[1-1: Explore the Certification App 3](#_Toc514686390)

[1-2: Prepare Your Training Org 6](#_Toc514686391)

[1-3: Create a Sandbox 8](#_Toc514686392)

[1-4: Download the Apex Developer’s Guide 9](#_Toc514686393)

[2-1: Create a Custom Object 10](#_Toc514686394)

[2-2: Create Custom Fields 13](#_Toc514686395)

[2-3: Create Relationship Fields 17](#_Toc514686396)

[3-1: Create a Formula Field 22](#_Toc514686397)

[3-2: Create a Roll-Up Summary Field 24](#_Toc514686398)

[3-3: Create a Formula Field that References Roll-Up Summary Fields 26](#_Toc514686399)

[3-4: Understand Record Types 28](#_Toc514686400)

[4-1: Logging into a Sandbox 30](#_Toc514686401)

[4-2: See Apex in Action 32](#_Toc514686402)

[4-3: Create and Use an Apex Class 34](#_Toc514686403)

[4-4: Observe the Effects of Versioning 36](#_Toc514686404)

[4-5: Take a Quick Tour of Apex 38](#_Toc514686405)

[4-6: Examine Implicit Operations 40](#_Toc514686406)

[4-7: Profile Limits Using the Developer Console 42](#_Toc514686407)

[4-8: Work with a Custom Object 43](#_Toc514686408)

[4-9: Use Record IDs to Access a Contact in the UI 44](#_Toc514686409)

[5-1: Create and Run a Query in the Developer Console 46](#_Toc514686410)

[5-2: Write a SOQL Query that Uses a WHERE Clause 48](#_Toc514686411)

[5-3: Write and Execute a SOQL Query in Apex 50](#_Toc514686412)

[5-4: Write a Dynamic Query in Apex 51](#_Toc514686413)

[6-1: Write and Test Child-to-Parent Relationship Queries 53](#_Toc514686414)

[6-2: Query Accounts and Related Contacts 55](#_Toc514686415)

[7-1: Execute DML Commands 57](#_Toc514686416)

[7-2: Handle DML Errors and Exceptions 59](#_Toc514686417)

[8-1: Define a Trigger 61](#_Toc514686418)

[8-2: Define the Trigger's Business Logic 62](#_Toc514686419)

[9-1: Define an Apex Class 64](#_Toc514686420)

[10-1: Explore the Implicit Firing of Triggers 66](#_Toc514686421)

[10-2: View the Events that Occur During a Rollback 68](#_Toc514686422)

[10-3: See the Save Order of Execution in Action 70](#_Toc514686423)

[11-1: Make Test Data Available to Test Methods 73](#_Toc514686424)

[11-2: Write and Run an Apex Test 75](#_Toc514686425)

[12-1: Explore Code Coverage 76](#_Toc514686426)

[13-1: Refactor a Trigger to Avoid SOQL Limits 78](#_Toc514686427)

[13-2: Refactor a Trigger to Avoid DML Limits 80](#_Toc514686428)

[14-1: Create a Formula Field to Eliminate a Query 82](#_Toc514686429)

[14-2: Create Fields for Counting Certification Elements 83](#_Toc514686430)

[14-3: Create Collections to Filter the Query 85](#_Toc514686431)

[14-4: Use a Map to Aggregate Results 87](#_Toc514686432)

[14-5: Create Certification Held Records 89](#_Toc514686433)

[14-6: Use a Workflow to Avoid Creation of Duplicate Records (Optional) 91](#_Toc514686434)

[15-1: Create a Simple Visualforce Page 94](#_Toc514686435)

[15-2: Display Data in a Visualforce Page 95](#_Toc514686436)

[16-1: Create a Simple Technician Status Page 98](#_Toc514686437)

[16-2: Refine Your Page and Add Navigational Links 101](#_Toc514686438)

[17-1: Reference a Controller Extension in a Visualforce Page 102](#_Toc514686439)

[17-2: Create a Simple Read-Only Property 105](#_Toc514686440)

[17-3: Creating a Read/Write Property in a Custom Controller 108](#_Toc514686441)

[17-4: Implement the Search Button 110](#_Toc514686442)

[17-5: Redirecting to a Results Page 112](#_Toc514686443)

[17-6: Handle Basic Save Errors in Your Method 114](#_Toc514686444)

[18-1: Create a Page to Display a List of Records 116](#_Toc514686445)

[18-2: Integrate SOSL Search in a Visualforce Page 117](#_Toc514686446)

[18-3: Create a Simple Search Page 118](#_Toc514686447)

[19-1: Determine Whether a Declarative Solution Exists 120](#_Toc514686448)

[19-2: Defend Against SOQL Injection 122](#_Toc514686449)

[19-3: Create a Custom Button that Uses JavaScript (Optional) 124](#_Toc514686450)

[20-1: Write the Test Method for the Constructor 126](#_Toc514686451)

[20-2: Write Unit Tests for Action Methods 128](#_Toc514686452)

[20-3: Write Unit Tests for Getters and Setters 130](#_Toc514686453)

1-1: Explore the Certification App

Goal:

Familiarize yourself with the custom certification app.

Tasks:

# 1. Locate the correct Service Vendor account.

# 2. Create a new technician record.

# 3. Sign your new technician up for training.

# 4. Add a certification attempt for your technician.

# 5. Document that your technician has earned the certification.

Time:

5 minutes

Instructions:

# 1. Locate the correct Service Vendor account.

## A. Click **App Launcher**, then click Certification app.

## B. Click the Accounts tab.

## C. Click **Recently Viewed**,then click Service Vendor Accounts list view.

## D. Click Windy City Network Solutions.

# 2. Create a new technician record.

## A. In the Contacts related list, click New.

## B. Click the Technician record type, then click Next.

## C. Enter the following information:

|  |  |
| --- | --- |
| First Name | Jonas |
| Last Name | Whittier |

## D. Click Save.

# 3. Sign your new technician up for training.

## A. Click the Courses tab.

## B. Click **Recently Viewed**, then click **All** list view.

## C. Click record named [101] AWCA Server.

## D. In the Course Deliveries related list, click record named **DELIVERY-00034**.

## E. In the Course Attendees related list, click New.

## F. Enter the following information:

|  |  |
| --- | --- |
| Student | Jonas Whittier |
| Course Delivery(should be pre-populated) | DELIVERY-00034 |
| Status | **Enrolled** |

## G. Click Save.

# 4. Add a certification attempt for your technician.

## A. In the Course Attendees related list, click record named Jonas Whittier.

## B. In the Certification Attempts related list, click New.

## C. Click the **Multiple Choice** record type, then click Next.

## D. Enter the following information:

|  |  |
| --- | --- |
| Certification Element | AWCA Server Multiple Choice |
| Certification Candidate (should be pre-populated) | Jonas Whittier |
| Attempt Date | **Today’s Date** |
| Status | **Complete/Pass** |

## E. Click Save.

# 5. Add a Certification Held record for your technician.

## A. In the Certification Attempts related list, click AWCA Server Multiple Choice.

## B. Click AWCA Server to view the Certification record.

## C. In the Certifications Held related list, click New.

## D. Enter the following information:

|  |  |
| --- | --- |
| Certification  (should be pre-populated) | AWCA Server |
| Certified Professional | Jonas Whittier |
| Date Achieved | **Today’s Date** |

## E. Click Save.

1-2: Prepare Your Training Org

Goal:

Prepare your org for classroom activities and access after class.

Tasks:

# 1. Update your user details in your training org.

# 2. Confirm email address.

# 3. Download your lab files from the Documents tab.

# 4. Verify the Developer Console settings.

Time:

5 minutes

Instructions:

# 1. Update your user details in your training org.

## A. Click Your Profile | Settings.

## B. Click Advanced User Details under My Personal Information.

## C. Click Edit.

## D. Enter the following information:

|  |  |
| --- | --- |
| First Name | (your first name) |
| Last Name | (your last name) |
| Email | (an email you can access from class) |
| Debug Mode | Select |
| Development Mode | Select |
| Show View State in Development Mode | Select |

## D. Click Save.

# 2. Download your lab files from the Files tab.

## A. Click the Files tab.

## B. Click **DEX450\_ProgrammaticDevelopmentUsingApexAndVF\_LabFiles** file.

## C. Download the zip file to your Desktop.

## D. From Windows Explorer, right-click the zip file and click 7-Zip | Extract Here.

## Note: If you do not see 7-Zip as an option, then you can download and install it from http://www.7-zip.org/.

# 3. Verify the Developer Console settings.

A. Click Setup gear icon | Developer Console.

## B. Click **Debug | Perspective Manager…**, and set the default perspective to Log Only (Predefined), if it is not already the default.

## C. Close the Developer Console window.

1-3: Create a Sandbox

Goal:

Create a sandbox to configure and test changes separate from the production environment.

Task:

# 1. Create a full sandbox named dev.

Time:

5 minutes

Instructions:

# 1. Create a full sandbox named **dev**.

## A. Click Setup gear icon | Setup.

## B. Type sand in the Quick Find textbox.

## C. Click Sandboxes under Environments.

## D. Click New Sandbox.

## E. Enter the following information:

|  |  |
| --- | --- |
| Name | dev |
| Description | This is a development environment. |

## F. In the Full column, click Next.

## G. Select the sandbox options.

|  |  |
| --- | --- |
| Object Data Included | All |
| Include Field Tracking History Data | Deselect |
| Include Chatter Data | Select |

H. Click Create.

1-4: Download the Apex Developer’s Guide

Goal:

Download the Apex Developer’s Guide to use as a resource.

Task:

# 1. Download the Apex Developer’s Guide.

Time:

5 minutes

Instructions:

# 1. Download the Apex Developer’s Guide.

## A. In a new browser tab, navigate to <https://developer.salesforce.com>.

## B. Type What is Apex? in the search window, then press ENTER.

## C. Click What is Apex? | Apex Developer Guide ...

## D. Right-click the PDF button in the upper-left hand corner and select Save Link As…

## E. Choose the Desktop as the location, and click Save to download the guide as a PDF.

2-1: Create a Custom Object

Goal:

Create a custom object to track customer success stories.

Tasks:

# 1. View the Account and Contact standard objects in Schema Builder.

# 2. Create a custom object from Schema Builder.

# 3. View the object detail page in the Setup menu.

# 4. Edit the object permissions on the Sales User and Marketing User profiles.

# 5. View the organization-wide default setting for the new object.

Time:

15 minutes

Instructions:

# 1. View the Account and Contact standard objects in Schema Builder.

## A. Click Setup gear icon | Setup.

## B. Type schema in the Quick Find textbox.

## C. Click Schema Builder under Objects and Fields.

## D. Click the **Objects** tab, then click Clear All.

## E. Select the Account and Contact objects.

## F. Click **Auto-Layout,** then arrange the two objects so they are visible in the canvas.

## G. Click View Options | Display Element Names if it is an available option.

# 2. Create a custom object from Schema Builder.

## A. Click the Elements tab.

## B. Drag and drop Object from the palette onto the canvas.

## C. Enter the following information:

|  |  |
| --- | --- |
| Label | Customer Story |
| Plural Label | Customer Stories |
| Starts With | Consonant |
| Object Name | Customer\_Story |
| Description | Used to track customer success stories. |
| Context-Sensitive Help Setting | Open the standard Salesforce.com Help & Training window |
| Record Name | Customer Story Name |
| Data Type | Text |
| Allow Reports | Select |
| Allow Activities | Deselect |
| Track Field History | Deselect |
| In Development | Deployed |
| Add Google Docs, Notes, and Attachments related list to default page layout | Deselect |

## D. Click Save.

## E. Rearrange the Customer Story object so it is visible in the canvas.

# 3. View the object detail page in Object Manager of the Setup menu.

## A. Click the gear icon on the Customer Story object, then click View Object.

### Click **Fields & Relationships**. What standard fields were automatically created?

## B. View the other sections of the Object Management Settings page like **Page Layouts** and **Buttons, Links, and Actions**. What other entities were automatically created?

# 4. Edit the object permissions on the Sales User and Marketing User profiles.

## A. Click Setup gear icon | Setup.

## B. Type profiles in the Quick Find textbox.

## C. Click Profiles under Users.

## D. Click Sales User, which is a custom profile.

## E. Click Object Settings.

## F. Click Customer Stories.

## G. Click Edit.

## H. Select Enabled for the Read, Create, and Edit permissions.

## I. Click Save.

## J. Click **Profiles** under Users.

## K. Click General Marketing User, which is a custom profile.

## L. Click Object Settings.

## M. Click Customer Stories.

## N. Click Edit.

## O. Select Enabled for the Read, Create, and Edit permissions.

## P. Click Save.

# 5. View the organization-wide default setting for the new object.

## A. Type sharing in the Quick Find textbox.

## B. Click Sharing Settings under Security.

### What is the organization-wide default setting for the Customer Story object?

2-2: Create Custom Fields

Goal:

Create custom fields on the Customer Stories object to track the story description, products, and installation time.

Tasks:

# 1. Using Schema Builder, add a text field to track the story description and view the field-level security.

# 2. Using the Setup menu, add the field to the page layout.

# 3. Using the Setup menu, add a multi-select picklist field to track products and a number field to track installation time.

Time:

15 minutes

Instructions:

# 1. Using Schema Builder, add a text field to track the story description and view the field-level security.

## A. Click Setup gear icon | Setup.

## B. Type schema in the Quick Find textbox.

## C. Click Schema Builder under Objects and Fields.

## D. From the Elements tab, drag and drop Text onto the Customer Story object.

## E. Enter the custom field details:

|  |  |
| --- | --- |
| Field Label | Story Description |
| Field Name | Story\_Description |
| Description | Description of the customer story |
| Help Text | Brief overview of the customer success story |
| Length | 255 |
| Default Value | Leave blank |
| Required | Deselect |
| Unique | Deselect |
| External ID | Deselect |

## F. Click Save.

## G. Hover over the Story Description field, right click, then click Manage Field Permissions.

### Which users can view and edit this field?

### How would you restrict users from editing this field?

### How would you restrict users from viewing this field?

## H. Click Cancel.

# 2. Using the Setup menu, add the field to the page layout.

## A. Click gear icon on the Customer Story object, then click View Page Layouts.

## B. Click **Customer Story Layout**.

## C. Drag and drop Story Description from the palette to below the Customer Story Name field (in the Information section).

## D. Click Save.

# 3. Using the Setup menu, add a multi-select picklist field to track products and a number field to track installation time.

## A. Click **Fields & Relationships**, then click **New**.

## B. Select Picklist (Multi-Select) as the data type, then click **Next**.

## C. Enter the following information:

|  |  |
| --- | --- |
| Field Label | Products |
| Values | **Enter values, with each value separated by a new line** |
| Enter values, with each value separated by a new line. | Desktops Laptops Printers Accessories Networking Equipment Servers |
| Display values alphabetically, not in the order entered | Select |
| Use first value as default value | Deselect |
| Restrict picklist to the values defined in the value set | Select |
| # Visible Lines: | 4 |
| Field Name | Products |
| Description | Products bought by the customer |
| Help Text | Select one or more products |

## D. Click Next.

## E. Leave the field-level security as-is and click Next.

## F. Choose to automatically add the field to the page layout and click Save & New.

## G. Select Number as the data type, then click **Next**.

## H. Enter the following information:

|  |  |
| --- | --- |
| Field Label | Installation Time (days) |
| Length | 3 |
| Decimal Places | 0 |
| Field Name | Installation\_Time |
| Description | Installation time |
| Help Text | Number of days from purchase to installation complete |
| Required | Deselect |
| Unique | Deselect |
| External ID | Deselect |
| Default Value | Leave blank |

## I. Click Next.

## J. Leave the field-level security as-is and click Next.

## K. Choose to automatically add the field to the page layout and click Save.

## L. Close the Custom Object browser tab.

## M. Refresh the Schema Builder browser tab to see the new fields.

### What are the differences between creating a field using the Setup menu and creating a field using Schema Builder?

2-3: Create Relationship Fields

Goal:

Relate the Customer Stories object to the Account and Contact objects.

Tasks:

# 1. Create a Master-Detail relationship field and add a filter to limit the records available to users.

# 2. Create a Lookup relationship field and view the lookup options.

# 3. Add the Customer Stories related list to the account page layout.

# 4. Create a customer story record to verify the object was configured properly.

Time:

20 minutes

Instructions:

# 1. Create a Master-Detail relationship field and add a filter to limit the records available to users.

## A. Click Setup gear icon | Setup.

## B. Type schema in the Quick Find textbox.

## C. Click Schema Builder under Objects and Fields.

## D. From the Elements tab, drag and drop Master-Detail onto the Customer Story object.

## E. Enter the following information:

|  |  |
| --- | --- |
| Field Label | Account |
| Field Name | Account |
| Description | The Account whose success story this is. |
| Help Text | Leave blank |
| Related To | Account |
| Child Relationship Name | Customer\_Stories |
| Sharing Setting | Read/Write: Allows users with at least Read/Write access to the Master record to create, edit, or delete related Detail records. |
| Related List Label | Customer Stories |
| Reparentable Master Detail | Deselect |

## F. Click Save.

## G. Hover over the Account field, right-click, then click View Field in New Window.

## H. Click Edit.

## I. Click Show Filter Settings and enter the following information:

|  |  |
| --- | --- |
| Field | Account: Account Record Type |
| Operator | equals |
| Value / Field | Value |
| Value | Customer |
| Filter Type | Required. The user-entered value must match filter criteria. |
| If it doesn’t, display this error message on save | Select a customer account. |
| Lookup Window Text | Leave blank |
| Active | Select |

## J. Click Save.

## K. Click Setup.

## L. Type sharing in the Quick Find textbox.

## M. Click Sharing Settings under Security.

### What is the organization-wide default setting for the Customer Story object?

## N. Close the Sharing Settings browser tab.

# 2. Create a Lookup Relationship field and view the lookup options.

## A. From the Elements tab, drag and drop Lookup onto the Customer Story object.

## B. Enter the custom field details:

|  |  |
| --- | --- |
| Field Label | Primary Contact |
| Field Name | Primary\_Contact |
| Description | The Primary Contact of the Account whose success story this is. |
| Help Text | Leave blank |
| Related To | **Contact** |
| Child Relationship Name | Customer\_Stories |
| Related List Label | Customer Stories |

## C. Click Save.

## D. Hover over the Primary Contact field, right-click, then click View Field in New Window.

## E. Click Edit.

### What are the lookup options?

## F. Click Cancel.

## G. Click **Setup**.

## H. Click the down arrow next to Object Manager, then click **Customer Story**.

## Click **Page Layouts**.

## Click **Customer Story Layout**.

## K. From the page layout, drag and drop Account field to below Currency field.

## L. From the palette, drag and drop Primary Contact field to below Account field.

## M. Click Save.

# 3. Add the Customer Stories related list to the account page layout.

## A. Click **Object Manager**.

## B. Click **Account**.

## C. Click Page Layouts.

## D. Click **Customer Account Layout**.

## E. From the palette, click Related Lists.

## F. Drag and drop Customer Stories to below the Cases related list.

## G. Click Save.

## H. Click Yes.

## I. Close the Account Page Layout browser tab.

# 4. Create a customer story record to verify the object was configured properly.

## A. In Schema Builder, click Close.

## B. Click **App Launcher**, then click Accounts.

## C. Click New.

## D. Click the **Customer** record type, then click Next.

## E. Enter the following information:

|  |  |
| --- | --- |
| Account Name | Test Account |

## F. Click Save.

## G. From the Contacts related list, click New.

## H. Click the **Standard** record type, then click **Next**.

## I. Enter the following information:

|  |  |
| --- | --- |
| First Name | Kate |
| Last Name | Hanson |

## J. Click Save.

## K. From the Customer Stories related list, click New.

## L. Enter the following information:

|  |  |
| --- | --- |
| Customer Story Name | Test Account Customer Story |
| Story Description | Major win at new customer against top competitor. |
| Products Chosen | **Laptops Networking Equipment Servers** |
| Installation Time (days) | 30 |
| Currency | **USD – U.S. Dollar** |
| Account | Test Account |
| Primary Contact | Kate Hanson |

## M. Hover over the info icon next to the Installation Time (days) field to view the help text.

## N. Click Save.

## O. From the Contacts related list, find the row for Kate Hanson and click the down arrow on the far right, then click **Delete** to delete the contact record.

## P. Click Delete to confirm.

## Q. From the Customer Stories related list, right-click Test Account Customer Story and click Open Link in New Tab. In the Customer Story tab, what happened to the value in the Primary Contact field? Why did this happen?

## R. Return to the Test Account record browser tab.

## S. Click Delete to delete the account record.

## T. Click Delete to confirm.

## U. Return to the Test Account Customer Story tab and refresh the page.

### What happened to the Test Account Customer Story record? Why did this happen?

3-1: Create a Formula Field

Goal:

Create a formula field on the Course Delivery object to calculate the end date.

Tasks:

# 1. Using the Setup menu, add a formula field to calculate the course delivery end date.

# 2. Test the formula field.

Time:

10 minutes

Instructions:

# 1. Using the Setup menu, add a formula field to calculate the course delivery end date.

## A. Click Setup gear icon | Setup.

## B. Click **Object Manager**.

## C. Click Course Delivery.

## D. Click **Fields & Relationships**, then click **New**.

## E. Select Formula as the data type, then click **Next**.

## F. Enter the following information:

|  |  |
| --- | --- |
| Field Label | End Date |
| Field Name | End\_Date |
| Formula Return Type | Date |

## G. Click Next.

## H. Click **Advanced Formula** tab, then enter the following formula:

Start\_Date\_\_c + Course\_\_r.Duration\_\_c - 1

## I. Click Check Syntax.

## J. Type End date of the course in the Description field.

## K. Click Next.

## L. Leave the field-level security as-is and click Next. Why is this field read only?

## M. Choose to automatically add the field to the page layout and click Save.

# 2. Test the formula field.

## A. From the Certification app, click the Courses tab.

## How many courses do you see?

## B. Click Recently Viewed, then click All list view.

## How many courses do you see?

## C. Click record named [401] Data Recovery.

### What is the course duration?

## D. In Course Deliveries related list, click record named DELIVERY-00012.

### What is the start date?

### What is the end date?

3-2: Create a Roll-Up Summary Field

Goal:

Create a roll-up summary field on the Course object to count the number of times a course was cancelled.

Tasks:

# 1. Using the Setup menu, add a roll-up summary field to count the number of cancellations.

# 2. Test the roll-up summary field.

Time:

10 minutes

Instructions:

# 1. Using the Setup menu, add a roll-up summary field to count the number of cancellations.

## A. Click Setup gear icon | Setup.

## B. Click **Object Manager**.

## C. Click Course.

## D. Click **Fields & Relationships**, then click New.

## E. Select Roll-Up Summary as the data type, then click **Next**.

## F. Enter the following information:

|  |  |
| --- | --- |
| Field Label | # of Courses Cancelled |
| Field Name | Number\_of\_Courses\_Cancelled |
| Description | Number of courses cancelled. |

## G. Click Next and define the summary calculation.

|  |  |
| --- | --- |
| Summarized Object | Course Deliveries |
| Select Roll-Up Type | Count |
| Filter Criteria | Only records meeting certain criteria should be included in the calculation |
| Field | **Status** |
| Operator | equals |
| Value | **Cancelled** |

## H. Click Next.

## I. Leave the field-level security as-is and click Next.

## J. Choose to automatically add the field to the page layout and click Save.

# 2. Test the roll-up summary field.

## A. From the Certification app, click the Courses tab.

## B. Click Recently Viewed, then click All list view.

## C. Click record named [102] AWCA Network.

### How many courses were cancelled?

### How many courses were delivered?

3-3: Create a Formula Field that References Roll-Up Summary Fields

Goal:

Create a formula field on the Course object to calculate the cancellation rate.

Tasks:

# 1. Using the Setup menu, add a formula field to calculate the course cancellation rate.

# 2. Test the formula field.

Time:

10 minutes

Instructions:

# 1. Using the Setup menu, add a formula field to calculate the course cancellation rate.

## A. Click Setup gear icon | Setup.

## B. Click **Object Manager**.

## C. Click Course.

## D. Click **Fields & Relationships**, then click New.

## E. Select Formula as the data type, then click **Next**.

## F. Enter the following information:

|  |  |
| --- | --- |
| Field Label | Course Cancellation Rate |
| Field Name | Course\_Cancellation\_Rate |
| Formula Return Type | Percent |
| Decimal Places | 2 |

## G. Click Next.

## H. Click the **Advanced Formula** tab.

## I. From the Functions drop down, choose **IF** and click Insert Selected Function.

## J. Replace logical\_test, value\_if\_true and value\_if\_false to write a formula that looks like this:

IF((Number\_of\_Courses\_Cancelled\_\_c + Number\_of\_Courses\_Delivered\_\_c) > 0, Number\_of\_Courses\_Cancelled\_\_c / ( Number\_of\_Courses\_Cancelled\_\_c + Number\_of\_Courses\_Delivered\_\_c ), null)

## K. Click Check Syntax.

## L. Type Cancellation rate of the course in the Description field.

## M. Click Next.

## N. Leave the field-level security as-is and click Next.

## O. Choose to automatically add the field to the page layout and click Save.

# 2. Test the formula field.

## A. From the Certification app, click the Courses tab.

## B. Click Recently Viewed, then click All list view.

## C. Click record named [102] AWCA Network.

### What is the course cancellation rate?

3-4: Understand Record Types

Goal:

Understand the capabilities of record types.

Tasks:

# 1. View account records for each record type.

# 2. Create an account record using the Service Vendor record type.

Time:

10 minutes

Instructions:

# 1. View account records for each record type.

## A. From the Certification app, click the Accounts tab.

## B. Click Recently Viewed, then click All list view.

## C. Click record named ABC Labs.

## D. Verify that the Industry field, the Opportunities related list, and the Cases related list are displayed on the page.

## E. Double click on the Support Level field to edit it. Verify that the picklist options are Silver, Gold, and Platinum.

## F. Click the Accounts tab.

## G. Click Recently Viewed, then click All list view.

## H. Click record named Alveswood Technologies.

## I. Verify that the Industry field, the Opportunities related list, and the Cases related list are not displayed on the page.

## J. Double click on the Support Level field to edit it. Verify that the picklist options are Standard Vendor and Premier Vendor.

# 2. Create an account record using the Service Vendor record type.

## A. Click the Accounts tab.

## B. Click New.

## C. Select the **Service Vendor** record type, and click **Next**.

## D. Enter the following information:

|  |  |
| --- | --- |
| Account Name | Test Service Vendor Account |

## E. Click **Save**.

4-1: Logging into a Sandbox

Goal:

Explore your sandbox.

Tasks:

# 1. Log in in to your sandbox.

# 2. Review the data.

# 3. Review user records.

# 4. Log out of your sandbox.

Time:

10 minutes

Instructions:

# 1. Log in in to your sandbox.

## A. Open a browser tab and browse to https://test.salesforce.com. This will open a page that allows you to log in to your sandbox.

## B. When you created your sandbox, you were asked to enter a name of dev. Your username for the sandbox should be the same as your regular username, followed by a period and the sandbox name. Note: the sandbox name is not case sensitive.

## For example, if your username is: yourname@trg.org

## then your sandbox username will be: yourname@trg.org**.dev**

## C. Type in the same password you used for your production training org.

## D. Click Log in to Sandbox.

## E. You should notice a banner at the top of the page that reads, “Sandbox: dev”.

# 2. Review the data.

## A. Click **App Launcher**, then select **Certification** app.

## B. Click the Courses tab.

## C. Notice that all the course data available in your training org is also in this sandbox instance.

# 3. Review user records.

## A. Click Setup gear icon | Setup.

## B. Type users in the Quick Find textbox.

## C. Click Users under Users.

## D. Click Edit for the first user listed.

## E. Notice that the email address has been modified, the original @ is replaced by = and the email address is appended with @example.com, so that production users don’t receive automatically generated email messages from the sandbox.

## Note: Sandboxes change Salesforce user email addresses, but don’t change other email addresses in Salesforce, such as email addresses in contact records. To avoid sending unsolicited email from your sandboxes, manually invalidate or delete all email addresses in your sandboxes that don’t belong to users of the sandbox.

# 4. Log out of your sandbox.

## A. In the Salesforce UI in the top right-hand corner, select Your Profile | Log Out.

## B. Close the browser tab. Note that, unless otherwise stated, all other exercises should be completed in your **production** org.

4-2: See Apex in Action

Goal:

Execute Apex to create a new Contact record in the database.

Tasks:

# 1. Configure the Developer Console.

# 2. Open Execute Anonymous window.

# 3. Enter the code to execute.

# 4. Examine the logs.

# 5. Examine the result in the user interface.

Time:

15 minutes

Instructions:

# 1. Configure the Developer Console.

## A. Click Setup gear icon | Developer Console.

## B. Click Debug, then deselect Show My Current Logs Only.

### C. Click Change Log Levels.

### D. Click **Add/Change** under DebugLevel Action.

### E. Set the DB, Apex Code and Profiling entries to Finest and System to Fine.

### F. Click **Done.**

### G. Click **Done**.

# 2. Open Execute Anonymous window.

## A. Click Debug | Open Execute Anonymous Window.

## B. Remove any existing code in the **Enter Apex Code** window.

## C. Select Open Log in the bottom right-hand corner.

Note: You can expand the window by clicking the Up arrow in the top right corner.

# 3. Enter the code to execute.

## A. Copy and paste the contents of **ContactExecuteAnonymous.txt** from the Exercises folder into the window, overwriting all existing text.

## B. Click Execute in the bottom right-hand corner.

# 4. Examine the logs.

## A. When the Execution Log opens, select Debug Only. As a result, only debug statements should be visible. It should display only one log entry with the ID of the inserted record.

## B. Deselect Debug Only.

## C. Enter DML into the Filter input box (case-sensitive).

## D. Notice the limits usage for DML statements and DML rows.

## E. Select File | Close All.

# 5. Examine the result in the Salesforce user interface.

## A. In the Salesforce UI, from the Certification app, click on the Contacts tab.

## B. Confirm that “June Morgan”, the record inserted using the code, appears in the Recent Contacts section.

4-3: Create and Use an Apex Class

Goal:

Create an Apex class called ContactManager and define a method within the class to create a Contact record in the database.

Tasks:

# 1. Create a class and save it.

# 2. Invoke the class method using code in the Execute Anonymous window.

# 3. Examine the logs to see the invocation of the class.

Time:

10 minutes

Instructions:

# 1. Create a class and save it.

## A. Click Setup gear icon | Developer Console.

### B. Click File | New | Apex Class.

### C. Type ContactManager for the name, then click **OK**.

### D. Copy and paste the contents of **ContactManagerClass.txt** from the Exercises folder into the window, overwriting all existing text.

### E. Press CTRL + S to save the file.

### F. Note the API Version of the class.

# 2. Invoke the class method using code in the Execute Anonymous window.

## A. Press CTRL + E to open the **Execute Anonymous** window.

## B. Copy the below code into the window, overwriting all existing text:

ID contactID = ContactManager.addContact('Rehman','Areil');

System.debug('Called from Execute Anonymous: ID=' + contactID);

## C. Ensure that Open Log is selected and click Execute.

# 3. Examine the logs to see the invocation of the class.

## A. In the debug log for your execution, select Debug Only. As a result, only debug statements should be visible.

## B. Notice that the newly created record has an ID, which was returned by the method.

## C. Deselect Debug Only.

## D. Enter ContactManager into the Filter input box (case-sensitive).

## E. Notice the invocation of the static addContact() method of the ContactManager class.

## F. Press CTRL + ALT + / to close all console tabs.

## G. Close the Developer Console window.

4-4: Observe the Effects of Versioning

Goal:

Change the version on an Apex class to see how it affects compilation.

Tasks:

# 1. Create a new class to test out versioning in Apex.

# 2. Discover what versions will compile FeedPost.

# 3. Discover what versions will compile FeedItem.

Time:

5 minutes

Instructions:

# 1. Create a new class to test out versioning in Apex.

## A. Click Setup gear icon | Setup.

## B. Type apex in the Quick Find textbox.

## C. Click Apex Classes under Custom Code.

## D. Click New.

## E. Copy the below code into the window, overwriting all existing text:

## public class ChatterVersion {

## }

## F. Click Save.

## 2. Discover what versions will compile FeedPost.

## Background: When Chatter was first inroduced, the original sObject that stored Chatter posts was called FeedPost. For example, inserting an sObject of type FeedPost would make a Chatter post to a user’s profile, another user’s profile, a Chatter group, or on a record.

## A. Click Edit to modify the ChatterVersion class.

## B. Select the Version Settings tab.

## C. Change the Salesforce.com API Version to 17.0.

## D. Click Quick Save.

## E. Select the Apex Class tab.

## F. Copy the below code into the window, overwriting all existing text:

## public class ChatterVersion {

## FeedPost fp;

## }

## G. Click Quick Save. You should see the error **Compile Error: Invalid type: FeedPost**.

## H. Select the Version Settings tab.

## I. Change the Salesforce.com API Version to 18.0.

## J. Click Save. You should not see an error.

# 3. Discover what versions will compile FeedItem.

## A. Click Edit to modify the ChatterVersion class.

## B. Select the Version Settings tab.

## C. Change the Salesforce.com API Version to 22.0.

## D. Click Quick Save. You should see the error **Compile Error: Invalid type: FeedPost**.

## E. Select the Apex Class tab.

## F. Copy the below code into the window, overwriting all existing text:

## public class ChatterVersion {

## // FeedPost fp;

## FeedItem fi;

## }

## G. Click Save. You should not see an error.

4-5: Take a Quick Tour of Apex

Goal:

Quickly learn some Apex fundamentals that will be familiar to you, and check out some simple differences.

Tasks:

# 1. Set up the classes.

# 2. Review the test classes.

# 3. Run the test.

# 4. View code coverage.

Time:

20 minutes

Instructions:

# 1. Set up the classes.

## A. Click Setup gear icon | Developer Console.

## B. Click File | New | Apex Class.

## C. Type CourseManager for the name, then click **OK**.

## D. Copy and paste the contents of **CourseManagerClass.txt** from the Exercises folder into the window, overwriting all existing text.

## E. Complete the TODOs.

## F. Press CTRL + S to save the file.

# 2. Review the test classes.

## A. Click File | New | Apex Class.

## B. Type CourseManager\_Test for the name, then click **OK**.

## C. Copy and paste the contents of **CourseManager\_Test.txt** from the Exercises folder into the window, overwriting all existing text.

## Note the following in the code:

### i. Lines 14 and 15 create two lists of records. There will be 3 overlapping records with names: course3, course4, and course5.

### ii. Line 17 checks whether the code that will be moved to production actually fulfills the business requirement. If the assertion fails, an exception will be caused, which will fail the test.

## D. Press CTRL + S to save the file.

# 3. Run the test.

## A. Click **Test |** New Run.

### i. Select the CourseManager\_Test class.

### ii. Click Add Selected.

### iii. Click Run.

## B. Switch to the Tests tab in the bottom panel and locate the most recent Test Run (it is usually at the bottom of the list).

## C. Expand the collapsed folder by clicking +.

## D. Ensure that all your methods executed successfully. A green tick indicates success.

# 4. View code coverage.

## A. Switch to the tab for the **CourseManager** class.

## B. Click Code Coverage: (next to API Version).

Note: The lines that are covered appear blue while those not covered appear red.

## C. Press CTRL + ALT + / to close all console tabs.

## D. Close the Developer Console window.

4-6: Examine Implicit Operations

Goal:

Examine the operations implicitly occurring when you perform a DML operation.

Tasks:

# 1. Create a Course Attendee record in the user interface.

# 2. View the logs to see operations implicitly invoked due to the DML operation

Time:

10 minutes

Instructions:

# 1. Create a Course Attendee record in the user interface.

## A. From the Certification app, click the Course Deliveries tab.

## B. Click Recently Viewed, then click All list view.

## C. Click record named DELIVERY-00025.

## D. From the Course Attendees related list, click **New**.

## E. Enter the following information:

|  |  |
| --- | --- |
| Student | Clara Petit |
| Status | **Enrolled** |

## F. Click Save.

# 2. View the logs to see operations implicitly invoked due to the DML operation.

## A. Click Setup gear icon | Developer Console.

## B. Click Debug, then deselect Show My Current Logs Only.

## C. Click the Logs tab in the bottom panel.

## D. Double-click on the most recent entry.

## E. Enter Trigger in the Filter input box (case-sensitive).

## F. Confirm that the CourseAttendeeTrigger was executed.

## G. Confirm that the provideAccessLMS method of the CourseAttendeeTriggerHandler class was executed.

## H. Enter WF in the Filter input box (case-sensitive), replacing existing text.

## I. Confirm that the **New Course Created** workflow rule was evaluated. The ON\_CREATE\_ONLY entry in the Details column shows that it was not executed because it did not meet the criteria.

## J. Press CTRL + ALT + / to close all console tabs.

## K. Close the Developer Console window.

4-7: Profile Limits Using the Developer Console

Goal:

Investigate limits using the Developer Console.

Task:

# 1. Review the limit profiling information in the log.

Time:

5 minutes

Instructions:

# 1. Review the limit profiling information in the log.

## A. Click Setup gear icon | Developer Console.

## B. Click Debug, then deselect Show My Current Logs Only.

## C. Click the Logs tab in the bottom panel.

## D. Double-click on the most recent entry.

## E. Click Debug | Switch Perspective | All (Predefined).

## F. In the **Execution Overview** panel, notice the Save Order of the various operations.

## G. Click the Timeline tab in the Execution Overview panel. Notice the entries in the Category, Millis, and % columns.

## H. Switch to the Limits tab in the Execution Overview panel. Enter STATEMENT in the Filter input box of the Execution Log panel (case-sensitive).

## I. Click the **first entry** in the Execution Log panel. This resets the “Used so far” column in the Limits tab. The Source panel jumps to the line of code that was executed at that point in time.

## J. Click the **third-to-last entry** in the Execution Log panel. The values for SOQL and SOQL\_ROWS Limits “Used so far” are updated. The Source panel jumps to the line of code that was executed at that point in time.

## K. Click the **last entry** in the Execution Log panel. All limits used now display.

## L. Press CTRL + ALT + / to close all console tabs.

## M. Close Developer Console window.

4-8: Work with a Custom Object

Goal:

Work with the fields of the Course object.

Tasks:

# 1. Write code to insert a record for the Course object.

# 2. Check the Salesforce UI to ensure that the record was inserted successfully.

Time:

15 minutes

Instructions:

# 1. Write code to insert a record for the Course object.

## A. Click Setup gear icon | Developer Console.

## B. Press CTRL + E to open the **Execute Anonymous** window.

## C. Copy and paste the contents of **CreateCustomObjRecord.txt** from the Exercises folder into the window, overwriting all existing text.

## D. Complete the TODOs.

## E. Ensure that Open Log is selected and click Execute.

## F. Press CTRL + ALT + / to close all console tabs.

## G. Close the Developer Console window.

# 2. Check the Salesforce UI to ensure that the record was inserted successfully.

## A. From the Certification app, click the Courses tab.

## B. Check that your record was created.

## 

4-9: Use Record IDs to Access a Contact in the UI

Goal:

Explore the use of record IDs.

Tasks:

# 1. Query the ID and name fields for Contact records in the database.

# 2. Copy the ID from a returned record.

# 3. Use the ID to display the associated Contact page in the UI.

Time:

5 minutes

Instructions:

# 1. Query the ID and name fields for Contact records in the database.

## A. Click Setup gear icon | Developer Console.

## B. Open the Contact object in the Object Inspector for reference.

### i. Click **File | Open**.

### ii. Select **Objects** as the Entity Type, type Contact into the filter input box, then double-click the entity **Contact**. As a result, you should see a list of the Contact sObject’s fields and their data types.

### iii. Click the Name header to sort the fields by name.

## C. Create a query.

### i. Holding the CTRL key, select the following fields from the Contact.obj tab:

|  |
| --- |
| Id |
| Name |

### ii. Click Query. The focus will shift to the Query Editor tab in bottom panel.

## D. Run a query.

### i. Ensure the **Use Tooling API** checkbox is not selected.

### ii. Click Execute. As a result, a Query Results tab opens.

# 2. Copy the ID from a returned record.

## A. Duble-click the ID of any record, then select it.

## B. Copy and paste the ID into a text editor such as Notepad++.

## C. Press CTRL + ALT + / to close all console tabs.

## D. Close the Developer Console window.

# 3. Use the ID to display the associated Contact page in the UI.

## A. From the Certification app, click the Home tab.

## B. In the URL, replace **lightning/page/home** with the ID of your record.

## C. Press ENTERto display the record.

## D. Determine the length of the ID in the browser address bar. Is it 15 or 18 characters?

5-1: Create and Run a Query in the Developer Console

Goal:

Retrieve Cases using the Query Editor in the Developer Console.

Task:

# 1. Run a query in the Query Editor in the Developer Console.

Time:

5 minutes

Instructions:

# 1. Run a query in the Query Editor in the Developer Console.

## A. Click Setup gear icon | Developer Console.

## B. Open the Case object in the Object Inspector for reference.

### i. Click **File | Open**.

### ii. Select **Objects** as the Entity Type, type Case into the filter the repository input, then double-click the entity **Case**. As a result, you should see a list of the Case sObject’s fields and their data types.

### iii. Click the Name header to sort the fields by name.

## C. Create a query.

### i. Holding the CTRL key, select the following fields from the Case.obj tab:

|  |  |
| --- | --- |
| CaseNumber | IsClosed |
| ClosedDate | Status |
| Id | Subject |

### ii. Click Query. The focus will shift to the Query Editor tab in bottom panel.

## D. Run a query.

### i. Ensure the **Use Tooling API** checkbox is not selected.

### ii. Click Execute. As a result, a Query Results tab opens.

## E. Explore row-level options.

### i. Select a single case in the Query Results tab.

### ii. Click Open Detail Page. As a result, the selected case’s detail page should appear.

### iii. Notice other actions available from the Query Results tab, such as creating, deleting, and updating sObject records.

## F. Press CTRL + ALT + / to close all console tabs.

## G. Close the Developer Console window.

5-2: Write a SOQL Query that Uses a WHERE Clause

Goal:

Retrieve Cases that meet specific criteria.

Tasks:

# 1. Run a query in the Query Editor in the Developer Console to retrieve all closed Cases.

# 2. Retrieve all Cases that do not have a specified type.

# 3. Retrieve all high-priority Cases that involve particular products.

# 4. Retrieve all Cases with a subject containing the word “printer.”

**Time**:

20 minutes

**Instructions**:

# 1. Retrieve all closed Cases.

## A. Click Setup gear icon | Developer Console.

## B. Open the Case object in the Object Inspector for reference.

### i. Click **File | Open**.

### ii. Select **Objects** as the Entity Type, type Case into the filter input box, then double-click the entity **Case**. As a result, you should see a list of the Case sObject’s fields and their data types.

### iii. Click the Name header to sort the fields by name.

## C. Create a query.

### i. Holding the CTRL key, select the following fields from the Case.obj tab:

|  |  |
| --- | --- |
| CaseNumber | Product\_Category\_\_c |
| Id | Status |
| IsClosed | Subject |
| Priority | Type |

### ii. Click Query. The focus will shift to the Query Editor tab in bottom panel.

## D. Run a query.

### i. Ensure the **Use Tooling API** checkbox is not selected.

### ii. Click Execute. As a result, a Query Results tab opens.iii. In the Query Results tab, click Status in the header row to sort the results by status. Notice that at least one Case’s status is ‘Closed’.

## E. Write and Execute a query with a WHERE clause.

### i. In the Query Editor tab in bottom panel, modify the query by manually adding a WHERE clause that specifies that only closed Cases should be retrieved.

### HINT: Review the WHERE clause operators in your Student Guide.

### ii. Click Execute. As a result, a Query Results tab opens.

## F. Close the Query Results tabs.

# 2. Retrieve all Cases that do not have a specified type.

## A. In the Query Editor tab in bottom panel, modify the query so it finds Cases that have no specified type.

## HINT: Type is a Nullable field, so a Case without a Type will have a NULL Type.

## B. Click Execute. As a result, a Query Results tab opens.

## C. Close the Query Results tab.

# 3. Retrieve all high-priority Cases that involve particular products.

## A. In the Query Editor tab in bottom panel, modify the query so it finds Cases that have a Priority of “High” and Product Category of “Printers”. HINT: Remember a WHERE clause supports multiple filter criteria using the logical operators: AND, OR.

## B. Click Execute. As a result, a Query Results tab opens.

## C. Close the Query Results tab.

# 4. Retrieve all Cases with a subject containing the word “printer.”

## A. In the Query Editor tab in bottom panel, modify the query so it finds Cases that have the word “printer” included somewhere in the Subject. The word “printer” can appear at the beginning, middle, or end of the Case Subject. HINT: Remember a WHERE clause supports the LIKE operator and wildcards when searching text values.

## B. Click Execute. As a result, a Query Results tab opens.

## C. Press CTRL + ALT + / to close all console tabs.

## D. Close the Developer Console window.

5-3: Write and Execute a SOQL Query in Apex

Goal:

Print cases into the debug log.

**Tasks**:

# 1. Print a single case to the debug log.

# 2. Print multiple cases to the debug log.

**Time**:

10 minutes

**Instructions**:

# 1. Print a single case to the debug log. When debugging, it can be helpful to query an sObject by its ID. Use with caution as record IDs may change between environments.

## A. Click Setup gear icon | Developer Console.

## B. Press CTRL + E to open the **Execute Anonymous** window.

## C. Copy and paste the contents of Task 1 in **SOQLQueriesInApex.txt** from the Exercises folder into the window, overwriting all existing text.

## D. Complete the TODOs.

## E. Ensure that Open Log is selected and click Execute.

## F. Select **Debug Only** checkbox, then review the Debug Log to confirm the results.

## G. Press CTRL + ALT + / to close all console tabs.

# 2. Print multiple cases to the debug log.

## A. Press CTRL + E to open the **Execute Anonymous** window.

## B. Copy and paste the contents of Task 2 in **SOQLQueriesInApex.txt** from the Exercises folder into the window, overwriting all existing text.

## C. Complete the TODOs.

## D. Ensure that Open Log is selected and click Execute.

## E. Select **Debug Only** checkbox, then review the Debug Log to confirm the results.

## F. Press CTRL + ALT + / to close all console tabs.

## G. Close the Developer Console window.

5-4: Write a Dynamic Query in Apex

Goal:

Retrieve cases that meet criteria specified at run time.

Tasks:

# 1. Use a bound variable to specify filter criteria.

# 2. Use Database.query() to repair the code from Task 1.

# 3. (Optional) Use a bound variable to query by record type.

Time:

10 minutes

Instructions:

# 1. Use a bound variable to specify filter criteria.

## A. Review considerations for working with Date Values in your Student Guide. In particular, notice that SOQL supports the use of date literals, such as LAST\_N\_DAYS in filter criteria for date values. You can find out more about date literals in the Apex documentation online.

## B. Click Setup gear icon | Developer Console.

## C. Press CTRL + E to open the **Execute Anonymous** window.

## D. Copy and paste the contents of Task 1 in **SOQLDynamicQueryInApex.txt** from the Exercises folder into the window, overwriting all existing text.

## E. Ensure that Open Log is selected and click Execute. Note that this causes an error because the bind variable is not supported with LAST\_N\_DAYS in a bracketed SOQL query.

## F. Click OK.

# 2. Use Database.query() to repair the code from Task 1.

## A. Copy and paste the contents of Task 2 in **SOQLDynamicQueryInApex.txt** from the Exercises folder into the window, overwriting all existing text.

## B. Complete the TODOs.

## C. Ensure that Open Log is selected and click Execute.

## D. Select **Debug Only** checkbox, then review the Debug Log to confirm the results.

## E. Press CTRL + ALT + / to close all console tabs.

# 3. (Optional) Use a bound variable to query by record type.

## A. Press CTRL + E to open the **Execute Anonymous** window.

## B. Copy and paste the contents of Task 3 in **SOQLDynamicQueryInApex.txt** from the Exercises folder into the window, overwriting all existing text.

## C. Complete the TODOs.

## D. Ensure that Open Log is selected and click Execute.

## E. Select **Debug Only** checkbox, then review the Debug Log to confirm the results.

## F. Press CTRL + ALT + / to close all console tabs.

## G. Close the Developer Console window.

6-1: Write and Test Child-to-Parent Relationship Queries

Goal:

Write child-to-parent relationship queries that explore relationships among sObjects in the Certification application.

Tasks:

# 1. Select all Contacts and their related Accounts.

# 2. Select Courses that have related Certifications.

Time:

20 minutes

Instructions:

# 1. Select all Contacts and their related Accounts.

## A. Click Setup gear icon | Developer Console.

## B. Open the Account and Contact objects in the Object Inspector for reference.

### i. Click **File | Open**.

### ii. Select **Objects** as the Entity Type, type Account into the filter input box, then double-click the entity **Account**.

### iii. Click **File | Open**.

### iv. Select **Objects** as the Entity Type, type Contact into the filter input box, then double-click the entity **Contact**.

## C. Use the Query Editor to write a query that selects all Contacts and their related Accounts.

### i. Select the Query Editor tab in bottom panel, and erase any existing text.

### ii. Write a query that selects all Contacts. You should select the following fields: the Name field from Contact and the related Name field from Account. The FROM clause of the query should be: FROM Contact.

## D. Run the query.

### i. Ensure that the Use Tooling API checkbox is not selected.

### ii. Click Execute. As a result, a Query Results tab opens.

### iii. Close the Query Results tab.

## E. Move the query into Apex, using a SOQL for Loop.

### i. Press CTRL + E to open the **Execute Anonymous** window.

### ii. Write Apex code that uses a SOQL for loop, which uses an iterator whose data type is List<Contact>, to print to the debug log the names of each Contact and related Account. Use the same SOQL query that you created in the previous step. HINT: If you have trouble remembering how to write SOQL for loops, consult module 5 in your Student Guide.

#### An example of what you need to print to the log follows:

#### Erica Neumann is related to the following Account: IntronCorporation

### iii. Ensure that the Open Log checkbox is selected and click Execute.

### iv. Select **Debug Only** checkbox, then review the Debug Log to confirm the results.

### v. Press CTRL + ALT + / to close all console tabs.

# 2. Select Courses that have related Certifications.

## A. View the API names of fields in sObjects.

### i. Click **File | Open**.

### ii. Select **Objects** as the Entity Type, type Course into the filter input box, then double-click the entity **Course\_\_c**.

### iii. Click **File | Open**.

### iv. Select **Objects** as the Entity Type, type Certification into the filter input box, then double-click the entity **Certification\_\_c**.

## B. Use the Query Editor to write and test a query that selects all Courses and their related Certifications.

### i. Select the Query Editor tab in bottom panel, and erase any existing text.

### ii. Write a query that selects only Courses with related Certifications. You should select the following fields: the Name from Course and the Name field from Certification. The FROM clause of the query should be: FROM Course\_\_c.

## C. Run the query.

### i. Ensure that the Use Tooling API checkbox is not selected.

### ii. Click Execute. As a result, a Query Results tab opens.

### iii. Press CTRL + ALT + / to close all console tabs.

### iv. Close the Developer Console window.

6-2: Query Accounts and Related Contacts

Goal:

Write parent-to-child relationship queries that explore relationships among sObjects in the Certification application.

Tasks:

# 1. Select all Accounts and their related Contacts.

# 2. Select all Certifications that have a related Course.

Time:

10 minutes

Instructions:

# 1. Select all Accounts and their related Contacts.

## A. Click Setup gear icon | Developer Console.

## B. Use Query Editor to write a query that selects all Accounts and their related Contacts.

### i. Select the Query Editor tab in bottom panel, and erase any existing text.

### ii. Write a query that selects all Accounts and displays their names, along with the last names of any related Contacts. The FROM clause of the query should be: FROM Account.

## C. Run the query.

### i. Ensure that the Use Tooling API checkbox is not selected.

### ii. Click Execute. As a result, a Query Results tab opens.

### iii. Close the Query Results tab.

# 2. Select all Certifications that have a related Course.

## A. Use Query Editor to write a query that selects all Certifications that have a related Course.

### i. Select the Query Editor tab in bottom panel, and erase any existing text.

## ii. Write a query that selects only Certifications with an associated Course. Be sure to also select the Name of the associated Course. The FROM clause of the query should be: FROM Certification\_\_c. Hint: You will need to use two nested queries.

## B. Run the query.

### i. Ensure that the Use Tooling API checkbox is not selected.

### ii. Click Execute. As a result, a Query Results tab opens.

### iii. Press CTRL + ALT + / to close all console tabs.

### iv. Close the Developer Console window.

7-1: Execute DML Commands

Goal:

Create and save Contacts.

Tasks:

# 1. Write Apex code to insert Contacts using a standalone insert statement.

# 2. Write Apex code to insert Contacts using a Database class method.

# 3. Test your code.

Time:

25 minutes

Instructions:

# 1. Write Apex code to insert Contacts using a standalone insert statement.

## A. Click Setup gear icon | Developer Console.

## B. Click File | New | Apex Class.

### C. Type ContactsDML for the name, then click **OK**.

### D. Copy and paste the contents of **ContactsDML.txt** from the Exercises folder into the window, overwriting all existing text.

### E. Complete the TODOs for Task 1, the standaloneDML method.

### F. Press CTRL + S to save the file.

# 2. Write Apex code to insert Contacts using a Database class method.

## A. Complete the TODOs for Task 2, the databaseMethodDML method.

## B. Complete the TODOs for Task 3, the databaseMethodDMLAllOrNone method.

## C. Press CTRL + S to save the file.

# 3. Test your code.

## A. Click File | New | Apex Class.

## B. Type ContactsDML\_Test for the name, then click **OK**.

## C. Copy and paste the contents of **ContactsDML\_Test.txt** from the Exercises folder into the window, overwriting all existing text.

## D. Complete the TODOs.

## E. Press CTRL + S to save the file.

### F. Click **Run Test**.

### G. Click the Tests tab in the bottom panel.

### H. Expand the collapsed folder by clicking on +.

### I. Ensure that all your methods executed successfully. A green tick indicates success.

### J. Press CTRL + ALT + / to close all console tabs.

### K. Close the Developer Console window.

7-2: Handle DML Errors and Exceptions

Goal:

Handle errors when inserting Contacts.

Tasks:

# 1. Print the list of reasons why Contacts could not be inserted into the database.

# 2. Test your code.

Time:

15 minutes

Instructions:

# 1. Print the list of reasons why Contacts could not be inserted into the database

## A. Click Setup gear icon | Developer Console.

### B. Click **File | Open**.

### C. Select **Classes** as the Entity Type, type Contacts into the filter input box, then double-click the entity **ContactsDML**.

### D. Copy and paste the contents of **ContactsDMLException.txt** from the Exercises folder into the window, overwriting all existing text.

## E. Complete the TODOs for the exceptionsDML method.

## F. Press CTRL + S to save the file.

# 2. Test your code.

### A. Click **File | Open**.

### B. Select **Classes** as the Entity Type, type Contacts into the filter input box, then double-click the entity **ContactsDML\_Test**.

### C. Copy and paste the contents of **ContactsDMLExceptionTest.txt** from the Exercises folder into the window, overwriting all existing text.

## D. Complete the TODOs for the exceptionsDMLTest method.

## E. Press CTRL + S to save the file.

# F. Click **Run Test**.

### G. Click the Tests tab in the bottom panel.

### H. Expand the collapsed folder by clicking on +.

### I. Ensure that all your methods executed successfully. A green tick indicates success.

### J. Press CTRL + ALT + / to close all console tabs.

### K. Close the Developer Console window.

8-1: Define a Trigger

Goal:

Define a trigger on the Course\_Delivery\_\_c sObject.

Task:

# 1. Define a trigger.

Time:

5 minutes

Instructions:

# 1. Define a trigger.

## A. Click Setup gear icon | Developer Console.

# B. Click **File | New | Apex Trigger**.

# C. Type CourseDeliveryTrigger for the name, select **Course\_Delivery\_\_c** as the sObject, then click **Submit**.

## D. Modify the trigger definition so that it runs before updates as well.

## E. Press CTRL + S to save the file.

### F. Press CTRL + ALT + / to close all console tabs.

### G. Close the Developer Console window.

8-2: Define the Trigger's Business Logic

Goal:

Define the business logic of a trigger that only allows a Course Delivery to be saved if is not scheduled to start on a holiday.

Tasks:

# 1. Create a Holiday.

# 2. Update a trigger.

# 3. Test the trigger’s logic.

Time:

15 minutes

Instructions:

# 1. Create a Holiday.

## A. Click Setup gear icon | Setup.

## B. Type holidays in the Quick Find textbox.

## C. Click Holidays under Company Settings.

## D. Click **New**.

## E. Enter the following information:

|  |  |
| --- | --- |
| Holiday Name | New Year's Eve |
| Date | **December 31st of the current year** |
| Time | **All Day** |
| Recurring Holiday | **Deselected** |

# F. Click **Save**.

# 2. Update a trigger.

## A. Click Setup gear icon | Developer Console.

## B. Click **File | Open**.

## C. Select **Triggers** as the Entity Type, type Course into the filter input box, then double-click the entity **CourseDeliveryTrigger**.

### D. Copy and paste the contents of **CourseDeliveryTrigger.txt** from the Exercises folder into the window, overwriting all existing text.

# E. Complete the TODOs.

## F. Press CTRL + S to save the file.

### G. Press CTRL + ALT + / to close all console tabs.

### H. Close the Developer Console window.

# 3. Test the trigger’s logic.

## A. In Salesforce, create and save a new Course Delivery with a start date of December 31 of the current year. Because its start date falls on a holiday, you should not be able to save the Course Delivery.

## B. In Salesforce, create and save a new Course Delivery with a start date of January 1 of the upcoming year. Because its start date does not fall on a holiday, you should be able to save the Course Delivery. Verify that the Course Delivery is saved (you can do this programmatically using SOQL or declaratively, by finding the Course Delivery in the UI).

## C. In Salesforce, find the Course Delivery you created in step 3.B. Update it so that its start date is on December 31 of the current year. You should not be able to update this Course Delivery’s start date to this day because it is a holiday.

9-1: Define an Apex Class

Goal:

Make a trigger easy to read and maintain by creating a helper class for it.

Tasks:

# 1. Create an Apex class.

# 2. Invoke the class from the trigger.

# 3. Test the trigger.

Time:

15 minutes

Instructions:

# 1. Create an Apex class.

## A. Click Setup gear icon | Developer Console.

## B. Click File | New | Apex Class.

### C. Type CourseDeliveryTriggerHandler for the name, then click **OK**.

### D. Copy and paste the contents of **CourseDeliveryTriggerHandler.txt** from the Exercises folder into the window, overwriting all existing text.

### E. Complete the TODOs.

### F. Press CTRL + S to save the file.

# 2. Invoke the class from the trigger.

### A. Click **File | Open**.

### B. Select **Triggers** as the Entity Type, type Course into the filter input box, then double-click the entity **CourseDeliveryTrigger**.

### C. Copy and paste the contents of **CourseDeliveryTriggerRefactored.txt** from the Exercises folder into the window, overwriting all existing text.

### D. Complete the TODOs.

### E. Press CTRL + S to save the file.

# 3. Test your code.

## A. Click File | New | Apex Class.

### B. Type CourseDeliveryTriggerRefactored\_Test for the name, then click **OK**.

### C. Copy and paste the contents of **CourseDeliveryTriggerRefactored\_Test.txt** from the Exercises folder into the window, overwriting all existing text.

## D. Press CTRL + S to save the file.

# E. Click **Run Test**.

### F. Click the Tests tab in the bottom panel.

### G. Expand the collapsed folder by clicking on +.

### H. Ensure that all your methods executed successfully. A green tick indicates success.

### I. Press CTRL + ALT + / to close all console tabs.

### J. Close the Developer Console window.

10-1: Explore the Implicit Firing of Triggers

Goal:

Determine the actions that occur when a Course record is saved.

Tasks:

# 1. Open the Developer Console.

# 2. Update the status of a Course record to “Retired” in the UI.

# 3. Review the results in the Developer Console.

Time:

10 minutes

Instructions:

Note: If you do not have a working copy of CourseDeliveryTrigger, then create or replace it in your org with the version **8.2-CourseDeliveryTrigger.txt** from the Solutions folder.

# 1. Open the Developer Console.

## A. Click Setup gear icon | Developer Console.

## B. Click **Debug**, then deselect Show My Current Logs Only.

# 2. Update the status of a Course record to Retired in the UI.

## A. Leaving the Developer Console open, switch to the Salesforce UI in another window.

## B. From the Certification app, click the Courses tab.

## C. Click Recently Viewed, then click All list view.

## D. Click record named **[401] Data Recovery**, then click **Edit**.

## E. Change the Status field to Retired.

## F. Click Save.

# 3. Examine the results in the Developer Console.

## A. Switch back to the Developer Console window.

## B. Click the Logs tab in the bottom panel.

## C. Double-click the most recent log entry.

## D. Click Debug | Switch Perspective | All (Predefined).

## E. Enter START in the Filter input box of the Execution Log panel (case-sensitive).

## F. Notice that AfterUpdate event was fired multiple times for CourseTrigger, followed by a BeforeUpdate event for CourseDeliveryTrigger, followed by the execution of a workflow rule.

## G. Looking at the **Save Order** tab in the Execution Overview panel also shows the triggers firing, as well as a workflow rule.

## H. Click the Executed Units tab in the Execution Overview panel and click the entry for the checkStatus method.

## I. Notice the various operations called during the execution of this method in the Execution Log panel.

## J. Press CTRL + ALT + / to close all console tabs.

## K. Close Developer Console window.

10-2: View the Events that Occur During a Rollback

Goal:

Determine the events that occur when an update action is rolled back.

Tasks:

# 1. Open the Developer Console.

# 2. Update the status of a Course record to “Retired” in the UI.

# 3. Review the results in the Developer Console and the UI.

Time:

10 minutes

Instructions:

# 1. Open the Developer Console.

## A. Click Setup gear icon | Developer Console.

## B. Click Debug, then deselect Show My Current Logs Only.

### C. Click Change Log Levels.

### D. Click **Add/Change** under DebugLevel Action.

### E. Set the DB, Apex Code and Profiling entries to Finest and System to Fine.

### F. Click **Done.**

### G. Click **Done**.

# 2. Update the status of a Course record to Retired in the UI.

## A. Leaving the Developer Console open, switch to the Salesforce UI in another window.

## B. From the Certification app, click the Courses tab.

## C. Click Recently Viewed, then click All list view.

## D. Click record named **[102] AWCA Network**, then click **Edit**.

## E. Change the Status field to Retired.

## F. Click Save. It will fail and display error, "Course has enrolled students".

# 3. Examine the results in the Developer Console and the UI.

## A. Switch back to the Developer Console window.

## B. Click the Logs tab in the bottom panel.

## C. Double-click the most recent log entry.

## D. Click Debug | Switch Persepctive | All (Predefined).

## E. Looking at the **Save Order** tab in the Execution Overview panel we see the same triggers and workflow that fired in the last exercise are firing again (although more often).

## F. Enter Retired in the Filter input box of the Execution Log panel (case-sensitive).

## G. Double-click on the first entry whose Details column lists triggerNew.

### i. In the popup window, validate that the value of Status\_\_c field is Retired.

### ii. Click OK. Note that this value was never committed to the database.

## H. Click the Executed Units tab in the Execution Overview panel and click the entry for the addError method. The entry in the log confirms that the addError method was executed.

## I. Press CTRL + ALT + / to close all console tabs.

## J. Close the Developer Console window.

## K. Click Cancel.

## L. Refresh the page.

## M. Notice that the Status of each record in the Course Deliveries related list has not changed and the Course’ Status is still Active, showing the save was rolled back.

10-3: See the Save Order of Execution in Action

Goal:

When saving a record, see and discuss the events that occur.

Tasks:

# 1. Open the Developer Console.

# 2. Create a Course Attendee record in the UI.

# 3. Using the log file generated by the interaction with the UI, answer the questions.

# 4. Run the test for the CourseAttendeeTrigger.

# 5. Using the log file generated by running the test class, answer the questions.

Time:

10 minutes

Instructions:

# 1. Open the Developer Console.

## A. Click Setup gear icon | Developer Console.

## B. Click Debug, then deselect Show My Current Logs Only.

### C. Click Change Log Levels.

### D. Click **Add/Change** under DebugLevel Action.

### E. Set the Apex Code entry to **Debug** and Profiling to Fine.

### F. Click **Done.**

### G. Click **Done**.

# 2. Create a Course Attendee record in the UI.

## A. Leaving the Developer Console open, switch to the Salesforce UI in another window.

## B. From the Certification app, click the Course Deliveries tab.

## C. Click Recently Viewed, then click All list view.

## D. Click the record named **DELIVERY-00029**.

## E. In the Course Attendees related list, click **New**.

## F. Enter the following information:

|  |  |
| --- | --- |
| Student | Alexa Delany |
| Status | **Enrolled** |
| Course Delivery  (should be pre-populated) | DELIVERY-00029 |

## G. Click Save.

# 3. Using the log file generated by the interaction with the UI, answer the questions.

## A. Switch back to the Developer Console window.

## B. Click the Logs tab in the bottom panel.

## C. Double-click the most recent log entry.

## D. Using the logs and your understanding of Triggers, answer the following questions:

### i. What DML operation occurred? (Hint: Enter DML in the Filter input box of the Execution Log panel)

### ii. Which event(s) does the Trigger CourseAttendeeTrigger support (before/after, insert/update/delete)? (Hint: Look at the actual code of the trigger)

# 4. Run the test for the CourseAttendeeTrigger trigger.

### A. Click **File | Open**.

### B. Select **Classes** as the Entity Type, type Course into the filter input box, then double-click the entity **CourseAttendeeTrigger\_Test**.

## C. Click Run Test.

### D. Click the Tests tab in the bottom panel.

### E. Expand the collapsed folder by clicking on +.

### F. Ensure that all your methods executed successfully. A green tick indicates success.

# 5. Using the log file generated by running the test class, answer the questions.

## A. Click the Logs tab in the bottom panel.

## B. Double-click the most recent log file to open it.

## C. Using the logs and your understanding of the Save Order of Execution and Triggers, answer these questions:

### i. How many times does the CourseAttendeeTrigger fire?

### ii. What is causing the CourseAttendeeTrigger to fire multiple times?

### iii. How often is the workflow firing? What determines the number of times it fires?

### D. Press CTRL + ALT + / to close all console tabs.

### E. Close the Developer Console window.

11-1: Make Test Data Available to Test Methods

Goal:

Create test data to test the Certification application.

Tasks:

# 1. Create test data using a Static Resource.

# 2. Create test data programmatically.

Time:

10 minutes

Instructions:

# 1. Create test data using a Static Resource.

## A. Click Setup gear icon | Developer Console.

## B. Click File | New | Static Resource.

### C. Enter the following information:

|  |  |
| --- | --- |
| Name | Test\_Holidays |
| MIME Type | **text/plain** |

## D. Click **Submit**.

### E. Copy and paste the contents of **HolidaysForStaticResource.txt** from the Exercises folder into the window, overwriting all existing text.

### F. Press CTRL + S to save the file.

### G. Press CTRL + ALT + / to close all console tabs.

# 2. Create test data programmatically.

## A. Click File | New | Apex Class.

### B. Type CourseDeliveryTriggerHandler\_Test2 for the name, then click **OK**.

### C. Copy and paste the contents of **CourseDeliveryTriggerHandler\_Test2.txt** from the Exercises folder into the window, overwriting all existing text.

### D. Complete the TODOs.

## E. Press CTRL + S to save the file.

# F. Click **Run Test**.

### G. Click the Tests tab in the bottom panel.

### H. Expand the collapsed folder by clicking on +.

### I. Ensure that all your methods executed successfully. A green tick indicates success.

### J. Press CTRL + ALT + / to close all console tabs.

### K. Close the Developer Console window.

11-2: Write and Run an Apex Test

Goal:

Determine if the Certification application properly prevents a course delivery from being scheduled on a holiday.

Tasks:

# 1. Write a test that tests the business logic of a trigger and a class.

Time:

10 minutes

Instructions:

# 1. Write a test that tests the business logic of a trigger and a class.

## A. Click Setup gear icon | Developer Console.

## B. Click **File | Open**.

## C. Select **Classes** as the Entity Type, type CourseDelivery into the filter input box, then double-click the entity **CourseDeliveryTriggerHandler\_Test2**.

### D. Copy and paste the contents of **CourseDeliveryTriggerHandler\_Test2\_TestMethods.txt** from the Exercises folder into the window, overwriting all existing text.

# E. Complete the TODOs.

## F. Press CTRL + S to save the file.

# G. Click **Run Test**.

### H. Click the Tests tab in the bottom panel.

### I. Expand the collapsed folder by clicking on +.

### J. Ensure that all your methods executed successfully. A green tick indicates success.

### K. Press CTRL + ALT + / to close all console tabs.

### L. Close the Developer Console window.

12-1: Explore Code Coverage

Goal:

Explore code coverage using the Developer Console.

Tasks:

# 1. Prepare the lab.

# 2. Run the test class.

# 3. View code coverage.

Time:

10 minutes

Instructions:

# 1. Prepare the lab.

## A. This lab requires the **CourseDeliveryTrigger**. If you haven’t already completed the exercises in the “Trigger Essentials” module, please do so now.

## B. This lab requires the static resource Test\_Holidays and the test class **CourseDeliveryTriggerHandler\_Test2**. If you haven’t already completed the exercises in the “Testing Essentials” module, please do so now.

# 2. Run the test class.

## A. Click Setup gear icon | Developer Console.

## B. Click Test | Clear Test Data to clear any existing test statistics.

## C. Click Test | New Run.

## D. Select CourseDeliveryTriggerHandler\_Test2, then click **Add Selected**.

## E. Click Run.

# 3. View code coverage.

## A. Click **File | Open**.

## B. Select **Classes** as the Entity Type, type Course into the filter input box, then double-click the entity **CourseDeliveryTriggerHandler**.

## C. Select the down-arrow next to **Code Coverage**, then click **CourseDeliveryTriggerHandler\_Test2.insertCourseDeliverySuccess**. This should cause the red and blue coverage highlighting to display.

## D. Notice that the code that was not covered by the test class (i.e., highlighted in red) is related to ‘bad data’ – i.e. a Course Deliveries that are scheduled on a Holiday.

## E. Select the down-arrow next to **Code Coverage**, then click **Code Coverage: None**. This should remove the red and blue coverage highlighting.

### F. Press CTRL + ALT + / to close all console tabs.

### G. Close the Developer Console window.

13-1: Refactor a Trigger to Avoid SOQL Limits

Goal:

Refactor the code for the Course Trigger to avoid the “Too many SOQL queries” error when running a test with several records.

Tasks:

# 1. Examine the current Course trigger.

# 2. Test the trigger with a single record in the Salesforce user interface.

# 3. Test the trigger with several records using unit test code.

# 4. Refactor the code to avoid a SOQL error.

Time:

15 minutes

Instructions:

# 1. Examine the current Course trigger.

## A. Click Setup gear icon | Developer Console.

## B. Click **File | Open**.

## C. Select **Classes** as the Entity Type, type CourseTrigger into the filter input box, then double-click the entity **CourseTriggerHandler**.

## D. Examine the code. What would cause it to receive a “Too many SOQL queries” error if run with multiple records?

# 2. Test the trigger with a single record in the Salesforce user interface.

## A. Leaving the Developer Console open, switch to the Salesforce UI in another window.

## B. From the Certification app, click the **Courses** tab.

## C. Click Recently Viewed, then click All list view.

## D. Click the record named **[101] AWCA Server**.

## E. Click **Edit**.

## F. Change the status to Retired, then click Save.

## G. If the Course does not have any enrolled Attendees, then it should allow you to change the status. If it does have enrolled attendees, then you will get a message telling you that the “Course has enrolled students.” Either outcome indicates that the trigger was able to complete execution.

## H. Click **Cancel**.

# 3. Test the trigger with several records using unit test code.

## A. Switch back to the Developer Console window.

## B. Click File | Open.

## C. Select **Classes** as the Entity Type, type Course into the filter input box, then double-click the entity CourseTrigger\_Test.

## D. Click Run Test.

### E. Click the Tests tab in the bottom panel. You should see it fail with a red x, and once you do, double-click the entry to bring up the results. Notice what error appears in the Errors column.

# 4. Refactor the code to avoid a SOQL error.

## A. Click the **CourseTriggerHandler** tab.

## B. Copy and paste the contents of **CourseTriggerHandler.txt** from the Exercises folder into the window, overwriting all existing text.

### C. Complete the TODOs. The class might not compile and save until you have done so.

## D. Press CTRL + S to save the file.

## E. Click the CourseTrigger\_Test tab.

## F. Click Run Test.

### G. Click the Tests tab in the bottom panel. You should see it fail again with a red x, and once you do, double-click the entry to bring up the results. Notice that the SOQL limit error is resolved but now a DML limit exceeded error appears in the Errors column. In the next exercise you will refactor the trigger to resolve this new error.

### H. Press CTRL + ALT + / to close all console tabs.

### I. Close the Developer Console window.

13-2: Refactor a Trigger to Avoid DML Limits

Goal:

Refactor the code for the Course Trigger to avoid the “Too many DML rows” error when running a test with several records.

Tasks:

# 1. Examine the current Course trigger.

# 2. Refactor the code to avoid a DML error.

# 3. Re-test the trigger with several records using unit test code.

Time:

15 minutes

Instructions:

# 1. Examine the current Course trigger.

## A. Click Setup gear icon | Developer Console.

### B. Click **File | Open**.

## C. Select **Classes** as the Entity Type, type CourseTrigger into the filter input box, then double-click the entity CourseTriggerHandler.

## D. Examine the code. What would cause it to receive an error if run with multiple records?

# 2. Refactor the code to avoid a DML error.

## A. Copy and paste the contents of **CourseTriggerHandler.txt** from the Exercises folder into the window, overwriting all existing text.

## B. Complete the TODOs.

## C. Press CTRL + S to save the file.

# 3. Re-test the trigger with several records using unit test code.

### A. Click **File | Open**.

## B. Select **Classes** as the Entity Type, type CourseTrigger into the filter input box, then double-click the entity CourseTrigger\_Test.

## C. Click Run Test.

## D. Click the Tests tab in the bottom panel. You should now see it pass.

### E. Press CTRL + ALT + / to close all console tabs.

### F. Close the Developer Console window.

14-1: Create a Formula Field to Eliminate a Query

Goal:

Create a formula field on Certification Attempt object of its related Certification Id.

Task:

# 1. Create a hidden formula field on the Certification Attempt object.

Time:

5 minutes

Instructions:

# 1. Create a hidden formula field on the Certification Attempt object.

## A. Click Setup gear icon | Setup.

## B. Click **Object Manager**.

## C. Click Certification Attempt.

## D. Click **Fields & Relationships**, then click **New**.

## E. Select Formula as the data type, then click **Next**.

## F. Enter the following information:

|  |  |
| --- | --- |
| Field Label | Certification Id |
| Field Name | Certification\_Id |
| Formula Return Type | Text |

## G. Click **Next**.

## H. Click **Advanced Formula** tab, then enter the following formula:

## CASESAFEID( Certification\_Element\_\_r.Certification\_\_c )

## I. Click **Check Syntax**.

## J. Type The 18-character record ID in the Description field.

## K. Click **Next**.

## L. Leave the field-level security as-is and click Next.

## M. Deselect **Add Field** so no page layout is selected, then click **Save**.

14-2: Create Fields for Counting Certification Elements

Goal:

Implement fields that will make the number of Certification Elements associated with a Certification available on the Certification Attempt Element.

Tasks:

# 1. Create a roll-up summary field on the Certification object.

# 2. Create a hidden formula field on the Certification Attempt object.

Time:

5 minutes

Instructions:

# 1. Create a roll-up summary field on the Certification object.

## A. Click Setup gear icon | Setup.

## B. Click **Object Manager**.

## C. Click Certification.

## D. Click **Fields & Relationships**, then click **New**.

## E. Select Roll-Up Summary as the data type, then click **Next**.

## F. Enter the following information:

|  |  |
| --- | --- |
| Field Label | Number of Certification Elements |
| Field Name | Number\_of\_Certification\_Elements |

## G. Click **Next**.

## H. Enter the following information:

|  |  |
| --- | --- |
| Summarized Object | **Certification Elements** |
| Roll-Up Type | **COUNT** |

## Click **Next**.

## J. Leave the field-level security as-is and click Next.

## K. Choose to automatically add the field to the page layout and click Save.

# 2. Create a hidden formula field on the Certification Attempt object.

## A. Click Setup gear icon | Setup.

## B. Click **Object Manager**.

## C. Click Certification Attempt.

## D. Click **Fields & Relationships**, then click **New**.

## E. Select Formula as the data type, then click **Next**.

## F. Enter the following information:

|  |  |
| --- | --- |
| Field Label | Number of Elements for Cert |
| Field Name | Number\_of\_Elements\_for\_Cert |
| Formula Return Type | Number |
| Decimal Places | 0 |

## G. Click **Next**.

## H. Click **Advanced Formula** tab, then enter the following formula:

## Certification\_Element\_\_r.Certification\_\_r.Number\_of\_Certification\_Elements\_\_c

## I. Click **Check Syntax**.

## J. Type The number of Certification Elements in the Description field.

## K. Click **Next**.

## L. Leave the field-level security as-is and click Next.

## M. Deselect **Add Field** so no page layout is selected, then click **Save**.

14-3: Create Collections to Filter the Query

Goal:

Create the query that will bring back Certification Attempt records.

Tasks:

# 1. Write code to create a query filter then loop over the SOQL results.

# 2. Change code to call the logic for the new class handler method.

# 3. Test your new trigger logic.

Time:

20 minutes

Instructions:

# 1. Write code to create a query filter then loop over the SOQL results.

## A. Click Setup gear icon | Developer Console.

### B. Click File | New | Apex Class.

### C. Type CertificationAttemptTriggerHandler for the name, then click **OK**.

### D. Copy and paste the contents of **CertificationAttemptTriggerHandler.txt** from the Exercises folder into the window, overwriting all existing text.

### E. Complete the TODOs.

### F. Press CTRL + S to save the file.

# 2. Change code to call the logic for the new class handler method.

# A. Click **File | New | Apex Trigger**.

# B. Type CertificationAttemptTrigger for the name, select **Certification\_Attempt\_\_c** as the sObject, then click **Submit**.

### C. Copy and paste the contents of **CertificationAttemptTrigger.txt** from the Exercises folder into the window, overwriting all existing text.

### D. Press CTRL + S to save the file.

# 3. Test your new trigger logic.

## A. Leaving the Developer Console open, switch to the Salesforce UI in another window.

# B. From the Certification app, click the **Certifications** tab.

## C. Click Recently Viewed, then click All list view.

## D. Click record named AWCA Network.

## E. In the Certification Elements related list, click the record named AWCA Network Multiple Choice.

## F. In the Certification Attempts related list, click New.

## G. Click the **Multiple Choice** record type, then click **Next**.

## H. Enter the following information:

|  |  |
| --- | --- |
| Certification Candidate | Clara Petit |
| Status | **Complete/Pass** |

## I. Click Save.

## J. Switch back to the Developer Console window.

## K. Click the **Logs** tab in the bottom panel.

## L. Double-click the most recent log entry.

## M. Click **Debug Only**. You should see several entries appear. Look for an entry that indicates that the createCertificationHeld method was called, followed by an entry that shows what Certification Attempt record was retrieved. This indicates that the new trigger logic was fired successfully.

### N. Press CTRL + ALT + / to close all console tabs.

### O. Close the Developer Console window.

14-4: Use a Map to Aggregate Results

Goal:

Aggregate the results of the query for use in determining if a candidate has passed all elements of a certification.

Tasks:

# 1. Construct and use a Map to aggregate query results.

# 2. Test your new trigger logic.

Time:

20 minutes

Instructions:

# 1. Construct and use a Map to aggregate query results.

## A. Click Setup gear icon | Developer Console.

### B. Click **File | Open**.

### C. Select **Classes** as the Entity Type, type Certification into the filter input box, then double-click the entity **CertificationAttemptTriggerHandler**.

### D. Copy and paste the contents of **CertificationAttemptTriggerHandler.txt** from the Exercises folder into the window, overwriting all existing text.

## E. Complete the TODOs.

## F. Press CTRL + S to save the file.

# 2. Test your new trigger logic.

## A. Leaving the Developer Console open, switch to the Salesforce UI in another window.

# B. From the Certification app, click the **Certifications** tab.

## C. Click Recently Viewed, then click All list view.

## D. Click record named AWCA Network.

## E. In the Certification Elements related list, click the record named AWCA Network Multiple Choice.

## F. In the Certification Attempts related list, click New.

## G. Click the **Multiple Choice** record type, then click **Next**.

## H. Enter the following information:

|  |  |
| --- | --- |
| Certification Candidate | Arthur Franz |
| Status | **Complete/Pass** |

## I. Click Save.

## J. Switch back to the Developer Console window.

## K. Click the **Logs** tab in the bottom panel.

## L. Double-click the most recent log entry.

## M. Click **Debug Only**. You should see several entries appear. Look for an entry that indicates that the createCertificationHeld method was called, followed by an entry that shows what Certification Attempt record was retrieved, and finally one that shows the size of passCounts equals one. This indicates that the new trigger logic was fired successfully.

### N. Press CTRL + ALT + / to close all console tabs.

### O. Close the Developer Console window.

14-5: Create Certification Held Records

Goal:

Complete the solution to create Certification Held records for qualified candidates.

Tasks:

# 1. Create Certification Held records for qualified candidates.

# 2. Test your solution.

Time:

15 minutes

Instructions:

# 1. Create Certification Held records for qualified candidates.

## A. Click Setup gear icon | Developer Console.

### B. Click **File | Open**.

### C. Select **Classes** as the Entity Type, type Certification into the filter input box, then double-click the entity **CertificationAttemptTriggerHandler**.

### D. Copy and paste the contents of **CertificationAttemptTriggerHandler.txt** from the Exercises folder into the window, overwriting all existing text.

## E. Complete the TODOs.

## F. Press CTRL + S to save the file.

# 2. Test your solution.

## A. Leaving the Developer Console open, switch to the Salesforce UI in another window.

# B. From the Certification app, click the **Certifications** tab.

## C. Click Recently Viewed, then click All list view.

## D. Click record named AWCA Network.

## E. In the Certification Elements related list, click the record named AWCA Network Multiple Choice.

## F. In the Certification Attempts related list, click New.

## G. Click the **Multiple Choice** record type, then click **Next**.

## H. Enter the following information:

|  |  |
| --- | --- |
| Certification Candidate | Clara Morales |
| Status | **Complete/Pass** |

## I. Click Save.

## J. Switch back to the Developer Console window.

## K. Click the **Logs** tab in the bottom panel.

## L. Double-click the most recent log entry.

## M. Click **Debug Only**. You should see several entries appear. Look for an entry that indicates that the createCertificationHeld method was called, followed by an entry that shows what Certification Attempt record was retrieved, one that shows the size of passCounts equals one and finally one that says a certification held record was added. This indicates that the new trigger logic was fired successfully.

### N. Press CTRL + ALT + / to close all console tabs.

### O. Close the Developer Console window.

14-6: Use a Workflow to Avoid Creation of Duplicate Records (Optional)

Goal:

Create a workflow rule that will prevent duplicate Certification Held records from being created.

Tasks:

# 1. Create a new hidden text field.

# 2. Create a workflow rule to populate the new field.

# 3. Execute the workflow rule for all existing records.

# 4. Test your workflow rule.

Time:

10 minutes

Instructions:

# 1. Create a new hidden text field.

## A. Click Setup gear icon | Setup.

## B. Click **Object Manager**.

## C. Click Certification Held.

## D. Click **Fields & Relationships**, then click **New**.

## E. Select Text as the data type, then click **Next**.

## F. Enter the following information:

|  |  |
| --- | --- |
| Field Label | Cert Held Dup Key |
| Field Name | Cert\_Held\_Dup\_Key |
| Length | 37 |
| Unique | Selected (case sensitive) |

## G. Click **Next**.

## H. Select **Read-Only** for all profiles, then click Next.

## I. Deselect **Add Field** so no page layout is selected, then click **Save**.

# 2. Create a workflow rule to populate the new field.

## A. Click **Home**.

## B. Type workflow in the Quick Find textbox.

## C. Click Workflow Rules under Process Automation.

## D. If you see “Understanding Worfklow” page, then click **Continue**.

## E. Click New Rule.

## F. Select **Certification Held** object, then click **Next**.

## G. Enter the following information:

|  |  |
| --- | --- |
| Rule Name | PopulateCertHeldDupKey |
| Evaluation Criteria | created, and every time it’s edited |
| Run this rule if the | formula evaluates to true |
| Formula | true |

## H. Click **Save & Next**.

## I. Click **Add Workflow Action | New Field Update**.

## J. Enter the following information:

|  |  |
| --- | --- |
| Name | Cert Held Dup Key |
| Unique Name | Cert\_Held\_Dup\_Key |
| Field to Update | Certification Held – Cert Held Dup Key |
| Text Options | Use a formula to set the new value |

## Click **Show Formula Editor**.

### L. Click Insert Field.

### M. Select Certification Held > Certification, then click **Insert**.

### N. Click Insert Operator | (&) Concatenate.

### O. Click Insert Field.

### P. Select Certification Held > Certified Professional, then click **Insert**.

### Q. Note, the formula value should read as: Certification\_\_c & Certified\_Professional\_\_c

### R. Click Save.

## S. Click Done.

## T. Click Activate.

# 3. Execute the workflow rule for all existing records.

## A. Click Setup gear icon | Developer Console.

## B. Click Debug | Open Execute Anonymous Window.

## C. Copy the below code into the window, overwriting all existing text:

## update [ SELECT Id FROM Certification\_Held\_\_c ];

# D. Ensure that Open Log is selected and click Execute.

## E. Press CTRL + ALT + / to close all console tabs.

## F. Close Developer Console window.

# 4. Test your workflow rule.

## A. From the Certification app, click the **Contacts** tab.

## B. Click Recently Viewed, then click All list view.

## C. Click the record named **Clara Morales**.

## D. In the Certifications Held related list, verify that Clara Morales already holds the certification named AWCA Network, then click **New**.

## E. Enter the following information:

|  |  |
| --- | --- |
| Certification | AWCA Network |
| Date Achieved | **Today’s Date** |

## F. Click Save. You should see the error message, “duplicate value found.” This indicates that the new workflow rule was fired successfully.

## G. Click **Cancel**.

15-1: Create a Simple Visualforce Page

Goal:

Create a simple Visualforce page that displays your name.

Tasks:

# 1. Create a Visualforce page using the inline editor.

# 2. Add static text to the page.

# 3. Add a reference to the global user variable to display your name.

Time:

10 minutes

Instructions:

# 1. Create a Visualforce page using the inline editor.

## A. Replace everything in the URL address bar after **force.com** with **/apex/HelloPage**, then press ENTER. For example:

## https://<instance>.lightning.force.com**/apex/HelloPage**

A warning will be displayed that the page does not exist.

## B. Click Create Page HelloPage.

# 2. Add static text to the page.

## A. Click HelloPage in the lower left corner of the screen.

## B. Replace lines 2-5 between <apex:page> and </apex:page> with the below code:

<b>Hello World</b>

## C. Press CTRL + S to save the file. Note that the web page content changes.

# 3. Add a reference to the global user variable to display your name.

## A. Replace line 3 with the below code:

<b>Hello {!$User.firstName}</b>

## B. Press CTRL + S to save the file. Note that the web page content changes.

# C. Remove everything in the URL address bar after **visual.force.com**, then press ENTER.

15-2: Display Data in a Visualforce Page

Goal:

Create a Visualforce Page that prints a simple Course Completion certificate.

Tasks:

# 1. Upload the pre-existing image file to be used as the certificate banner.

# 2. Create a new Visualforce page using Developer Console.

# 3. Create a custom button to launch the new Visualforce page.

# 4. Test the new page.

Time:

15 minutes

Instructions:

# 1. Upload the pre-existing image file to be used as the certificate banner.

## A. Click Setup gear icon | Setup.

## B. Type static in the Quick Find textbox.

## C. Click Static Resources under Custom Code.

## D. Click **New**.

### E. Enter the following information:

|  |  |
| --- | --- |
| Name | CourseCertificateImage |
| Choose File | Click then locate and select the **CourseCertificateImage.png** file in your Exercises folder. |
| Cache Control | **Public** |

## F. Click **Save**.

# 2. Create a new Visualforce page using the Developer Console.

## A. Click Setup gear icon | Developer Console.

## B. Click File | New | Visualforce Page.

### C. Enter the following information:

|  |  |
| --- | --- |
| Name | CourseCertificatePage |

## D. Click **OK**.

### E. Copy and paste the contents of **CourseCertificatePage.txt** from the Exercises folder into the window, overwriting all existing text.

### F. Complete the TODOs.

### G. Press CTRL + S to save the file.

### H. Press CTRL + ALT + / to close all console tabs.

### I. Close the Developer Console window.

# 3. Create a custom button to launch the new Visualforce page.

## A. Click **Object Manager**.

## B. Click Course.

## C. Click **Buttons, Links, and Actions**, then click **New Button or Link**.

## D. Enter the following information:

|  |  |
| --- | --- |
| Label | Course Certificate |
| Name | Course\_Certificate |
| Display Type | Detail Page Button |
| Behavior | Display in existing window without sidebar or header.  ([Known Issues](https://success.salesforce.com/issues_view?id=a1p3A000000jkrUQAQ)) |
| Content Source | Visualforce Page |
| Content | CourseCertificatePage |

## E. Click **Save**.

## F. Click **OK** to the warning about adding the button to the page layout.

## G. Click **Page Layouts**.

## H. Click **Course Layout**.

## I. From the palette, click **Mobile & Lightning Actions**.

## J. Drag and drop **Course Certificate** from the palette to the left of New Event in the **Salesforce Mobile and Lightning Experience Actions** page layout section.

## K. Click **Save**.

# 4. Test the new page.

## A. From the Certification app, click the **Courses** tab.

## B. Click Recently Viewed, then click All list view.

## C. Click the record named **[102] AWCA Network**.

## D. On the Course detail page, click the Course Certificate button.

## E. Verify that the certificate is rendered and that the course name is listed on the certificate.

## F. Click the browser back button.

16-1: Create a Simple Technician Status Page

Goal:

Create a simple technician status page to display technician name, account name, and courses they’ve registered for. Launch the page with a custom button.

Tasks:

# 1. Provide a record to be used as context during development.

# 2. Create a technician status page.

# 3. Create a custom button on the Technician Contact Layout to launch the page.

# 4. Test your new page.

Time:

15 minutes

Instructions:

# 1. Provide a record to be used as context during development.

## A. From the Certification app, click the Contacts tab.

## B. Click Recently Viewed, then click All list view.

## C. Click the record named **Arthur Franz**.

## D. Copy and paste the 18-character Contact record ID (starts with 003) from the URL into a text editor.

# 2. Create a Visualforce page using the inline editor.

## A. Replace everything in the URL address bar after **force.com** with **/apex/TechnicianStatusPage**, then press ENTER. Your new URL will look like:

## https://<instance>--c.visual.force.com**/apex/TechnicianStatusPage**

A warning will be displayed that the page does not exist.

## B. Click Create Page TechnicianStatusPage.

## C. Click TechnicianStatusPage in the lower left corner of the screen.

## D. Append to the end of the URL ?id=RecordId, replacing RecordId with the Contact record ID copied earlier, then press ENTER. Your new URL will look like:

## https://<instance>--c.visual.force.com/apex/TechnicianStatusPage**?id=003xxx**

### E. Copy and paste the contents of **SimpleTechnicianStatusPage.txt** from the Exercises folder into the window, overwriting all existing text.

### F. Complete the TODOs.

### G. Press CTRL + S to save the file. Note that the web page content changes.

### H. Remove everything in the URL address bar after **visual.force.com**, then press ENTER to return to the Salesforce UI.

# 3. Create a custom button on the Technician Contact Layout to launch the page.

## A. Click Setup gear icon | Setup.

## B. Click **Object Manager**.

## C. Click Contact.

## D. Click **Buttons, Links, and Actions**, then click **New Button or Link**.

## E. Enter the following information:

|  |  |
| --- | --- |
| Label | Technician Status |
| Name | Technician\_Status |
| Display Type | Detail Page Button |
| Behavior | Display in existing window without sidebar or header.  ([Known Issues](https://success.salesforce.com/issues_view?id=a1p3A000000jkrUQAQ)) |
| Content Source | Visualforce Page |
| Content | TechnicianStatusPage |

## F. Click **Save**.

## G. Click **OK** to the warning about adding the button to the page layout.

## H. Click **Page Layouts**.

## I. Click **Technician Contact Layout**.

## J. From the palette, click **Mobile & Lightning Actions**.

## K. Drag and drop **Technician Status** from the palette to the left of New Event in the **Salesforce Mobile and Lightning Experience Actions** page layout section.

## L. Click **Save**.

# 4. Test your new page.

## A. From the Certification app, click the **Contacts** tab.

## B. Click Recently Viewed, then click All list view.

## C. Click the record named **Arthur Franz**.

## D. On the Contact detail page, click Technician Status.

## E. Notice the Course Listing section at the bottom of the page, which displays the additional information provided by the code.

## F. Click the browser back button.

16-2: Refine Your Page and Add Navigational Links

Goal:

Refine your TechnicianStatusPage Visualforce page to display only the necessary fields, then add navigational links so users can easily edit related records.

Tasks:

# 1. Update the Visualforce page to only display the necessary fields and navigational links.

# 2. Test the new page.

Time:

10 minutes

Instructions:

# 1. Update the Visualforce page to only display the necessary fields and navigational links.

## A. Replace everything in the URL address bar after **force.com** with **/apex/TechnicianStatusPage**, then press ENTER. Your new URL will look like:

## https://<instance>--c.visual.force.com**/apex/TechnicianStatusPage**

## B. Append to the end of the URL ?id=RecordId, replacing RecordId with the Contact record ID copied earlier, then press ENTER. Your new URL will look like:

## https://<instance>--c.visual.force.com/apex/TechnicianStatusPage?id=003xxx

### C. Copy and paste the contents of **NavigationalLinksPage.txt** from the Exercises folder into the window, overwriting all existing text.

### D. Complete the TODOs.

### E. Press CTRL + S to save the file. Note that the web page content changes.

### F. Remove everything in the URL address bar after **visual.force.com**, then press ENTER.

# 2. Test your new page.

## A. From the Certification app, click the **Contacts** tab.

## B. Click Recently Viewed, then click All list view.

## C. Click the record named **Arthur Franz**.

## D. On the Contact detail page, click Technician Status.

## E. Click **Edit Technician Record**, then click **Cancel**.

## F. Click the browser back button.

17-1: Reference a Controller Extension in a Visualforce Page

Goal:

Use the provided controller extension to display all Certification Held records related to the current account in a Visualforce page embedded on the account page layout.

Tasks:

# 1. Create an apex controller extension that queries all Certification Held records.

# 2. Create a page to display all Certification Held records.

# 3. Create a section on the Account Page Layout to display the new page.

# 4. Test your new page.

Time:

15 minutes

Instructions:

# 1. Create an apex controller extension that queries all Certification Held records.

## A. Click Setup gear icon | Developer Console.

## B. Click File | New | Apex Class.

### C. Enter the following information:

|  |  |
| --- | --- |
| Name | AccountDisplayCertsHeld\_CX |

## D. Click **OK**.

### E. Copy and paste the contents of **AccountDisplayCertsHeld\_CX.txt** from the Exercises folder into the window, overwriting all existing text.

### F. Press CTRL + S to save the file.

# 2. Create a page to display the Certification Held records.

## A. Click File | New | Visualforce Page.

### B. Enter the following information:

|  |  |
| --- | --- |
| Name | AccountDisplayCertsHeldPage |

## C. Click **OK**.

### D. Copy and paste the contents of **AccountDisplayCertsHeldPage.txt** from the Exercises folder into the window, overwriting all existing text.

### E. Complete the TODOs.

### F. Press CTRL + S to save the file.

### G. Press CTRL + ALT + / to close all console tabs.

### H. Close the Developer Console window.

# 3. Create a section on the Account page layout to display the new page.

## A. Click Setup gear icon | Setup.

## B. Click **Object Manager**.

## C. Click **Account**.

## D. Click **Page Layouts**.

## E. Click **Service Vendor Account Layout**.

## F. From the palette, click **Visualforce Pages**.

## G. Drag and drop **Section** from the palette to below the **Account Information** section.

## H. Enter the following information:

|  |  |
| --- | --- |
| Section Name | Certifications Held |
| Display Section Header On | **Select** Detail Page  **Deselect** Edit Page |
| Layout | 1-Column |

## I. Click **OK**.

## J. Drag and drop **AccountDisplayCertsHeldPage** from the palette to the new Certifications Held section.

## K. Click the **Tool** icon in the top-right area of the AccountDisplayCertsHeldPage element to open the Visualforce Page Properties.

## L. Enter the following information:

|  |  |
| --- | --- |
| Width (in pixels or %) | 100% |
| Height (in pixels) | 100 |
| Show scrollbars | Selected |
| Show label | Deselected |

## M. Click **OK**.

## N. Click **Save.**

# 4. Test your new page.

## A. From the Certification app, click the **Accounts** tab.

## B. Click Recently Viewed, then click All list view.

## C. Click the record named Alveswood Technologies.

## D. Verify that the new section appears and that you can scroll through all the records.

17-2: Create a Simple Read-Only Property

Goal:

Create a simple controller extension that uses a property with a get block to allow you to display a table showing all currently in-progress certification attempts associated with an account on the account page layout.

Tasks:

# 1. Write the code necessary for the controller extension.

# 2. Create a page to display in-progress Certification Attempt records.

# 3. Create a section on the Account Page Layout to display the new page.

# 4. Test your new page.

Time:

15 minutes

Instructions:

# 1. Write the code necessary for the controller extension.

## A. Click Setup gear icon | Developer Console.

## B. Click File | New | Apex Class.

### C. Enter the following information:

|  |  |
| --- | --- |
| Name | AccountDisplayCertAttempts\_CX |

## D. Click **OK**.

### E. Copy and paste the contents of **AccountDisplayCertAttempts\_CX.txt** from the Exercises folder into the window, overwriting all existing text.

### F. Complete the TODOs.

### G. Press CTRL + S to save the file.

# 2. Create a page to display in-progress Certification Attempt records.

## A. Click File | New | Visualforce Page.

### B. Enter the following information:

|  |  |
| --- | --- |
| Name | AccountDisplayCertAttemptsPage |

## C. Click **OK**.

### D. Copy and paste the contents of **AccountDisplayCertAttemptsPage.txt** from the Exercises folder into the window, overwriting all existing text.

### E. Complete the TODOs.

### F. Press CTRL + S to save the file.

### G. Press CTRL + ALT + / to close all console tabs.

### H. Close the Developer Console window.

# 3. Create a section on the Account page layout to display the new page.

## A. Click Setup gear icon | Setup.

## B. Click **Object Manager**.

## C. Click **Account**.

## D. Click **Page Layouts**.

## E. Click **Service Vendor Account Layout**.

## F. From the palette, click **Visualforce Pages**.

## G. Drag and drop **Section** from the palette to below the **Certifications Held** section.

## H. Enter the following information:

|  |  |
| --- | --- |
| Section Name | Certification Attempts |
| Display Section Header On | **Select** Detail Page  **Deselect** Edit Page |
| Layout | 1-Column |

## I. Click **OK**.

## J. Drag and drop **AccountDisplayCertAttemptsPage** from the palette to the new Certifications Attempts section.

## K. Click the **Tool** icon in the top-right area of the AccountDisplayCertAttemptsPage element to open the Visualforce Page Properties.

## L. Enter the following information:

|  |  |
| --- | --- |
| Width (in pixels or %) | 100% |
| Height (in pixels) | 100 |
| Show scrollbars | Selected |
| Show label | Deselected |

## M. Click **OK**.

## N. Click **Save.**

# 4. Test your new page.

## A. From the Certification app, click the **Accounts** tab.

## B. Click Recently Viewed, then click All list view.

## C. Click the record named Alveswood Technologies.

## D. Verify that the new section appears and that you can scroll through all the records.

17-3: Creating a Read/Write Property in a Custom Controller

Goal:

Create a Visualforce page and custom controller to display a list of courses with corresponding checkboxes. Begin writing the action method that will trigger the search.

Tasks:

# 1. Write the code necessary for the custom controller.

# 2. Create a page to display all courses.

# 3. Test your new page.

# 

Time:

25 minutes

Instructions:

# 1. Write the code necessary for the custom controller.

## A. Click Setup gear icon | Developer Console.

## B. Click File | New | Apex Class.

### C. Enter the following information:

|  |  |
| --- | --- |
| Name | SearchCourses\_CC |

## D. Click **OK**.

### E. Copy and paste the contents of **SearchCourses\_CC.txt** from the Exercises folder into the window, overwriting all existing text.

### F. Complete the TODOs.

### G. Press CTRL + S to save the file.

# 2. Create a page to display all courses.

## A. Click File | New | Visualforce Page.

### B. Enter the following information:

|  |  |
| --- | --- |
| Name | SearchCoursesPage |

## C. Click **OK**.

### D. Copy and paste the contents of **SearchCoursesPage.txt** from the Exercises folder into the window, overwriting all existing text.

### E. Complete the TODOs.

### F. Press CTRL + S to save the file.

### G. Leave the Developer Console window open.

# 3. Test your new page.

## A. Click **Preview**. This will open a new browser tab displaying the Visualforce page.

## B. Select one or more of the checkboxes next to the courses listed, then click See Upcoming Course Deliveries. The page will refresh but no other visual changes.

## C. Close the browser tab displaying the Visualforce page.

## D. Switch back to the Developer Console window.

## E. Click the **Logs** tab in the bottom panel.

## F. Double-click the most recent log entry.

## G. Click Debug Only. You should see an entry printed for each course you selected with the phrase “Course Selected: <course name>”.

### H. Press CTRL + ALT + / to close all console tabs.

### I. Close the Developer Console window.

17-4: Implement the Search Button

Goal:

Create action method to create a map of just the selected courses. Your method should fire when the user clicks the search button.

Tasks:

# 1. Write the code necessary to create map of selected courses.

# 2. Test the code changes.

Time:

10 minutes

Instructions:

# 1. Write the code necessary to create map of selected courses.

## A. Click Setup gear icon | Developer Console.

### B. Click **File | Open**.

### C. Select **Classes** as the Entity Type, type Search into the filter input box, then double-click the entity **SearchCourses\_CC**.

### D. Copy and paste the contents of **SearchCourses\_CC.txt** from the Exercises folder into the window, overwriting all existing text.

## E. Complete the TODOs.

## F. Press CTRL + S to save the file.

# 2. Test the code changes.

### A. Click **File | Open**.

### B. Select **Pages** as the Entity Type, type Search into the filter input box, then double-click the entity **SearchCoursesPage**.

## C. Click **Preview**. This will open a new browser tab displaying the Visualforce page.

## D. Select one or more of the checkboxes next to the courses listed, then click See Upcoming Course Deliveries. The page will refresh but no other visual changes.

## E. Close the browser tab displaying the Visualforce page.

## F. Switch back to the Developer Console window.

## G. Click the **Logs** tab in the bottom panel.

## H. Double-click the most recent log entry.

## I. Click Debug Only. As like the previous exercise, you should see an entry printed for each course you selected with the phrase “Course Selected: <course name>”, and a new entry that printed the number of courses selected.

### J. Press CTRL + ALT + / to close all console tabs.

### K. Close the Developer Console window.

17-5: Redirecting to a Results Page

Goal:

You must redirect the user to show the results of the search they have set up.

Tasks:

# 1. Create a new results page.

# 2. Write the code necessary to redirect the user to a new page.

# 3. Add code to the new page to display the course deliveries.

# 4. Test your new page.

Time:

25 minutes

Instructions:

# 1. Create a new results page.

## A. Click Setup gear icon | Developer Console.

## B. Click File | New | Visualforce Page.

### C. Enter the following information:

|  |  |
| --- | --- |
| Name | ListCourseDeliveriesPage |

## D. Click **OK**.

## Note: We will return to this page in a later step to complete the code for it.

# 2. Write the code necessary to redirect the user to a new page.

## A. Click Setup gear icon | Developer Console.

### B. Click **File | Open**.

### C. Select **Classes** as the Entity Type, type Search into the filter input box, then double-click the entity **SearchCourses\_CC**.

### D. Copy and paste the contents of **SearchCourses\_CC.txt** from the Exercises folder into the window, overwriting all existing text.

## E. Complete the TODOs.

## F. Press CTRL + S to save the file.

# 3. Add code to the new page to display the course deliveries.

## A. Switch to the tab for the **ListCourseDeliveriesPage** page.

## B. Copy and paste the contents of **ListCourseDeliveriesPage.txt** from the Exercises folder into the window, overwriting all existing text.

### C. Complete the TODOs.

### D. Press CTRL + S to save the file.

# 4. Test your new page.

### A. Click **File | Open**.

### B. Select **Pages** as the Entity Type, type Search into the filter input box, then double-click the entity **SearchCoursesPage**.

## C. Click **Preview**. This will open a new browser tab displaying the Visualforce page.

## D. Select one or more of the checkboxes next to the courses listed, then click See Upcoming Course Deliveries. The page will refresh and show you the search results.

## E. Verify that the Upcoming Course Deliveries page displays course deliveries related to the course(s) you selected. Also verify that you can click the New Search button and return to the original search page.

## F. Close the browser tab displaying the Visualforce page.

## G. Switch back to the Developer Console window.

### H. Press CTRL + ALT + / to close all console tabs.

### I. Close the Developer Console window.

17-6: Handle Basic Save Errors in Your Method

Goal:

Add simple error handling to make sure the user has selected courses to search for before clicking Search.

Tasks:

# 1. Add conditional logic to check for selected course.

# 2. Add the <apex:pageMessages/> tag to your Visualforce page.

# 3. Test the code changes.

Time:

10 minutes

Instructions:

# 1. Add conditional logic to check for selected course.

## A. Click Setup gear icon | Developer Console.

### B. Click **File | Open**.

### C. Select **Classes** as the Entity Type, type Search into the filter input box, then double-click the entity **SearchCourses\_CC**.

### D. Copy and paste the contents of **SearchCourses\_CC.txt** from the Exercises folder into the window, overwriting all existing text.

## E. Complete the TODOs.

## F. Press CTRL + S to save the file.

# 2. Add conditional logic to check for selected course.

### A. Click **File | Open**.

### B. Select **Pages** as the Entity Type, type Search into the filter input box, then double-click the entity **SearchCoursesPage**.

### C. Copy and paste the contents of **SearchCoursesPage.txt** from the Exercises folder into the window, overwriting all existing text.

## D. Complete the TODOs.

## E. Press CTRL + S to save the file.

# 3. Test the code changes.

## A. Click **Preview**. This will open a new browser tab displaying the Visualforce page.

## B. Select NONE of the checkboxes next to the courses listed, then click See Upcoming Course Deliveries. The page will display a message to select at least one course.

## C. Close the browser tab displaying the Visualforce page.

## D. Switch back to the Developer Console window.

### E. Press CTRL + ALT + / to close all console tabs.

### F. Close the Developer Console window.

18-1: Create a Page to Display a List of Records

Goal:

Create a Visualforce page that lets you paginate and filter Account records.

Tasks:

# 1. Create a page to display Account records.

# 2. Test your new page.

Time:

15 minutes

Instructions:

# 1. Create a page to display Account records.

## A. Click File | New | Visualforce Page.

### B. Enter the following information:

|  |  |
| --- | --- |
| Name | AccountDisplayPage |

## C. Click **OK**.

### D. Copy and paste the contents of **AccountDisplayPage.txt** from the Exercises folder into the window, overwriting all existing text.

### E. Complete the TODOs.

### F. Press CTRL + S to save the file.

### G. Leave the Developer Console window open.

# 2. Test your new page.

## A. Click **Preview**. This will open a new browser tab displaying the Visualforce page.

## B. Paginate through the records by clicking the **>** and **<** buttons, and test filtering the records by setting **Filter by** to different values.

## C. Close the browser tab displaying the Visualforce page.

## D. Switch back to the Developer Console window.

## E. Press CTRL + ALT + / to close all open windows.

## F. Close the Developer Console window.

18-2: Integrate SOSL Search in a Visualforce Page

Goal:

Search for text across multiple sObjects.

Tasks:

# 1. Construct a simple SOSL search.

# 2. Create a code block to cycle through the results.

Time:

15 minutes

Instructions:

# 1. Construct a simple SOSL search.

## A. Click Setup gear icon | Developer Console.

## B. Click the **Query Editor** tab in the bottom panel.

## C. Write a SOSL search for a person named Frank. This search should be made on the Contact and User objects, returning the found person’s FirstName and LastName. Note: The search string must be delimited by curly braces without any quotes in the Query Editor tab. However, single quotes are required in an Anonymous block.

## D. Click Execute. As a result, a Query Results tab opens.

# 2. Create a code block to cycle through the results.

## A. Click Debug | Open Execute Anonymous Window.

## B. Copy and paste the contents of **ParseSOSLSearch.txt** from the Exercises folder into the window, overwriting all existing text.

## C. Complete the TODOs.

## D. Ensure that Open Log is selected and click Execute.

## E. Select **Debug Only** checkbox, then review the Debug Log to confirm the results.

## F. Press CTRL + ALT + / to close all console tabs.

## G. Close the Developer Console window.

18-3: Create a Simple Search Page

Goal:

Create a Visualforce page and custom controller to display a list of contacts based on the results of a SOSL search.

Tasks:

# 1. Write the code necessary for the custom controller.

# 2. Create a page to search for contacts.

# 3. Test your new page.

Time:

15 minutes

Instructions:

# 1. Write the code necessary for the controller extension.

## A. Click Setup gear icon | Developer Console.

## B. Click File | New | Apex Class.

### C. Enter the following information:

|  |  |
| --- | --- |
| Name | DisplayContacts\_CC |

## D. Click **OK**.

### E. Copy and paste the contents of **DisplayContacts\_CC.txt** from the Exercises folder into the window, overwriting all existing text.

### F. Complete the TODOs.

### G. Press CTRL + S to save the file.

# 2. Create a page to display all courses.

## A. Click File | New | Visualforce Page.

### B. Enter the following information:

|  |  |
| --- | --- |
| Name | DisplayContactsPage |

## C. Click **OK**.

### D. Copy and paste the contents of **DisplayContactsPage.txt** from the Exercises folder into the window, overwriting all existing text.

### E. Complete the TODOs.

### F. Press CTRL + S to save the file.

# 3. Test your new page.

## A. Click **Preview**. This will open a new browser tab displaying the Visualforce page.

## B. Type ar into the Search Text input box, then click **Search**. You should see a list of contacts whose names include the search term.

## C. Paginate through the records by clicking the **>** and **<** buttons.

## D. Erase the value in the Search Text input box, then click **Search**. You should see an error that your search term must be at least two characters long.

## E. Close the browser tab displaying the Visualforce page.

## F. Switch back to the Developer Console window.

### G. Press CTRL + ALT + / to close all console tabs.

### H. Close the Developer Console window.

19-1: Determine Whether a Declarative Solution Exists

Goal:

Determine which scenarios are best solved using Visualforce.

Task:

# Review each scenario and determine if Visualforce would be the best solution.

Time:

5 minutes

Scenarios:

# 1. The certification team would like to include a Chatter feed on each Certification record.

# 2. The certification team would like the Start Date field to display only when the Status is set to Scheduled on a Course Delivery.

# 3. The certification team would like the Course Deliveries related list to be the only related list displayed on the Course record.

# 4. The Contact object has a lot of fields and requires scrolling to see all the information. The certification team would like every section and related list to display as an individual tab that can be viewed when clicked on.

# 5. The certification team would like the Course list view to match the look and feel of their corporate website.

19-2: Defend Against SOQL Injection

Goal:

Modify the controller of a Visualforce page to defend against SOQL Injection.

Tasks:

# 1. Create a custom Visualforce Controller.

# 2. Create a Visualforce page.

# 3. Search for an existing record.

# 4. Sanitize the code to defend against SOQL Injection attacks.

Time:

10 minutes

Instructions:

# 1. Create a Controller Extension and save it.

## A. Click Setup gear icon | Developer Console.

## B. Click File | New | Apex Class.

### C. Enter the following information:

|  |  |
| --- | --- |
| Name | SOQLController |

## D. Click **OK**.

### E. Copy and paste the contents of **SOQLControllerClass.txt** from the Exercises folder into the window, overwriting all existing text.

### F. Press CTRL + S to save the file.

# 2. Create a Visualforce page.

## A. Click File | New | Visualforce Page.

### B. Enter the following information:

|  |  |
| --- | --- |
| Name | CertificationHeldQuickInfoPage |

## C. Click **OK**.

### D. Copy and paste the contents of **CertificationHeldQuickInfoPage.txt** from the Exercises folder into the window, overwriting all existing text.

### E. Press CTRL + S to save the file.

# 3. Search for an existing record.

## A. Click **Preview**. This will open a new browser tab displaying the Visualforce page.

## B. Type CERTIFICATION-00001 in the textbox, then click **Query**.

## C. Verify that the correct information is displayed.

# 4. Sanitize the code to defend against SOQL Injection attacks.

## A. Type CERTIFICATION in the textbox, then click **Query**. Try CERTIFICATION-0000\*, CERTIFICATION-0000?, CERTIFICATION-0000\_ or any other text to confirm that the query is not displaying unintended information.

## B. Simulate a SOQL Injection attack by typing:

### test' OR name like '%

## This has the effect of displaying all Certification Held records.

## C. Close the browser tab displaying the Visualforce page.

## D. In the Developer Console window, switch to the tab for the **SOQLController** class.

### E. Copy and paste the contents of **SOQLControllerClassRefactored.txt** from the Exercises folder into the window, overwriting all existing text.

### F. Press CTRL + S to save the file.

## G. Switch to the tab for the **CertificationHeldQuickInfoPage** page.

## H. Click **Preview**. This will open a new browser tab displaying the Visualforce page.

## I. Simulate a SOQL Injection attack by typing:

### test' OR name like '%

## There should be no results displayed because the attack was mitigated.

## J. Close the browser tab displaying the Visualforce page.

## K. Switch back to the Developer Console window.

### L. Press CTRL + ALT + / to close all console tabs.

### M. Close the Developer Console window.

19-3: Create a Custom Button that Uses JavaScript (Optional)

Goal:

Modify the TechnicianStatusPage Visualforce page to include a custom button.

Tasks:

# 1. Upload the JavaScript as a static resource.

# 2. Open the TechnicianStatusPage Visualforce page and complete the TODOs.

# 3. Test the button.

Time:

10 minutes

Instructions:

# 1. Upload the JavaScript as a static resource.

## A. Click Setup gear icon | Setup.

## B. Type static in the Quick Find textbox.

## C. Click Static Resources under Custom Code.

## D. Click New.

## E. Enter the following information:

|  |  |
| --- | --- |
| Name | CustomCancelButton |
| Description | A custom cancel button that creates a dialog box to ask users if they are sure they want to cancel. |
| Choose File | Click then locate and select the **CustomCancelButton.js** file in your Exercises folder. |
| Cache Control | **Public** |

## F. Click **Save**.

# 2. Open the TechnicianStatusPage Visualforce page.

## A. Click Setup gear icon | Developer Console.

### B. Click **File | Open**.

### C. Select **Pages** as the Entity Type, type Technician into the filter input box, then double-click the entity **TechnicianStatusPage**.

### D. Copy and paste the contents of **TechnicianStatusPage.txt** from the Exercises folder into the window, overwriting all existing text.

## E. Complete the TODOs.

## F. Press CTRL + S to save the file.

### G. Press CTRL + ALT + / to close all console tabs.

### H. Close the Developer Console window.

# 3. Test the button.

## A. From the Certification app, click the Contacts tab.

## B. Click Recently Viewed, then click All list view.

## C. Click record named **Aaryn Patel**.

## D. On the Contact detail page, click **Technician Status**.

## E. Click **Cancel** and confirm that the JavaScript displays confirmation prompt.

## F. Click **OK**. You should be redirected back to the contact’s detail page.

20-1: Write the Test Method for the Constructor

Goal:

Write a test method to verify that the Technician Status page's controller extension constructor is invoked successfully.

Tasks:

# 1. Upload controller code for a new extended version of Technician Status.

# 2. Upload markup code for a new extended version of Technician Status.

# 3. Create a unit test method to test the extension constructor.

# 4. Test your new unit test logic.

Time:

20 minutes

Instructions:

# 1. Upload controller code for a new extended version of Technician Status.

## A. Click Setup gear icon | Developer Console.

### B. Click File | New | Apex Class.

### C. Type TechnicianStatus\_CX for the name, then click **OK**.

### D. Copy and paste the contents of **TechnicianStatus\_CX.txt** from the Exercises folder into the window, overwriting all existing text.

### E. Press CTRL + S to save the file.

# 2. Upload markup code for new extended version of Technician Status.

### A. Click **File | Open**.

### B. Select **Pages** as the Entity Type, type Technician into the filter input box, then double-click the entity **TechnicianStatusPage**.

### C. Copy and paste the contents of **TechnicianStatusPage.txt** from the Exercises folder into the window, overwriting all existing text.

## D. Press CTRL + S to save the file.

# 3. Create a unit test method to test the extension constructor.

### A. Click File | New | Apex Class.

### B. Type TechnicianStatus\_Test for the name, then click **OK**.

### C. Copy and paste the contents of **TechnicianStatus\_Test.txt** from the Exercises folder into the window, overwriting all existing text.

### D. Complete the TODOs.

### E. Press CTRL + S to save the file.

# 4. Test your new unit test logic.

# A. Click **Run Test**.

### B. Click the Tests tab in the bottom panel.

### C. Expand the collapsed folder by clicking on +.

### D. Ensure that all your methods executed successfully. A green tick indicates success.

### E. Press CTRL + ALT + / to close all console tabs.

### F. Close the Developer Console window.

20-2: Write Unit Tests for Action Methods

Goal:

Write the necessary unit tests for the two custom action methods in the controller extension.

Tasks:

# 1. Examine the controller extension code for the Technician Status Page.

# 2. Create unit test methods to test the two custom action methods.

# 3. Test your new unit test logic.

Time:

20 minutes

Instructions:

# 1. Examine the controller extension code for the Technician Status Page.

## A. Click Setup gear icon | Developer Console.

### B. Click **File | Open**.

### C. Select **Classes** as the Entity Type, type Technician into the filter input box, then double-click the entity TechnicianStatus\_CX.

### D. Review the code for the action methods: editContact2 and editContact3.

# 2. Create unit test methods to test the two custom action methods.

### A. Click **File | Open**.

### B. Select **Classes** as the Entity Type, type Technician into the filter input box, then double-click the entity TechnicianStatus\_Test.

### C. Copy and paste the contents of TechnicianStatus\_Test**.txt** from the Exercises folder into the window, overwriting all existing text.

### D. Complete the TODOs.

## E. Press CTRL + S to save the file.

# 3. Test your new unit test logic.

# A. Click **Run Test**.

### B. Click the Tests tab in the bottom panel.

### C. Expand the collapsed folder by clicking on +.

### D. Ensure that all your methods executed successfully. A green tick indicates success.

### E. Press CTRL + ALT + / to close all console tabs.

### F. Close the Developer Console window.

20-3: Write Unit Tests for Getters and Setters

Goal:

Test whether the getter method in the controller extension returns results including the string "Attendees: ".

Tasks:

# 1. Examine the controller extension code for the Technician Status Page.

# 2. Create unit test methods to test the getAttendeeList getter method.

# 3. Test your new unit test logic.

Time:

20 minutes

Instructions:

# 1. Examine the controller extension code for the Technician Status Page.

## A. Click Setup gear icon | Developer Console.

### B. Click **File | Open**.

### C. Select **Classes** as the Entity Type, type Technician into the filter input box, then double-click the entity TechnicianStatus\_CX.

### D. Review the code for the getter method: getAttendeeList.

# 2. Create unit test methods to test the two custom action methods.

### A. Click **File | Open**.

### B. Select **Classes** as the Entity Type, type Technician into the filter input box, then double-click the entity TechnicianStatus\_Test.

### C. Copy and paste the contents of TechnicianStatus\_Test**.txt** from the Exercises folder into the window, overwriting all existing text.

### D. Complete the TODOs.

## E. Press CTRL + S to save the file.

# 3. Test your new unit test logic.

# A. Click **Run Test**.

### B. Click the Tests tab in the bottom panel.

### C. Expand the collapsed folder by clicking on +.

### D. Ensure that all your methods executed successfully. A green tick indicates success.

### E. Press CTRL + ALT + / to close all console tabs.

### F. Close the Developer Console window.

## C:\Users\lmann\Documents\_Training_Certification\classes\Developerville\Actual Design\Master\Certification App Design.png