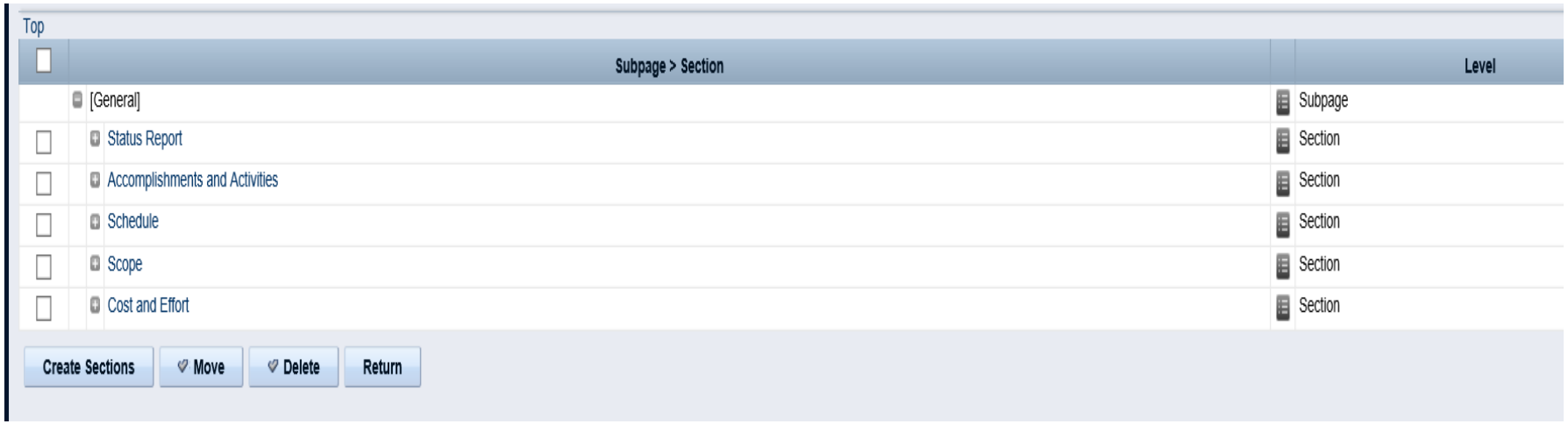
# Objects: Subpage vs. Section (Home)




Subpage shows up in the menu bar as a new page whereas a Section is an area on the same page

# Objects: Subpage vs. Section (Home)



# Objects: Views

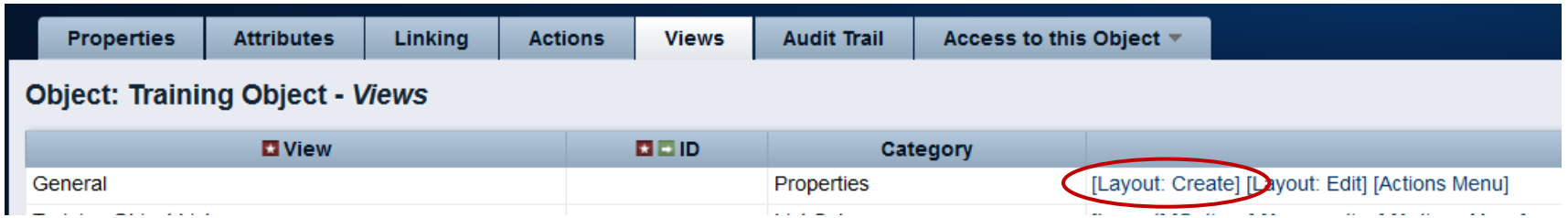
- You can show or hide views based on display conditions or securing the page.  The differences in these two are defined below.
- **Display Conditions**
  - Defines a set of conditions that determine when a subpage appears.  The expression builder is used to create these conditions and utilizes attributes 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

# Exercise #3: Objects And Views

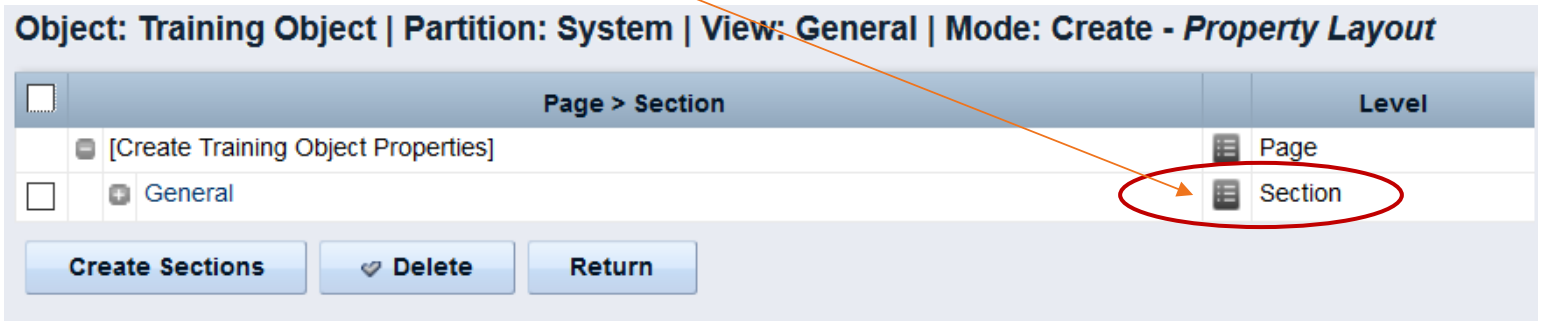
## Using the new object do the following:

➢ Modify the create view to add 1 or 2 attributes to General section of General subpage

- Navigate to Administration -> Objects -> Views -> Layout:Create

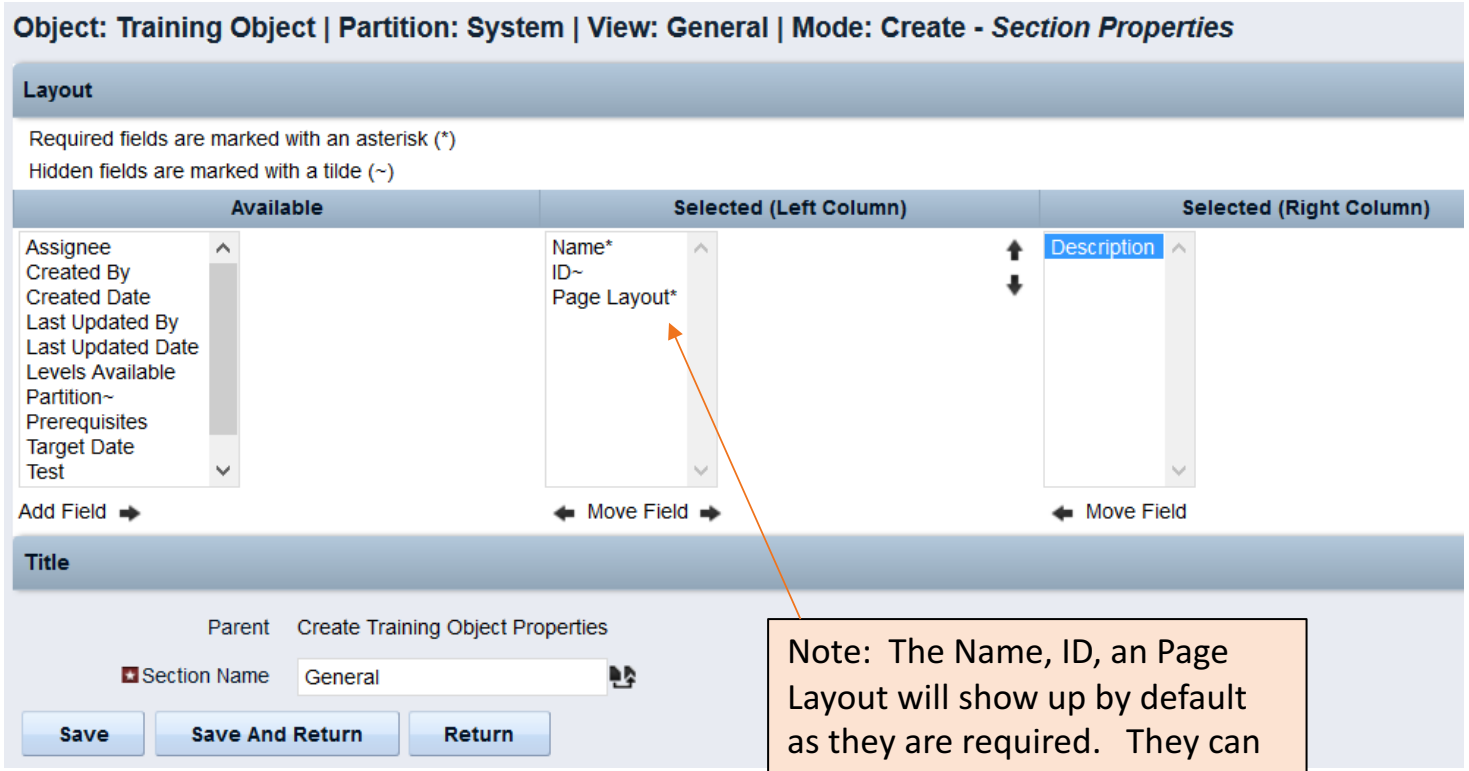



b. Select the properties icon to the left of the Section to alter the layout



# Exercise #3: Objects And Views

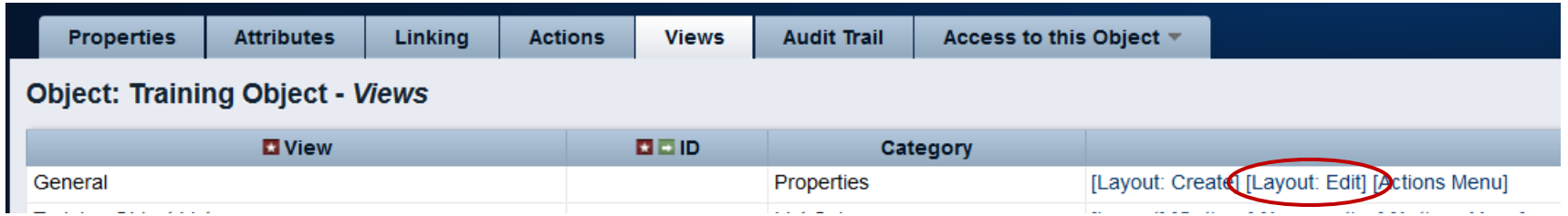
c. Move an attribute to the 2nd column using the arrows underneath the columns below



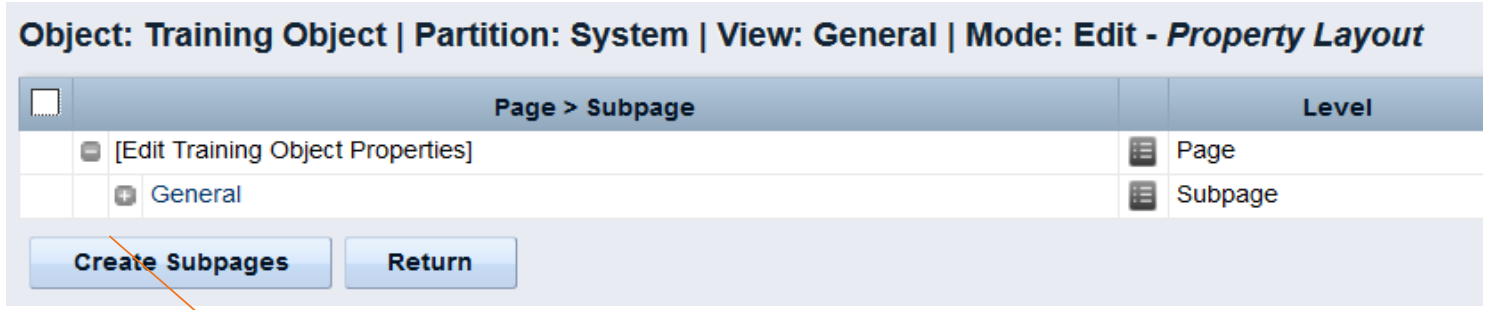
# Exercise #3: Objects And Views

➢ Create a separate subpage within the Edit View

- Navigate to Administration -> Objects -> Views -> Layout:Edit

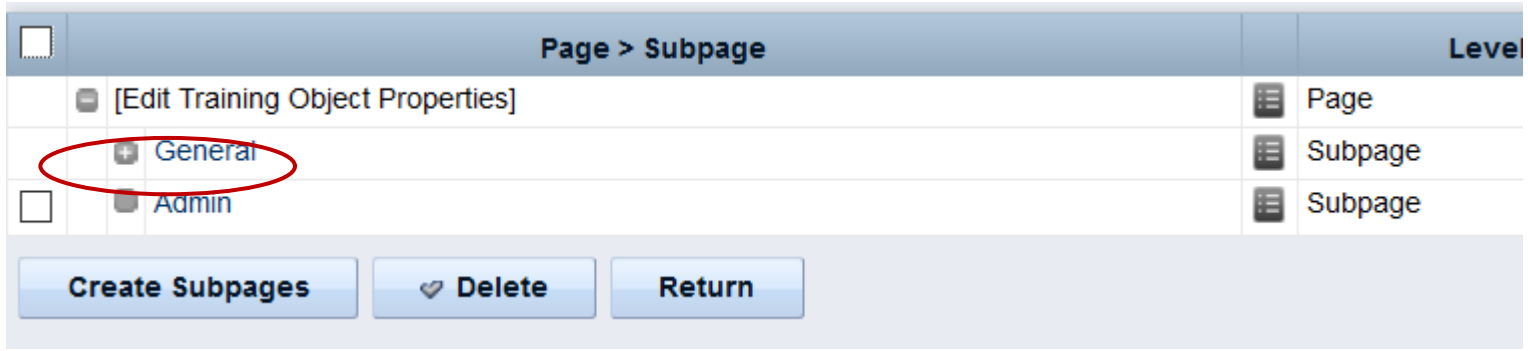



b. Select “Create Subpages” and fill out subpage “Name” and “ID”. Click Save and Return.



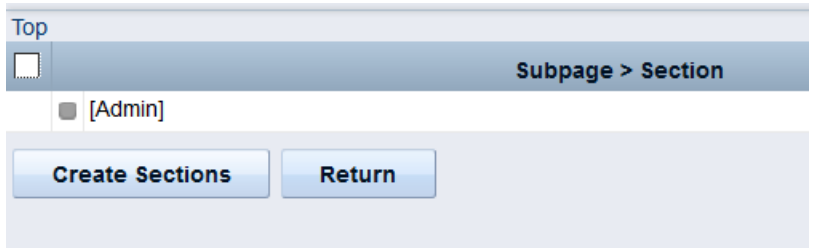
# Exercise #3: Objects And Views

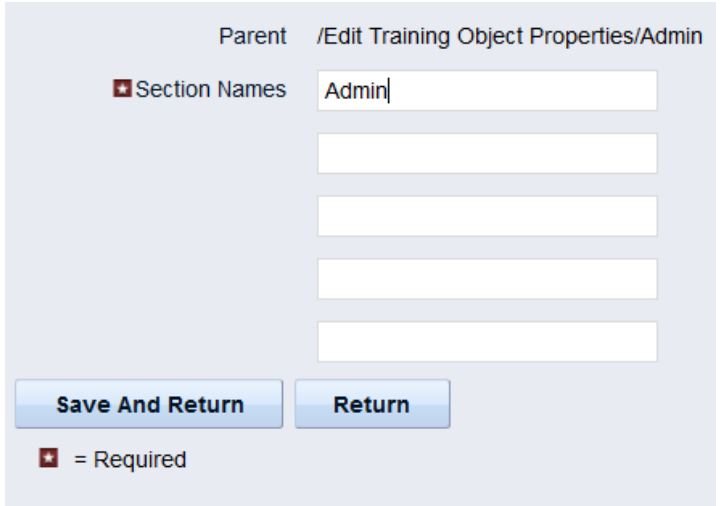
- Add a section within the subpage by clicking on the subpage link first




b. Select “Create Subpages” and fill out subpage “Name” and “ID”

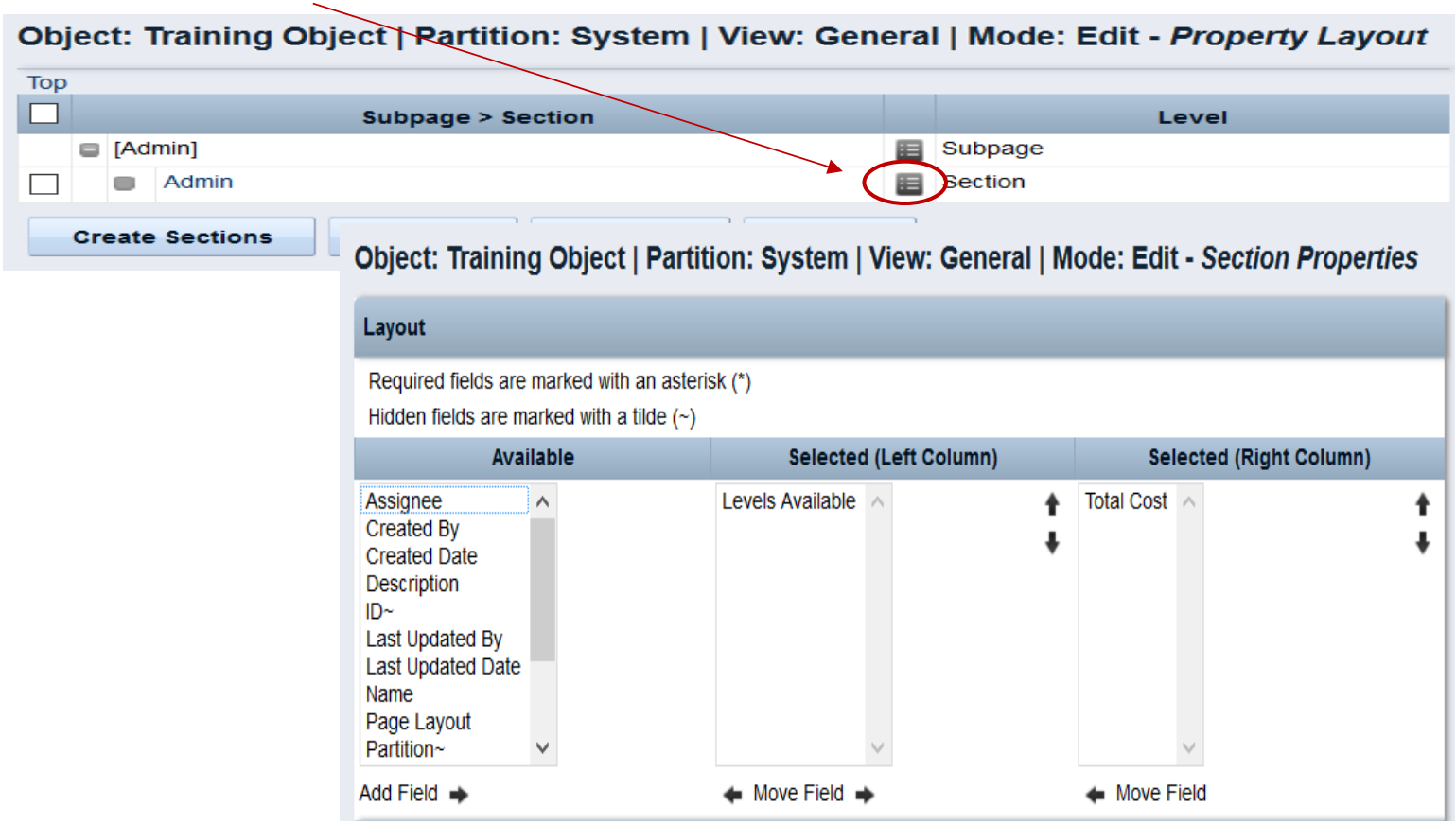
d. Select “Create Sections” and type the name of each section.
You can add up to 5 at time
Save and Return



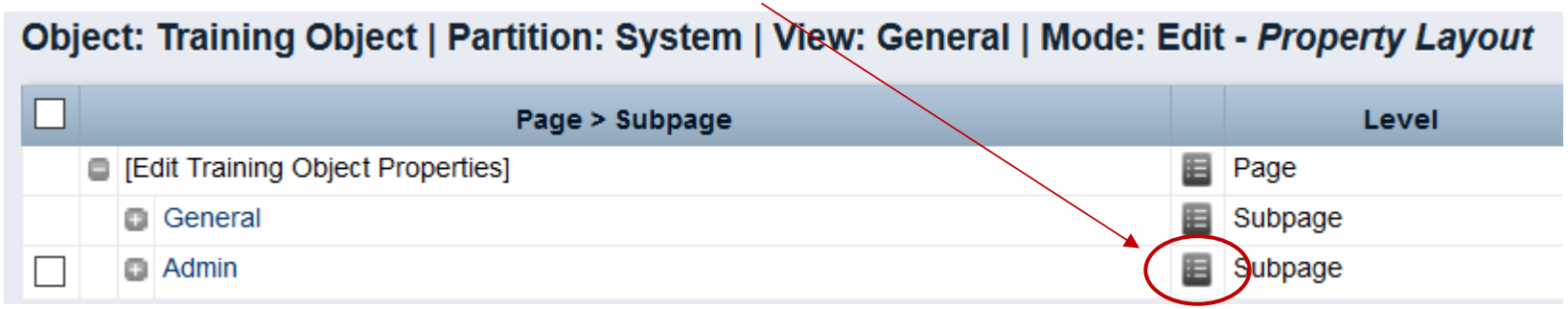
# Exercise #3: Objects And Views

- 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)



# Exercise #3: Objects And Views

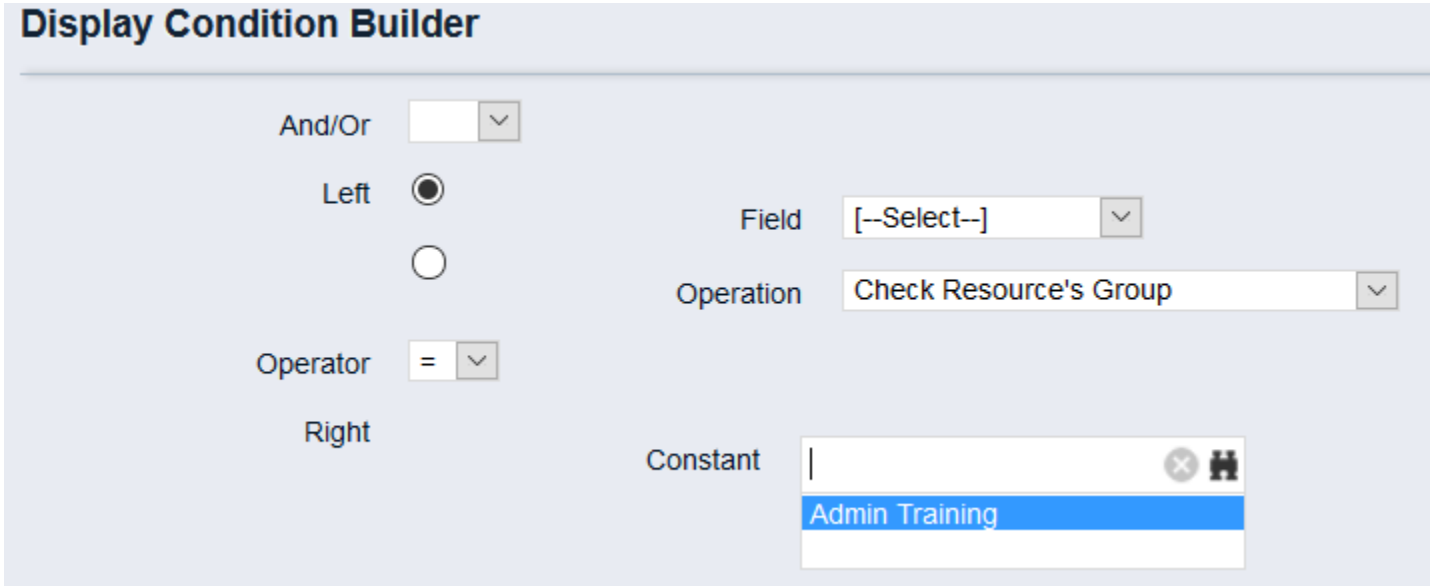
- Create a display condition on the new sub-page
  - Navigate to Administration -> Objects -> Views -> Layout:Edit-> Admin Subpage and select the properties icon




b. Select “Define Display Conditions” within the “Display Conditions” section

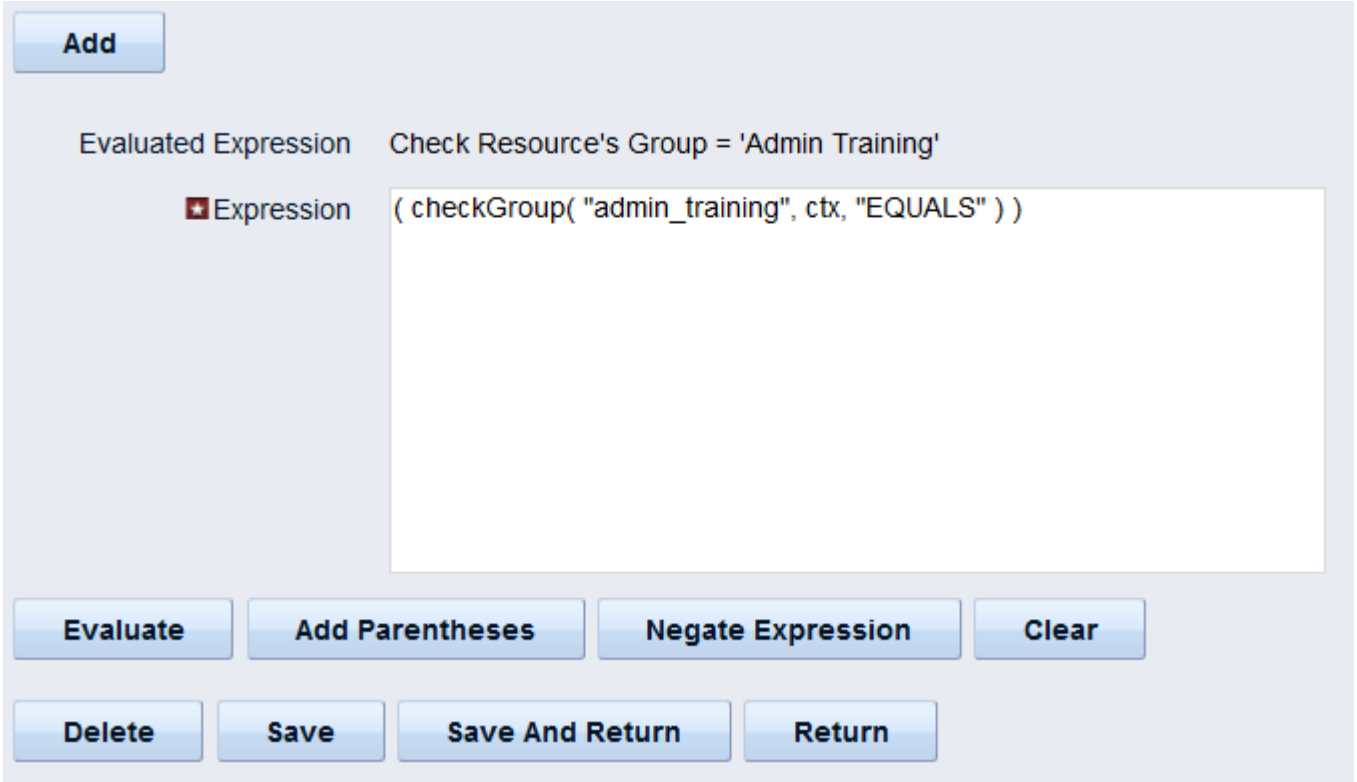
# Exercise #3: Objects And Views

- 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



# Exercise #3: Objects And Views

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



# Additional Fields Properties

- Fields are attributes and are available to be placed onto views, however they have some additional / different properties that allow flexibility in view configuration
- Properties of field attributes control how the field appears within the view itself. The following additional properties can be configured within the field:
  - Enter Once:  Select this check box to prevent users from changing the attribute's value after it has been entered
  - Required:  Select this check box to require that users enter a value

    ** Note:  The attribute itself does not have to have the Required checkbox selected

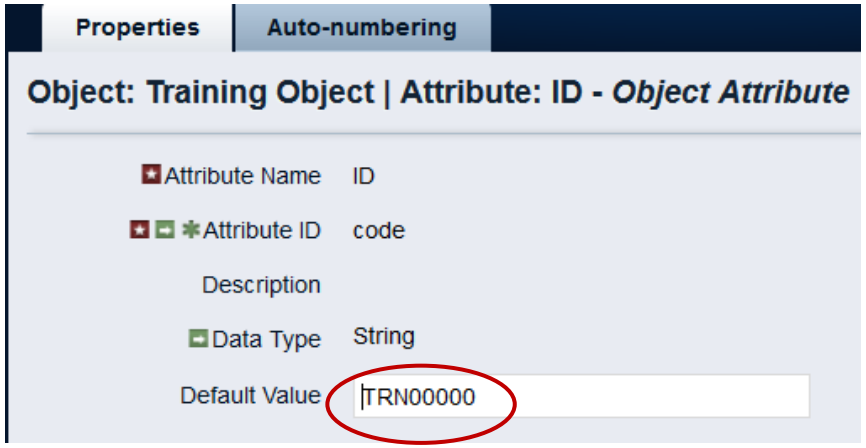
# Additional Fields Properties

- Additional properties can be configured within the field (continued):
  - Hidden:  Select this check box to prevent the attribute from displaying on user views.  Use hidden attributes to add data that is used in calculations but does not display on the page. You must define a default for hidden attributes.
  - Hints / Tooltips:  Enter a message that helps the user. The maximum length for a hint is 512 characters.  Hints display as a static value above/below the field. Tooltips appear when the user hovers over the field.
  - Height:  Enter the number of lines allowed for a text box.  For a notes or description type text field choose “Text Area” from “Display Type” and both the height and width of the text box can be set.

# Exercise #4: Fields And Views

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.

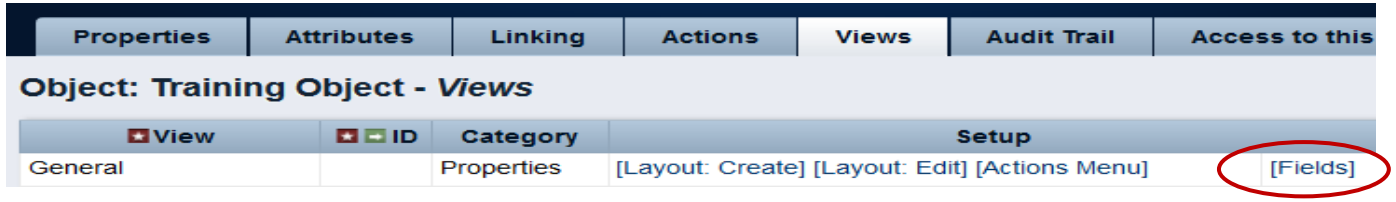



- Select Views

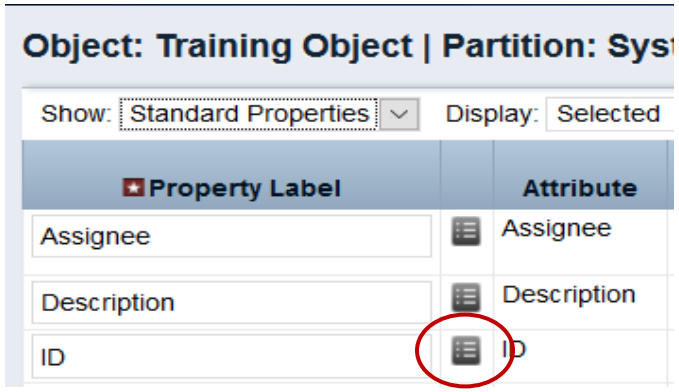
# Exercise #4: Fields And Views

## Using the new object do the following:

- From the General -> Properties line click on “Fields”



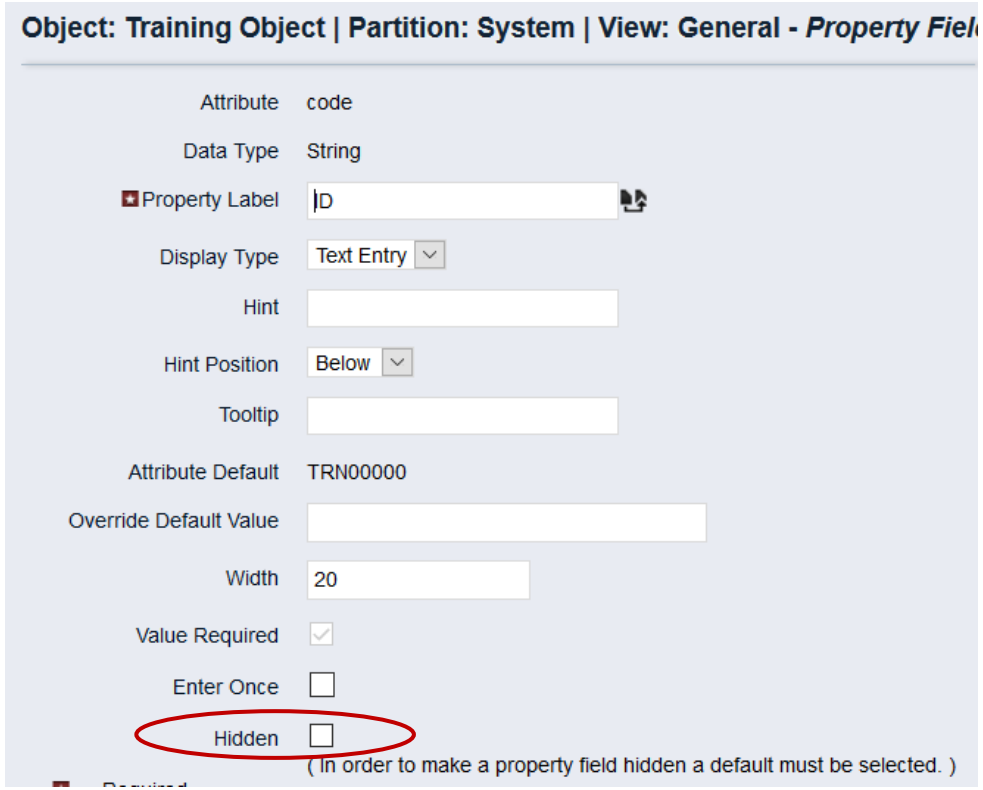

- Click on the properties icon next to the “ID” field



# Exercise #4: Fields And Views

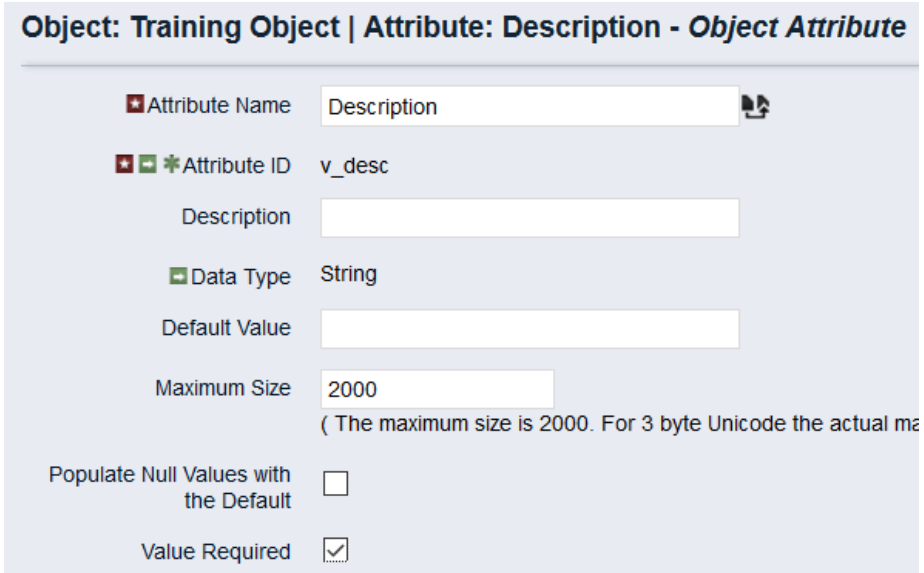
## Using the new object do the following:

- Select the “Hidden” checkbox



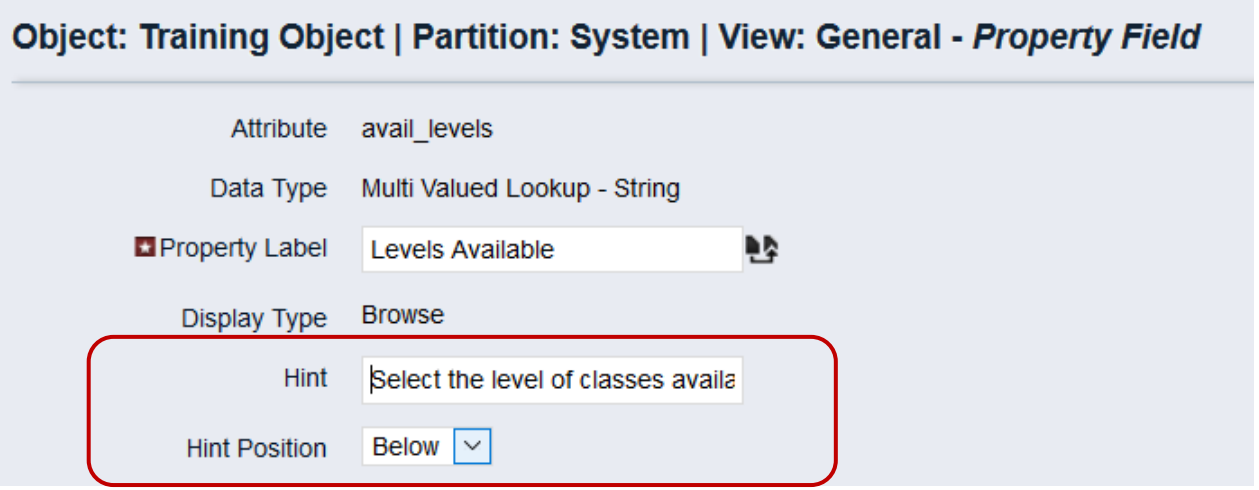
# Exercise #4: Fields And Views

- Make one of the new attributes created “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”.



# Exercise #4: Fields And Views

- 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



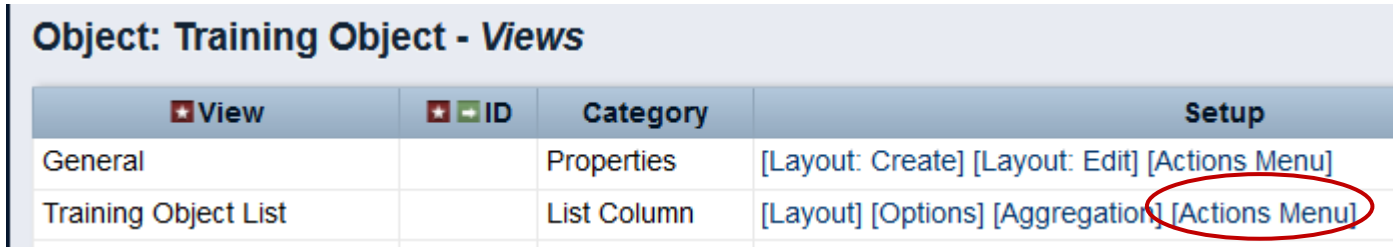
# Objects: Actions

- 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
- Both menus and actions can be renamed to fit the business need

# Exercise #5: Action Menu

## Using the new sub-object do the following:

- Add the “New Action Item” action to the “Actions Menu” on the list page
  - Navigate to Administration -> Objects -> <Object Name> -> Views
  - Select “Actions Menu” from the “XXX List” view

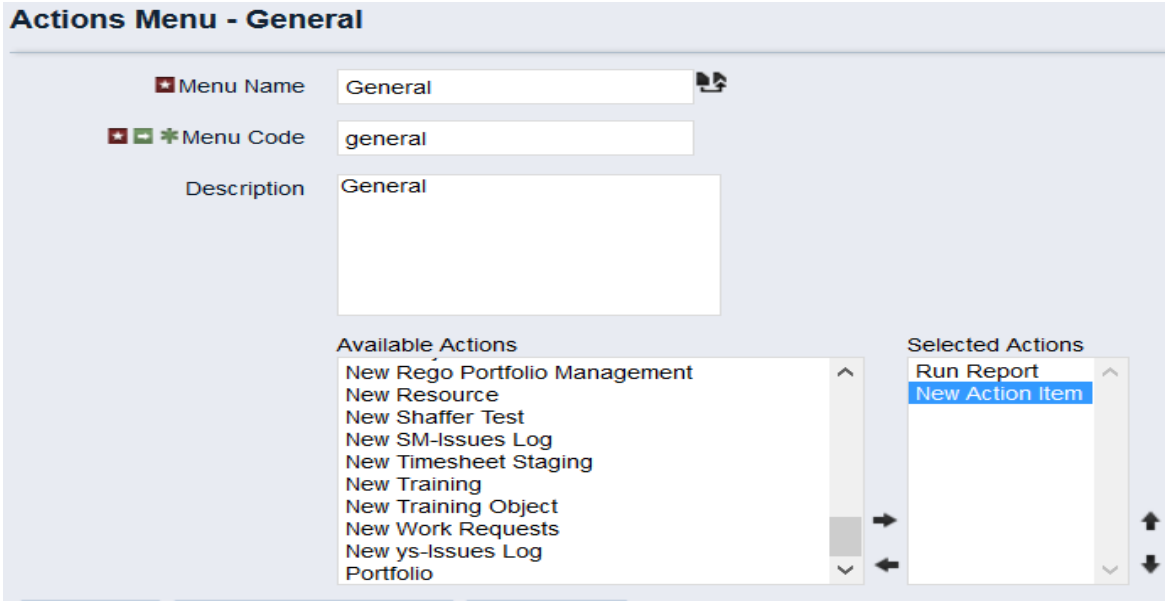



  - Use the existing “General” menu or add a new menu that represents the action

# Exercise #5: Action Menu

## Using the new sub-object do the following:

- Add the “New Action Item” action to the “Actions Menu” on the list page
  - Move the “New Action Item” option over to the “Selected Actions” column
  - Click Save and Return



# Questions?

**Let Rego be your guide.**

# SQL

regoUniversity 2017

# SQL: Course Outline

- What is SQL?
- Using SQL within CA PPM
- Basic SQL Syntax
- SQL Concepts

# 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 on 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

# SQL Types

- SQL is broken down is as follows:
  - Data Manipulation Language (DML)
    - Used to work with the data stored in the database
    - Ie. Select/Update/Insert/Delete statements
  - Data Definition Language (DDL)
    - Used to build and modify the tables (and other objects) in the database
    - Ie. Create/Drop/Alter/Rename table statements
  - Data Control Language (DCL)
    - Used to administer privleges within the database
    - i.e. Grant / Revoke
  - Transaction Control (TCL)
    - Used to manage the changes made by DML statements
    - i.e. Commit/Rollback

# Using SQL With CA PPM

- SQL is used in multiple facets within CA PPM:
  - To extract ad-hoc data
  - As a basis for NSQL in portlet writing
  - To get data within a process for data manipulation

- CA PPM 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 summarized data by daily, weekly, monthly, etc.
    - DataMart Tables:  Provides summary and rollup data

# 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

# 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*;

# 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. **Note:** 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 |

# Exercise #1: Basic Query

- 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 Concepts

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

  - E.g.

    SELECT *column_name,column_name*
    FROM *table_name tbl*

  - In the above example “tbl” is the alias and will be used to reference the table, “table_name” throughout the rest of the query

# SQL Concepts

- Column Names
  - Make your results more presentable by renaming your columns
  - The easiest way to rename a column is without spaces using something like an underscore
  - If you want to rename a column using spaces you must enclose the name in double quotes (“”)
  - E.g.

    SELECT *user_name as “User Name”, email as “Email Address”*
    FROM *cmn_sec_users*

# SQL Concepts Cont.

- 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

# SQL Concepts Cont.

- 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 Concepts Cont.

- Functions
  - Used to perform processing on string and numeric data
  - Types
    - Aggregate
      - Return a single value, calculated from values in a column
        - Ie. 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
        - Ie. ROUND, UCASE (Uppercase), LCASE (Lowercase), FORMAT

# SQL Concepts Cont.

- Sub-Queries
  - Used to return data that would be used in the main query
  - Nested inside SELECT, INSERT, UPDATE or DELETE statements.
    - Must be enclosed by parentheses

# Exercise #2: Query Using Concepts

- Write a query to pull all active projects starting in 2015 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 2015

```
 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('2015-01-01','yyyy-mm-dd')
```

# Exercise #2: Query Using Concepts

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
```

# Exercise #2: Query Using Concepts

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
```

# Questions?

**Let Rego be your guide.**

# Object Portlets

regoUniversity 2017

# Object Portlets: Course Outline

- What is a Portlet? Object Portlet?
- Design Basics
- Example/Exercise
- Questions

# Object Portlets: Overview

- What is a Portlet?
  - Portlets are snapshots of CA PPM data and can consist of grids, graphs, or snippets of HTML
  - You select data to display in a Portlet
  - Portlets do not replace CA PPM reports, but can be considered mini-reports
  - Portlets obtain information and business intelligence from CA PPM, 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
  - A Portlet Page is a set of Portlets that automatically present to users with the appropriate access privileges
  - Users can personalize their Portlet Pages by showing, hiding, and positioning Portlets on the page

# Object Portlets: Overview

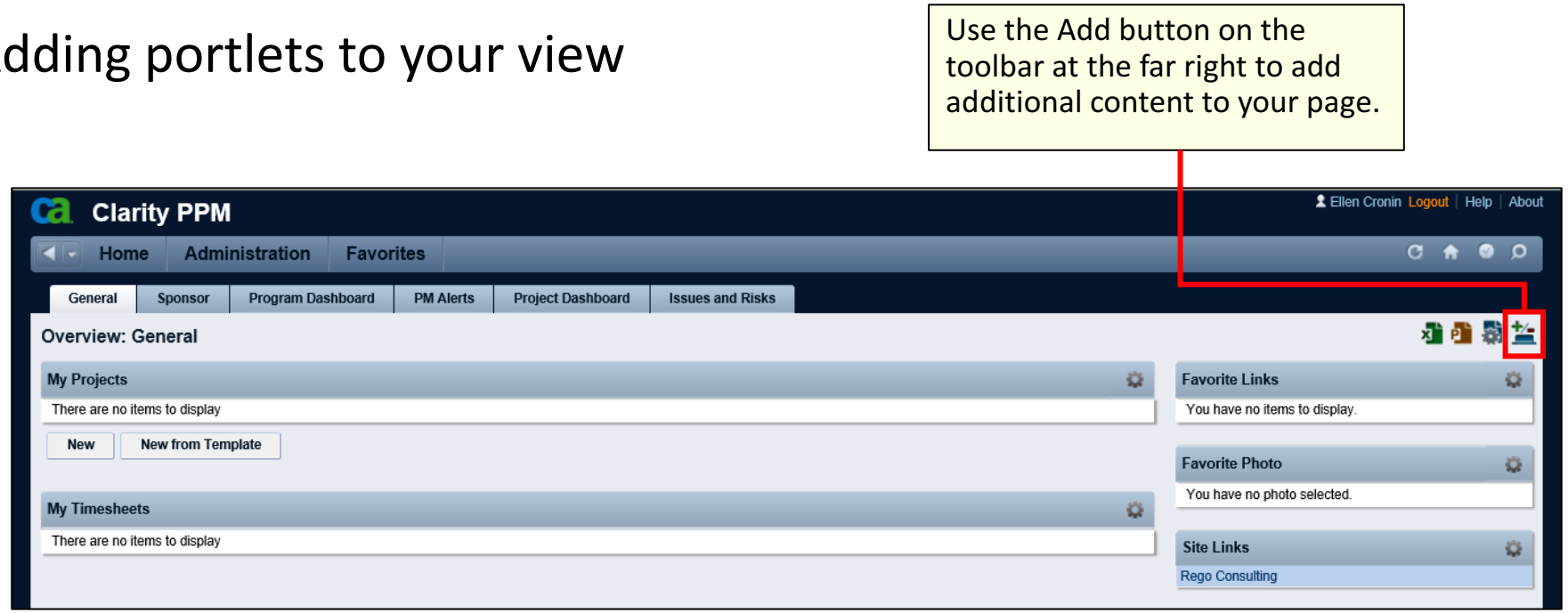
- **Chart Portlet –** A graphical view of CA PPM data (for example, pie and line charts)
- **Grid Portlet –** A list or table of data you can filter in real time
- **HTML Portlet –** Displays information on a CA PPM page from internal or external web sites formatted as HTML
- **Filter Portlet –** Applies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

# Object Portlets: Overview

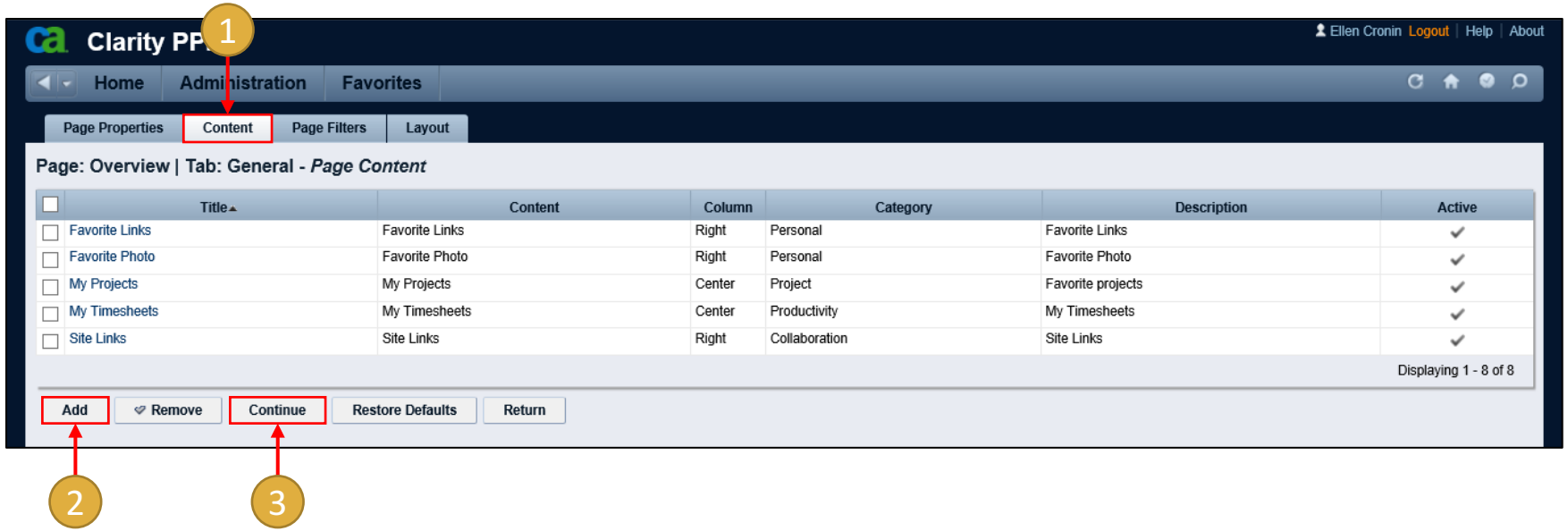
- What is an Object Portlet?
  - Portlets created using an Object (i.e. Projects, Task, Resource) instead of a query to define the data set gathered.
  - Pros
    - Customizable
    - Default Security
    - In-Line Editing
    - Time Scaled Values
  - Cons
    - Multiple Objects Difficulty
    - No Custom Logic

# Object Portlets

- Adding portlets to your view



# Object Portlets




Click the tab where you want the portlet to appear

1. Click the *Content* tab
2. Click *Add* and browse for the desired portlet
3. Click *Continue*

# Object Portlets: Exercise #1

- 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: Project Details
    - Data Provider: Project Object
    - List Layout:
      - Project ID, Project Name, Start Date, Finish Date
    - Filter Layout:
      - Project ID, Is Active?

# Questions?

**Let Rego be your guide.**

# Basic Processes

regoUniversity 2017

# Basic Processes: Course Outline

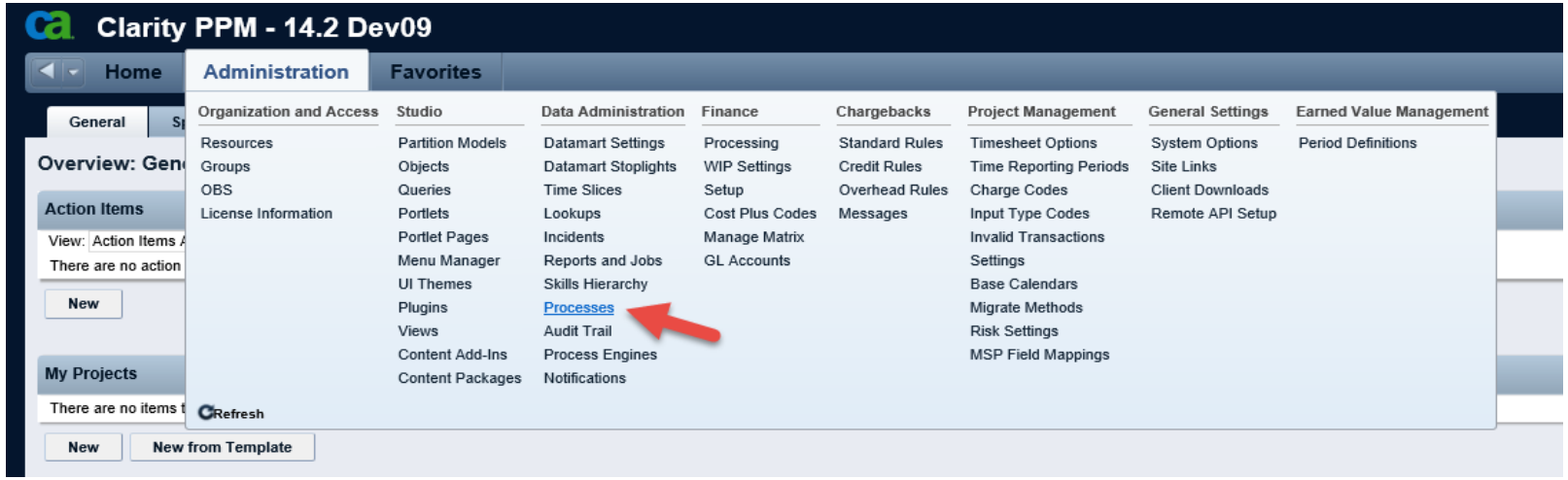
- What is a Process?
- Design Basics
- Example/Exercise
- Questions

# Basic Processes: Overview

- What is a process?
- Processes automate repetitive steps that you would otherwise perform manually though the user interface
  - To accurately reproduce a user action, the process impersonates the process initiator to perform the process steps
  - A process includes a series of steps that result in an end
  - Each step performs one or more actions that move the process toward completion
  - All processes have a start and finish step
  - 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
  - Implement scenarios

# Basic Processes: Overview

- *Administration > Data Administration > Processes*

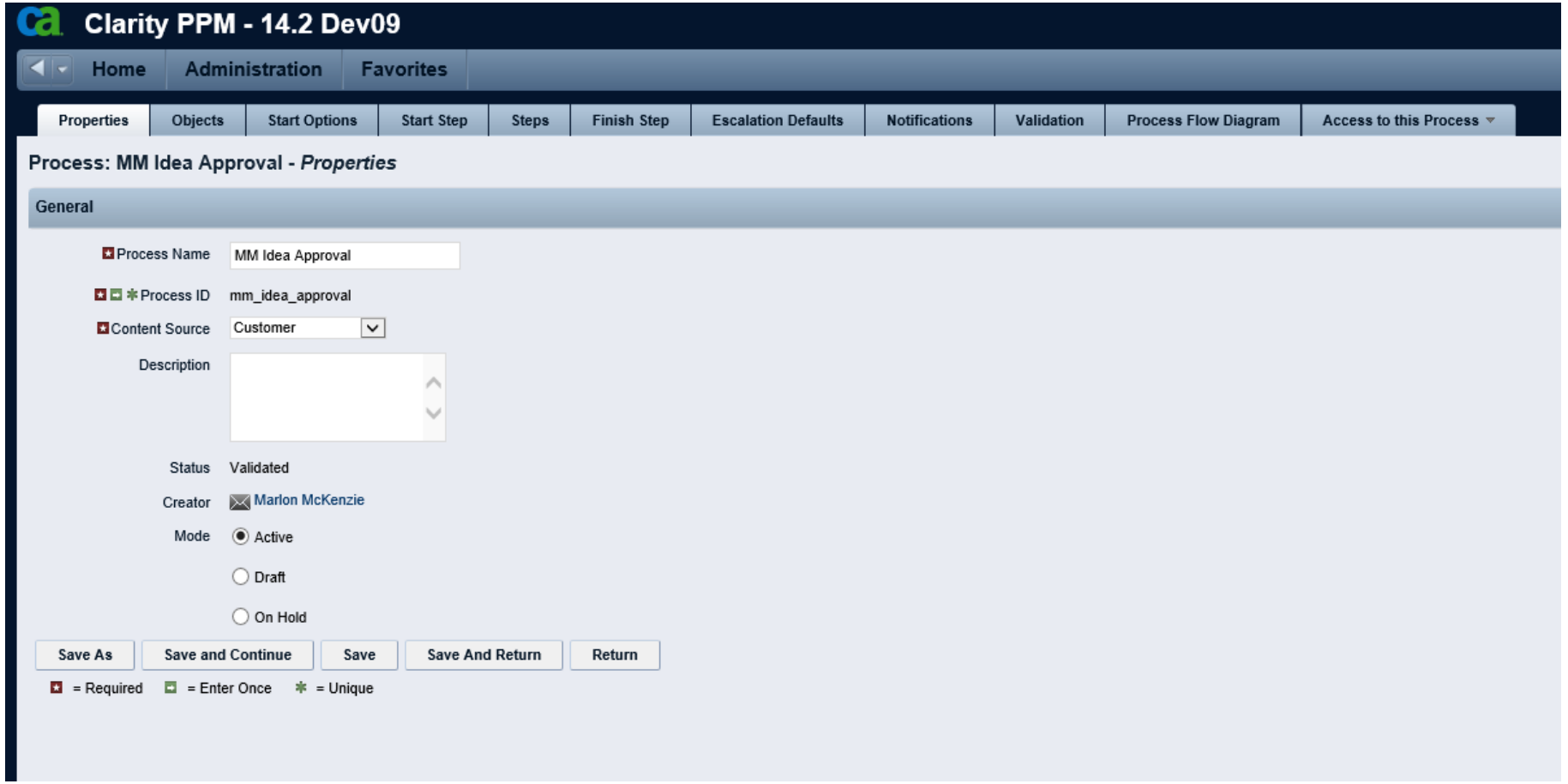


# Basic Processes: Design Basics

1. Create
    a. Properties
    b. Associate Object
    c. Start Options
    d. Steps
        1) Pre-condition
        2) Action
        3) Post-condition
2. Validate / Activate

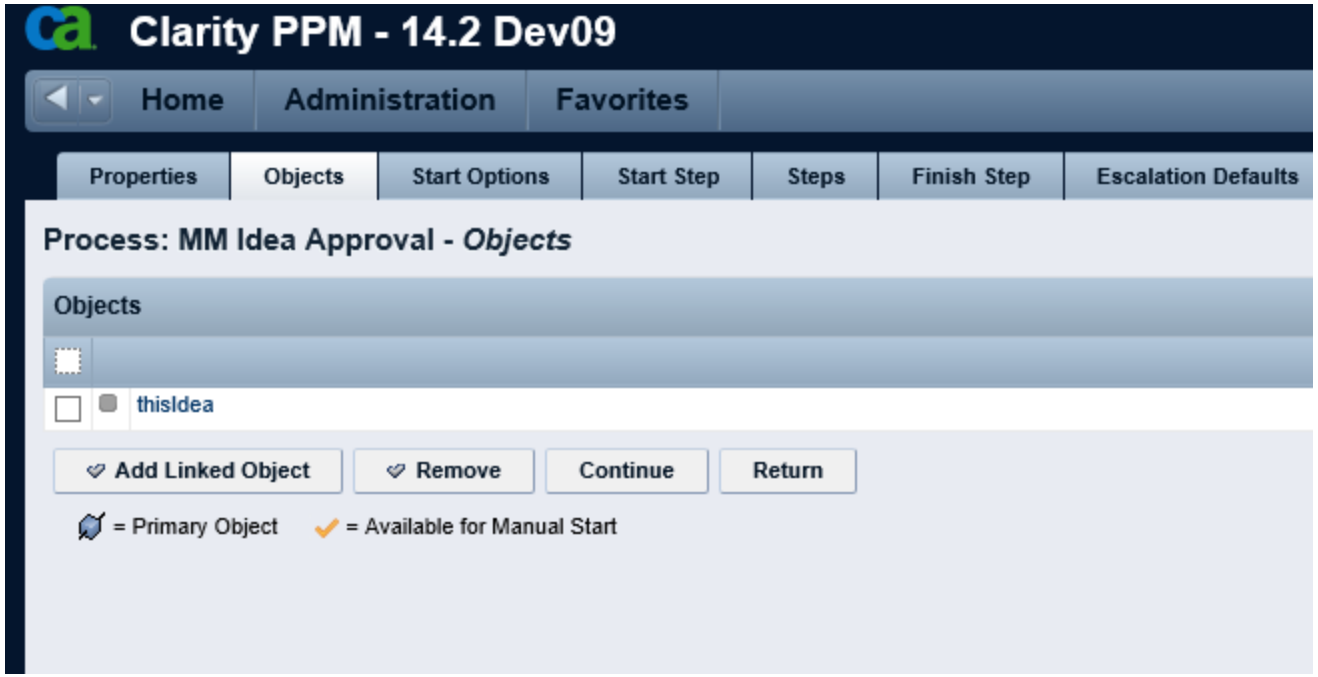
# Basic Processes: Design Basics

- Properties



# Basic Processes: Design Basics

- Associated Object



# Basic Processes: Design Basics

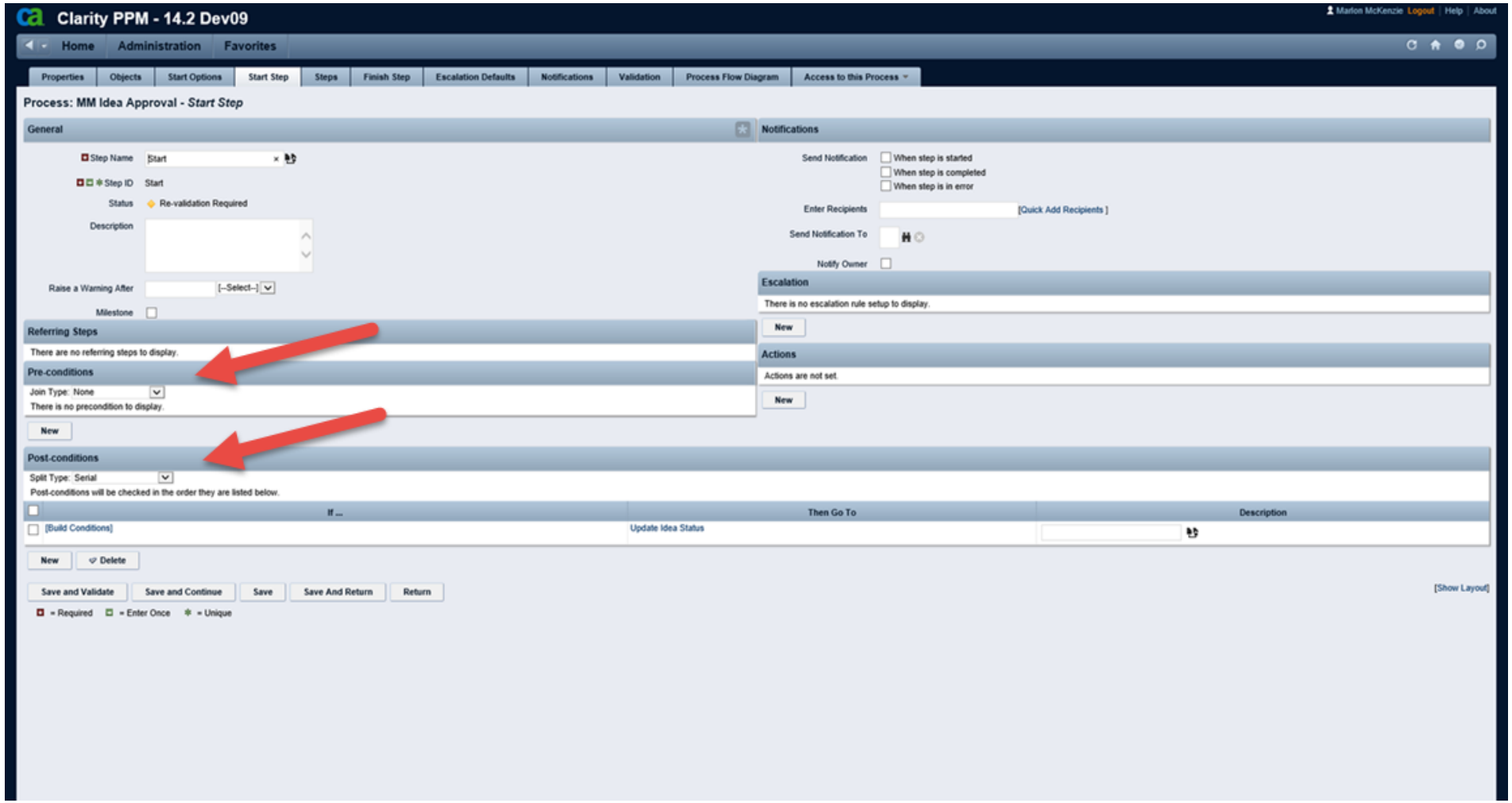
- Start Options

# Basic Processes: Design Basics

- 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 multiple objects added to the process; for example, you can:
  - Check the status of action items
  - Check between object attribute values
  - Wait for a sub-process to complete before joining the master process
- After defining the pre-conditions that trigger a step, you must define post-conditions that connect this step to the next step or the end step
- Examples of 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

# Basic Processes: Design Basics

- Steps



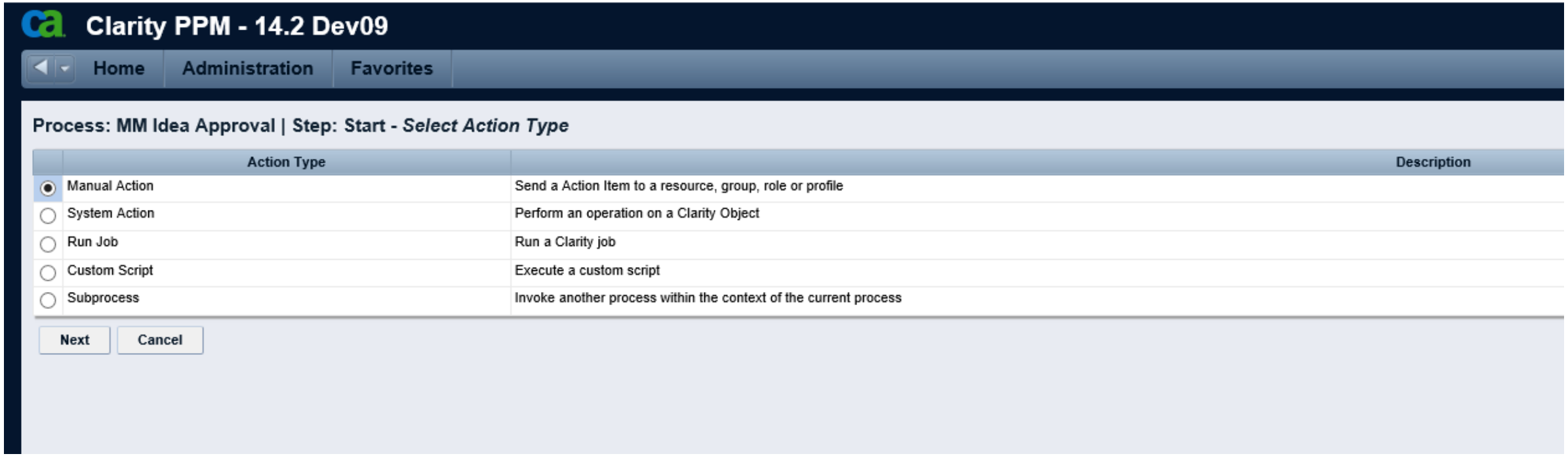
# Basic Processes: Design Basics

- Post Conditions or Splits
  - Serial
  - Parallel
  - Decision Point
  - Multi choice
- Preconditions or Joins
  - Rendezvous (AND)
  - Merge (XOR)
  - Wait and Merge
  - Multi-thread
  - First in Line

# Basic Processes: Design Basics

- Process actions can be defined as
  - Manual Actions
    - Example(s): Actions Items
  - System Actions
    - Example(s): Lock/Unlock Attributes, Set Attribute Values
  - Running of a Job
    - Example(s): Post Timesheets
  - Custom Scripts (GEL)
    - Example(s): Notification Scripts
  - Subprocess

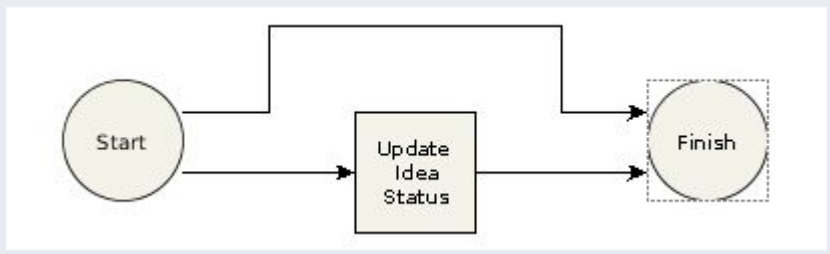
# Basic Processes: Design Basics



# Basic Processes: Design Basics

- Validate and Activate Process

# Basic Processes: Exercise #1




1. Create a new process using your initials as identifier
2. Link the process to the idea object
3. Make the process auto-start with status set to submitted for Approval
4. Create one steps in addition to the Start and Finish steps
5. Add the actions to each step that sets the idea status to Approved
6. Once all steps are created, create links between steps
7. Ensure the conditions point to the correct step

# Questions?

**Let Rego be your guide.**

# XOG

regoUniversity 2017

# XOG Overview

- Introduction
- Requirements
- Installation and Setup
- XOG Files and Procedure
  - XML Files (Read/Write)
  - Properties File
- Best Practices
- Other XOG Methods and XML Creation

# XOG: Overview

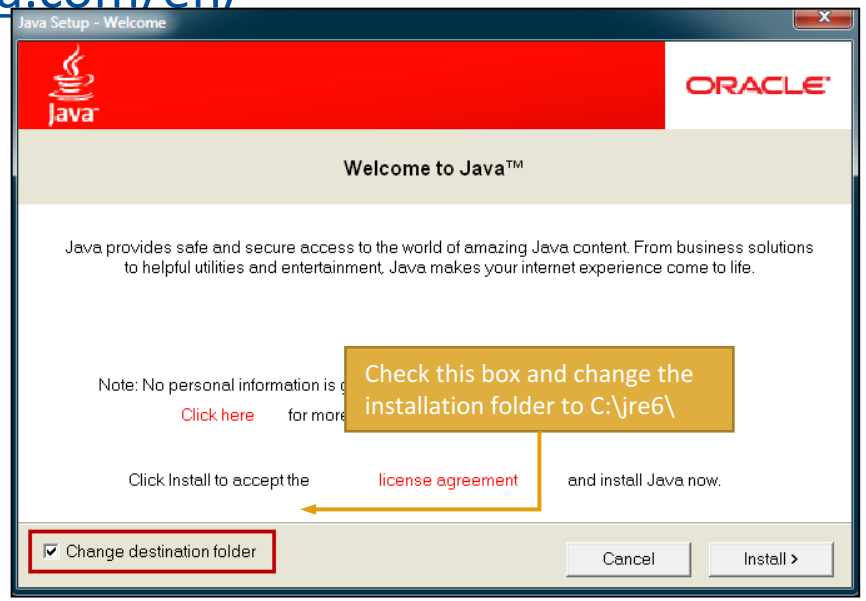
- CA PPM has a Web Services interface called XML Open Gateway (XOG)
- What XOG Does
  - Export data and configuration
  - Import data and configuration
- When to Use XOG
  - Move configuration or data from one environment to another
  - Handle data imports via batch processing

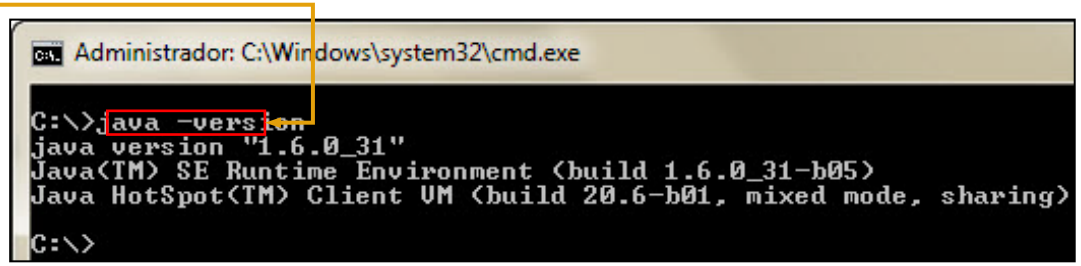
# XOG: Requirements

- 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 6 Runtime Environment version 1.6.0_15
  - Oracle Java 7 Runtime Environment version 1.7.0_25 or Higher
- CA PPM
- XOG client version matching the version of CA PPM (Recommended)
- CA PPM user with XOG Global Rights
  - Administration – XOG
  - Administration – Access
  - XOG rights for individual objects (for example, Resource, Project, OBS)

# XOG: Installation

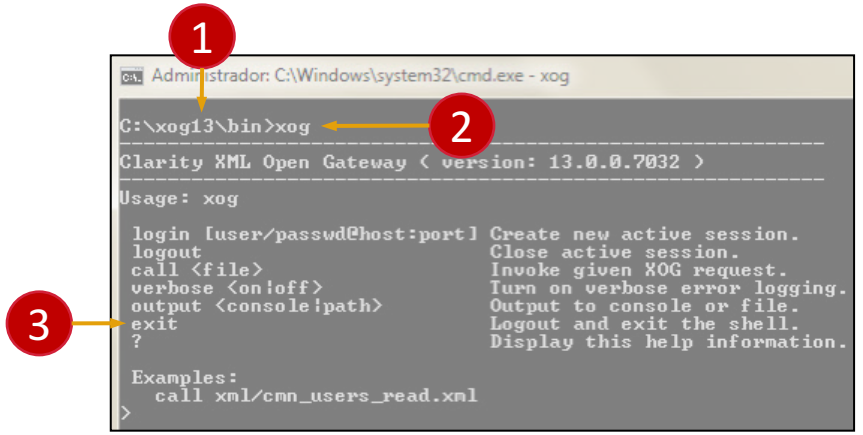
- Download compatible JRE from http://www.java.com/en/
  - Installation folder: *C:\jre6*
- Set the environment variables:
  - **JAVA_HOME=**C:\jre6
  - **PATH=**;%JAVA_HOME%\bin
- Test for Java using a command prompt
  - java -version



# XOG: Installation

- Download the XOG client from the Admin tool
  - Download the cross-platform client
  - Extract the ZIP file to C:\xog13\
- Test for XOG client using a command prompt
  1. Access C:\xog13\bin\ folder
  2. Type *xog* and press *enter*
  3. Type

# XOG: Verify Connectivity

Use XOG client shell commands to test the connection between the XOG client and CA PPM server as follows:

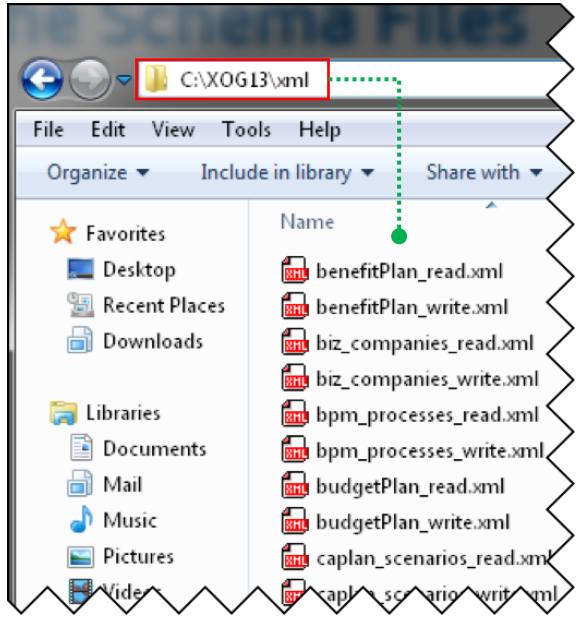
1. Open a command prompt
2. Type **cd** *C:\xog13\bin* and press *enter*
3. Type **xog** (**xog  -sslenabled true** if your connection is using HTTPS) and press *enter*
4. Type **login** *<username>***/***<mypassword>***@***<myserver>***:***<port>* and press *enter*

   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: The 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

dev.txt

```
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

1. Create a *.properties* file with the default values for the XOG parameters
2. Create an input XML file with the necessary header information, arguments, and query filters
3. Navigate to the “bin” folder under the XOG client installation folder by typing **cd** C:\xog13\bin\ and pressing *enter*
4. Type **xog -propertyfile** *<properties.txt>*
5. 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

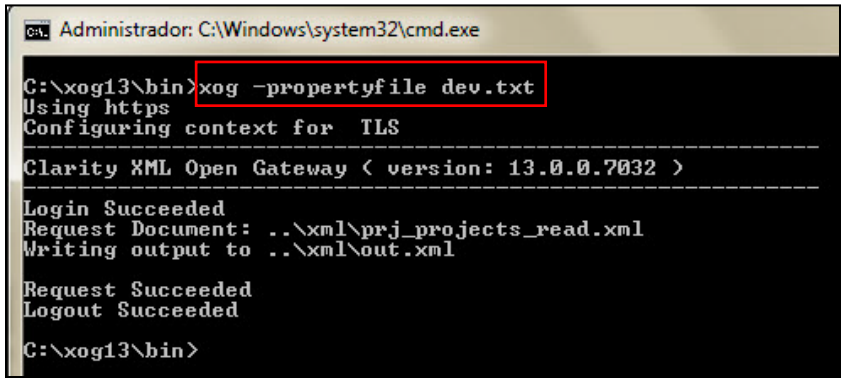
- #default port number for ssl
- portnumber=443

- #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



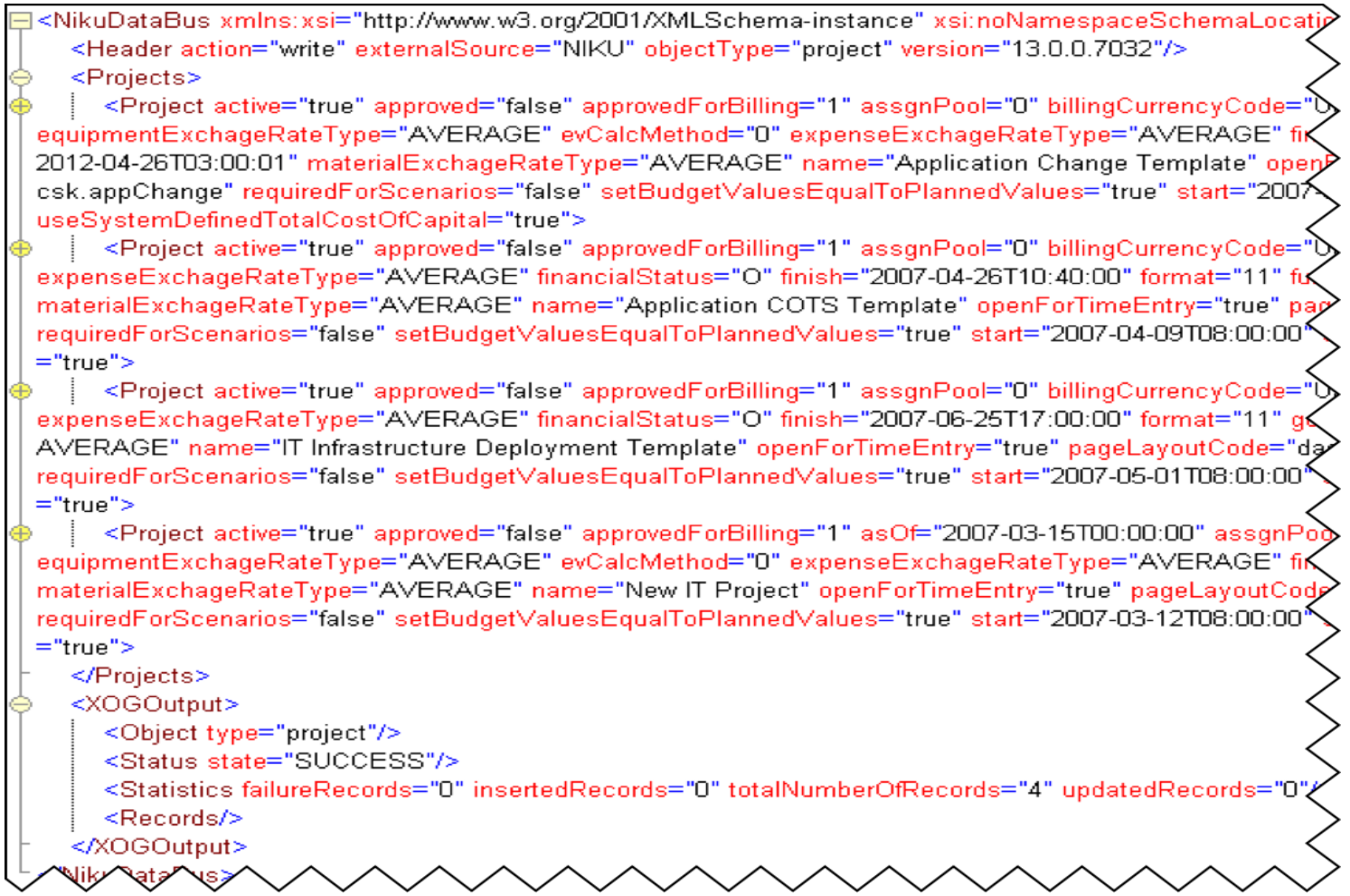
# XOG: Exercise #1 - Export Data Cont.

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 Cont.

Output File



```
<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocati
    <Header action="write" externalSource="NIKU" objectType="project" version="13.0.0.7032"/>
    <Projects>
      <Project active="true" approved="false" approvedForBilling="1" assgnPool="0" billingCurrencyCode="U
equipmentExchageRateType="AVERAGE" evCalcMethod="0" expenseExchageRateType="AVERAGE" fi
2012-04-26T03:00:01" materialExchageRateType="AVERAGE" name="Application Change Template" openF
csk.appChange" requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-
useSystemDefinedTotalCostOfCapital="true">
      <Project active="true" approved="false" approvedForBilling="1" assgnPool="0" billingCurrencyCode="U
expenseExchageRateType="AVERAGE" financialStatus="O" finish="2007-04-26T10:40:00" format="11" fu
materialExchageRateType="AVERAGE" name="Application COTS Template" openForTimeEntry="true" pag
requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-04-09T08:00:00
="true">
      <Project active="true" approved="false" approvedForBilling="1" assgnPool="0" billingCurrencyCode="U
expenseExchageRateType="AVERAGE" financialStatus="O" finish="2007-06-25T17:00:00" format="11" g
AVERAGE" name="IT Infrastructure Deployment Template" openForTimeEntry="true" pageLayoutCode="da
requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-05-01T08:00:00
="true">
      <Project active="true" approved="false" approvedForBilling="1" asOf="2007-03-15T00:00:00" assgnPoo
equipmentExchageRateType="AVERAGE" evCalcMethod="0" expenseExchageRateType="AVERAGE" fi
materialExchageRateType="AVERAGE" name="New IT Project" openForTimeEntry="true" pageLayoutCode
requiredForScenarios="false" setBudgetValuesEqualToPlannedValues="true" start="2007-03-12T08:00:00
="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

1. Create a *.properties* file with the default values for the XOG parameters
2. Create an input XML file with the necessary header and import data
3. Navigate to the XOG client installation “bin” folder by typing **cd** C:\xog13\bin\ and pressing *enter*
4. Type **xog -propertyfile** *<properties.txt>* and press *enter*
5. 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

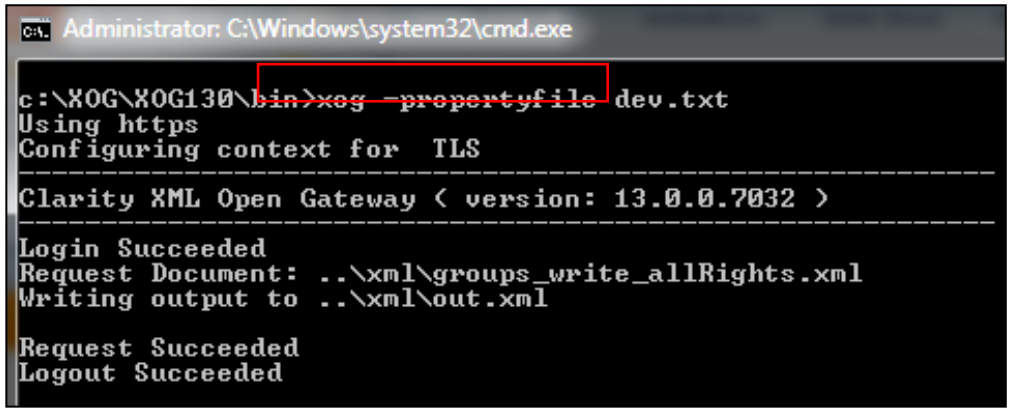
- #identify the path to the input and output files
- input=../xml/groups_write_allRights.xml
- output=../xml/out.xml

## 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




## 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: Clean Up XML Write Files

- 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 | & | & |
| Left angle bracket | < | < |
| Right angle bracket | > | > |
| Straight quotation mark | " | " |
| Apostrophe | ' | '' |

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

# Questions?

**Let Rego be your guide.**

# Follow Up Contact Information

If you have any follow up questions, please feel free to reach out via email.

- Anthony Alcala
  - Email: anthony.alcala@regoconsulting.com
- Marlon McKenzie
  - Email: marlon.mckenzie@regoconsulting.com
- Jenn Rinella
  - Email: jenn.rinella@regoconsulting.com

# Thank You For Attending regoUniversity

## Instructions for PMI credits

- Access your account at pmi.org
- Click on **Certification**
- Click on **Maintain My Certification**
- Scroll down to **Report PDU's**
- Click on Course Training (or other appropriate category)
- Enter **Rego Consulting**
- Enter Activity- **Enter Name of Course**
- Enter **Description**
- Enter **Date Started**
- Enter **Date Completed**
- Provide Contact Person **Name of Person to Contact**
- Provide Contact E-Mail **E-Mail of Person to Contact**
- Enter Number of **PDU's Claimed** (1 PDU per course hour)
- Click on the **I agree this claim is accurate box**
- Click **Submit** button

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

**Phone**
888.813.0444

**Email**
info@regouniversity.com

**Website**
www.regouniversity.com