TRAILHEAD

Student Guide

Programmatic Development Using Apex and Visualforce

DEV450-S18-V1-SG

salesforce

AGENDA

DEV450: Programmatic Development Using Apex and Visualforce

Day One

15 minutes	**Introductions**
60 minutes	**Welcome to AW Computing** *Watch Me 1-1 (5 min): Explore the Certification App* *Join Me 1-2 (5 min): Prepare Your Training Org* *Join Me 1-3 (5 min): Create a Sandbox* *Join Me 1-4 (5 min): Download the Apex Developer's Guide*
120 minutes	**Building Objects and Fields** ▪ Understanding Objects on the Force.com Platform ▪ Creating Custom Objects ▪ Creating Custom Fields *Join Me 2-1 (15 min): Create a Custom Object* *Join Me 2-2 (15 min): Create Custom Fields* ▪ Creating Relationships Between Objects *Join Me 2-3 (20 min): Create Relationship Fields*
100 minutes	**Working Effectively with Objects and Fields** ▪ Creating Formula Fields *Join Me 3-1 (10 min): Create a Formula Field* ▪ Creating Roll-Up Summary Fields *Join Me 3-2 (10 min): Create a Roll-Up Formula Field* *Your Turn 3-3 (10 min): Create a Formula Field that References Roll-Up Summary Fields* ▪ Understanding Record Types *Watch Me 3-4 (10 min): Understand Record Types* ▪ Building a Data Model on the Force.com Platform
90 minutes	**Programming with Apex** ▪ Getting Started with Apex *Join Me 4-1 (10 min): Logging into a Sandbox* *Join Me 4-2 (15 min): See Apex in Action* *Join Me 4-3 (10 min): Create and Use an Apex Class* *Watch Me 4-4 (5 min): Observe the Effects of Versioning* *Join Me 4-5 (20 min): Take a Quick Tour of Apex*

AGENDA

DEV450: Programmatic Development Using Apex and Visualforce

Day Two

90 minutes	**Programming with Apex (cont.)**

- What Makes Apex Different?
  Join Me 4-6 (10 min): Examine Implicit Operations
  Watch Me 4-7 (5 min): Profile Limits Using Developer Console
- Working with sObjects
  Your Turn 4-8 (15 min): Work with a Custom Object
  Join Me 4-9 (5 min): Use Record Ids to Access an Account in the UI

115 minutes	**Use SOQL to Query Your Org's Data**

- Using SOQL to Query Data
  Watch Me 5-1 (5 min): Create and Run Query in the Developer Console
  Your Turn 5-2 (20 min): Write a SOQL Query that Uses a WHERE Clause
- Writing and Processing a SOQL Query in Apex
  Your Turn 5-3 (10 min): Write and Execute a SOQL Query in Apex
- Creating a Dynamic Query at Run Time
  Your Turn 5-4 (10 min): Write a Dynamic Query in Apex

80 minutes	**Use SOQL to Query Parent-Child Relationships**

- Understanding Relationship Queries
- Querying Child-to-Parent Relationships
  Your Turn 6-1 (15 min): Write and Test Child-to-Parent Relationship Queries
- Querying Parent-to-Child Relationships
  Your Turn 6-2 (20 min): Query Account and Related Contacts

95 minutes	**DML Essentials**

- Options for Persisting Data
- Invoking DML Events
  Your Turn 7-1 (25 min): Execute DML Commands
- Handling DML Errors and Exceptions
  Your Turn 7-2 (15 min): Handle DML Errors and Exceptions

AGENDA

DEV450: Programmatic Development Using Apex and Visualforce

Day Three

80 minutes	**Trigger Essentials** ▪ Automating Logic ▪ Defining a Trigger *Your Turn 8-1 (5 min): Define a Trigger* ▪ Defining Trigger Logic *Join Me 8-2 (15 min): Define the Trigger's Business Logic*
70 minutes	**Apex Class Essentials** ▪ Using an Apex Class ▪ Defining an Apex Class *Your Turn 9-1 (15 min): Define an Apex Class* ▪ Determining Data Access for an Apex Class
80 minutes	**The Save Order of Execution and Apex Transactions** ▪ Exploring the Save Order of Execution *Watch Me 10-1 (10 min): Explore the Implicit Firing of Triggers* *Your Turn 10-2 (10 min): View the Events that Occur During a Rollback* ▪ Working with Apex Transactions *Watch Me 10-3 (10 min): See the Save Order of Execution in Action*
75 minutes	**Testing Essentials** ▪ Describing Apex's Testing Framework ▪ Creating Test Data *Your Turn 11-1 (10 min): Make Test Data Available to Test Methods* ▪ Writing and Running an Apex Test *Your Turn 11-2 (10 min): Write and Run an Apex Test*
50 minutes	**Testing Strategies** ▪ Understanding the Side Effects of Testing *Your Turn 12-1 (10 min): Explore Code Coverage* ▪ Testing Using Best Practices
95 minutes	**Strategies for Designing Efficient Apex Solutions** ▪ Working Efficiently with the Database *Your Turn 13-1 (15 min): Refactor a Trigger to Avoid SOQL Limits* *Your Turn 13-2 (15 min): Refactor a Trigger to Avoid DML Limits*

AGENDA

DEV450: Programmatic Development Using Apex and Visualforce

Day Four

95 minutes	**Strategies for Designing Efficient Apex Solutions (cont.)** ▪ Designing Triggers ▪ Designing Classes
120 minutes	**Trigger Design Strategy** ▪ Analyzing the Problem ▪ Creating a Solution *Your Turn 14-1 (5 min): Create a Formula Field to Eliminate a Query* *Your Turn 14-2 (5 min): Create Fields for Counting Certifications Elements* *Your Turn 14-3 (15 min): Create Collections to Filter the Query* *Your Turn 14-4 (20 min): Use a Map to Aggregate Results* *Your Turn 14-5 (15 min): Create Certification Held Records* *Your Turn 14-6 (10 min): Use a Workflow to Avoid Creation of Duplicate Records (optional)*
55 minutes	**Creating Visualforce Pages** ▪ Understanding Visualforce *Join Me 15-1 (10 min): Create a Simple Visualforce Page* ▪ Creating a Visualforce Page ▪ Displaying Record Data and Launching a Visualforce Page *Your Turn 15-2 (15 min): Display Data in a Visualforce Page*
50 minutes	**Exploring the View and Controller Layers** ▪ Accessing Data on Related Records *Join Me 16-1 (15 min): Create a Simple Technician Status Page* ▪ Exploring Visualforce Tags and Built-in Styling *Your Turn 16-2 (10 min): Refine Your Page and Add Navigational Links*
140 minutes	**Working with Custom Controllers and Controller Extensions** ▪ Referencing a Custom Controller *Your Turn 17-1 (5 min): Reference a Controller Extension in a Visualforce Page* ▪ Working with Getters, Setters, and Properties *Your Turn 17-2 (10 min): Create a Simple Read-Only Property*

AGENDA

DEV450: Programmatic Development Using Apex and Visualforce

Day Five

140 minutes	**Working with Custom Controllers and Controller Extensions (cont.)** ▪ Working with Action Methods *Your Turn 17-3 (15 min): Writing a Read/Write Property in a Custom Controller* *Your Turn 17-4 (10 min): Implementing the Search Button* *Your Turn 17-5 (10 min): Redirecting to a Results Page* ▪ Handling Basic Errors *Your Turn 17-6 (10 min): Handle Basic Save Errors in Your Method*
120 minutes	**Working with Lists Controllers and SOSL Queries** ▪ Working with Standard List Controllers *Join Me 18-1 (15 min): Create a Page to Display a List of Records* ▪ Writing a Simple SOSL Query *Your Turn 18-2 (15 min): Integrate SOSL Search in a Visualforce Page* ▪ Creating a Custom List Controller *Your Turn 18-3 (15 min): Create a Simple Search Page*
60 minutes	**Visualforce Development Considerations** ▪ When to Use Visualforce *Your Turn 19-1 (5 min): Determine Whether a Declarative Solution Exists* ▪ Visualforce and Governor Limits ▪ Security Considerations for Visualforce *Your Turn 19-2 (10 min): Defend Against SOQL Injection* ▪ Developing Pages for Mobile Devices ▪ JavaScript in Visualforce *Your Turn 19-3 (10 min): Create a Custom Button that Uses JavaScript*
100 minutes	**Testing Visualforce Controllers** ▪ Understanding Visualforce Controller Testing *Your Turn 20-1 (20 min): Write the Test Methods for the Constructor* ▪ Testing a Visualforce Controller Constructor ▪ Testing Action Methods *Your Turn 20-2 (20 min): Write Unit Tests for Action Methods* ▪ Testing Getters, Setters, and Properties *Your Turn 20-3 (20 min): Write Unit Tests for Getters and Setters*
15 minutes	**Wrap Up**

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

@Trailhead

Safe harbor statement under the Private Securities Litigation Reform Act of 1995:

This document and other items we publish, including through social media outlets, may contain forward-looking statements, the achievement or success of which involves risks, uncertainties, and assumptions. If any such risks or uncertainties materialize or if any of the assumptions proves incorrect, the results of salesforce.com, inc. could differ materially from the results expressed or implied by the forward-looking statements we make.

The risks and uncertainties referred to above include – but are not limited to – risks associated with possible fluctuations in our financial and operating results; our rate of growth and anticipated revenue run rate, including our ability to convert deferred revenue and unbilled deferred revenue into revenue and, as appropriate, cash flow, and our ability to grow deferred revenue and unbilled deferred revenue; errors, interruptions or delays in our service or Web hosting; breaches of our security measures; the financial impact of any previous and future acquisitions; the nature of our business model; our ability to continue to release, and gain customer acceptance of, new and improved versions of our service; successful customer deployment and utilization of our existing and future services; changes in our sales cycle; competition; various financial aspects of our subscription model; unexpected increases in attrition or decreases in new business; our ability to realize benefits from strategic partnerships; reliance on third-party computer hardware and software; the emerging markets in which we operate; unique aspects of entering or expanding in international markets; our ability to hire, retain and motivate employees and manage our growth; changes in our customer base; technological developments; regulatory developments; litigation related to intellectual property and other matters, and any related claims, negotiations and settlements; unanticipated changes in our effective tax rate; factors affecting our outstanding convertible notes and credit facility; fluctuations in the number of shares we have outstanding and the price of such shares; foreign currency exchange rates; collection of receivables; interest rates; factors affecting our deferred tax assets and ability to value and utilize them, including the timing of achieving profitability on a pre-tax basis; the potential negative impact of indirect tax exposure; the risks and expenses associated with our real estate and office facilities space; and general developments in the economy, financial markets, and credit markets.

Further information on these and other factors that could affect the financial results of salesforce.com, inc. is included in the reports on Forms 10-K, 10-Q and 8-K and in other filings we make with the Securities and Exchange Commission from time to time, including our most recent Form 10-K. These documents are available on the SEC Filings section of the Investor Information section of our website at www.salesforce.com/investor.

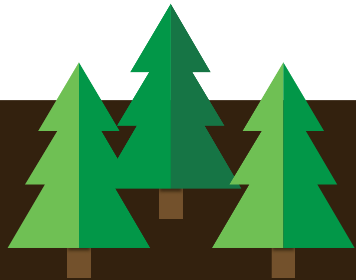
Any unreleased services or features referenced in this or other presentations, press releases or public statements are not currently available and may not be delivered on time or at all. Customers who purchase our services should make their purchase decisions based upon features that are currently available.

Logistics

- Class etiquette and participation
- Breaks

Courseware and Agenda

- Agenda for this class
- Layout of the manual and exercises



Your Fellow Students

- Your name
- Goals for your time in this class

COURSE AGENDA

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

MODULE 1:
WELCOME TO AW COMPUTING

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

Programme de révision :
http://www.salesforceben.com/platform-developer-certification-guide-tips/

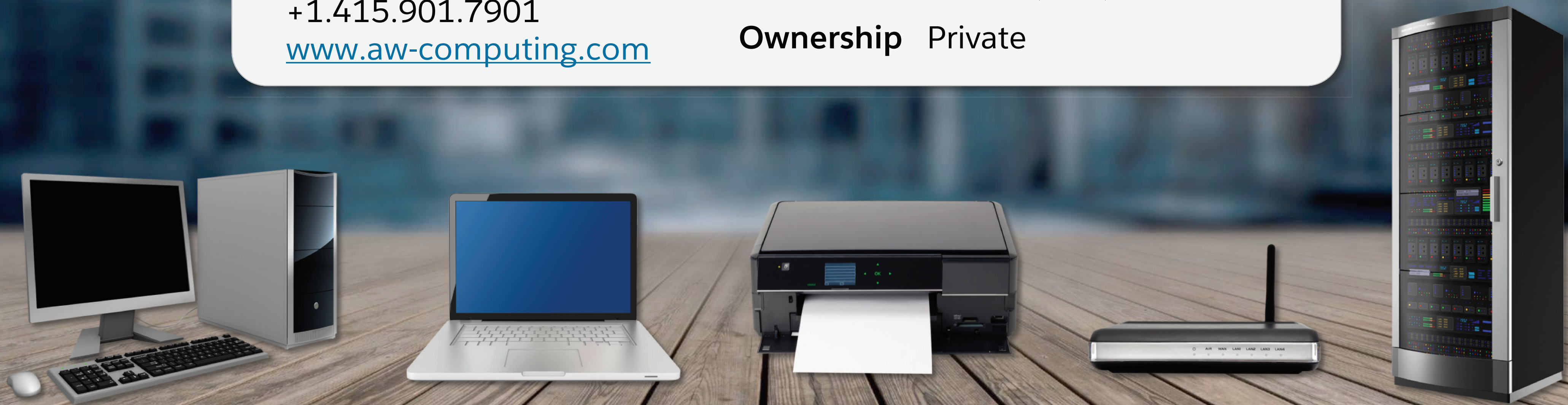
WELCOME

A.W. Computing

1 Market St.
San Francisco, CA 94105
United States

+1.415.901.7901
www.aw-computing.com

Industry	Computers & Electronics
Employees	750
Revenue	USD 30,000,000
Ownership	Private



Emely Adjei, Sales Rep – "I use the Sales app to keep track of my customer accounts and contacts, and to track my sales deals as they move through the pipeline and become wins."

Manage Customer Accounts

Track account information.

Manage Contacts

Track key contacts for each account.

Track Opportunities

Move deals through the sales cycle.

Deals Won

USING THE CALL CENTER APP

Herve Lopitaux, Support Rep – “I use the Call Center app to keep track of my customer accounts and contacts. I also use it to track, escalate, and resolve customer issues. ”

Manage Customer Accounts

Track account information.

Manage Contacts

Track key contacts for each account.

Track Cases

Work on or escalate customer issues.

Issues Resolved

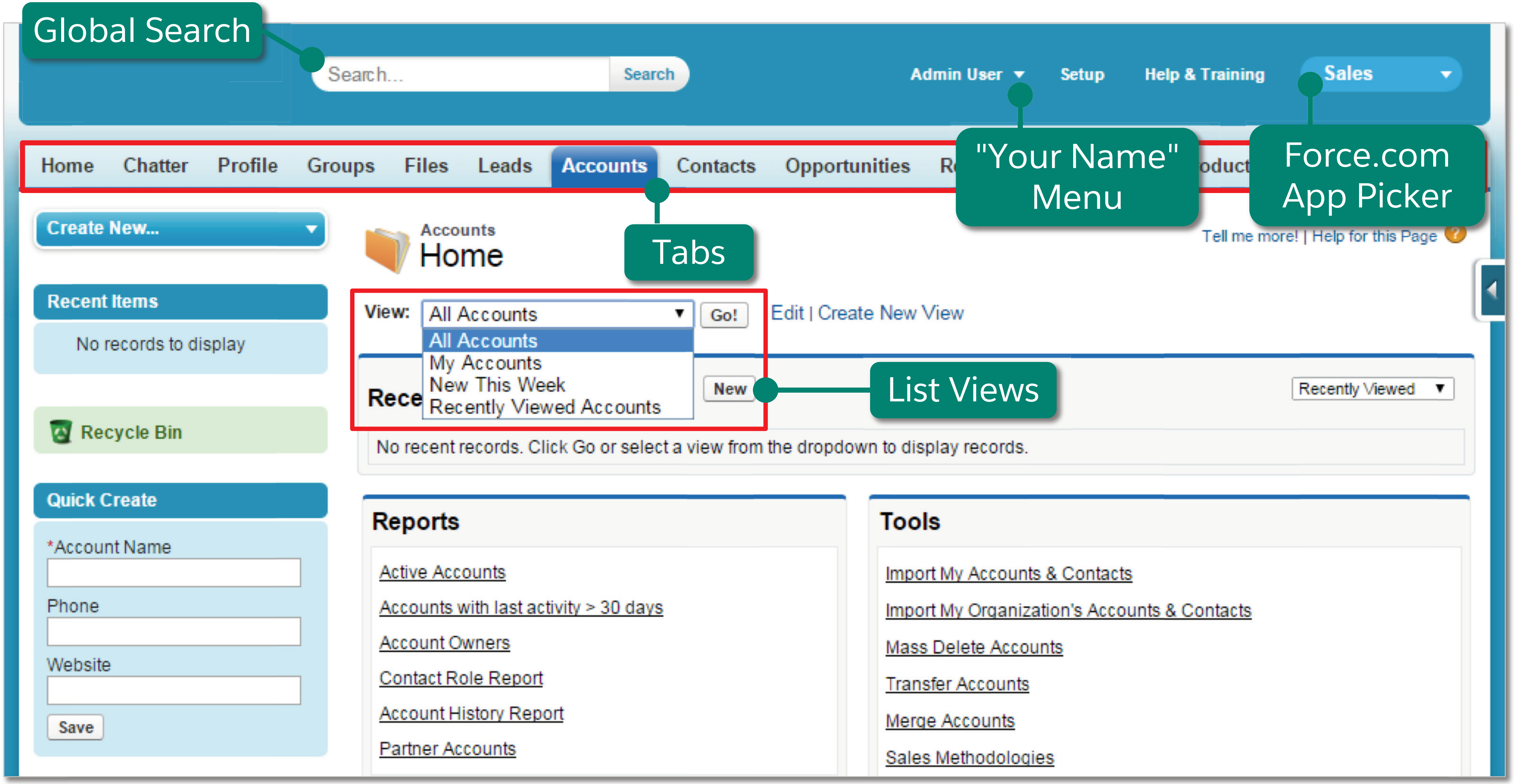
Story

USING THE CERTIFICATION APP

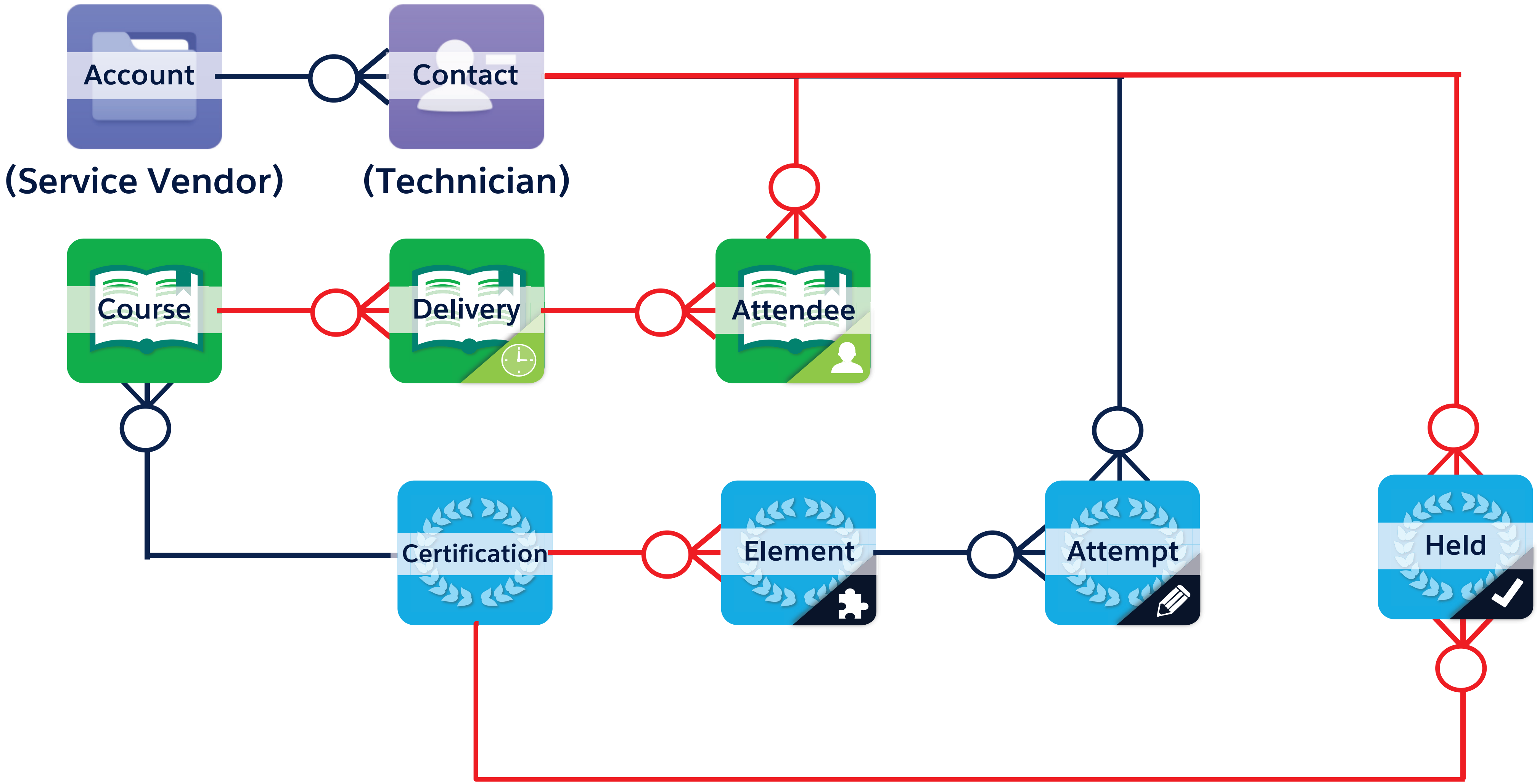
salesforce

Samantha Duncan, VP of Professional Services – “Our team sends contractor technicians out into the field to install and repair our server and networking equipment. The custom Certification app lets us manage the training and certification of all technicians.”

Manage Vendor Accounts	Manage Technician Contacts	Manage Courses and Deliveries	Manage Certifications and Attempts	
Track vendor information.	Track technicians for each vendor.	Manage training delivery schedule and enroll technicians in courses.	Track where technicians are in the certification process.	Technicians Certified


Global Search
Search...
Search
Admin User
Setup
Help & Training
Sales
Home
Chatter
Profile
Groups
Files
Leads
Accounts
Contacts
Opportunities
"Your Name" Menu
Force.com App Picker
Create New...
Accounts
Home
Tabs
Tell me more! | Help for this Page
Recent Items
No records to display
View:
All Accounts
Go!
Edit | Create New View
All Accounts
My Accounts
New This Week
Recently Viewed Accounts
New
List Views
Recently Viewed
Recycle Bin
No recent records. Click Go or select a view from the dropdown to display records.
Quick Create
*Account Name
Phone
Website
Save
Reports
Active Accounts
Accounts with last activity > 30 days
Account Owners
Contact Role Report
Account History Report
Partner Accounts
Tools
Import My Accounts & Contacts
Import My Organization's Accounts & Contacts
Mass Delete Accounts
Transfer Accounts
Merge Accounts
Sales Methodologies

UNDERSTANDING THE CERTIFICATION APPLICATION



WATCH ME

1-1: EXPLORE THE CERTIFICATION APP

5 minutes

Goal:

Before you begin customizing your Salesforce org, you should understand the existing configuration, and how it is used.

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.

Story

MEET THE DEVELOPER TEAM

Ryan Jackson
Lead Salesforce Programmatic Developer

Cassie Evans
Salesforce Programmatic Developer

Jason Beck
Salesforce Programmatic Developer

As our new Salesforce Programmatic Developer, we need you to:

- Understand and extend the declarative capabilities of Salesforce.
- Customize business logic using Apex classes and triggers.
- Develop custom user interfaces using Visualforce pages and controllers.
- Access data in Salesforce using SOQL and SOSL.
- Manipulate records in Salesforce using DML.
- Develop and test code using development best practices.

A.W. Computing

HOW CAN SALESFORCE BE CUSTOMIZED?

User Interface

Business Logic

Data Model

Each of these layers can be configured declaratively or customized programmatically.

Declarative
Using a point-and-click interface which does not require coding skills.

Programmatic
Using code, including Apex, Visualforce, and the APIs.

JOIN ME

1-2: PREPARE YOUR TRAINING ORG

5 minutes

Goal:
Prepare your org for classroom activities and access after class.

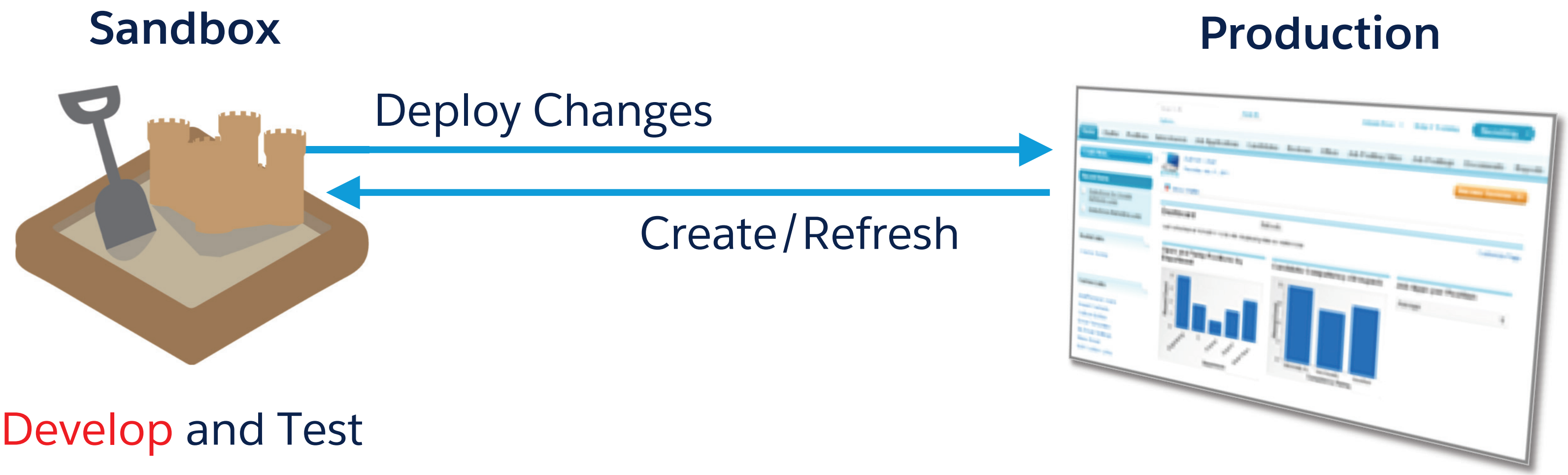
Tasks:

1. Log in to the training org and reset your user details.
2. Confirm your email address.
3. Download your lab files from the Documents tab.
4. Verify the Developer Console settings.

WHAT IS A SANDBOX?

DEFINITION:

A **sandbox** is a replica of your production organization that allows you to develop and test in a separate environment without risking or compromising data.



The time it takes to create or refresh a sandbox depends on:

- The amount (and complexity) of metadata and data to be copied from production to the sandbox.
- The amount of activity on your sandbox server.

JOIN ME

1-3: CREATE A SANDBOX

5 Minutes

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

Task:

Create a full sandbox named Dev.

FINDING MORE INFORMATION

You can find more resources to help you with declarative and programmatic customization:

- The eBook for this class
- Help & Training
  - Documentation
  - Knowledge Articles
  - Training
- http://developer.salesforce.com
  - Documentation
  - Forums
  - Code share
  - Sign up for a free developer edition org

JOIN ME

1-4: DOWNLOAD THE APEX DEVELOPER'S GUIDE

salesforce

5 Minutes

Goal:
Download the Apex Developer's Guide to use as a resource.

Task:

Download the Apex Developer's Guide.

MODULE 2:
BUILDING OBJECTS AND FIELDS

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

TRAILHEAD

We need you to create a new object for the Sales app to track customer success stories. These stories will highlight customer wins and be used by sales and marketing to win similar deals with other customers.

To accomplish this, you need to:

- Describe the capabilities of objects on the Force.com platform.
- Create a custom object.
- Create custom fields.
- Create relationship fields.

MODULE AGENDA

MODULE 2: BUILDING OBJECTS AND FIELDS

- **Understanding Objects on the Force.com Platform**
- Creating Custom Objects
- Creating Custom Fields
- Creating Relationships Between Objects

WHY ARE WE DOING THIS?

User Interface

View

Business Logic

Controller

Data Model

Model

WHAT IS AN OBJECT ON THE FORCE.COM PLATFORM?

salesforce

DEFINITION: In very simplistic terms, an **object** on the Force.com platform is similar to a database table.

Field

Account Name	Phone	Industry
ABC Labs	(408) 555-2091	Biotechnology
American Bank	(800) 555-2277	Banking
AMP Industries	(415) 555-4887	Manufacturing
Cable Inc.	(914) 555-1000	Entertainment

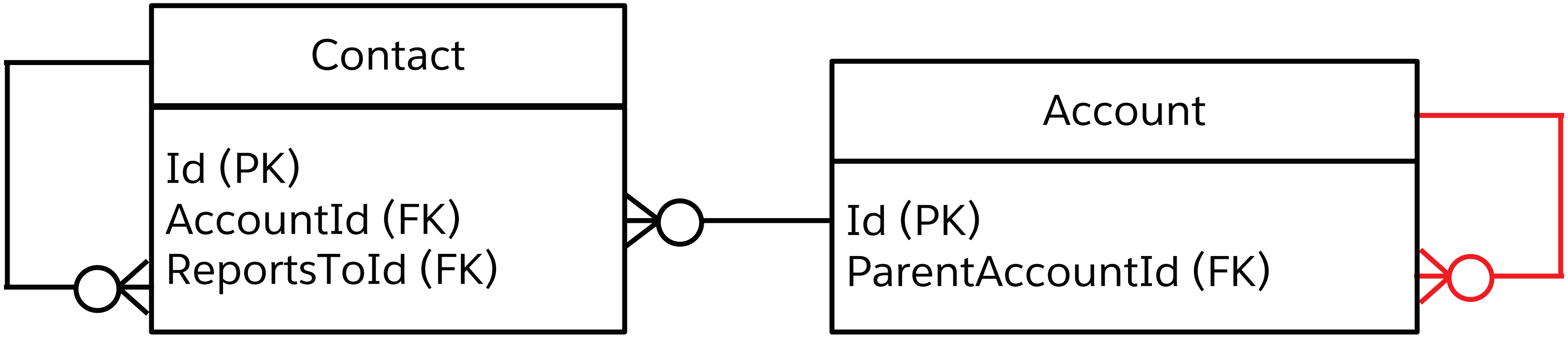
Record

Data Value

Pas une vrai table de donnée, on a des infos en plus

DATA MODEL: ACCOUNT AND CONTACT OBJECTS

Modèle de données SALES OBJECT : https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/sforce_api_erd_majors.htm



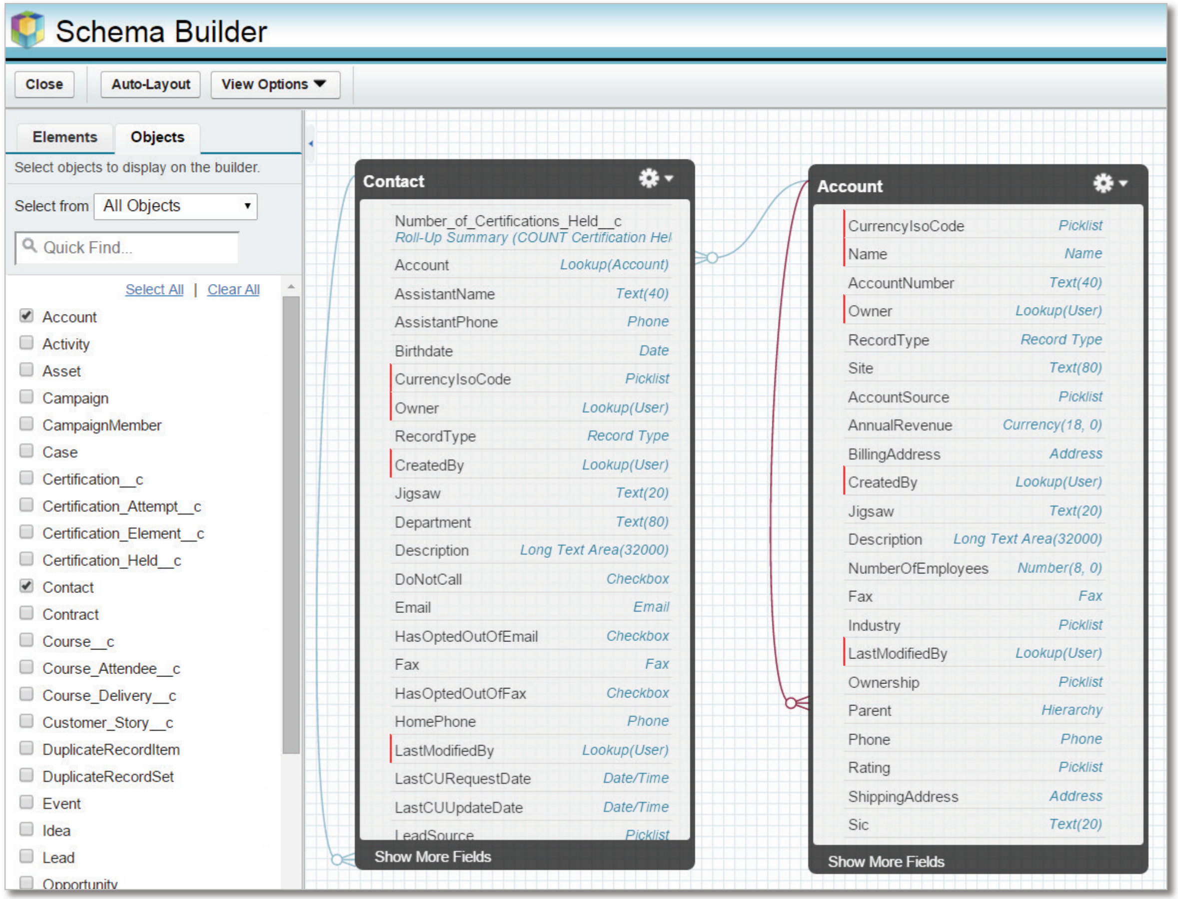
Aller voir Guide : Reference | Data Model

RESOURCE:

https://developer.salesforce.com/

Soap API Developer's Guide: **Reference | Data Model**

USING SCHEMA BUILDER TO VIEW OBJECTS AND RELATIONSHIPS



1. What information does Schema Builder display about an object?
2. Is the Id field displayed for Contact? For Account?
3. What type of relationship exists between Contact and Account?
4. Which field holds the foreign key?

CLICK PATH:

Setup | Schema Builder

MODULE 2: BUILDING OBJECTS AND FIELDS

- Understanding Objects on the Force.com Platform
- **Creating Custom Objects**
- Creating Custom Fields
- Creating Relationships Between Objects

TRACKING CUSTOMER SUCCESS STORIES

We need you to create a custom object to track customer success stories.

Custom Object

Customer Story

Help for this Page

Standard Fields [5] | Custom Fields & Relationships [0] | Validation Rules [0] | Page Layouts [1] | Field Sets [0] | Compact Layouts [1] | Search Layouts [4] | Buttons, Links, and Actions [8] | Record Types [0] | Apex Sharing Reasons [0] | Apex Sharing Recalculation [0] | Object Limits [10]

Custom Object Definition Detail Edit Delete

Singular Label	Customer Story	Description	Used to track customer success stories.
Plural Label	Customer Stories	Enable Reports	✓
Object Name	Customer_Story	Track Activities	
API Name	Customer_Story__c	Allow in Chatter Groups	
		Allow Sharing	✓
		Allow Bulk API Access	✓
		Allow Streaming API Access	✓
		Track Field History	
		Deployment Status	Deployed
		Help Settings	Standard salesforce.com Help Window
Created By	Admin User, 6/11/2015 2:23 PM	Modified By	Admin User, 6/11/2015 2:23 PM

Standard Fields Standard Fields Help

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
	Created By	CreatedBy	Lookup(User)		
Edit	Currency	CurrencyIsoCode	Picklist		
Edit	Customer Story Name	Name	Text(80)		✓
	Last Modified By	LastModifiedBy	Lookup(User)		
Edit	Owner	Owner	Lookup(User,Queue)		✓

Jason Beck
Developer

WHAT IS A STANDARD OBJECT?

DEFINITION:

A **standard object** is an object that is predefined by the Force.com platform.

Account: A company with which you do business.

Each standard object comes with a predefined set of standard fields.

Account Standard Fields

Action	Field Label	Field Name	Data Type
Edit	Account Currency	CurrencyIsoCode	Picklist
	Account Name	Name	Name
Edit	Account Number	AccountNumber	Text(40)
Edit	Account Owner	Owner	Lookup(User)
Edit	Account Record Type	RecordType	Record Type
Edit	Account Site	Site	Text(80)
Replace | Edit	Account Source	AccountSource	Picklist
Edit	Annual Revenue	AnnualRevenue	Currency(18, 0)
	Billing Address	BillingAddress	Address
	Created By	CreatedBy	Lookup(User)

Other objects:

Contact: An individual associated with your business accounts.

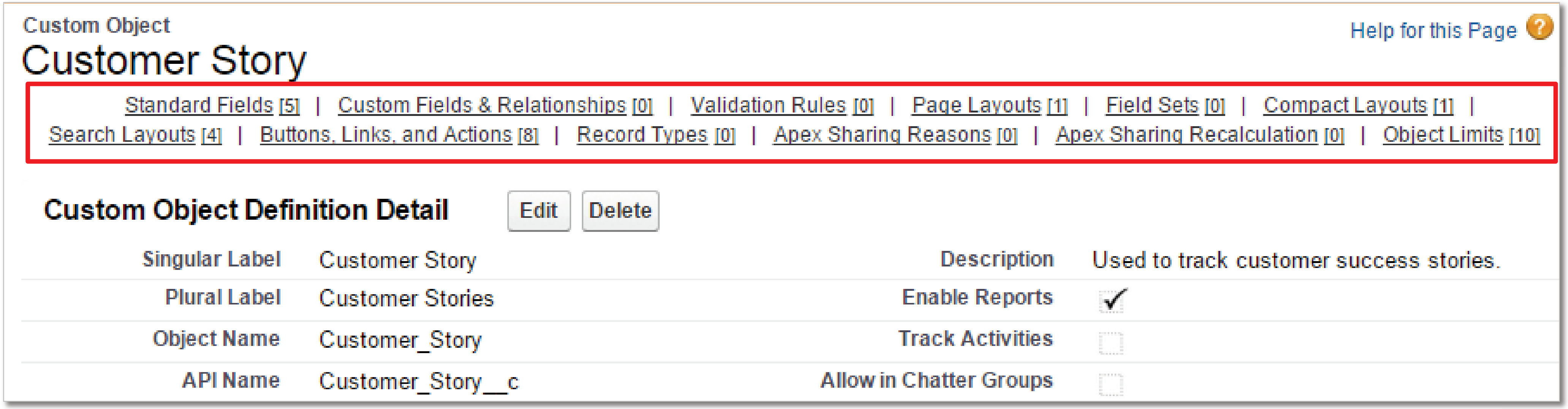
Opportunity: A sales deal.

Case: A customer issue.

WHAT IS A CUSTOM OBJECT?

DEFINITION:

A **custom object** is created by a developer to capture and manage additional data based on specific business requirements.



Custom objects have the same features and functionality as standard objects.

OBJECT ACCESS

When you create a custom object, you must give users access to the object. This is done by setting the object permissions on a custom profile. A permission set can also be used to provide a user access to an object.

Un objet dans SalesForce est l'ensemble de ses champs, mais aussi des relations, des layouts etc : c'est plus qu'une bdd habituelle.

Object Permissions

Permission Name	Enabled
Read	
Create	
Edit	
Delete	
View All	
Modify All	

Object permissions determine whether users can view, create, edit, or delete records in an object.

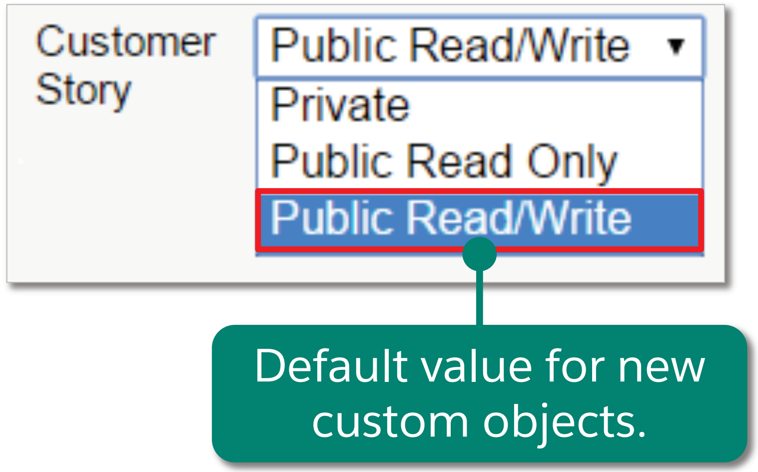
RECORD ACCESS

Record access determines which individual records users can view and edit in each object they have access to on their profile.

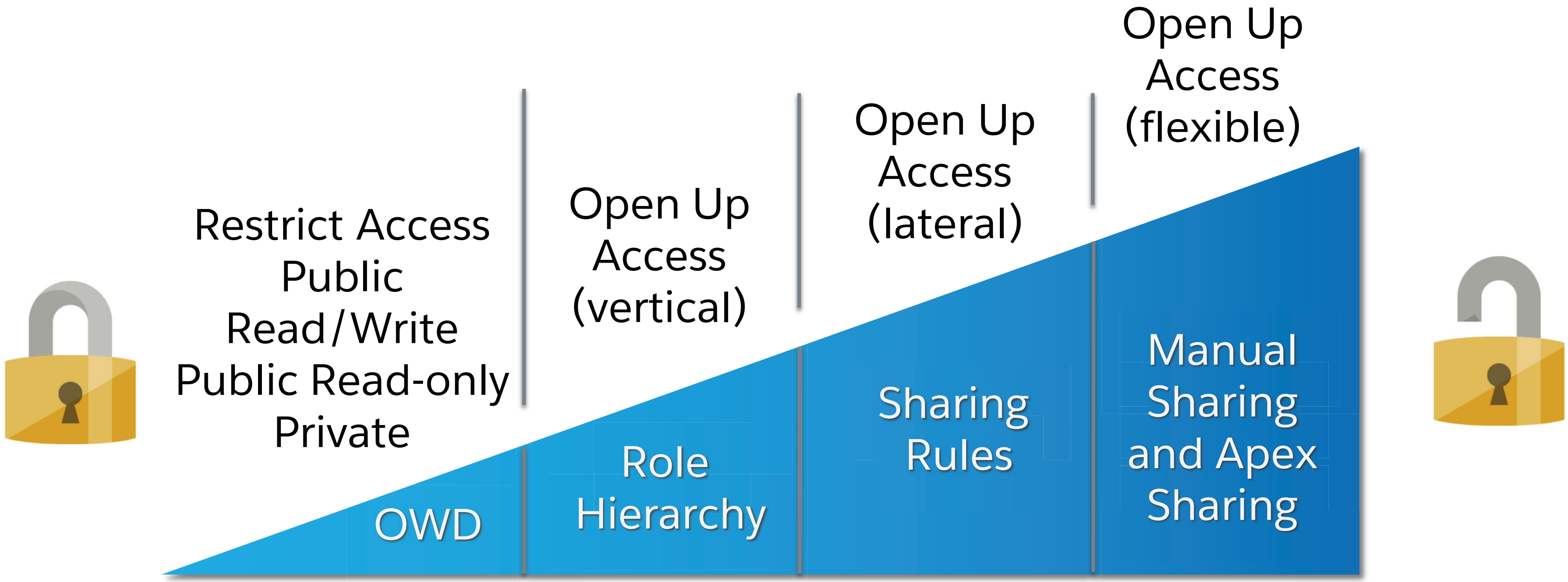
Given the appropriate object permissions, the user who owns a record can always:

- View and edit the record.
- Transfer the record to a different owner.
- Delete the record.
- Share the record.

Organization-wide defaults set the default level of access users have to records they do not own, in each object.



If the organization-wide default setting for an object is private or public read-only, you can open up access to records using a variety of tools.



JOIN ME

2-1: CREATE A CUSTOM OBJECT

salesforce

15 minutes

Goal:

Create a custom object to track customer success stories.

Tasks:

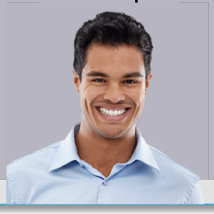
1. View the Account and Contact standard objects in Schema Builder.
2. Create a custom object from Schema Builder.
3. View the object detail page in the Setup menu.
4. Edit the object permissions on the Sales User and Marketing User profiles.
5. View the organization-wide default setting for the new object.

MODULE AGENDA

MODULE 2: BUILDING OBJECTS AND FIELDS

- Understanding Objects on the Force.com Platform
- Creating Custom Objects
- **Creating Custom Fields**
- Creating Relationships Between Objects

Jason Beck
Developer



For each customer story, we need you to capture a brief description of the story, the products involved in the sale, and the number of days from purchase to installation completed.

Custom Fields & Relationships New | Field Dependencies | Custom Fields & Relationships Help ?

Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By
Edit | Del	Installation Time (days)	Installation_Time__c	Number(3, 0)			Admin User, 6/22/2015 5:54 AM
Edit | Del | Replace	Products	Products__c	Picklist (Multi-Select)			Admin User, 6/22/2015 5:39 AM
Edit | Del	Story Description	Story_Description__c	Text(255)			Admin User, 6/23/2015 5:12 AM

WHAT IS A STANDARD FIELD?

DEFINITION:

A **standard field** is a field that is predefined by the Force.com platform.

You can customize some aspects of standard fields, including:

- Values in picklists.

Case Priority Picklist Values

Action	Values	Default
Edit | Del	High	☐
Edit	Medium	✓
Edit | Del	Low	☐

- Format of auto number fields.

Display Format	{00000000} **Example: A-{0000}** What Is This?
Next Number	1004

- Lookup filters on relationship fields.

Field Information

Field Label	Contact Name	Field Name	Contact
Data Type	Lookup(Contact)	Child Relationship Name	Cases
Help Text			

Lookup Filter

Filter Criteria	Contact Name: Account Name ID EQUALS Case: Account Name ID
Filter Type	**Required.** The user-entered value must match filter criteria.
Error Message	You must select a contact related to the same account as the case.
Lookup Window Text	
Active	✓

NOTE:

You cannot delete standard fields, but you can hide them from users.

STANDARD FIELDS FOR CUSTOM OBJECTS

When creating a custom object, the Force.com platform automatically generates these standard fields.

Standard Fields

Action	Field Label	Field Name	Data Type
	Created By	CreatedBy	Lookup(User)
Edit	Currency	CurrencyIsoCode	Picklist
Edit	Customer Story Name	Name	Text(80)
	Last Modified By	LastModifiedBy	Lookup(User)
Edit	Owner	Owner	Lookup(User,Queue)

The Force.com platform also generates a standard Id field.

Id à 15 chiffres.

WHAT IS A CUSTOM FIELD?

DEFINITION:

A **custom field** is created by a developer to capture a specific piece of information.

Custom Fields & Relationships New | Field Dependencies | Custom Fields & Relationships Help ?

Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By
Edit | Del	Installation Time (days)	Installation_Time__c	Number(3, 0)			Admin User, 6/22/2015 5:54 AM
Edit | Del | Replace	Products	Products__c	Picklist (Multi-Select)			Admin User, 6/22/2015 5:39 AM
Edit | Del	Story Description	Story_Description__c	Text(255)			Admin User, 6/23/2015 5:12 AM

Because custom fields are not built in, you can customize and delete them.

FIELD TYPES

Numeric
- Number
- Currency
- Percent

Calculation
- Auto Number
- Formula
- Roll-Up Summary

Calendar
- Date
- Date/Time

Formatted Text
- Email
- Phone
- URL
- Geolocation

Text
- Text
- Text (Encrypted)
- Text Area
- Long Text Area
- Rich Text Area

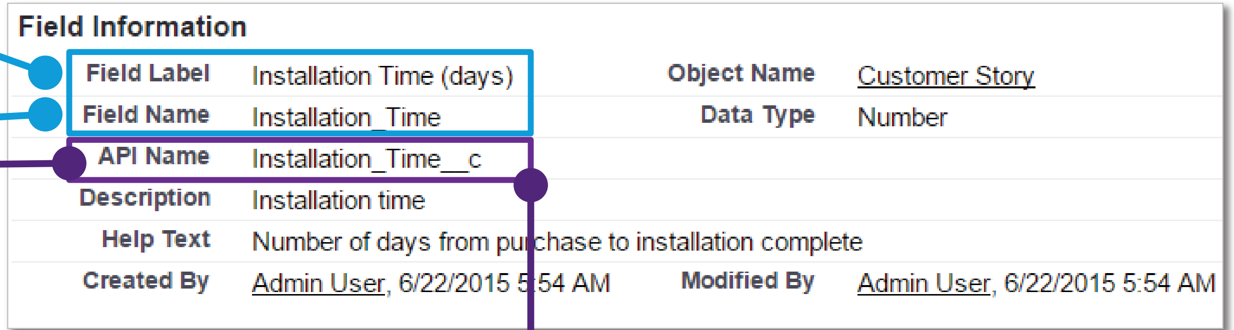
Limited Option
- Checkbox
- Picklist
- Picklist (Multi-Select)

Relationship
- Lookup
- Hierarchy
- Master-Detail

FIELD LABELS VS. API NAMES

Name displayed in the user interface. Field label and field name can be specified during field creation.

Used to automatically generate the API name by appending "__c".



Field Information

Field Label	Installation Time (days)	**Object Name**	Customer Story
Field Name	Installation_Time	**Data Type**	Number
API Name	Installation_Time__c		
Description	Installation time		
Help Text	Number of days from purchase to installation complete		
Created By	Admin User, 6/22/2015 5:54 AM	**Modified By**	Admin User, 6/22/2015 5:54 AM

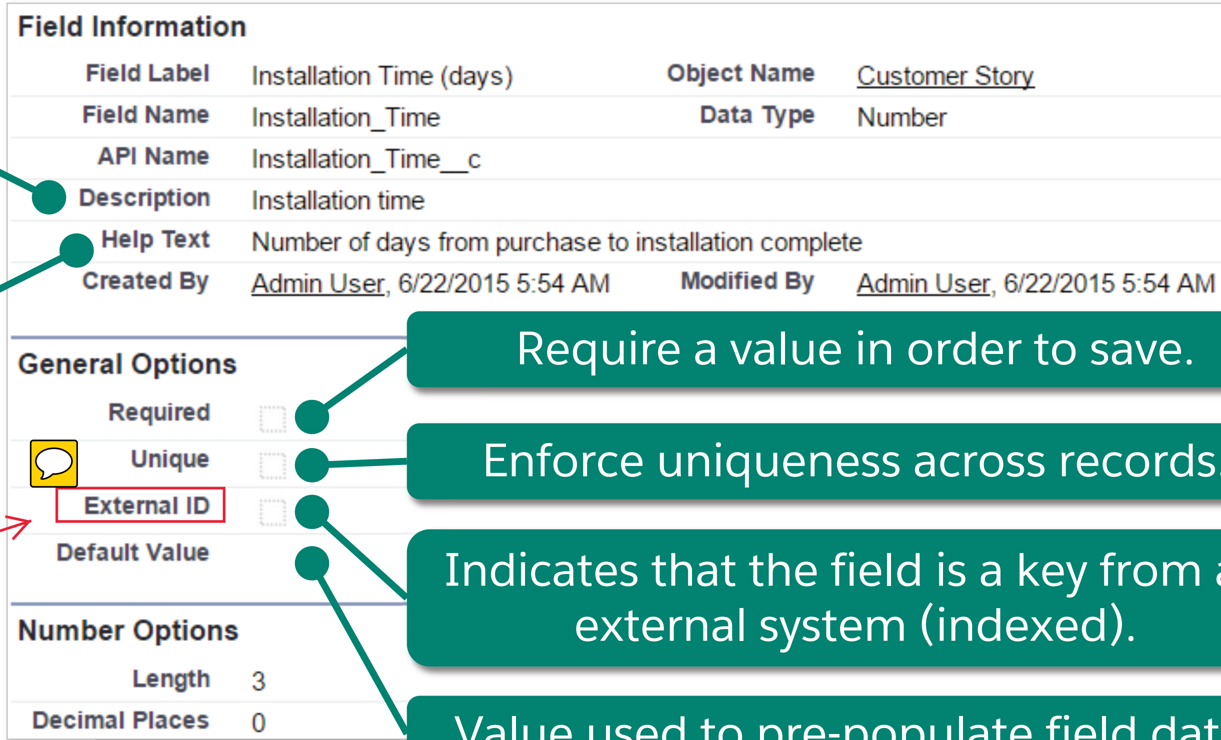
Name used in code and by the API.

Automatically generated by the system based on field name when the definition is saved.

OTHER FIELD ATTRIBUTES

Description (administrative purposes).

Text displayed when users hover over the Info icon.



Require a value in order to save.

Enforce uniqueness across records.

Indicates that the field is a key from an external system (indexed).

Value used to pre-populate field data.

Pour lier aux systèmes externes, permettre la connexion.
Utile pour les opérations d'upsert aussi.

NOTE:

The field attributes vary depending on the field type.

JOIN ME

2-2: CREATE CUSTOM FIELDS

15 minutes

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.

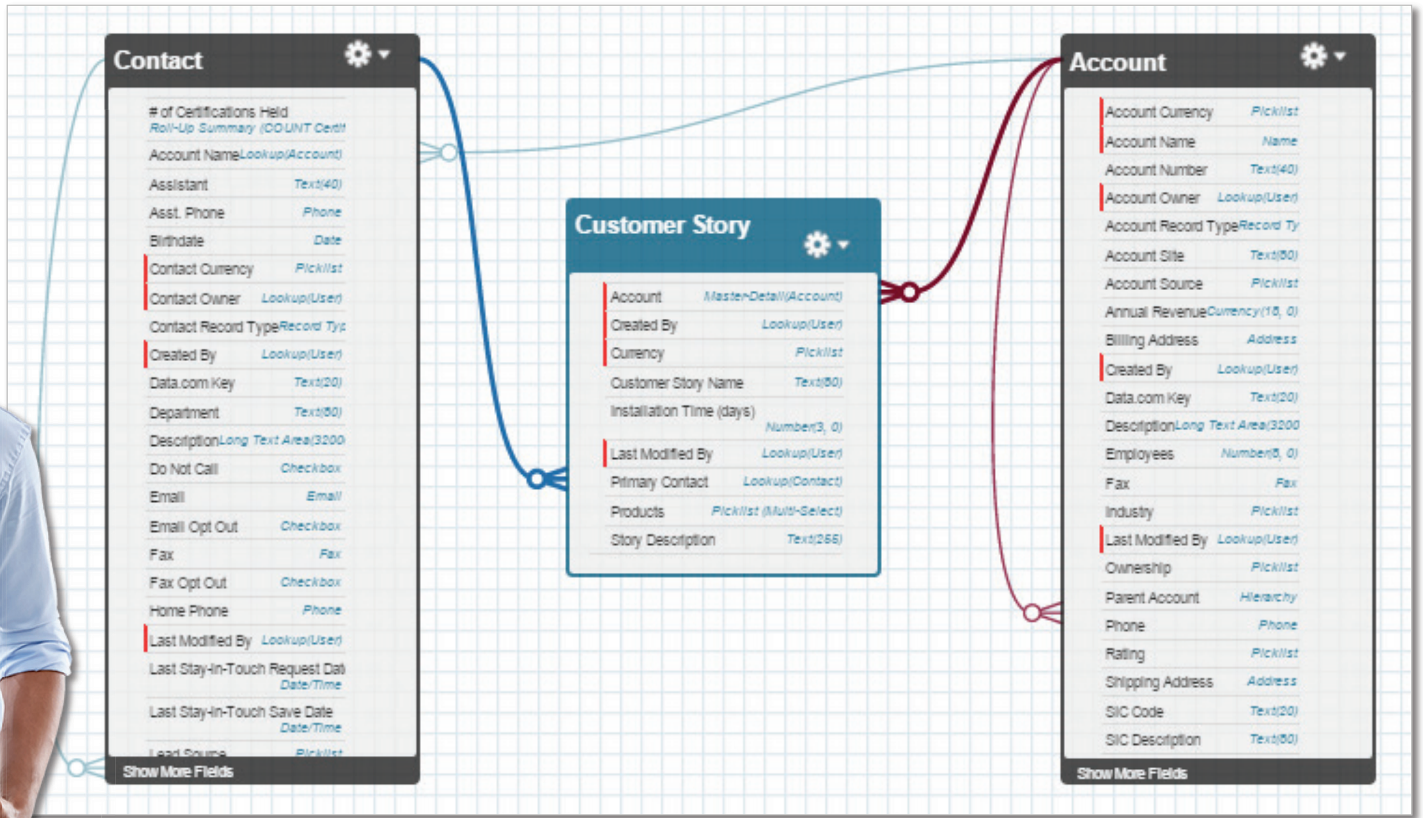
MODULE AGENDA

MODULE 2: BUILDING OBJECTS AND FIELDS

- Understanding Objects on the Force.com Platform
- Creating Custom Objects
- Creating Custom Fields
- **Creating Relationships Between Objects**

We need you to create relationship fields to associate the account and primary contact with each customer story.

Jason Beck
Developer



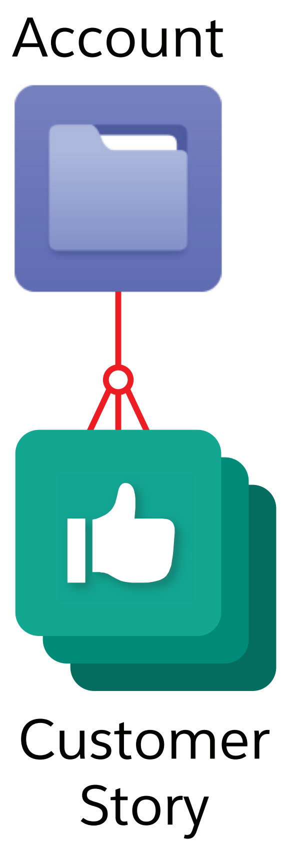
You can create a one-to-many relationship between two objects.

The relationship is defined on the child object using a custom field.

These are the types of relationship fields:

- Master-Detail
- Lookup

The Owner field on the detail and subdetail records is not available, and is automatically set to the owner of the master record.



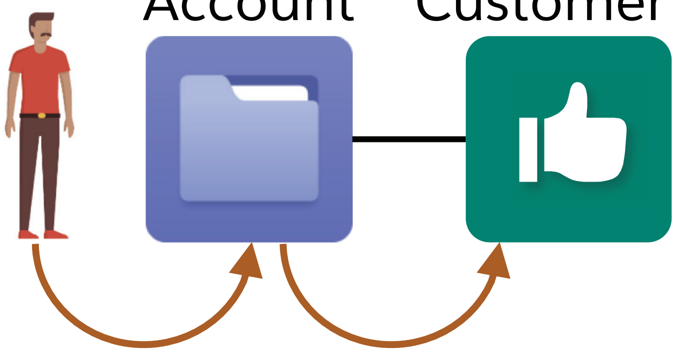
The platform also supports Hierarchical Relationships, which is a special lookup relationship available only for the user object. It allows users to use a lookup field to associate one user with another that does not directly or indirectly refer to itself. For example, you can create a custom hierarchical relationship field to store each user's direct manager.

Custom objects on the “detail” side of a master-detail relationship can't have sharing rules, manual sharing, or queues, as these require the Owner field

MASTER-DETAIL RELATIONSHIPS

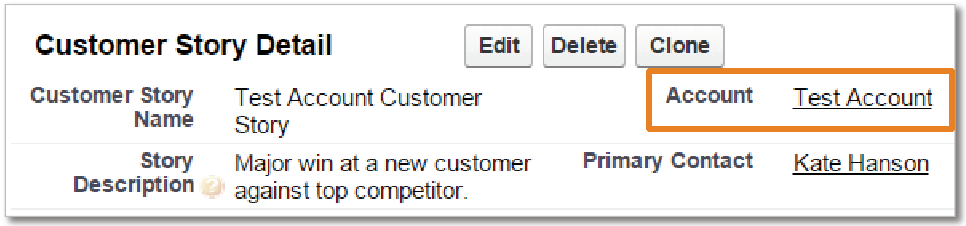
- Access to a detail record is inherited from the master record.

Account Customer Story



- The detail record is automatically deleted when the parent is deleted.

- The parent reference is always required on the child record.



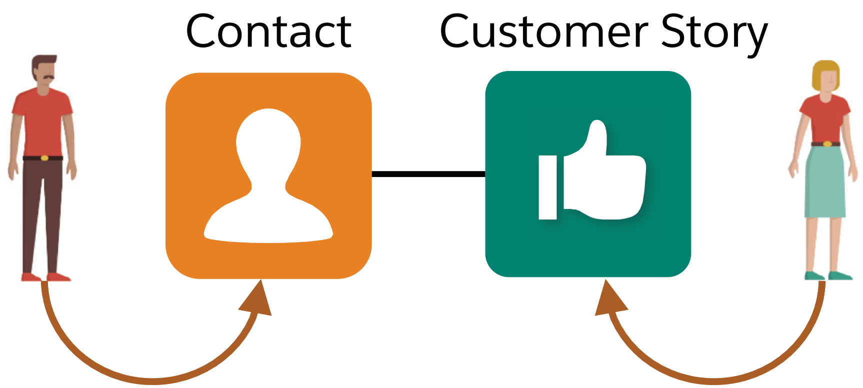
- You can add a lookup filter.
- You can choose whether or not the detail record can be reparented.

NOTE:

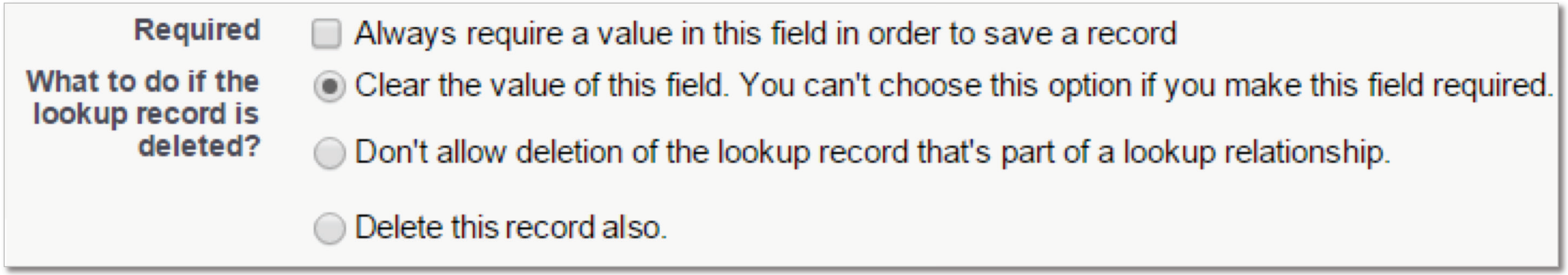
The detail side of a master-detail relationship must be a custom object.

LOOKUP RELATIONSHIPS

- The child record and parent record have independent sharing.



- The lookup field on the child record can be optional or required.



Required ☐ Always require a value in this field in order to save a record

What to do if the lookup record is deleted?
- ◉ Clear the value of this field. You can't choose this option if you make this field required.
- ○ Don't allow deletion of the lookup record that's part of a lookup relationship.
- ○ Delete this record also.

- You can add a lookup filter.

Deletion du Master-side :
- Soit on vide le champs lookup dans le detail
- Soit on interdit la deletion
- soit on efface le detail aussi

JOIN ME

2-3: CREATE RELATIONSHIP FIELDS

20 minutes

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.

LOOKUP VS. MASTER-DETAIL

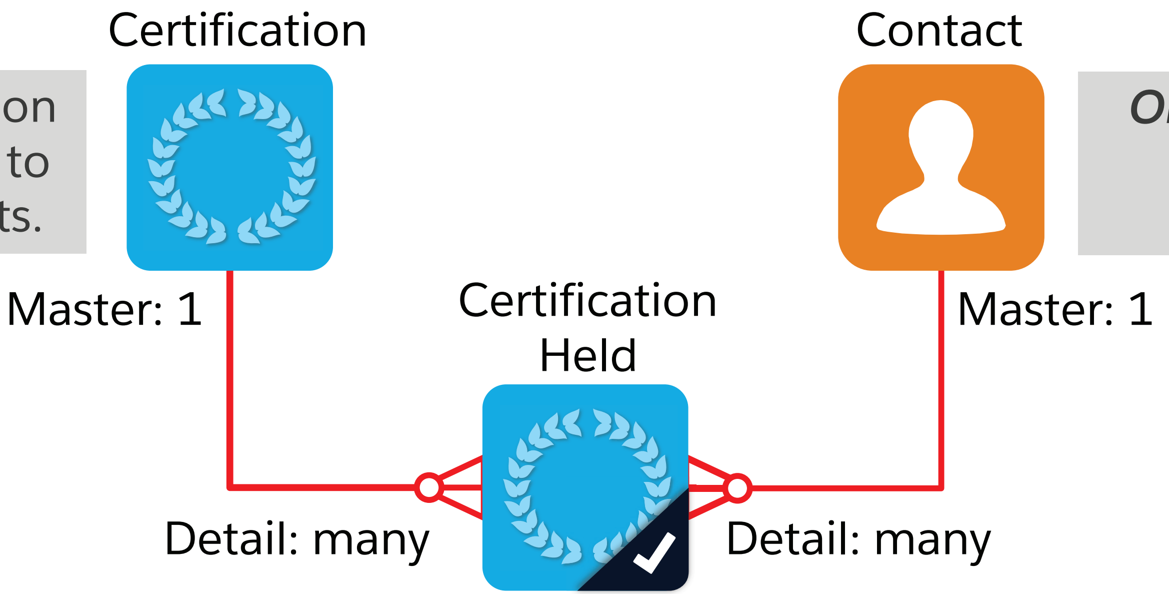
Lookup Relationships	Master-Detail Relationships
Parent field on child can be required or optional.	Parent field on child is required.
No impact on security and access.	Access to parent determines access to children.
If parent field is *required*, you cannot delete parent when referenced by child. If parent field is *optional*, choose one of three delete behaviors.	Deleting parent automatically deletes children.
Lookup field on page layout depends on required/optional choice.	Lookup field on page layout is required.

WHAT IS A JUNCTION OBJECT?

DEFINITION:

A **junction object** is a custom object with two master-detail relationships. It allows you to model a “many-to-many” relationship between two objects.

One certification can be linked to ***many*** contacts.



One contact can be linked to ***many*** certifications.

Junction object that can link multiple certification records to multiple contact records.

METADATA VS. DATA

Metadata contains the information about the look and feel of the application, along with its functionality.

Data is the value of fields in the records.

Custom Fields & Relationships New | Field Dependencies

Action	Field Label	API Name	Data Type
Edit | Del	Account	Account__c	Master-Detail(Account)
Edit | Del	Installation Time (days)	Installation_Time__c	Number(3, 0)
Edit | Del	Primary Contact	Primary_Contact__c	Lookup(Contact)
Edit | Del | Replace	Products	Products__c	Picklist (Multi-Select)
Edit | Del	Story Description	Story_Description__c	Text(255)

Metadata

Customer Story Detail Edit | Delete | Clone

Customer Story Name	Test Account Customer Story	Account	Test Account
Story Description	Major win at new customer against top competitor,	Primary Contact	Kate Hanson
Products	Laptops; Networking Equipment; Servers		
Installation Time (days)	30		

Data

This configuration data (such as field label, API Name, and Data Type) is known as metadata.

WHAT DO YOU GET WITH AN OBJECT?

Objects on the Force.com platform:

- Provide a predefined set of **standard fields** to capture common business information.
- Allow you to create **custom fields** to capture additional business information.
- Allow you to create **custom relationships** to link objects together.
- Allow you to create **validation rules** to verify that the data in one or more fields meets the specified criteria before the record is saved.
- Allow you to define **page layouts** and **record types** to control what a user sees when they view or edit a record.
- Allow you to automate business processes using **workflow rules**, **processes**, **flows**, and **approval processes**.
- Allow you to control **record access** and **field-level security**.

KEY TAKEAWAYS

- Objects represent database tables that contain your organization's information.
- Objects created by Salesforce are called standard objects.
- A custom object is an object you create to capture and manage additional data based on your specific business requirements.
- Object access determines which objects users can view and edit.
- Record access determines which individual records users can view and edit in each object on which they have been granted appropriate permissions.
- Standard and custom fields store data on individual records.
- Create lookup or master-detail relationships to model one-to-many relationships in Salesforce.
- Use junction objects to model many-to-many relationships.

1. How can a developer customize an object on the Force.com platform? Schema builder or object manager
2. How can a developer reference a field programmatically? Le nom API (obj__c)
3. What is a consideration for creating a master-detail relationship between two objects? Notion de visibilité, ou pour utiliser les roll-up summaries
4. What is a consideration for creating a lookup relationship between two objects?

- An object comes with a predefined set of standard fields. You can create **custom fields** to **capture additional business information** and link objects together. You can **create validation rules** to ensure valid data. You can **define page layouts** and **record types** to control what a user sees when they view or edit a record. You can also automate business processes and control record access and field-level security.
- A developer can use the **API Name** to reference a field programmatically.
- When creating a master-detail relationship between two objects, the master record controls **access to detail records**, the **related lookup field is a required field** on the detail record, and the **detail record is deleted** when the master record is deleted.
- When creating a lookup relationship between two objects :

- the child record and parent record have **independent sharing**
- the **parent record in a lookup relationship cannot be deleted it the parent field is required**
- a developer can choose one of three delete behaviors **if the parent field is optional**, and a developer can add **a lookup filter** to limit the records available to users.

TRAILHEAD HOMEWORK

Developer Beginner | Salesforce Platform Basics
(1 hour, 35 minutes)

Developer Beginner | Data Modeling
(1 hour)

MODULE 3:

WORKING EFFECTIVELY WITH OBJECTS AND FIELDS

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

Cassie Evans

Developer

We need you to understand how to use the features of the Force.com platform to build efficient and effective applications.

To accomplish this, you need to:

- Create formula fields.
- Create roll-up summary fields.
- Describe the capabilities of record types.

TAKING ADVANTAGE OF DECLARATIVE CUSTOMIZATION

The declarative customizations in this module are important tools that make programmatic solutions simpler and more efficient.

- Let Salesforce do the work for you!
- Don't reinvent the wheel.

MODULE 3: WORKING EFFECTIVELY WITH OBJECTS AND FIELDS

- **Creating Formula Fields**
- Creating Roll-Up Summary Fields
- Understanding Record Types
- Building a Data Model on the Force.com Platform

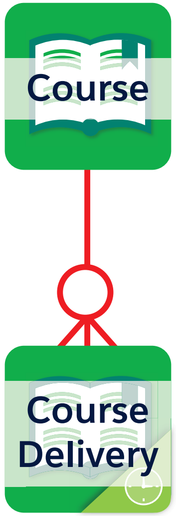
Story

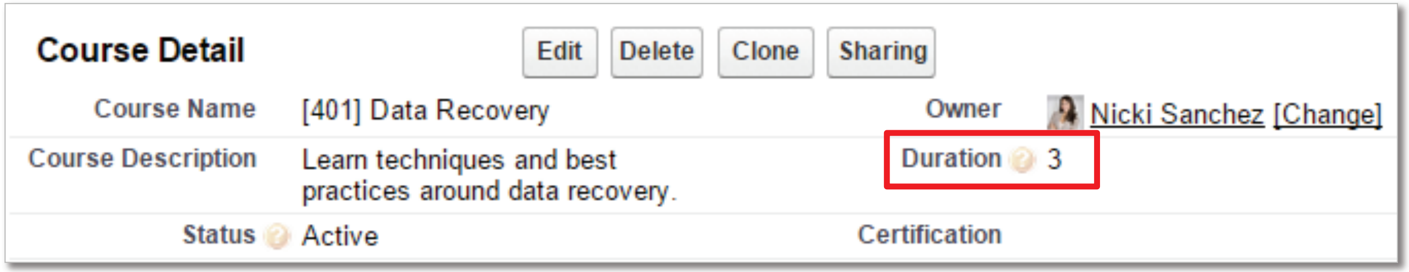
DERIVING DATA

Cassie Evans

Developer

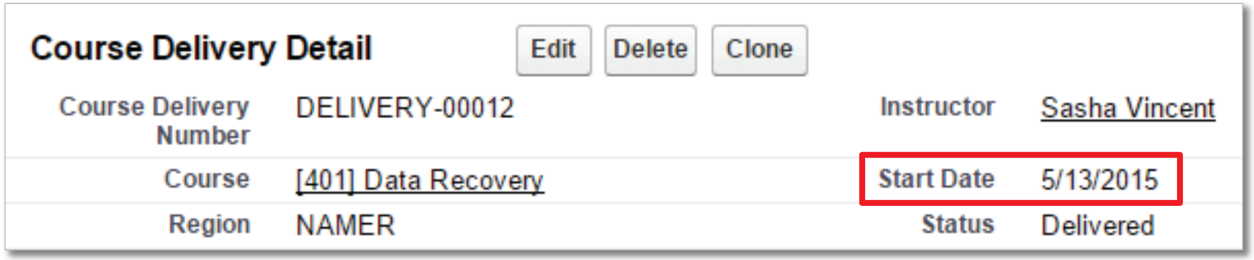
We need you to track the end date for course deliveries based on the start date and duration of the course.





Course Detail Edit Delete Clone Sharing

Course Name	[401] Data Recovery	Owner	Nicki Sanchez [Change]
Course Description	Learn techniques and best practices around data recovery.	Duration	3
Status	Active	Certification	



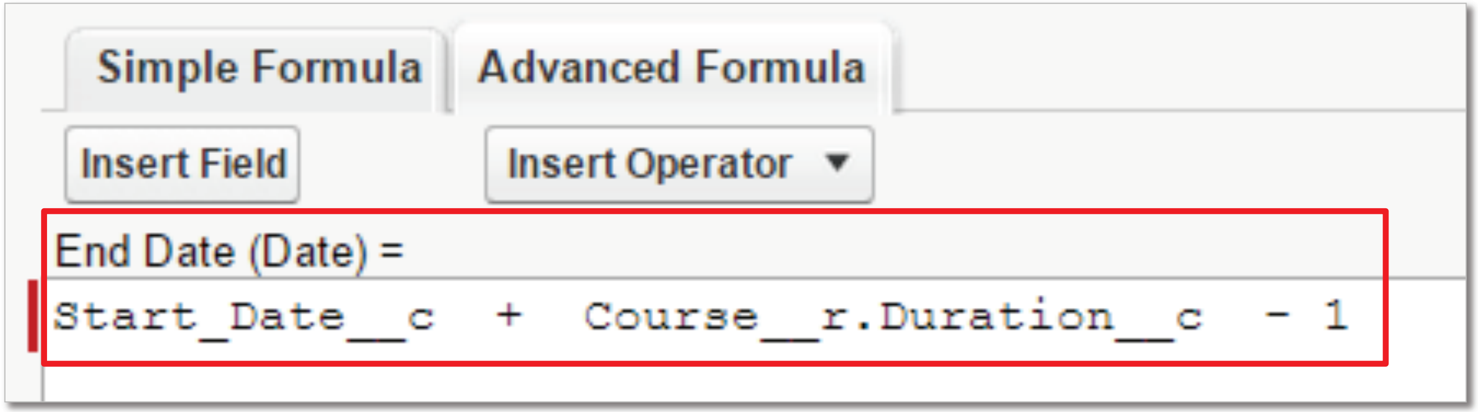
Course Delivery Detail Edit Delete Clone

Course Delivery Number	DELIVERY-00012	Instructor	Sasha Vincent
Course	[401] Data Recovery	Start Date	5/13/2015
Region	NAMER	Status	Delivered

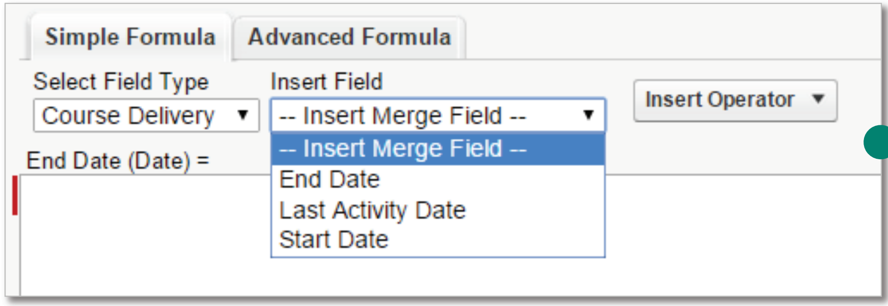
WHAT IS A FORMULA FIELD?

DEFINITION:

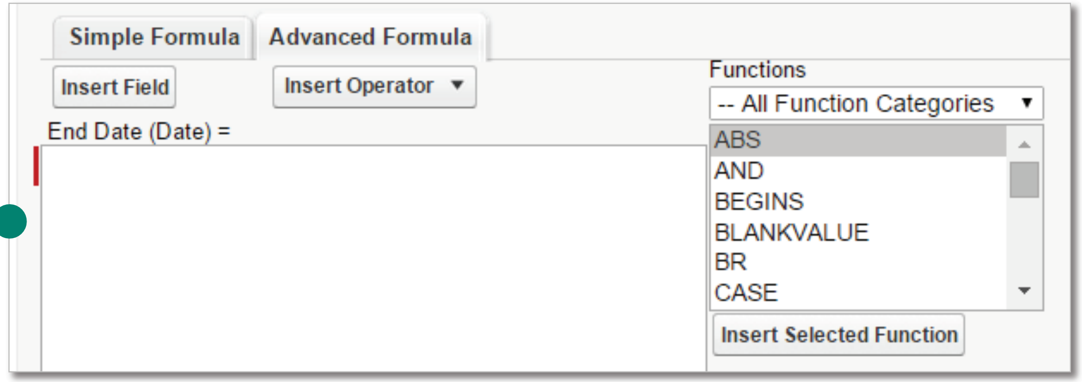
A **formula field** is a field that derives its value from other fields, expressions, or values.



On ne peut pas rechercher un champs formule (puisqu'il est calculé à la volée



The Simple Formula editor lets you create basic calculations involving fields from the same object or global variables.



The Advanced Formula editor lets you create complex calculations involving fields from parent objects.

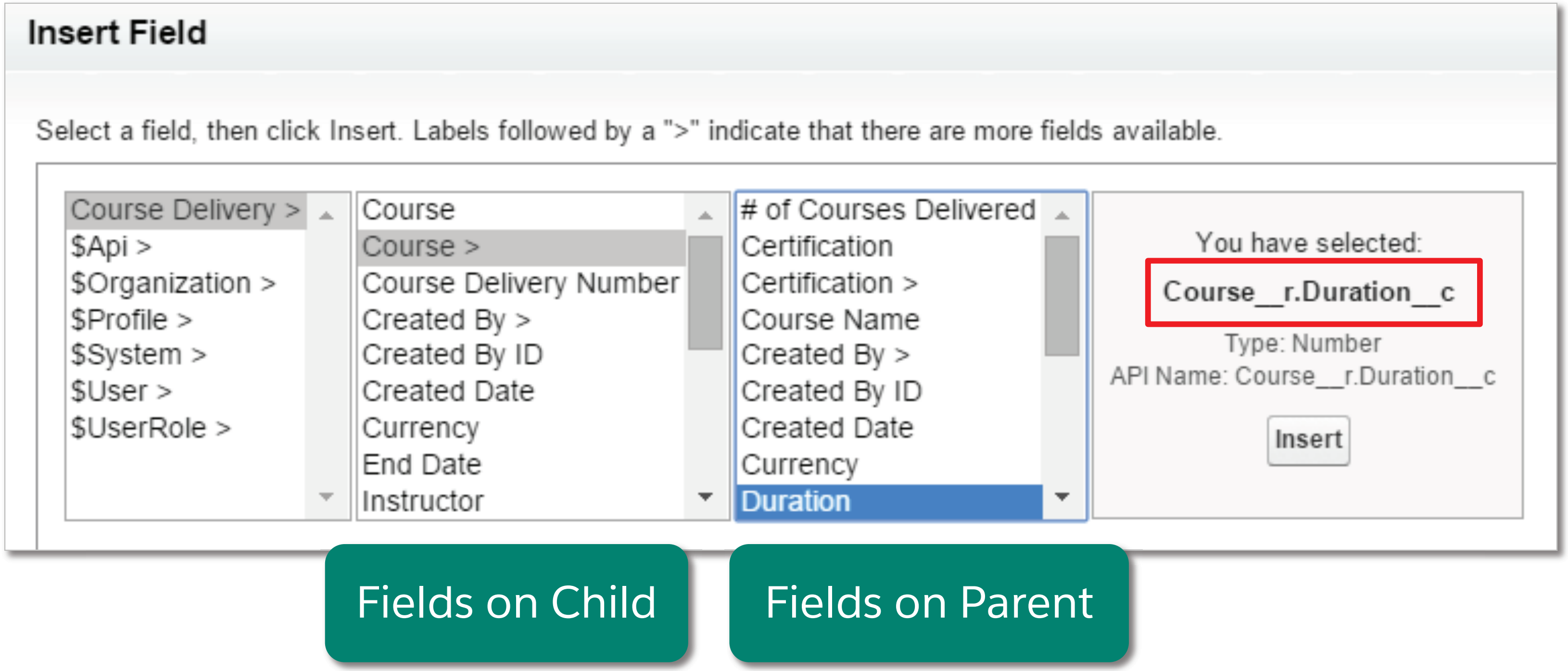
toujours insérer avec l'éditeur, pour éviter fautes de frappe.

NOTE:

Schema Builder only supports the Simple Formula editor.

CROSS-OBJECT FORMULAS

You can create a cross-object formula on a child object to reference data from parent objects, up to 10 relationships away.



JOIN ME

3-1: CREATE A FORMULA FIELD

salesforce

10 minutes

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.

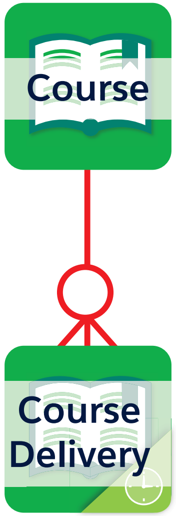
MODULE 3: WORKING EFFECTIVELY WITH OBJECTS AND FIELDS

- Creating Formula Fields
- **Creating Roll-Up Summary Fields**
- Understanding Record Types
- Building a Data Model on the Force.com Platform

Cassie Evans

Developer

We need you to track the number of times a course delivery was cancelled so we can calculate the course cancellation rate.



Course Deliveries

[102] AWCA Network

Course: [102] AWCA Network

Printable View | Help for this Page

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other |

Action	Course Delivery Number ↑	Region	Location	Start Date	Status	Instructor
Edit | Del	DELIVERY-00003	APAC	Singapore, SG	2/4/2015	Cancelled	Kim Tran
Edit | Del	DELIVERY-00004	EMEA	London, GB	1/28/2015	Delivered	Heidi Rosen
Edit | Del	DELIVERY-00005	NAMER	Chicago, US	12/31/2014	Delivered	Sasha Vincent
Edit | Del	DELIVERY-00018	APAC	Singapore, SG	4/22/2015	Delivered	Kim Tran
Edit | Del	DELIVERY-00024	NAMER	Toronto, CA	6/10/2015	Cancelled	Patrick Hughes
Edit | Del	DELIVERY-00026	EMEA	Berlin, DE	7/1/2015	Scheduled	Eugene Peters

WHAT IS A ROLL-UP SUMMARY FIELD?

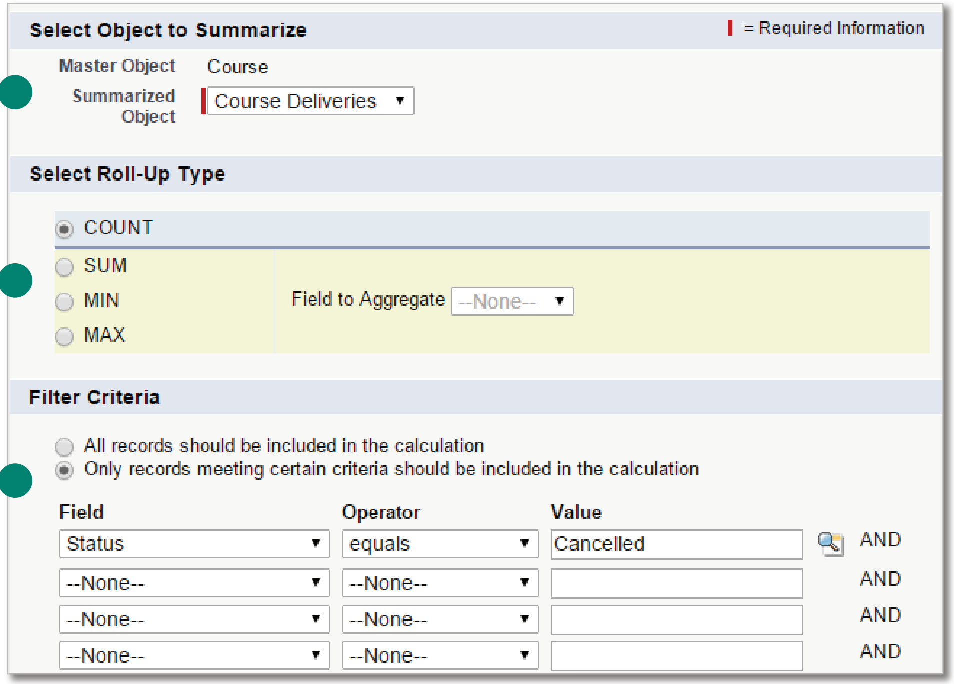
DEFINITION:

A **roll-up summary field** is a field on a master record that summarizes date or numerical data from detail records.

Select the detail object to summarize.

Set the roll-up type to count, sum, min, or max.

Determine which records to include in the calculation.



You can reference roll-up summary fields in a formula field, which allows you to perform calculations on the summarized data.

RESOURCE:

For considerations and best practices, search for `Roll-Up Summary Field` in Help & Training.

Intérêt : pas de code, pas de maintenance, et mis-à-jour à chaque ajout d'un detail-side.

Déclenche trigger, validation rules, workflow processes.

JOIN ME

3-2: CREATE A ROLL-UP SUMMARY FIELD

10 minutes

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.

YOUR TURN

3-3: CREATE A FORMULA FIELD THAT REFERENCES ROLL-UP SUMMARY FIELDS

salesforce

10 minutes

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.

MODULE AGENDA

MODULE 3: WORKING EFFECTIVELY WITH OBJECTS AND FIELDS

- Creating Formula Fields
- Creating Roll-Up Summary Fields
- **Understanding Record Types**
- Building a Data Model on the Force.com Platform

CAPTURING DIFFERENT TYPES OF DATA

We have two different types of accounts: customer and service vendor. We need to track some information that is common to both types and some information that is type-specific. You need to understand how to accomplish this on the Force.com platform.

Common Information	Customer-Specific	Service Vendor-Specific
Account Owner	Industry	Status
Account Name	Related Opportunities	
Parent Account	Related Cases	
Phone		
Website		
Support Level		
Employees		
Annual Revenue		
Billing & Shipping Address		
Related Contacts		

Cassie Evans
Developer

WHAT IS A PAGE LAYOUT?

DEFINITION: A **page layout** controls the fields, sections, related lists, buttons, and custom links that appear when a user views or edits a record.

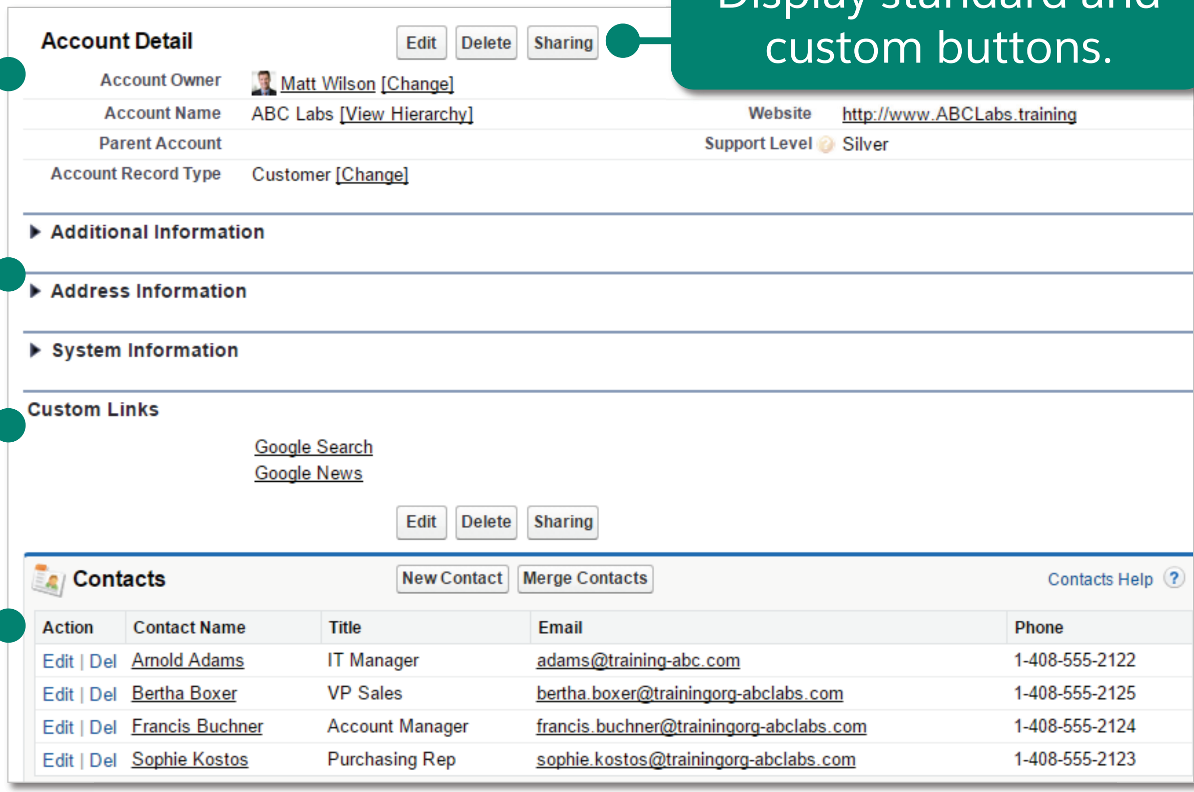
Add, remove, and move fields. Make fields read-only or required.

Create and move sections.

Display custom links.

Add, remove, and move related lists. Change which columns are displayed.

Display standard and custom buttons.

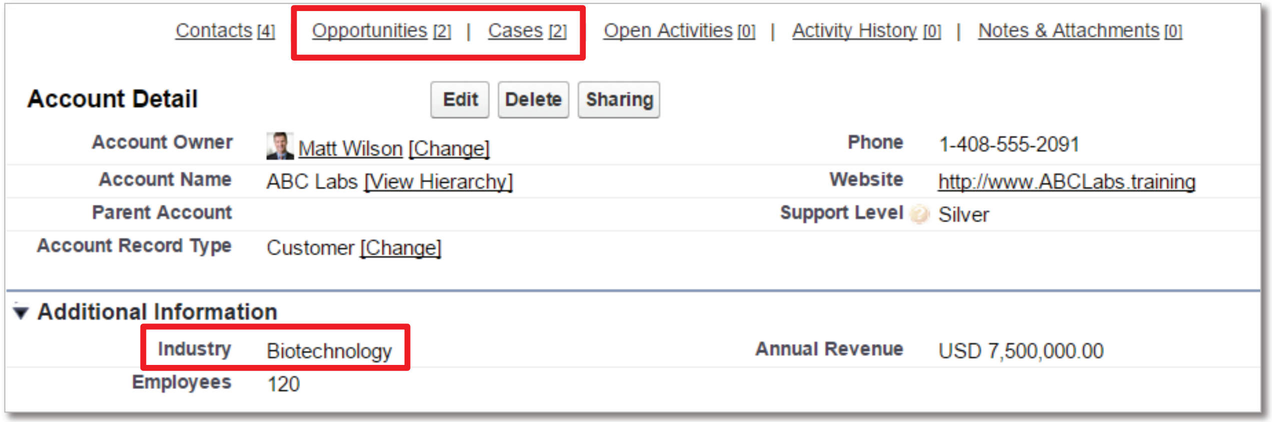


WHAT IS A RECORD TYPE?

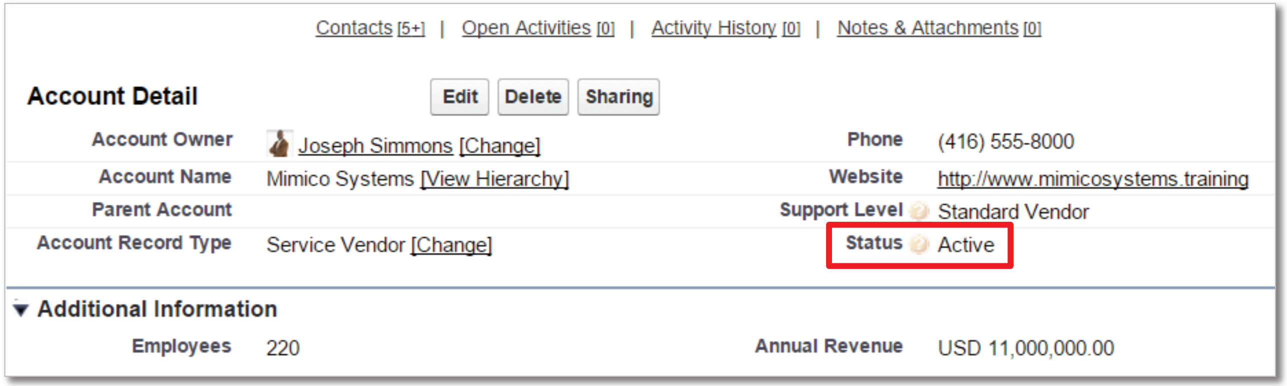
DEFINITION:

A **record type** enables a single user to view multiple page layouts for records of the same object.

Customer



Service Vendor

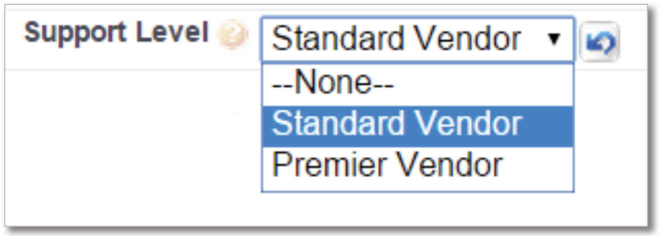


SPECIFYING PICKLIST VALUES

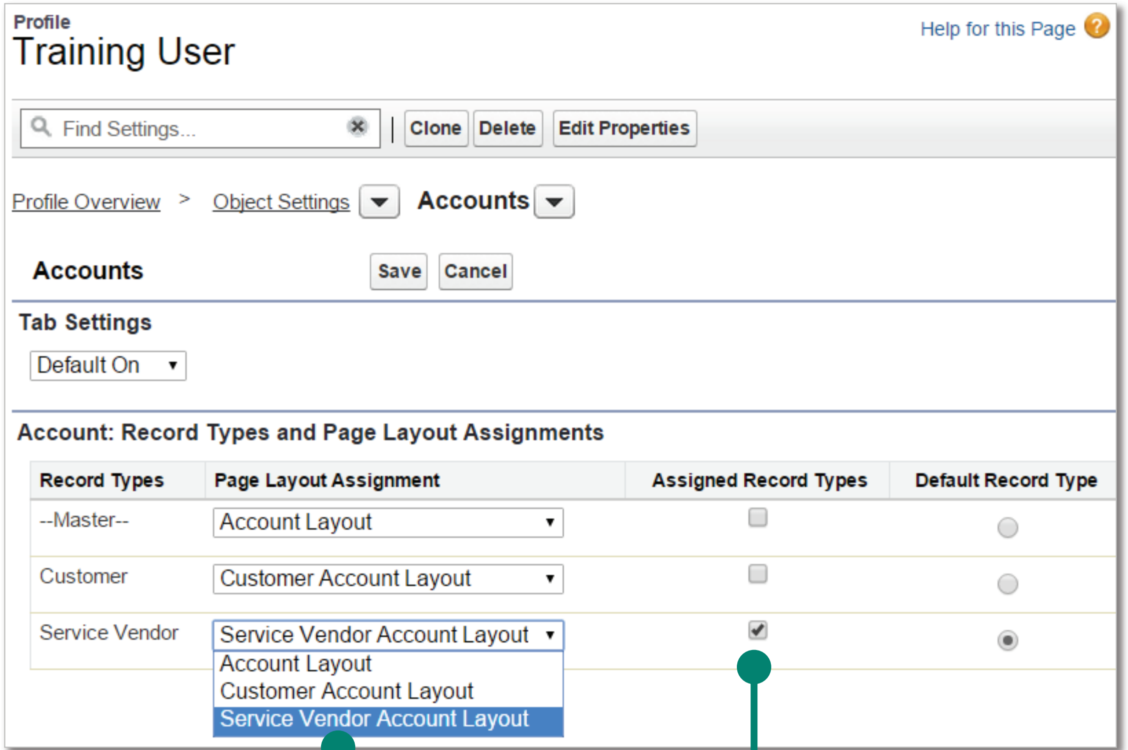
You can specify different picklist values for each record type.

Customer

Service Vendor



ASSIGNING RECORD TYPES AND PAGE LAYOUTS



Page layout assignment determines which page layout a user sees when viewing a record of a given record type.

Record type assignment determines which record types a user can select when creating a new record.

WATCH ME

3-4: UNDERSTAND RECORD TYPES

10 minutes

Goal:
Understand the capabilities of record types.

Tasks:

1. View the account records for each record type.
2. Create an account record using the Service Vendor record type.

MODULE AGENDA

MODULE 3: WORKING EFFECTIVELY WITH OBJECTS AND FIELDS

- Creating Formula Fields
- Creating Roll-Up Summary Fields
- Understanding Record Types
- **Building a Data Model on the Force.com Platform**

USE CASE #1

salesforce

As part of the Certification app, professional services needs to track this information about service technicians:

- Name
- Account Name
- Title
- Phone
- Mobile
- Other Phone
- Email
- Status (active or inactive)

How would you meet this requirement?

	Option A	Option B
Object	Use the standard Contact object	Create a custom object called `Technician`
Custom Fields	Only the Status field	All fields except Name
Relationships	None	A relationship from Technician to Account
Page Layouts & Record Types	2 page layouts 2 record types	None

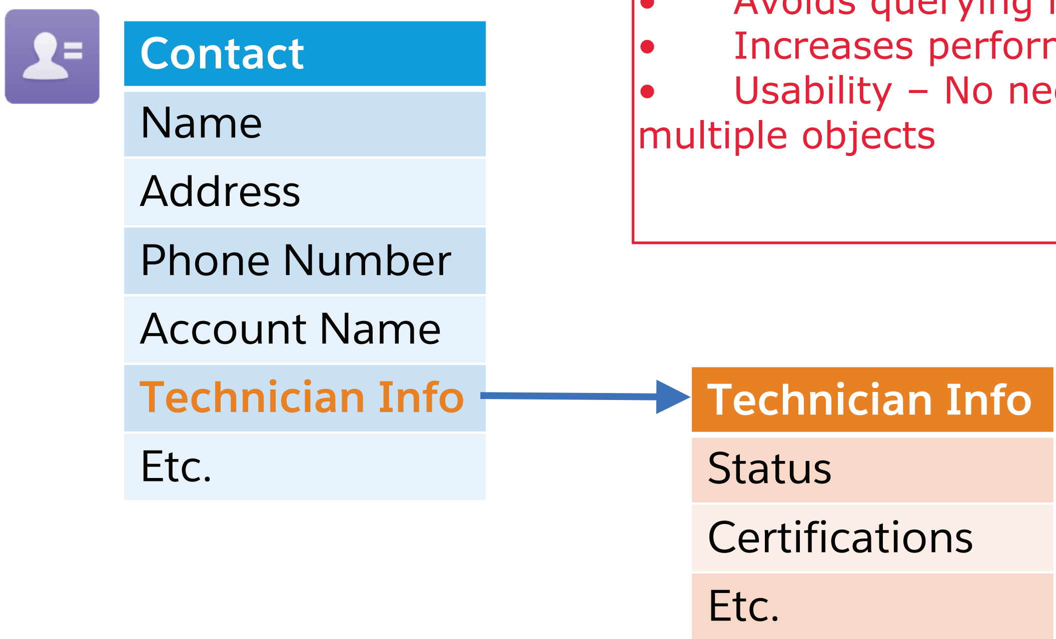
Licences communities pour les clients, licences moins chères,
Comme elles se basent sur l'objet CONTACT c'est beaucoup plus avantageux de faire sur Contact (obj standard).

C'est un moyen de donner à un technicien l'accès à notre plateforme.
Donc la bonne pratique est la A, d'autres raisons aussi (dont maintenance)

De plus, intéressant pour ne pas dupliquer les traitements éventuellement

Why not solve Use Case #1 using a separate object called "Technician Info" that is related to Contact using a lookup field?

- Performance!
- Avoids querying multiple objects
- Increases performance on reports
- Usability – No need to work on multiple objects



USE CASE #2

As part of the Certification app, professional services needs to track the region and location of each course delivery.

The supported regions are: APAC, EMEA, and NAMER.

The supported locations are:

- Tokyo, JP
- Singapore, SG
- Paris, FR
- London, GB
- Berlin, DE
- San Francisco, US
- Chicago, US
- Toronto, CA

How would you meet this requirement?

Region
APAC
EMEA
NAMER

Location
Tokyo, JP
Singapore, SG
Paris, FR
London, GB
Berlin, DE
San Francisco, US
Chicago, US
Toronto, CA

What if the professional services team also needed to track room capacity, site coordinator, and handicap accessibility?

Custom Object: Location

Custom Fields: Region, Room Capacity, Coordinator, Accessibility

KEY TAKEAWAYS

salesforce

- A formula field derives its value from other fields, expressions, or values (which prevents the need to manually duplicate data).
- Formula fields are calculated on retrieval.
- A roll-up summary field is a field on a master record that summarizes date or numerical data from detail records.
- A page layout controls the fields, sections, related lists, buttons, and custom links that appear when a user views or edits a record.
- Record types allow you to specify different page layouts and picklist values for different types of records.

Le roll up summary permet d'afficher une stat des détails sur le layout du master, dans une relation master-detail

KNOWLEDGE CHECK

salesforce

1. What is a capability of formula fields?
2. What is a capability of roll-up summary fields? Max, Min, Count (pas AVERAGE !)
3. What do record types determine?
4. AW Computing uses the Certification Held object to track if a service technician contact has passed a certification. There is a master-detail relationship between the Contact object (master) and the Certification Held object (detail). The professional services team wants to track the total number of certifications held by each service technician contact. How can a developer meet this requirement?

   On est en master Detail sur Contact, on peut y faire un Count (rollup summary)
5. AW Computing populates the Industry field on customer account records. The sales team wants the industry information displayed on related opportunity records and updated when the value is updated on the account record. How can a developer meet this requirement?

TRAILHEAD HOMEWORK

Developer Intermediate | Formulas & Validations
(45 minutes)

Admin Advanced | Advanced Formulas
(2 hours, 55 minutes)

MODULE 4: PROGRAMMING WITH APEX

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

We need you to get up to speed on Apex as soon as possible.

To lay the foundation for understanding Apex, you need to:

- Describe key aspects of Apex that differentiate it from other languages, such as Java and C#.
- Describe why Apex transactions and governor limits must be considered when writing Apex.
- Execute simple Apex.
- Use the sObject data type, the primitive data types, and basic control statements in Apex.

MODULE 4: PROGRAMMING WITH APEX

- **Getting Started with Apex**
- What Makes Apex Different?
- Working with sObjects

JOIN ME

4-1: LOGGING IN TO A SANDBOX

10 minutes

Goal:

Explore your sandbox.

Tasks:

1. Log in to your sandbox.
2. Review the data.
3. Review user records.
4. Log out of your sandbox.

JOIN ME

4-2: SEE APEX IN ACTION

salesforce

15 minutes

Goal:

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

Tasks:

1. Open the Developer Console.
2. Open Execute Anonymous.
3. Enter the execute code.
4. Examine the logs.
5. Examine the result in the user interface.

EXAMINING THE CODE

1. What is the outcome of executing this code?
2. What looks familiar in the code?
3. Which line of code shows Apex interacting with the database?

```
Contact contactToAdd = new Contact();
contactToAdd.firstName = 'June';
contactToAdd.lastName = 'Morgan';
insert contactToAdd;
System.debug('contactToAdd recordID is: ' + contactToAdd.id);
```

WHAT IS APEX?

DEFINITION:

Apex is Salesforce's cloud-based, object-oriented programming language, specifically designed for customizing and extending apps on the Force.com platform.

- Tailored for data access and manipulation.
- Works in conjunction with declarative features.
- Has access to your org's metadata.
- Designed to work effectively and efficiently in a multi-tenant environment.

APEX CLASSES AND TRIGGERS

All Apex code is saved in one of two forms:

DEFINITION:

class: Apex code with members such as inner classes, variables, methods, properties, etc.

DEFINITION:

trigger: Apex code whose execution is automatically triggered during the processing of DML events.

JOIN ME

4-3: CREATE AND USE AN APEX CLASS

salesforce

10 Minutes

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

Tasks:

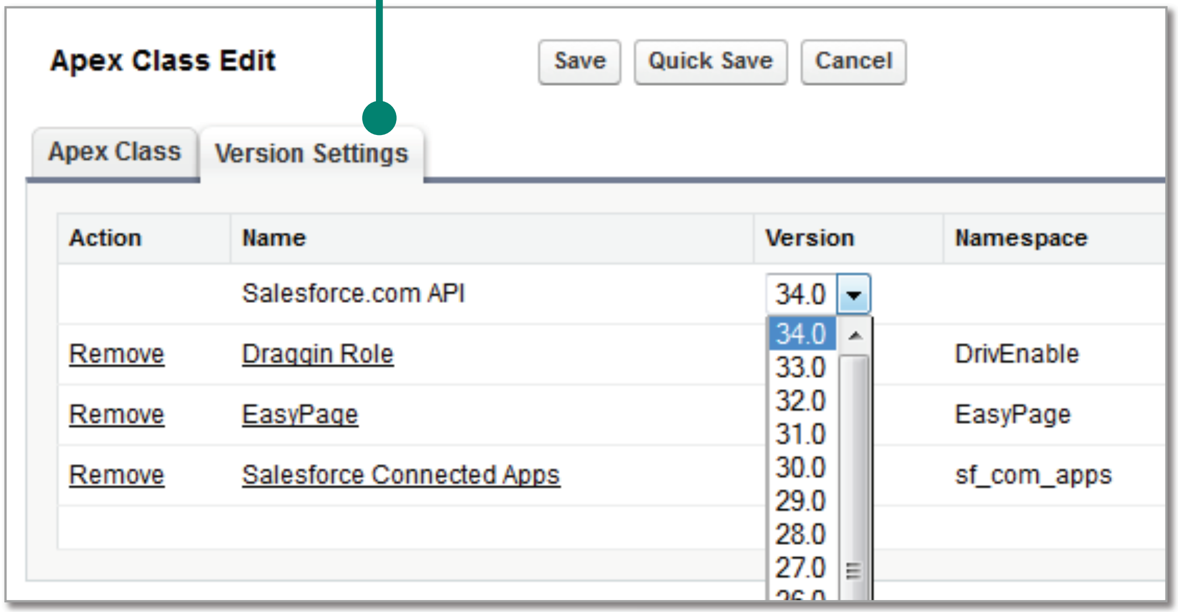
1. Create a class and save it.
2. Invoke the class method using code in the Execute Anonymous window.
3. Examine the logs to see the invocation of the class.

REVIEWING THE CODE

1. What do you think is the purpose of the API version when you saved your code?
2. What do you recognize in this code segment?
3. What is different from Java or C#?

```
public class ContactManager {
    public static ID addContact(String lastNameToInsert, String firstNameToInsert) {
        Contact contactToAdd = new Contact(firstName=firstNameToInsert,
                                           lastName=lastNameToInsert);
        insert contactToAdd;
        return contacttoadd.Id;
    }
}
```

Apex classes, triggers, Visualforce pages, and Visualforce components are saved with information about the specific Salesforce API version with which they will be compiled and executed.



ALERT:

Version Settings are not related to file versioning provided with tools such as source control tools.

WATCH ME

4-4: OBSERVE THE EFFECTS OF VERSIONING

5 minutes

Goal:

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

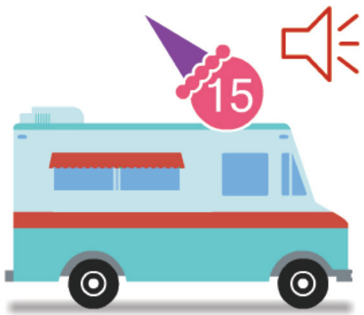
Tasks:

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.

VERSIONING DISCUSSION

1. How did Salesforce handle deprecating FeedPost and introducing FeedItem?

2. Why might a developer choose to update an existing Apex class to a later version?



3. How might this benefit Apex developers?

WHERE IS YOUR CODE?

- Successfully compiled code is saved as metadata to the Force.com database.
- The Apex environment retrieves the compiled code when it is needed.

NOTE:

When using execute anonymous, your successfully compiled code is not saved, but instead is immediately sent to the Apex environment for execution.

JOIN ME

4-5: TAKE A QUICK TOUR OF APEX

20 Minutes

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 class.
3. Run the test.
4. View code coverage.

When we load courses from our previous system, we want to filter the incoming courses to eliminate duplicates.

Jason Beck
Developer

DATA TYPES: PRIMITIVES

Blob	Decimal	Long
Boolean	Double	String
Date	ID	Time
Datetime	Integer	

```
//Sample String method
Boolean validString = userString.isAlphanumeric();

//Sample Datetime method
Date curUserDate = curDateTime.date();
```

Apex initializes all variables, regardless of type, to `null`.

NOTE:

Although called primitives in Apex, these data types are similar to wrapper classes in Java.

CONDITIONAL AND LOOPS

- Conditionals
  - If
  - If-else

- Loops
  - While
  - Do-while
  - For Loops
    - Traditional
    - List or set iteration
    - Iterate over SOQL result

```
1A  List<Account> accounts = new List<Account>();
2A  accounts.add(new Account(name='Account 1'));
3A  accounts.add(new Account(name='Account 2'));
4A  System.debug('First account is ' + accounts.get(0));
5A  System.debug('Second account is ' + accounts [1]);
```

An ordered, indexed collection of elements.

```
1B  Set<String> names = new Set<String>();
2B  names.add('Acme');
3B  names.add('Salesforce');
4B  names.add('Salesforce');
5B  System.debug('Does the set contain Pardot? ' + names.contains('Pardot'));
6B  System.debug('The size of the set is: ' + names.size());
```

Ecrase la valeur préexistante : pas d'erreur

An unordered collection of elements that does not contain duplicates.

```
1C  Map<String, Integer> counts= new Map<String, Integer>()
2C  counts.put('Acme', 200);
3C  counts.put('Salesforce', 400);
4C  counts.put('NewCorp', 200);
5C  counts.put('Acme', 600);
6C  System.debug('The size of the map is: ' + counts.size());
7C  System.debug('The count for Acme is ' +
                            (counts.containsKey('Acme')? counts.get('Acme'): 0));
```

Pareil, ça écrase

A collection of key value pairs where each unique key maps to a single value.

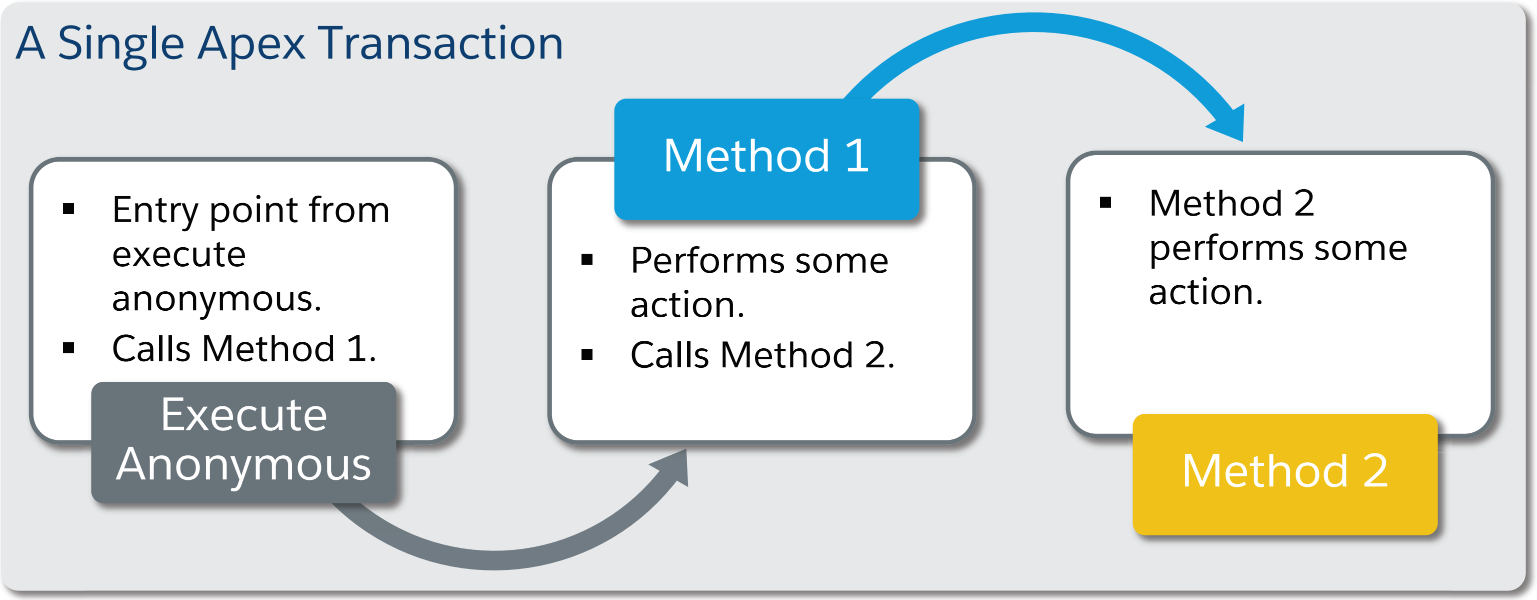
MODULE AGENDA

MODULE 4: PROGRAMMING WITH APEX

- Getting Started with Apex
- **What Makes Apex Different?**
- Working with sObjects

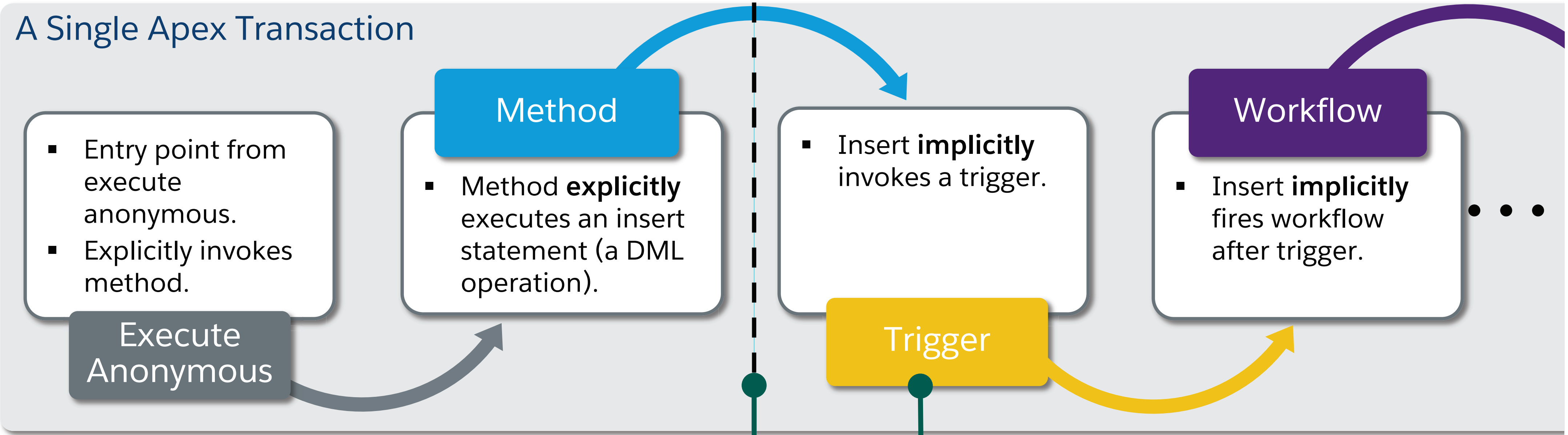
DEFINITION:

An **Apex transaction** represents a set of operations that are executed as a single unit.



APEX TRANSACTIONS THAT OPERATE ON THE DATABASE

An Apex transaction that contains the execution of a database-modifying statement also contains the execution of logic, such as a trigger, implicitly executed by that modification.



From this point, the system processes configured business logic, such as triggers and workflow, following the steps in the order of execution.

Cannot know when persisted code will be invoked, and not easy to determine who or what invoked it.

WATCH ME

4-6: EXAMINE IMPLICIT OPERATIONS

10 minutes

Goal:

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

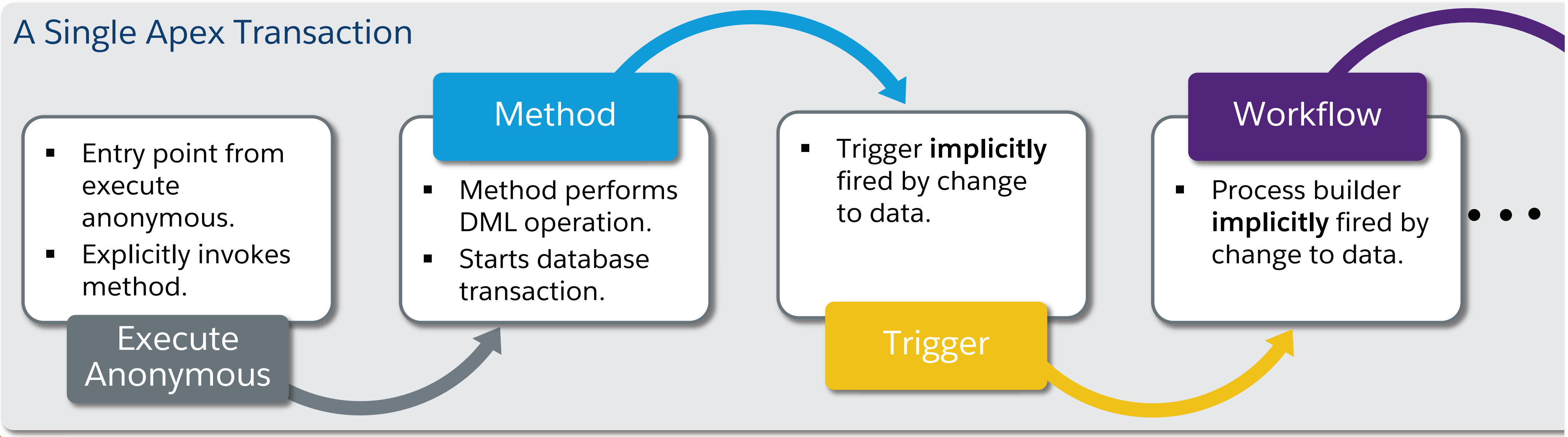
Tasks:

1. Create a Course Attendee record in the UI.
2. View logs to see operations implicitly invoked due to the DML operation.

WHAT ARE APEX GOVERNOR LIMITS?

DEFINITION:

All Apex execution is bound by **governor limits** that the system enforces on operations to ensure resources are available for all tenants.



Per Transaction: Queries Database operations Heap space CPU time Other…

ALERT:

If you exceed a governor limit, your code will terminate with an unhandleable and hence unrecoverable exception.

WATCH ME

4-7: PROFILE LIMITS USING THE DEVELOPER CONSOLE

salesforce

5 minutes

Goal:

Investigate limits using the Developer Console.

Task:

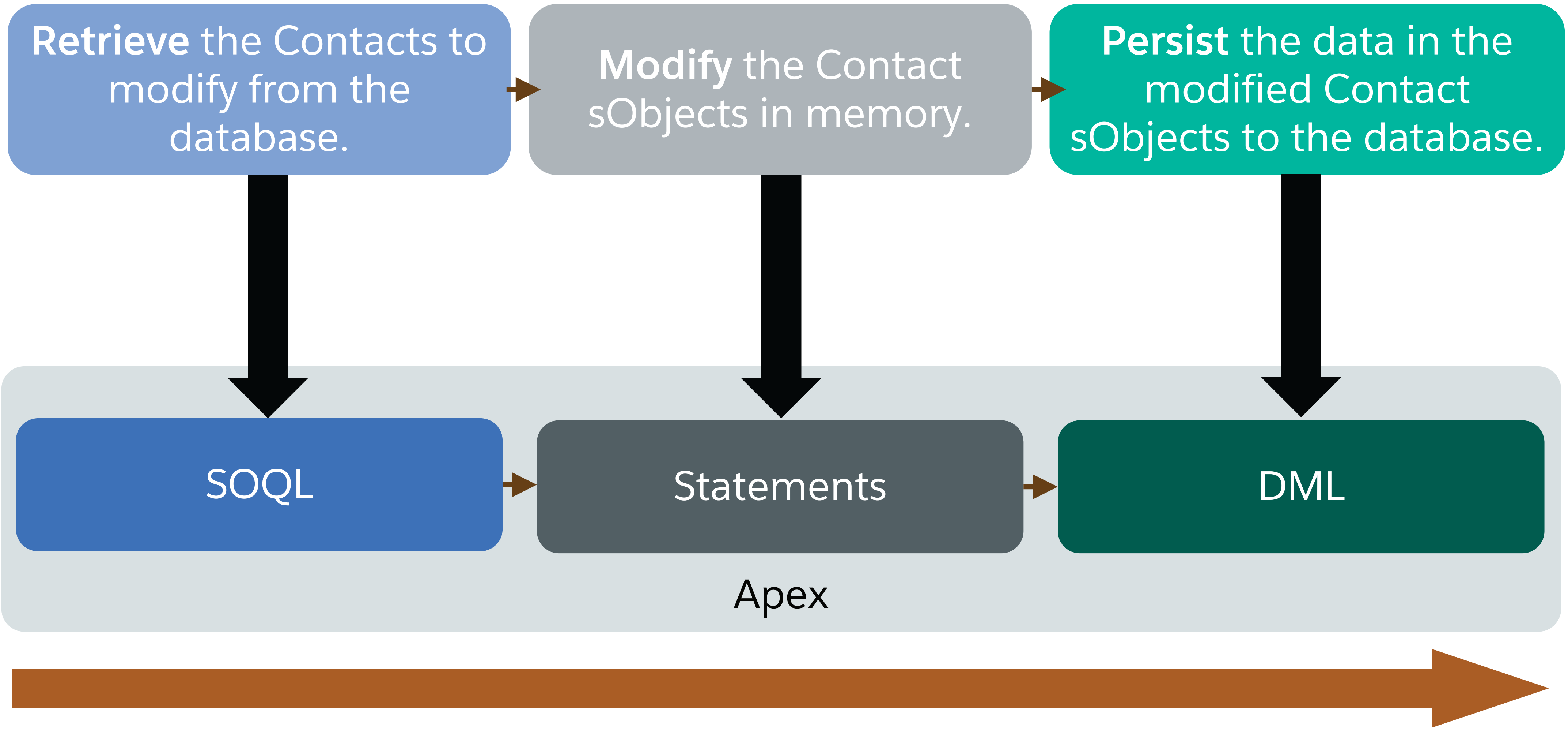
Review the limit profiling information in the log.

MODULE AGENDA

MODULE 4: PROGRAMMING WITH APEX

- Getting Started with Apex
- What Makes Apex Different?
- **Working with sObjects**

WORKING WITH DATA IN APEX: A COMMON PATTERN



WHAT IS AN sObject?

DEFINITION:

sObject is a generic abstract type that can be used to refer to any object that can be stored in the Force.com platform database.

```
Account acctToAdd = new Account(name = 'Feldman Associates');
insert acctToAdd;
```

```
public void updateLocation(List<Course_Delivery__c> deliveriesToCheck){
    List<Course_Delivery__c> deliveriesToUpdate = new List<Course_Delivery__c>();
    for (Course_Delivery__c delivery: deliveriesToCheck) {
        if (delivery.location__c == 'Tokyo') {
            delivery.region__c = 'JAPA';
            deliveriesToUpdate.add(delivery);
        }
    }
    update deliveriesToUpdate;
}
```

A data object provided or defined in the user interface is represented as an sObject in Apex.

COMMON TERMINOLOGY

A Salesforce administrator creates and modifies an Object declaratively.

Data is saved as a **record**.

Data is manipulated in memory as an **sObject**.

The configuration for the Object is stored as **Metadata**.

WHAT'S FAMILIAR IN THIS CODE SEGMENT?

1. What is this code segment declaring?
2. What do you recognize in this code segment?
3. What is different from Java or C#?

```
public void updateLocation(List<Course_Delivery__c> deliveriesToCheck){
    List<Course_Delivery__c> deliveriesToUpdate = new List<Course_Delivery__c>();
    for (Course_Delivery__c delivery: deliveriesToCheck) {
        if (delivery.location__c == 'Tokyo') {
            delivery.region__c = 'JAPA';
            deliveriesToUpdate.add(delivery);
        }
    }
    update deliveriesToUpdate;
}
```

DEFINITION:

Use the == operator for String equality as well as between two sObjects to determine if all fields are equal. The === operator, when used with sObjects, checks that the operands are referring to the same sObject.

sObjects AND THE DECLARATIVE CONFIGURATION

The Force.com platform tracks dependencies between sObjects used in Apex and declarative object definitions.

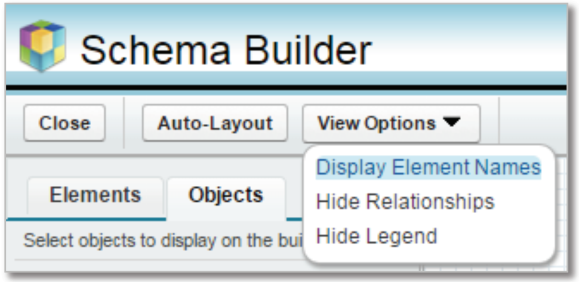
```
public void updateLocation(List<Course_Delivery__c> deliveriesToCheck){
    List<Course_Delivery__c> deliveriesToUpdate = new List<Course_Delivery__c>();
    for (Course_Delivery__c delivery: deliveriesToCheck) {
        if (delivery.location__c == 'Tokyo') {
            delivery.region__c = 'JAPA';
            deliveriesToUpdate.add(delivery);
        }
    }
    update(deliveriesToUpdate);
}
```

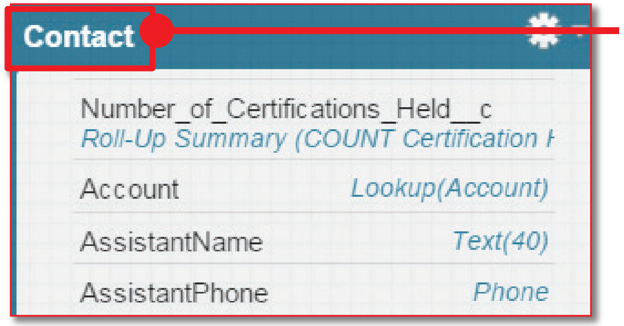
Custom Fields & Relationships New

Action	Field Label	API Name
Edit | Del	Course	Course__c
Edit | Del	Instructor	Instructor__c
Edit | Del | Replace	Location	Location__c

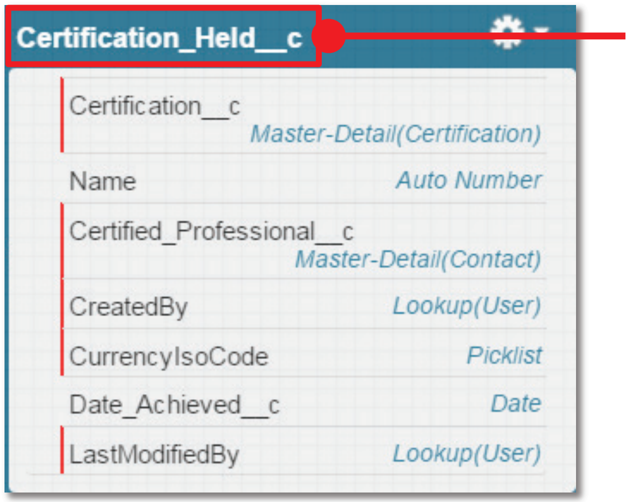
HOW IS AN sObject NAMED IN APEX?

The API name of an object is the name used in Apex to refer to an sObject.





- Standard object API names are often the same as the object name.



- Custom object names end with the `__c` suffix by default.

USING THE sObject API NAME

Example - Standard Object

```
1A Contact contactToAdd = new Contact(lastName='Jensen',firstName='Sam');
2A contactToAdd.firstName = 'John';
3A contactToAdd.lastName = 'Test2';
4A insert(contactToAdd);
5A System.debug('contactToAdd recordID is: ' + contactToAdd.Id);
```

When constructing an sObject, you have the option of specifying field values in the constructor.

Example - Custom Object

```
1B  public Contact UpdateCertificationHeld (Certification_Held__c userCert){
2B      …
3B  }
```

On which line…

1. … is `Contact` used as a variable's data type?
2. … is `Contact` the name of a constructor?
3. 1B is `Contact` used as a method's return data type?
4. 1B is `Certification_Held__c` used as a data type of a parameter?

REFERENCING FIELDS WITHIN AN sObject

DEFINITION:

The **API name of an object's field** is the name used in Apex to refer to a field within an sObject.

The names of custom fields end with the `__c` suffix in Apex by default.

```
Certification__c certificationToAdd
    = new Certification__c(Name='AWCI',
                           status__c = 'Active',
                           certification_description__c = 'AW Computing Certified Network');
insert certificationToAdd;
System.debug(' certificationToAdd recordID is: ' + certificationToAdd.Id);
```

```
List<Contact> contactsToAmend = [SELECT Id, Name, Status__c
                                  FROM Contact
                                  WHERE Email = ''];
```

A SOQL query enclosed in [] is an expression. Two possible return types are `sObject` and `List<sObject>`.

The sObject and field names in a SOQL query use the same names you use in Apex.

4-8: WORK WITH A CUSTOM OBJECT

15 Minutes

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.

REFERENCING A RELATIONSHIP FIELD ON A CHILD sObject

Each relationship field on a child object is represented by two fields of an Apex sObject.

A field acting as a foreign key holds the ID of the parent sObject record.

```
//Populated from the database.
List<Contact> contactsToModel = [SELECT id, AccountId, Account.Name
                                 FROM contact];
Contact contactToModel = contactsToModel[0];
//Have the new contact associated with the same account as contactToModel using
//the ID AccountId field.
Contact ContactToAdd = new Contact(AccountID=contactToModel.AccountId);

//Output the name of the Account just added to the new contact using the Account field
System.debug('Account for new contact is ' + contactToModel.Account.Name);
```

A field containing a reference to the parent sObject.

REFERENCING A RELATIONSHIP FIELD ON A CHILD sObject (CONT.)

salesforce

Standard Relationship

```
1A  Contact contactToChange = [SELECT AccountId, Account.Name FROM Contact LIMIT 1];
2A  Id oldAccountId = contactToChange.AccountId;
3A  Account oldAccount = contactToChange.Account;
4A  String accountName = contactToChange.Account.Name;
```

Id: `Child.ParentId`

Reference (sObject): `Child.Parent`

Custom Relationship

```
1B  Course__c courseToChange = [SELECT Certification__c, Certification__r.name
                                FROM Course__c LIMIT 1];
2B  Id oldCertificationId = courseToChange.Certification__c;
3B  Certification__c oldCertification = courseToChange.Certification__r;
4B  String certificationName = courseToChange.Certification__r.name;
```

Id: `Child.Parent__c`

Reference (sObject): `Child.Parent__r`

WHAT FIELDNAMES SHOULD YOU USE?

For each blank, what fieldname must be specified if you want to:

1. Select the name of the Certification Attempt?
2. Select the parent Certification Element's Id?
3. Select the parent Certification Element's Name?
4. Select the parent Contact's Name?
5. Select the parent Contact's associated Account Name?
6. Retrieve only records whose status is Scheduled?

```
List<Certification_Attempt__c> oldCertAttempts =
            [SELECT id, ___1____, ___2____, ___3____, ___4____, ___5____
             FROM Certification_Attempt__c
             WHERE ___6___ = 'Scheduled'];
```

WHAT DATA TYPE SHOULD YOU USE WHEN GOING FROM CHILD TO PARENT?

For each blank, what type must be specified?

```
if (oldCertAttempts.length() > 0) {
    Certification_Attempt__c certificationAttempt = oldCertAttempts[0];
    ___1_____ var1 = certificationAttempt.Certification_Element__c;
    ___2_____ var2 = certificationAttempt.Certification_Candidate__r;
    ___3_____ var3 = certificationAttempt.Certification_Candidate__r.Account;
}
```

2 : un Contact

3 : Un account

REFERENCING A RELATIONSHIP FIELD ON A PARENT sObject

Standard Relationship

```
1A  Account acctToProcess = [SELECT Id, (SELECT Id FROM Contacts) FROM Account LIMIT 1];
2A  List<Contact> contacts = acctToProcess.Contacts;
```

List of sObjects:
`Parent.Children`

Custom Relationship

```
1B  Certification__c certToProcess= [SELECT Id, (SELECT Id FROM Courses__r)
                                   FROM Certification__c LIMIT 1];
2B  List<Course__c> courses = certToProcess.Courses__r;
```

List of sObjects:
`Parent.Children__r`

NOTE: The default name for the field that refers to a list of children uses the parent object's plural label.

WHAT DATA TYPE SHOULD YOU USE WHEN GOING FROM PARENT TO CHILD?

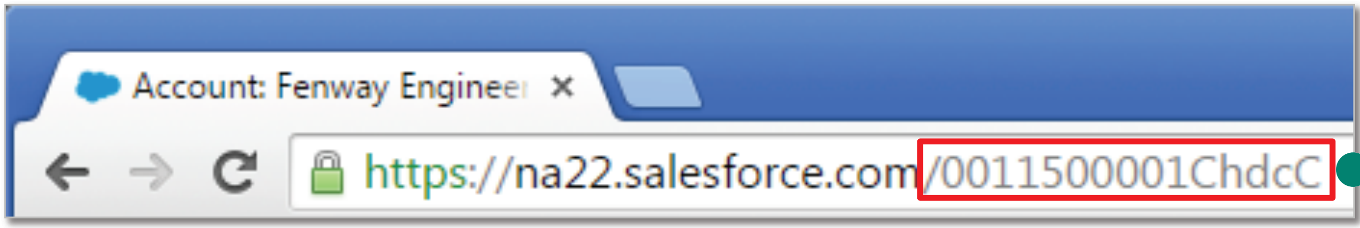
What must be specified if you want to select Certification Attempt records associated with a Certification Element?

```
1A  List<Certification_Element__c> oldCertElements =
                [SELECT Id, Name, (SELECT Id, Name
                                  FROM __1__)   Certification_Attempts__r
                 FROM Certification_Element__c
                 WHERE Type__c = 'Lab'];
```

What type should be used to store the child sObjects returned?

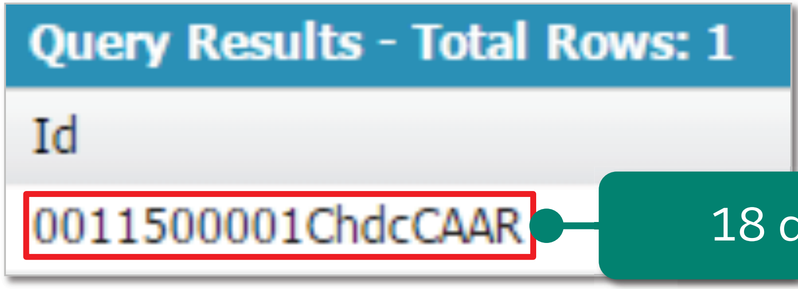
```
1B  if (oldCertElements.size() > 0) {
2B      Certification_Element__c certificationElement = oldCertElements[0];
3B      __2__ certAttempts = certificationElement.Certification_Attempts__r;
4B  }         List <Certification_Attempts__c>
```

Each object has a standard field of type `Id` for holding each record's unique system-generated Id.



15 digit ID

IHM, formules, etc



18 digit ID

IL FAUT COMPARER LES ID SUR LE MEME FORMAT

Case insensitive

ALERT:

Never hardcode an Id value in any code.

JOIN ME

4-9: USE RECORD IDS TO ACCESS AN ACCOUNT IN THE UI

Goal:

5 Minutes

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.

KEY TAKEAWAYS

- Apex is an object-oriented programming language that enables you to customize and create applications on the Force.com platform.
- As an integral part of the platform, Apex interacts with and has dependencies on other platform features.
- Apex transactions are subject to governor limits that monitor and enforce the use of system resources.
- sObjects are the Apex representation of objects provided and created in the Setup menu.

1. What would correctly complete the final statement?

```
Certification_Attempt__c certAttempt = …; //Assume this populates from the database.
String certificationElementName = ______________________________;
```

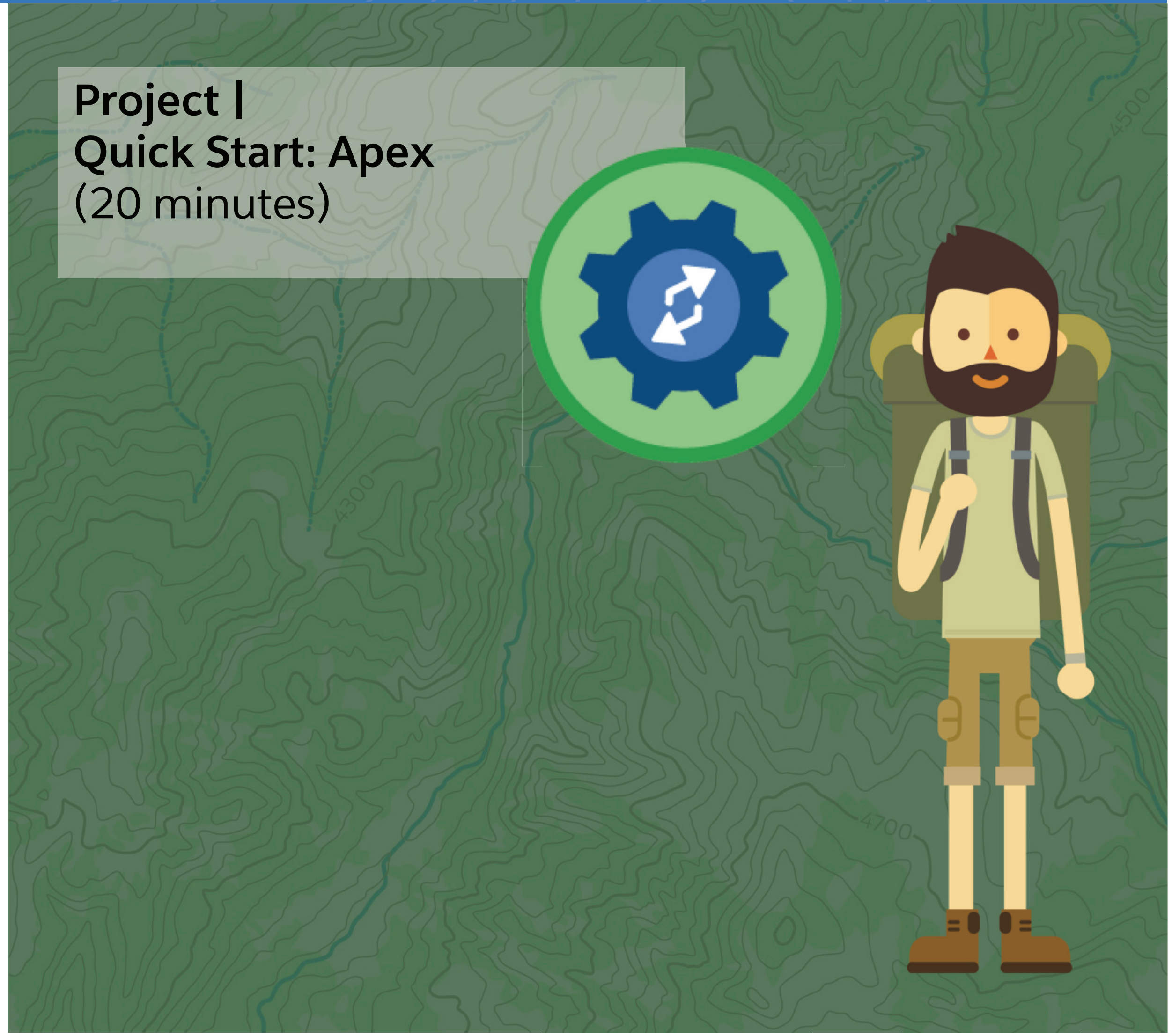
2. What would correctly complete the final statement?

```
Certification_Element__c certElement = …; //Assume this populates from the database.
List<Certification_Attempt__c> certificationAttempts = ________________________________;
```

3. What conditional expression would be used to determine if the contents of two sObjects are equal?
   a) sObj1 == sObj2
   b) sObj1.equals(sObj2)
   c) sObj1===sObj2
   d) sObj1.eq(sObj2)

4. What happens if an operation in code exceeds a governor limit?
   a) The system will throw an unrecoverable exception.
   b) The system will throw an exception the code can catch.
   c) The system will not allow you to migrate the code from a sandbox to production.
   d) The system will prevent the offending code from executing until it is fixed.

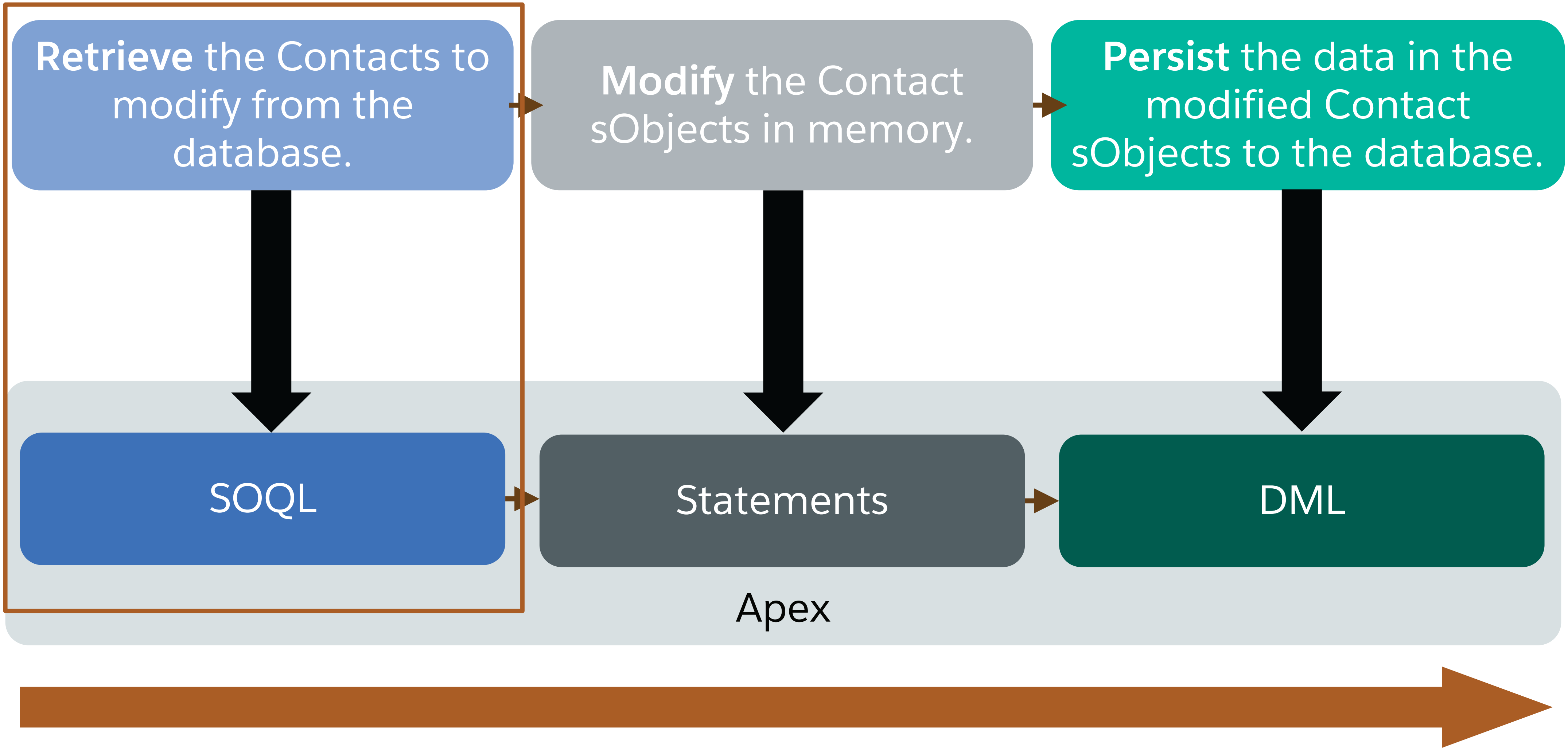
Project |
Quick Start: Apex
(20 minutes)



MODULE 5:
USE SOQL TO QUERY YOUR ORG'S DATA

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

WORKING WITH DATA IN APEX: A COMMON PATTERN



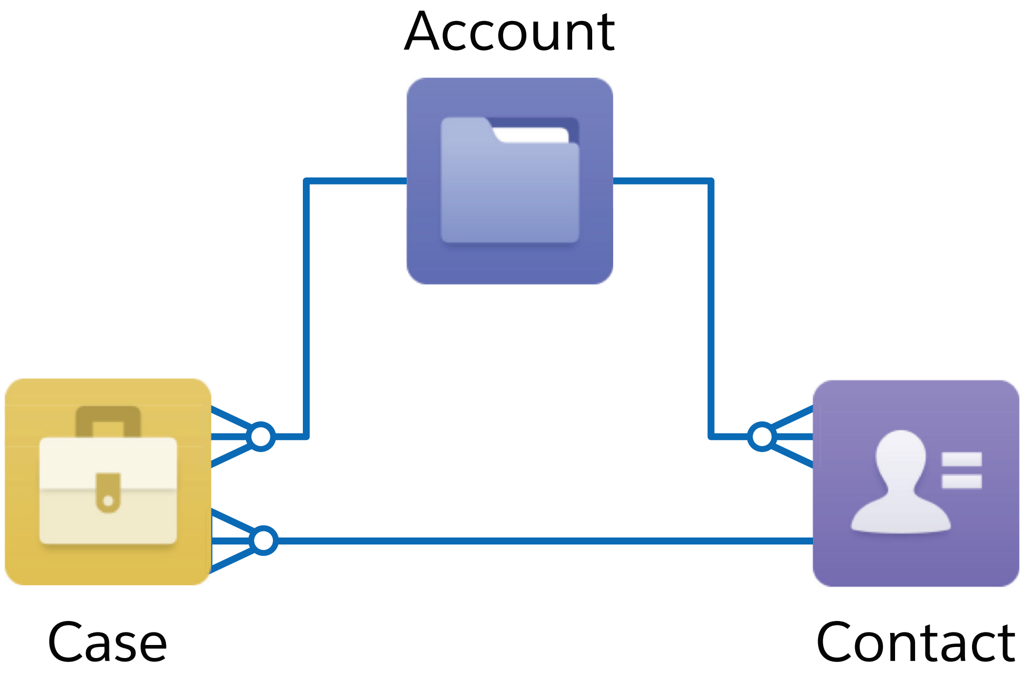
Cassie Mitchell
Beginning Developer

For the first project you'll be working on, you need to access data related to cases in AW's org.

To accomplish this, you need to:

- Write a basic query using Salesforce’s query language, SOQL.
- Process the result of a query in Apex.
- Create a query dynamically at run-time.

THE CASE OBJECT

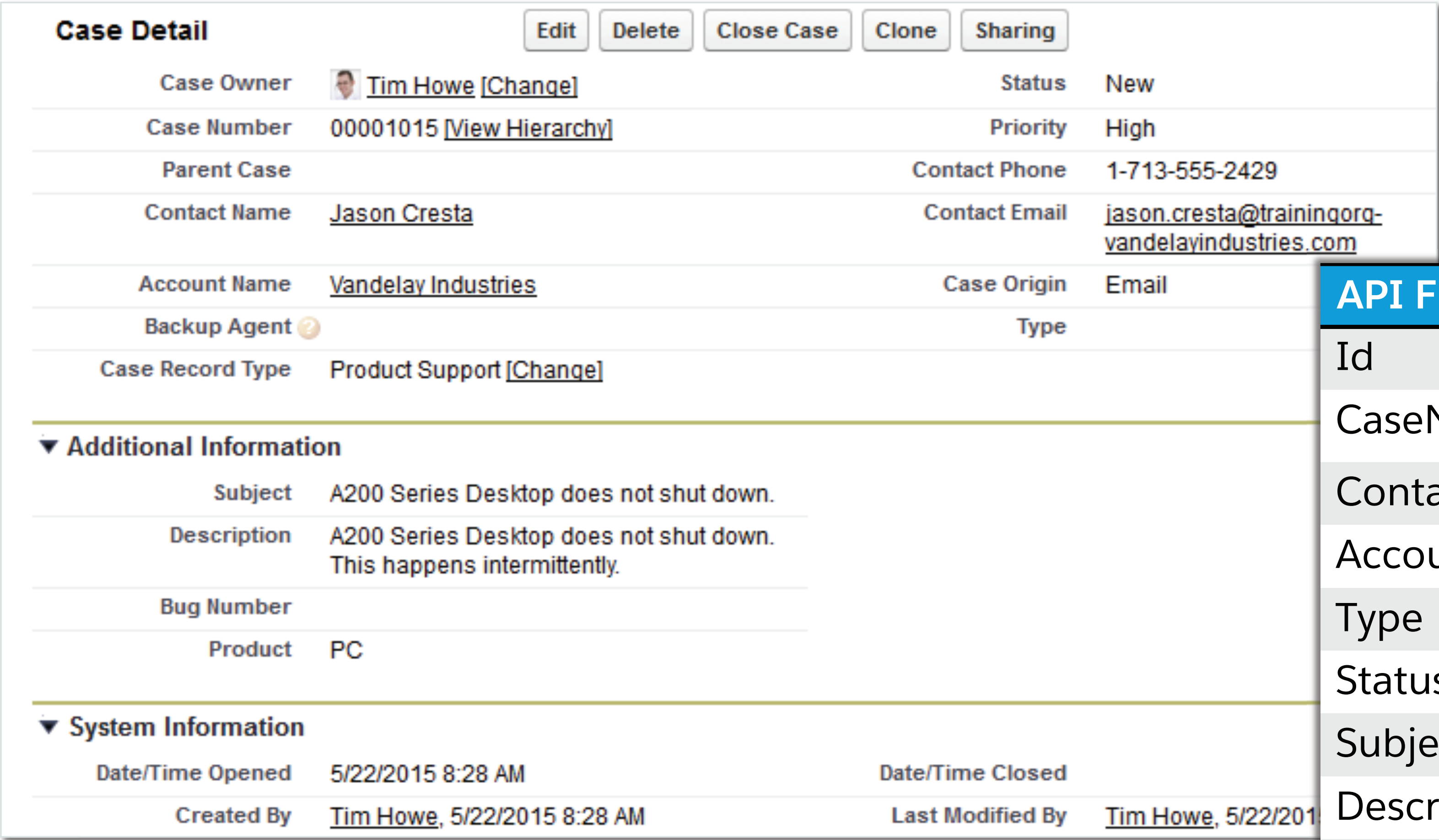


MODULE AGENDA

MODULE 5: USE SOQL TO QUERY YOUR ORG'S DATA

- **Using SOQL to Query Data**
- Writing and Processing a SOQL Query in Apex
- Creating a Dynamic Query at Run Time

WHICH FIELDS CAN YOU QUERY IN THE CASE OBJECT?



API Field Name	API Field Type
Id	id
CaseNumber	string
ContactId	reference
AccountId	reference
Type	string
Status	string
Subject	string
Description	textarea
IsClosed	boolean
ClosedDate	datetime

WATCH ME

5-1: CREATE AND RUN A QUERY IN THE DEVELOPER CONSOLE

salesforce

5 minutes

Goal:

Retrieve Cases using the Query Editor in the Developer Console.

Task:

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

WHAT IS SOQL?

DEFINITION:

SOQL is the Salesforce Object Query Language.

SOQL allows developers to query (using user-defined selection criteria) data in the Salesforce database.

SOQL queries can be performed:

- On an ad-hoc basis in tools such as the Developer Console.
- Within your Apex/API code.

SOQL IS NOT SQL

salesforce

Most SQL Variants	SOQL
Support statements for querying, CRUD, transaction control, schema definition, and more	Only supports query statements
Support SELECT *	Does not support SELECT *
Support joins, which are written using “left” and “right” keywords	Supports “relationship queries,” which are written using parent-child syntax
Do not support dot notation syntax to traverse foreign key relationships	Supports dot notation syntax to traverse object relationships
Are not governed by limits	Is multi-tenant aware (therefore, governed by limits)

SOQL SELECT SYNTAX


```
SELECT field1, field2, …
FROM object
[WHERE conditionExpression]
[LIMIT numberOfRows]
[Other options, such as GROUP BY]
```

A SOQL EXAMPLE

salesforce

Available Cases		
CaseNumber	**Subject**	**Status**
00001001	Laptop screen cracked	New
00001002	Keyboard keys are missing	Working
00001003	Print driver is out of date	New
00001004	Battery will not charge	Bug Fixed

```
SELECT CaseNumber, Subject
FROM Case
WHERE Status = 'New'
LIMIT 1
```

Return only 1 record

1. Which fields are being fetched?
2. From which Object is data being fetched?
3. What condition do all fetched sObjects need to meet?

WHERE CLAUSE OPERATORS

```
SELECT
   CaseNumber,
   Subject
FROM Case
WHERE __________
```

Consider the following separate scenarios, and then fill in the blank for the WHERE clause.

1. You want to query for the case whose Case Number is '00001001.'
2. You want to query for all cases whose Subject contains the word 'printer' (HINT: Use the LIKE operator with wildcards).

WHERE Clause Operator	Use
=	Equals
!=	Not equals
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
LIKE	Wildcard search in String fields; '%' matches 0+ characters. '_' matches exactly 1 character
IN / NOT IN	Inclusion / Exclusion
INCLUDES / EXCLUDES	Inclusion and exclusion for multi-select picklists
AND	Logical AND
OR	Logical OR
NOT	Negation

TYPE-SPECIFIC CONSIDERATIONS FOR THE WHERE CLAUSE

Date values:

- The Date format is: YYYY-MM-DD.

```
… WHERE BirthDate = 1999-01-30
```

Les dates ne sont pas entre quotes dans les queries !

- In Apex, DateTime field values are in the Coordinated Universal Time (UTC). You may need to offset DateTime values to your local time zone (ex: -08:00).

```
… WHERE ClosedDate > 2005-10-08T10:15:03-08:00
```

- You can write a query using date literals.

```
… WHERE ClosedDate != LAST_N_DAYS:365
```

Boolean values can be used in SOQL.

```
… WHERE IsClosed = TRUE
```

YOUR TURN

5-2: WRITE A SOQL QUERY THAT USES A WHERE CLAUSE

20 Minutes

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.”

MODULE AGENDA

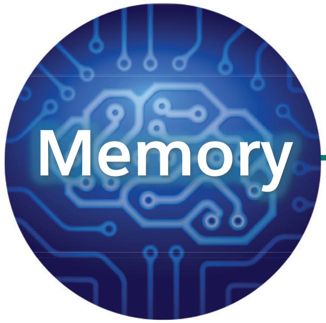
MODULE 5: USE SOQL TO QUERY YOUR ORG'S DATA

- Using SOQL to Query Data
- **Writing and Processing a SOQL Query in Apex**
- Creating a Dynamic Query at Run Time

WORKING WITH EXISTING CASE DATA IN APEX

Use SOQL to fetch data that exists in the database.


```
List<Case> cases = [SELECT CaseNumber FROM Case];
```



Store retrieved Case data in an sObject List in memory.

SYNTAX FOR USING A SOQL QUERY IN APEX

You can execute SOQL inside:

API calls.

Apex statements, using:

- Bracket notation.

```
[SELECT Fields FROM sObject]
```

- The `System.Database` class and its static `query` method.

```
Database.query(String query)
```

RETURN TYPES FOR A SOQL QUERY IN APEX: sObject LIST

Records fetched by SOQL queries are commonly stored in a list.

C'est même une bonne pratique.

```
List<Case> cases = [SELECT CaseNumber FROM Case];

for (Case aCase : cases) {
    System.debug(aCase);
}

// alternative loop with integer iterator
Integer numCases = cases.size();
for (Integer i = 0; i < numCases; i++) {
    System.debug(cases[i]);
}
```

Use a for-loop to iterate over the resulting sObjects.

NOTE:

Apex does not have arrays, but you can use array notation with lists.

RETURN TYPES FOR SOQL QUERIES IN APEX: sObject

You can assign the results of a query to a single sObject variable.

```
Case aCase = [SELECT Subject
              FROM Case
              WHERE CaseNumber = '00001007'];
```

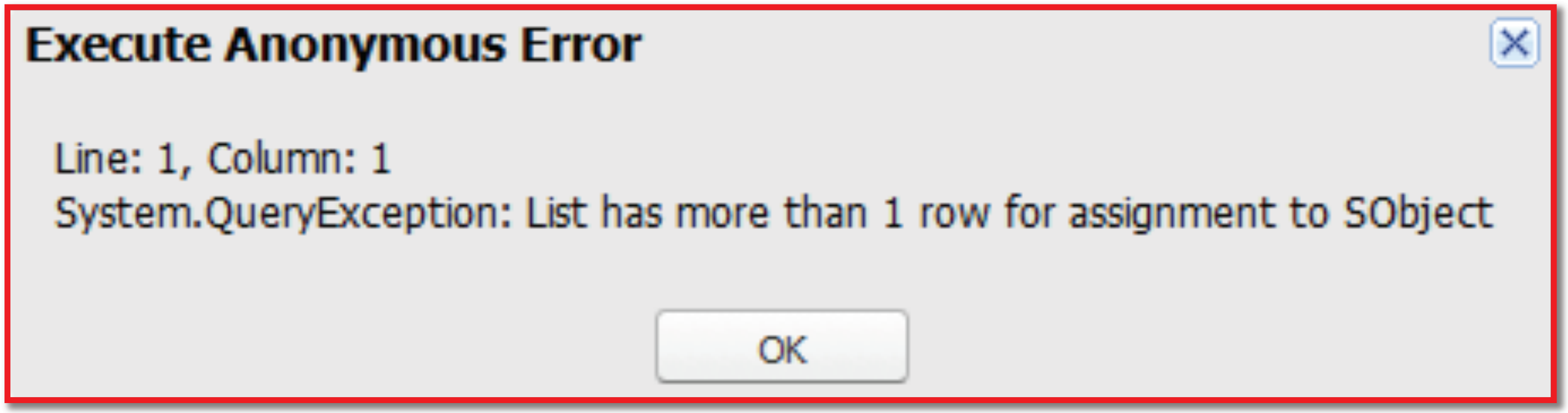
Préférer une liste List <Case> cListe qui fonctionne si 0 ou plus d'un résultats

NOTE: If you cannot guarantee that a single sObject is returned *each* time the query is run, you should assign the query to an sObject list.

THE WRONG RETURN TYPE

In Apex, any problem with executing a SOQL query will result in a Query Exception. Why is the exception raised in this instance? How would you get past this exception? What do you think would happen if the query didn't return any sObjects?

```
Case aCase = [SELECT Subject FROM Case];
```



5-3: WRITE AND EXECUTE A SOQL QUERY IN APEX

10 minutes

Goal:

Print cases into the debug log.

Tasks:

1. Print a single case into the debug log.
2. Print multiple cases into the debug log.

WORKING WITH SELECTED FIELD VALUES

- The Id field is always returned, even if not specified explicitly.
- All other fields must be explicitly selected to be evaluated.
- Fields of a selected sObject are assignable, even if the field wasn't explicitly selected in the query.

Which of these FOR loops results in an exception?

```
List<Case> cases = [SELECT CaseNumber FROM Case];
// Assume the list "cases" contains at least 1 case.

// FOR Loop 1
for (Case aCase : cases) {
    System.debug(aCase.Id + ' ' + aCase.CaseNumber);
}

// FOR Loop 2
for (Case theCase : cases) {
    System.debug(theCase.Subject);
}

// FOR Loop 3
for (Case currentCase : cases) {
    currentCase.Subject = 'Printer is jammed';
}
```

Simple affectation aucune raison que ça pose probleme.

The amount of data returned may cause your query to exceed the governor limit on heap size. Assume that, in the following code, the query returns so many Cases that you exceed the heap size governor limit. How can you overcome this issue?

```
List<Case> cases = [SELECT Description FROM Case];

//… process cases
```

For loop

On ne requête que les éléments dont on a vraiment besoin

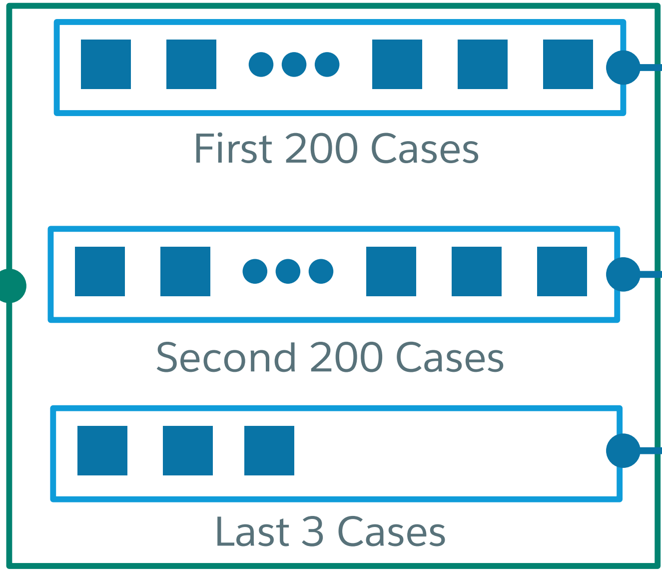
USING List<sObject> ITERATION VARIABLE


```
for (List<Case> cases : [SELECT CaseNumber FROM Case]) {
    for (Case aCase : cases) {
        //… process a single case
    }
}
```

The outer loop executes once per 200 sObjects fetched by the SOQL query.

The outer loop executes once per 200 sObjects fetched by the SOQL query

Query outside of loop would retrieve 403 records at once.



Processed in the first iteration of the outer loop.

Processed in the second iteration of the outer loop.

Processed in the third iteration of the outer loop.

NOTE:

A list iteration variable can help prevent you from exceeding the heap size limit when working with large volumes of data fetched by a query.

WHAT HAPPENS WITH A List<sObject> ITERATION VARIABLE?

salesforce

1. How many times will each loop iterate if there are 5 Cases in the database?
2. How many times will each loop iterate if there are 205 Cases in the database?
3. If there are 205 Cases in the database, what will Line 2 print to the log?

Important à comprendre !

```
1 for (List<Case> cases : [SELECT Id FROM Case]) {
2       System.debug(cases.size());
3 }
```

3. Ca affiche d'abord 200, puis 5, car bulkified (pack de 200) !

MODULE 5: USE SOQL TO QUERY YOUR ORG'S DATA

- Using SOQL to Query Data
- Writing and Processing a SOQL Query in Apex
- **Creating a Dynamic Query at Run Time**

BINDING IN THE `WHERE` CLAUSE

SOQL supports the binding operator (:) in the `WHERE` clause.

Example of a bound variable:

```
Set<String> caseSet = new Set<String>{'00001003', '00001005'};
List<Case> selectCases = [SELECT CaseNumber
                          FROM Case
                          WHERE CaseNumber IN :caseSet];
```

Use the colon operator to specify binding.

CREATING AND EXECUTING A QUERY AT RUN TIME

Dynamic SOQL: Build a query string at run time and pass it into the `Database.query` method.

```
String criteria = '';
Boolean lookForOpenCases = false;

if (lookForOpenCases) {
    criteria = 'Status = \'Working\'';
} else {
    criteria = 'ClosedDate < TODAY';
}

List<Case> theCases =
    Database.query('SELECT Id, Subject FROM Case WHERE ' + criteria);
System.debug(theCases);
```

ALERT:

You will not know about syntax errors in the query string until run time.

YOUR TURN

5-4: WRITE A DYNAMIC QUERY IN APEX

salesforce

10 minutes

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.

KEY TAKEAWAYS

- SOQL is the query language of the platform.
- Limits govern SOQL queries in a transaction.
- SOQL can be run ad-hoc or within Apex.
- A query in Apex commonly evaluates to an sObject List.
- When a SOQL FOR loop has a list iteration variable, the maximum number of sObjects processed in one iteration is 200.
- SOQL supports WHERE clauses with bound variables and bound expressions.
- `Database.query()` can be used to evaluate a query created at run time.

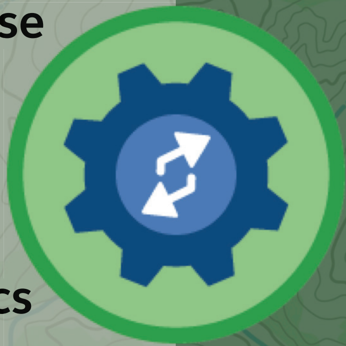
1. If a developer executes this query in the Query Editor of the Developer Console, what would be the result?

```
SELECT * FROM Case
```

2. Which WHERE clause operator supports wildcards?
3. Which SOQL clause supports variable binding?
4. The variable queryString is assigned the string literal: 'SELECT Name FROM Account'. How do you specify queryString as an argument to Database.query()?

Developer Beginner | Apex Basics & Database
(2 hours, 45 minutes)

Developer Beginner | Database & .NET Basics
(1 hours, 5 minutes)



MODULE 6:
USE SOQL TO QUERY PARENT-CHILD RELATIONSHIPS

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

You need to write more complex queries that utilize the relationships among the objects in the Certification custom app.

To accomplish this, you need to:

- Describe a relationship query.
- Write a query that traverses a child-to-parent relationship.
- Write a query that traverses a parent-to-child relationship.

MODULE 6: USE SOQL TO QUERY PARENT-CHILD RELATIONSHIPS

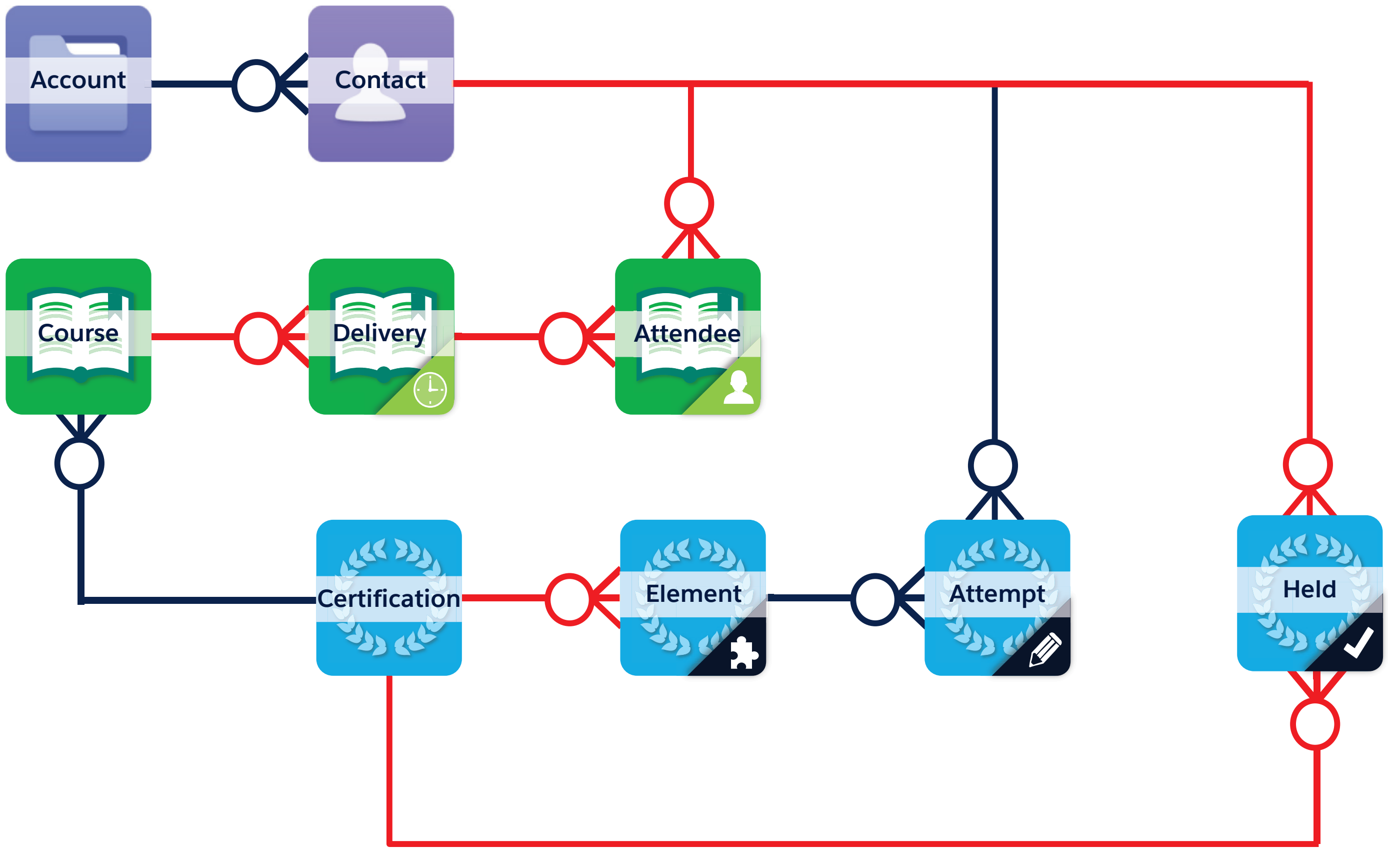
- **Understanding Relationship Queries**
- Querying Child-to-Parent Relationships
- Querying Parent-to-Child Relationships

UNDERSTANDING THE CERTIFICATION APPLICATION

salesforce

Which types of records would be created or updated by these scenarios?

1. A vendor relationship manager adds a new service provider and their technicians to the org.
2. A training coordinator signs a technician up for a course delivery.
3. A training coordinator changes the status of an Attempt from In Progress to Complete/Passed. This is the final element needed to hold the certification.

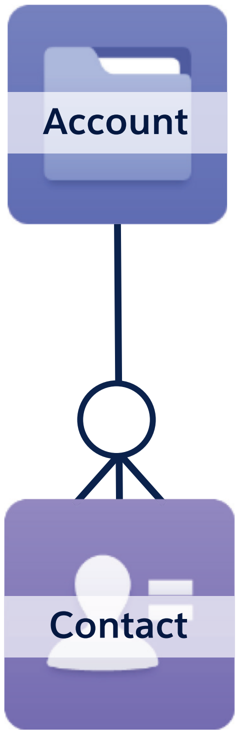


REMINDER: WHAT IS AN sObject RELATIONSHIP?

DEFINITION:

An **sObject relationship** associates sObjects through foreign keys, thereby establishing one as the "parent" and the other as the "child."

Objet père



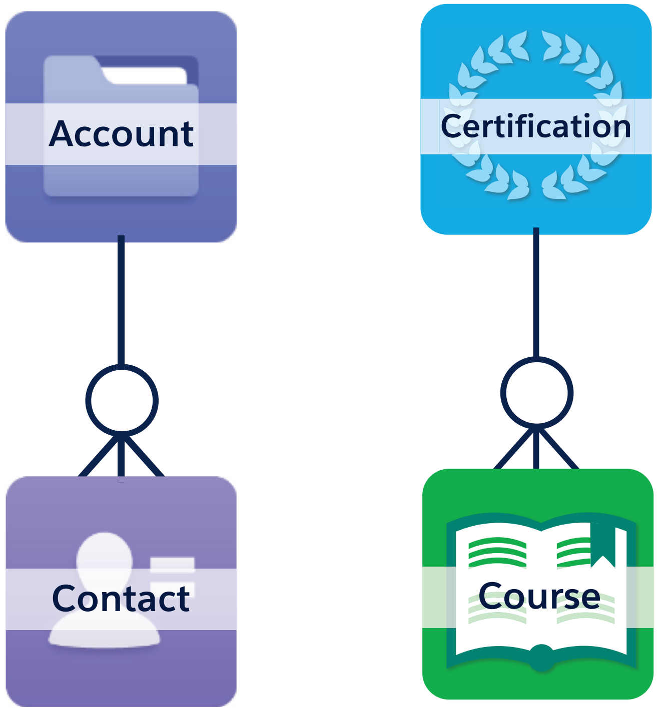
Objet fiston

The Account sObject is a parent of the Contact sObject.

An Account may be related to many Contacts.

A Contact may be related to a single Account.

Which relationship could be a standard relationship? Which is necessarily custom?



WHAT IS A RELATIONSHIP QUERY?

DEFINITION:

A **relationship query** traverses one or many sObject relationships to retrieve data related by foreign keys.

For example, use a relationship query to determine:

Which courses have deliveries in June of this year?

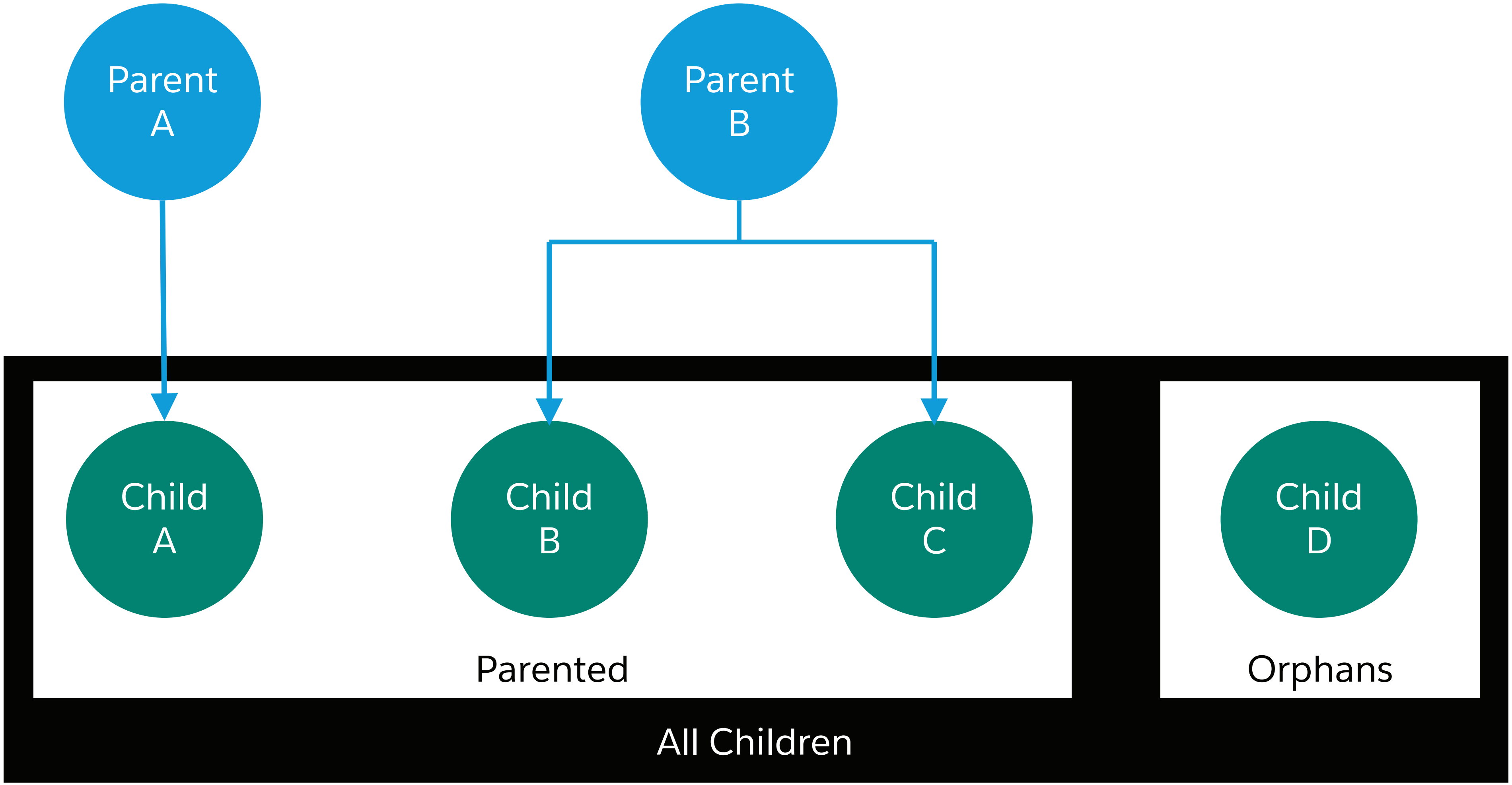
Which courses did attendees attend?

course_Delivery__r.Course__r.Name

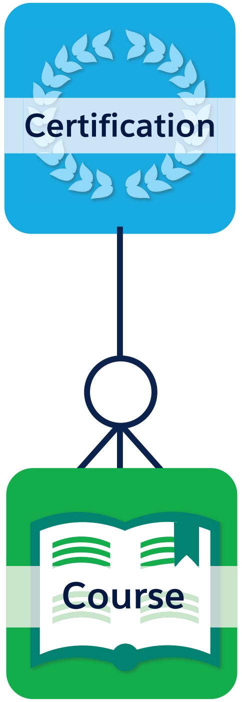
MODULE 6: USE SOQL TO QUERY PARENT-CHILD RELATIONSHIPS

- Understanding Relationship Queries
- **Querying Child-to-Parent Relationships**
- Querying Parent-to-Child Relationships

TERMINOLOGY: TYPES OF CHILD-TO-PARENT RELATIONSHIP QUERIES



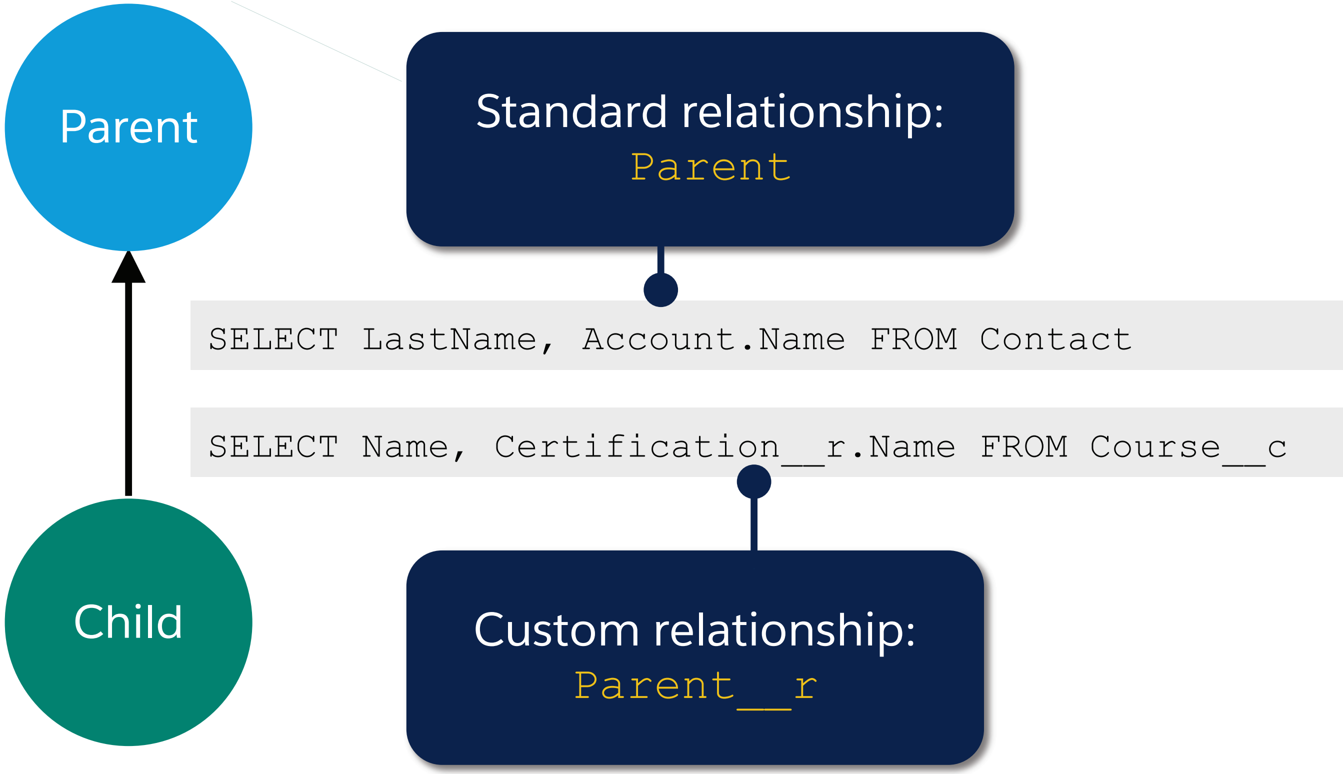
WHICH ROWS WILL BE FETCHED?



Field Values from Course	
Name	**Certification__c**
[202] AWCP Network	a03U00000091uRIIAY
[401] Data Recovery	<null>
[402] Managing Network Load	<null>

1. Certification and Course are related. Which is the parent? Which is the child?
2. Given the data above, how many rows would be fetched:
   a. If you selected all Courses, regardless of whether the Courses have related Certifications (i.e., "all children")? 3
   b. If you selected only Courses that have related Certifications (i.e., "parented children")? 1
   c. If you selected only Courses that don't have related Certifications (i.e., "orphans")? 2

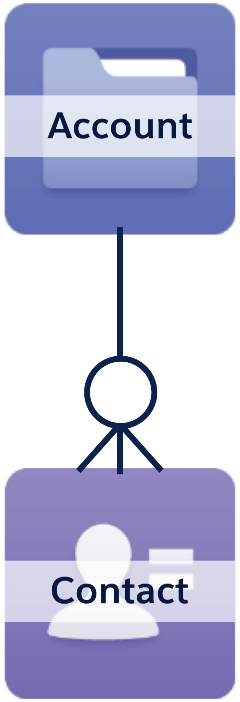
REFERENCING THE CHILD-TO-PARENT RELATIONSHIP



NOTE:

You can access five levels of ancestors from a child using dot notation.

IMPORTANT !
On peut remonter de 5 niveaux




```
SELECT FirstName, LastName, ___1___, ___2___
FROM Contact
```

Account.Name

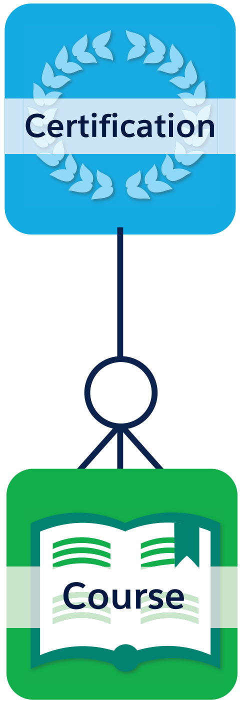
The query above selects <u>all Contacts</u>.

1. What fieldname would you specify in the first blank to select the parent Account's Id?
2. How would you use dot notation to fetch the parent Account's Name in the second blank?
3. What would happen if you specified `Accounts.Name` in the second blank?

Ca pète : inverse la relation, un contact n'a qu'un seul compte.

Field Values from Contact			... from Account
First Name	Last Name	Account	Parent Account's Name
Aaryn	Patel	0011400001fV45bAAC	Alveswood Technologies
Alexa	Delany	0011400001fV45bAAD	Enmore Installations

ALL CHILDREN – CUSTOM RELATIONSHIP



Certification__r.Name

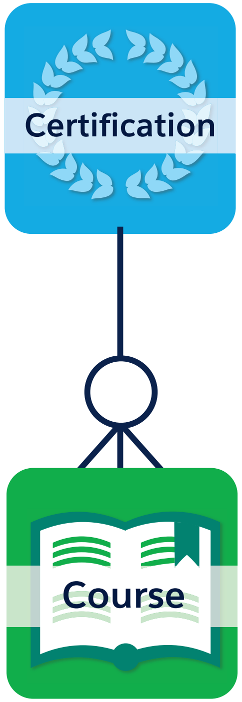
```
SELECT Name, _1__, _2__
FROM Course__c
```

The query above selects all Courses.

1. What fieldname must be specified in the first blank to select the associated Certification's Id?
2. How would you use dot notation to fetch the associated Certification's name in the second blank?
3. How does a reference to a parent custom relationship differ from a reference to a parent standard relationship?

Field Values from Course		... from Certification
Course's Name	**Certification's Id**	**Parent Certification's Name**
[202] AWCP Network	a03U00000091uRIIAY	AWCP Network
[401] Data Recovery	<null>	<null>

PARENTED CHILDREN – CUSTOM RELATIONSHIP




```
SELECT Name, Certification__r.Name
FROM Course__c
WHERE Certification__c <> NULL
```

The query above selects only parented Courses.

1. How would you change the WHERE clause so that only orphaned Courses were selected?

En changeant la whereClause à = NULL

Field Values from Course	... from Certification
Name	**Parent Certification's Name**
[202] AWCP Network	AWCP Network

YOUR TURN

6-1: WRITE AND TEST CHILD-TO-PARENT RELATIONSHIP QUERIES

salesforce

20 Minutes

Goal:

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

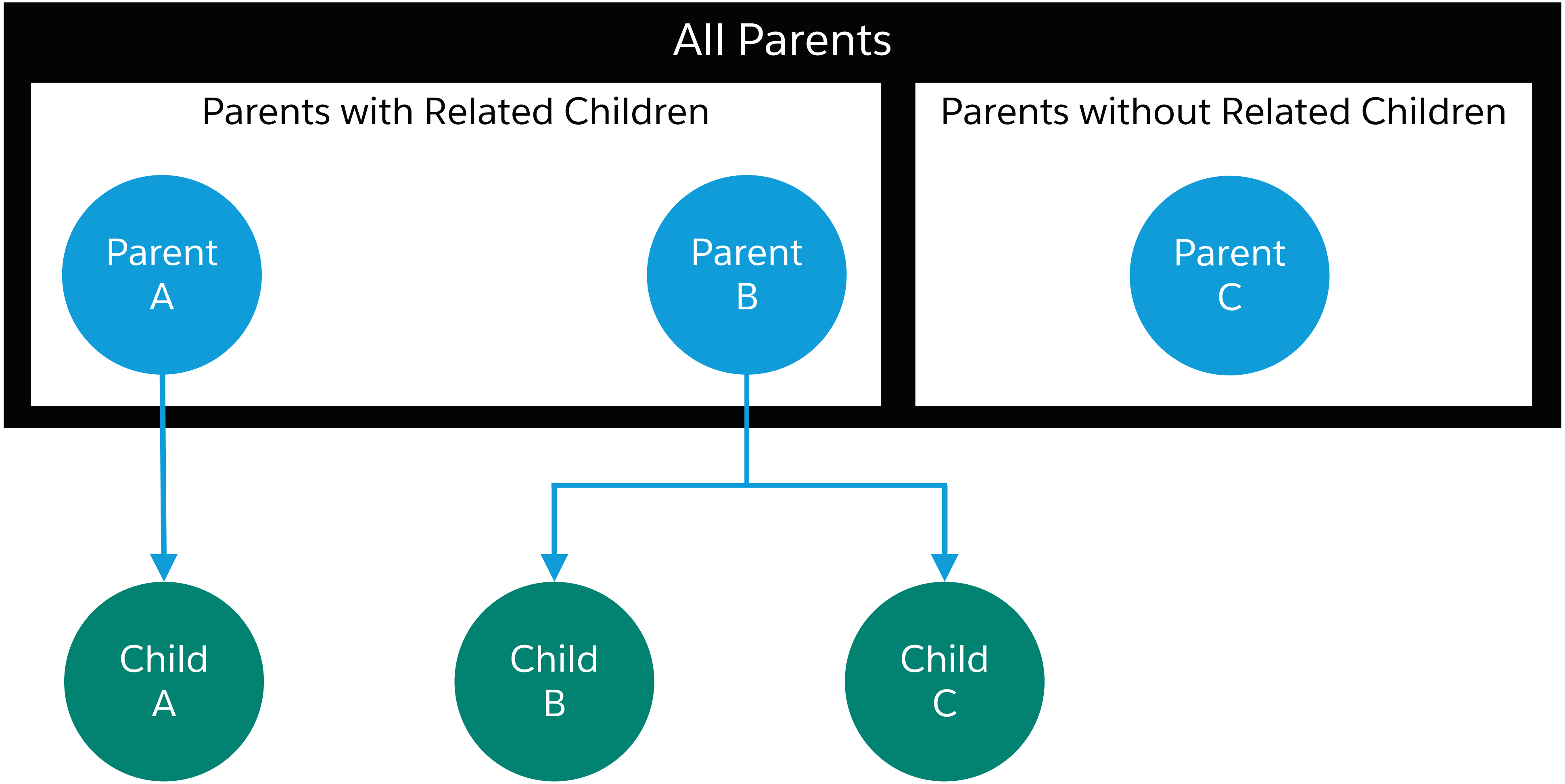
Tasks:

1. Select all Contacts and their related Accounts.
2. Select Courses that have related Certifications.

MODULE 6: USE SOQL TO QUERY PARENT-CHILD RELATIONSHIPS

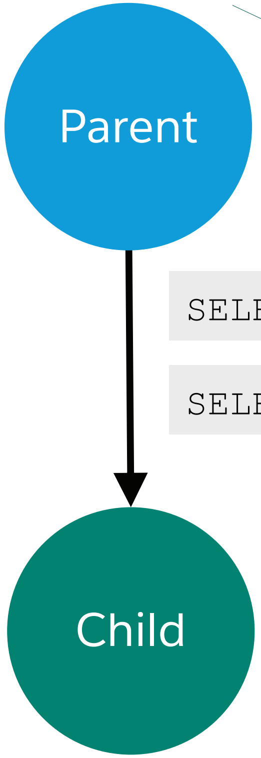
- Understanding Relationship Queries
- Querying Child-to-Parent Relationships
- **Querying Parent-to-Child Relationships**

TERMINOLOGY: TYPES OF PARENT-TO-CHILD RELATIONSHIPS



REFERENCING THE PARENT-TO-CHILD RELATIONSHIP

salesforce



Standard relationship:
`Children`

```
SELECT Name, (SELECT Lastname, Firstname FROM Contacts) FROM Account
SELECT Name, (SELECT Name, Duration__c FROM Courses__r) FROM Certification__c
```

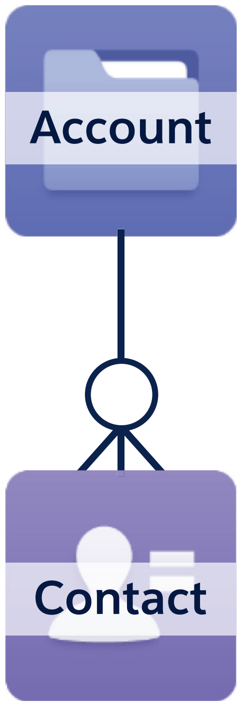
Custom relationship:
`Children__r`

un seul niveau de requêtes imbriquées est autorisé, dans la partie SELECT de la query

NOTE:

Only one level of nested queries is allowed in a SELECT clause.

ALL PARENTS – STANDARD RELATIONSHIP



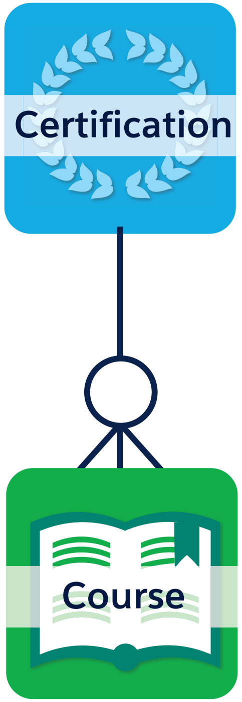

```
SELECT Name,
     (SELECT LastName FROM Contacts)
FROM Account
```

This query selects <u>all Accounts</u>.

1. What does the nested SELECT do?
2. In the nested SELECT,
   why is "Contacts" specified and
   not "Contact"?

Field Values from Account	... from Contact
Name	**List of Contacts**
Enmore Installations	[{"LastName":"Morelli"},{"LastName":"Parkinson"}, {"LastName":"Curran"},{"LastName":"Yang"},{"LastName":"Delany"}]

ALL PARENTS – CUSTOM RELATIONSHIP




```
SELECT Name,
     (SELECT Name FROM Courses__r )
FROM
     Certification__c
```

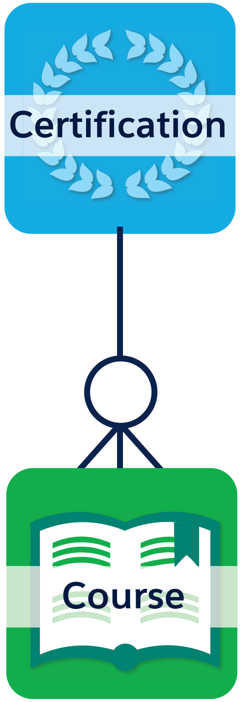
This query selects <u>all Certifications</u>.

1. How does a reference to a child custom relationship differ from a reference to a child standard relationship?

   append __r

Field Values from Certification	... from Course
Certification Name	**List of Courses**
AWCP Server	[{"Name":"[201] AWCP Server"}]

PARENTS WITH RELATED CHILDREN – CUSTOM RELATIONSHIP




```
SELECT Name
FROM Certification__c
WHERE Id IN
    (SELECT Certification__c FROM Course__c)
```

This query selects <u>only Certifications that have related Courses</u>.

1. How would you change this query to only select Certifications without related Courses?

Field Values from Certification
Certification Name
AWCP Server

WATCH ME

6-2: QUERY ACCOUNT AND RELATED CONTACTS

10 minutes

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.

- Use dot notation to traverse child-to-parent relationships in a query.
- Use a nested SELECT to retrieve children's fields when traversing a parent-to-child relationship.
- You can access the fields of 5 levels of ancestors or 1 level of children in the SELECT clause of a SOQL query.

MODULE 7:
DML ESSENTIALS

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

TRAILHEAD

Cassie Evans
Developer

You need to make changes to Contacts using Apex and then save those changes to the database.

To accomplish this, you need to:

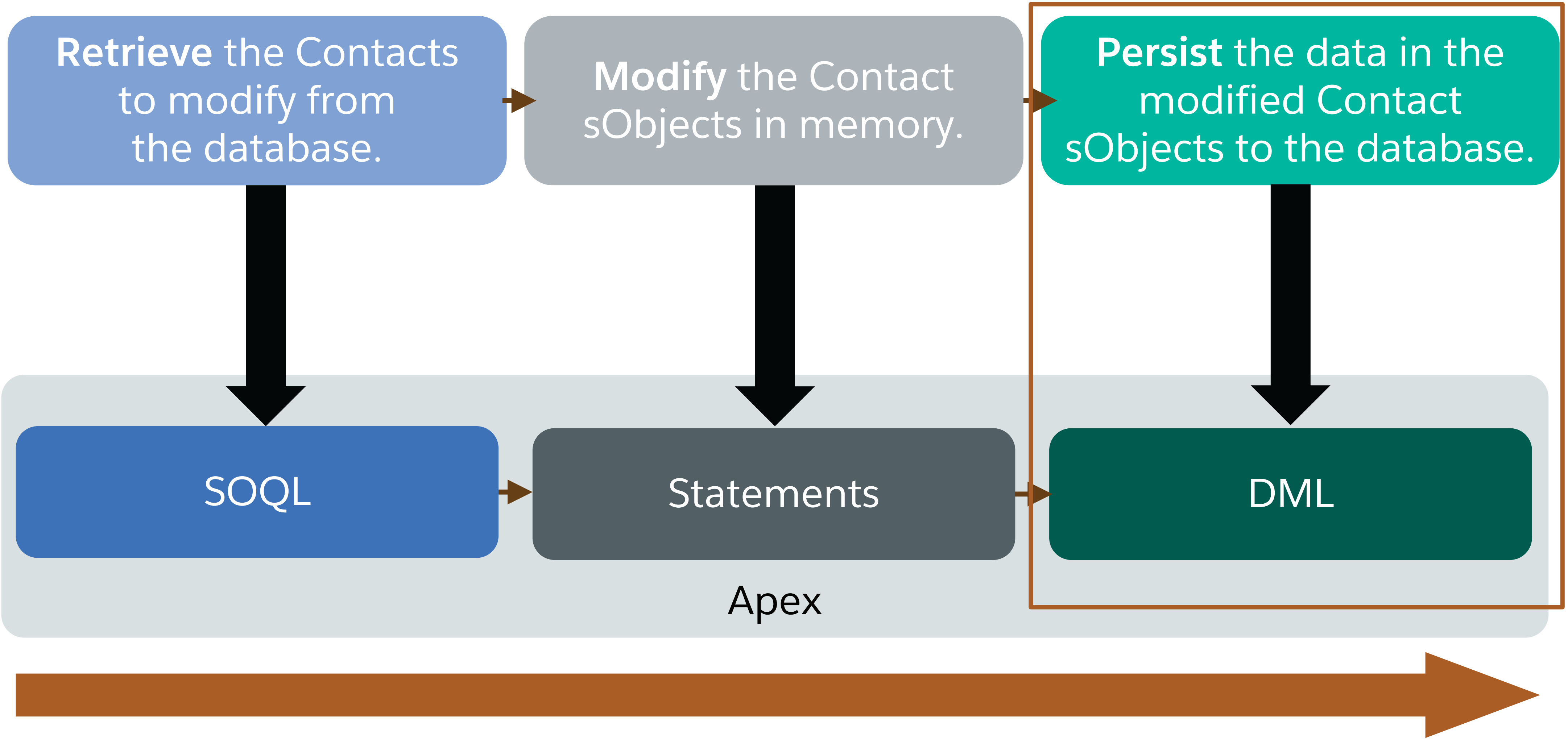
- List the differences between the way you can invoke DML operations.
- Write Apex to invoke DML operations and handle DML errors.

MODULE AGENDA

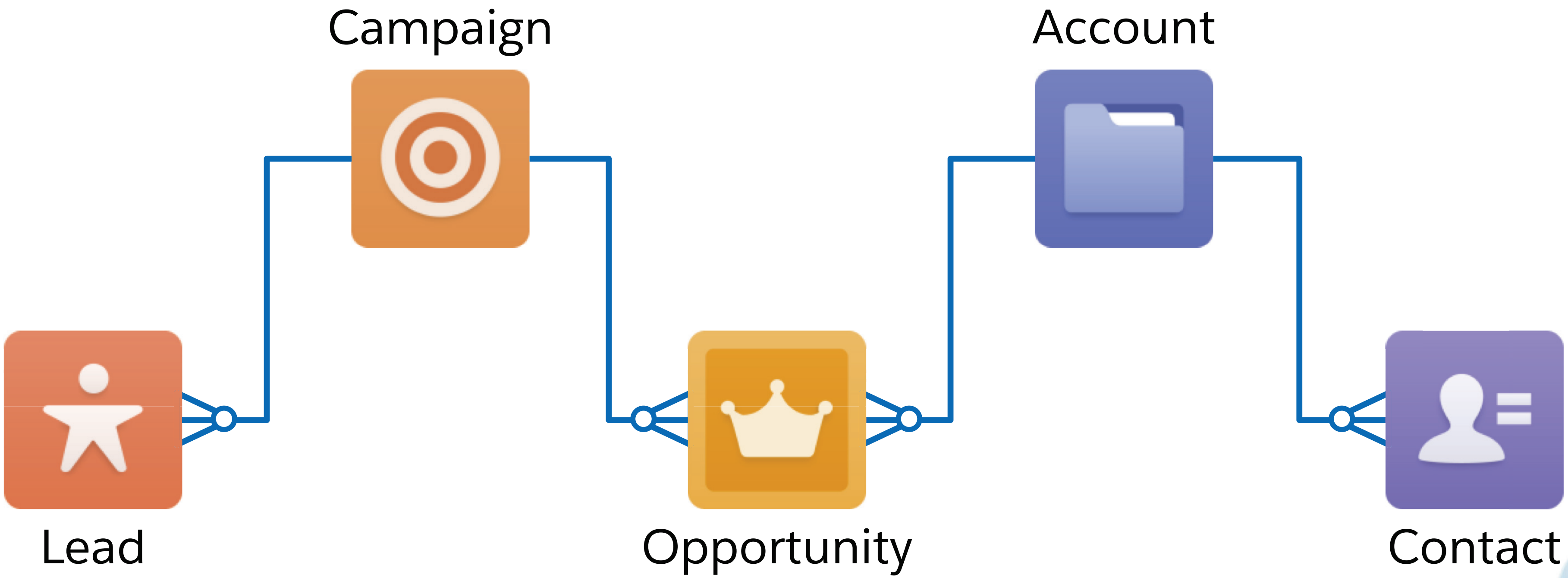
MODULE 7: DML ESSENTIALS

- **Options for Persisting Data**
- Invoking DML Events
- Handling DML Errors and Exceptions

WORKING WITH DATA IN APEX: A COMMON PATTERN

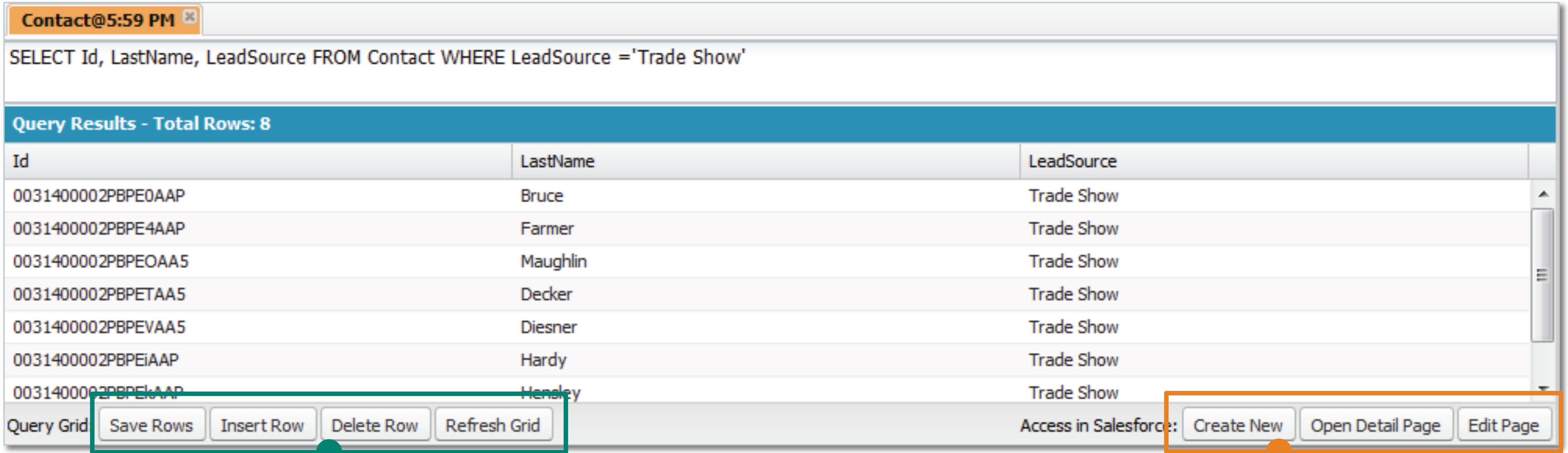


SALES CRM DATA MODEL



A.W. Computing

DEMO: SAVING CHANGES TO CONTACTS IN THE DEVELOPER CONSOLE



Use the Developer Console to persist changes.

Use the Developer Console to open the UI to make and persist changes.

EXAMPLE: SAVING CHANGES TO CONTACTS PROGRAMMATICALLY


```
List<Contact> oldLeadSourceContacts =
    [SELECT LeadSource FROM Contact WHERE LeadSource ='Trade Show'];

Set<Id> oldLeadSourceContactsIds = new Set<Id>();
for (Contact c : oldLeadSourceContacts) {
    c.LeadSource = 'Other';
    oldLeadSourceContactsIds.add(c.Id);
}

update oldLeadSourceContacts;

List<Contact> updatedLeadSourceContacts =
    [SELECT LeadSource FROM Contact WHERE Id IN :oldLeadSourceContactsIds];
for (Contact c : updatedLeadSourceContacts) {
    if (c.LeadSource <> 'Other')
        System.debug('Update failed');
}
```

Fetch rows from the database.

Update fetched rows in memory.

Persist updates to the database (DML).

Fetch the updated rows from the database.

Verify the updates.

DEFINITION:

DML: Apex's **Data Manipulation Language** allows you to persist the creation of or modifications to an instance of an sObject.

MATCH THE PROGRAMMING SCENARIO TO THE DML COMMAND

salesforce

Scenario	DML Command
1. Retrieve into memory a Contact whose LeadSource is 'Trade Show.' Modify its LeadSource to 'Other.' Persist this modification to your org. UPDATE	A. Insert
2. Create a new instance of a Contact in memory. Persist this instance to your org. INSERT	B. Update
3. Users were incorrectly entered as Contacts through the UI. Retrieve those Contacts into memory, and use their Id values to remove them from the org. DELETE	C. Upsert
4. Create new Contacts in memory. Also modify existing Contacts that were loaded into memory using SOQL. Issue a single UPSERT command to persist new and modified Contacts to the org.	D. Delete
5. Actually, the Contacts removed in Scenario 3 were created correctly. Use SOQL to retrieve those Contacts from the Recycle Bin into memory. Then, restore those Contact records. UNDELETE	E. Undelete

MODULE 7: DML ESSENTIALS

- Options for Persisting Data
- **Invoking DML Events**
- Handling DML Errors and Exceptions

TWO WAYS OF WRITING DML COMMANDS

Standalone DML

```
1A  Contact withName =
        new Contact(LastName = 'Hines');
2A  Contact noName = new Contact();
3A  List<Contact> contacts = new List<Contact>();
4A  contacts.add(withName);
5A  contacts.add(noName); // this won't work because LastName is required
6A  insert contacts;
```

Database.method(sObject List)

```
1B  Contact withName =
        new Contact(LastName = 'Hines');
2B  Contact noName = new Contact();
3B  List<Contact> contacts = new List<Contact>();
4B  contacts.add(withName);
5B  contacts.add(noName);
6B  Database.insert(contacts);
```

DEMO: WHAT HAPPENS WHEN YOU INSERT INCOMPLETE CONTACTS?


```
Contact withName = new Contact(LastName = 'Santoyo');
Contact noName = new Contact();
List<Contact> contacts = new List<Contact>();
contacts.add(withName);
contacts.add(noName);
insert contacts;  // Inserted contacts need a last name
```

1. Does the incomplete Contact get inserted?
2. Does the complete Contact get inserted?

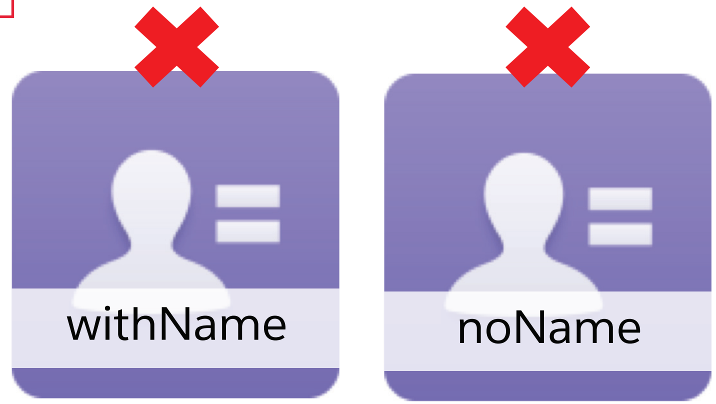
Aucun des deux, rollback

PARTIAL PROCESSING OF RECORDS

Standalone DML

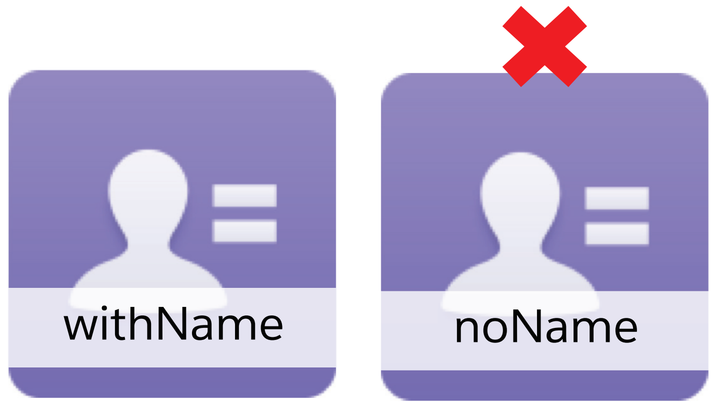
Aucun insert ne rentre si un ou plus ne marche pas

```
1A Contact withName =
       new Contact(LastName = 'Soto');
2A Contact noName = new Contact();
3A List<Contact> contacts = new List<Contact>();
4A contacts.add(withName);
5A contacts.add(noName);
6A insert contacts;
```



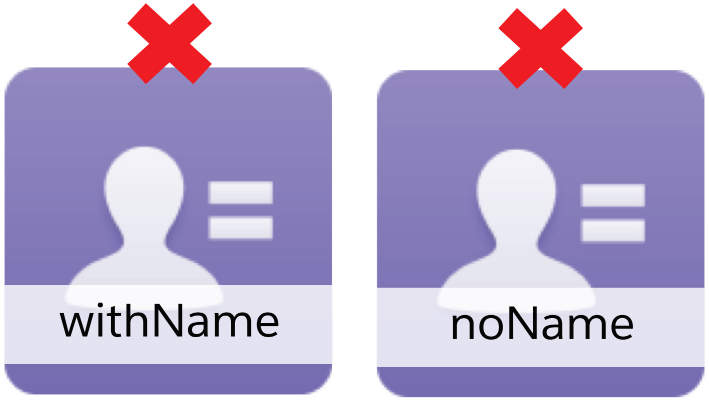
Database.method(sObject List, false)

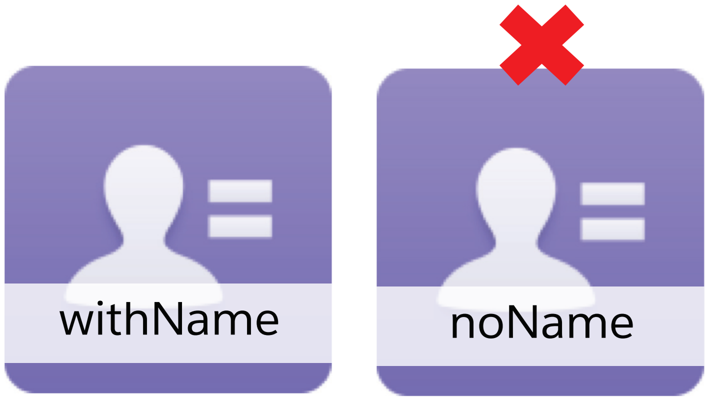
```
1B Contact withName =
       new Contact(LastName = 'Soto');
2B Contact noName = new Contact();
3B List<Contact> contacts = new List<Contact>();
4B contacts.add(withName);
5B contacts.add(noName);
6B Database.insert(contacts, false);
```



Database.method(param1, param2) :
Si on ne met pas param2 (AllOrNone) il est par défaut à TRUE donc soit ça passe en entier, soit ça casse. Pour du partiel, spécifier à FALSE.

ARE THERE OTHER OPTIONS FOR PARTIAL PROCESSING?





Standalone DML: `insert contacts;`

Database method: `Database.insert(contacts);`

Par défaut à TRUE !

Database method: `Database.insert(contacts, TRUE);`

All of these options result in "all or none" behavior.

Database method: `Database.insert(contacts, FALSE);`

Partial processing only occurs when the optional `AllorNone` parameter is FALSE. This also means that if the DML operation fails for an sObject, an error is recorded but an exception is not raised.

FINDING THE ID OF A SUCCESSFULLY INSERTED RECORD


```
Contact withName = new Contact(LastName = 'Okoye');
List<Contact> contacts = new List<Contact>();
contacts.add(withName);
insert contacts;
System.debug(withName.Id);
```

After a successful insert, the variable that holds the instance of the inserted sObject is updated with the Id.

YOUR TURN

7-1: EXECUTE DML COMMANDS

25 minutes

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.

MODULE 7: DML ESSENTIALS

- Options for Persisting Data
- Invoking DML Events
- **Handling DML Errors and Exceptions**

WHICH CODE BLOCKS RESULT IN AN EXCEPTION?

```
1A  List<Contact> contacts = new List<Contact>();
2A  insert contacts;
```

```
1B  List<Contact> contacts = new List<Contact>();
2B  Contact noName = new Contact();
3B  Contacts.add(noName);
4B  insert contacts;
```

```
1C  List<Contact> contacts = new List<Contact>();
2C  Contact newContact = new Contact(LastName = 'Benett');
3C  Contacts.add(newContact);
4C  Contacts[0] = null;
5C  insert contacts;
```

```
1D  for (Integer i = 0; i<175; i++) {
2D      Contact testContact = new Contact(LastName = 'Test' + i);
3D      insert testContact; // LIMIT for DML commands in a single transaction = 150
4D  }
```

85 caractères

```
1E  List<Contact> contacts = new List<Contact>();
2E  Contact longName = new Contact(LastName =
        '00085chars00085chars00085chars00085chars00085chars00085chars00085chars00085chars00085');
3E  Contacts.add(longName);
4E  insert contacts; //LastName is a Text(80) field
```

OVERCOMING EXCEPTIONS


```
List<Contact> contacts = new List<Contact>();
insert contacts;
```

No exception.

```
List<Contact> contacts = new List<Contact>();
Contact noName = new Contact();
Contacts.add(noName);
insert contacts;
```

Ensure all the required fields are populated.

```
List<Contact> contacts = new List<Contact>();
Contact newContact = new Contact(LastName = 'Benett');
Contacts.add(newContact);
Contacts[0] = null;
insert contacts;
```

Don't run DML operations on NULL elements.

```
for (Integer i = 0; i<175; i++) {
    Contact testContact = new Contact(LastName = 'Test' + i);
    insert testContact; // LIMIT for DML commands in a single transaction = 150
}
```

Stay within the DML Limits.

```
List<Contact> contacts = new List<Contact>();
Contact longName = new Contact(LastName =
    '00085chars00085chars00085chars00085chars00085chars00085chars00085chars00085chars00085');
Contacts.add(longName);
insert contacts; //LastName is a Text(80) field
```

Ensure that field type restrictions are respected.

CATCHING EXCEPTIONS


```
List<Contact> contacts = new List<Contact>();
Contact noName = new Contact();
Contact anotherNoName = new Contact();

Contacts.add(noName);
Contacts.add(anotherNoName);

try {
    insert contacts;
} catch (DMLException e){
    System.debug('Caught exception: ' + e);
}
```

Throws an exception.

Catches the exception.

NOTE:

If an end user started the chain of events that resulted in an exception, the exception will be displayed in the user interface.

CAPTURING THE RESULT OF PARTIAL PROCESSING


```
Contact withName= new Contact(LastName = 'Lee');
Contact noName= new Contact();
List<Contact> contacts = new List<Contact>();
contacts.add(withName);
contacts.add(noName);
List<Database.SaveResult> srs =
      Database.insert(contacts, false);
```

You can capture the per-sObject result of a DML operation that is written as for partial processing.

What is in a `SaveResult`?

`SaveResult`	Which sObject?	`isSuccess()`	`getId()`	`getErrors()`
`srs[0]`	`withName`	True	An 18-digit Id	Empty list
`srs[1]`	`noName`	False	Null, because noName didn't get saved to the database	A list of type `Database.Error` that captures why the insert failed for this sObject

PROCESSING A List<Database.SaveResult>


```
Contact withName= new Contact(LastName = 'Lee');
Contact noName= new Contact();
List<Contact> contacts = new List<Contact>();
contacts.add(withName);
contacts.add(noName);
List<Database.SaveResult> srs =  Database.insert(contacts, false);

for (Database.SaveResult sr : srs ) {

    if(sr.isSuccess() == FALSE) {

        List<Database.Error> errors = sr.getErrors();

        Integer i = 1;
        String debugString = 'Errors: ';
        for (Database.Error e : errors) {
            debugString += i + '. ' + e.getMessage() + '; ';
            i++;
         }

        System.debug(debugString);
    }
}
```

Loop processes one SaveResult per sObject inserted.

If an error was detected...

... start processing the errors.

getMessage() gets error info.

YOUR TURN

7-2: HANDLE DML ERRORS AND EXCEPTIONS

salesforce

15 minutes

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.

STAYING WITHIN GOVERNOR LIMITS WHEN USING DML (1)

salesforce

```
1  for (Contact aContact : [SELECT Id FROM Contact]) {
2      //modify aContact
3      Database.update(aContact);
4  }
```

1. What is the current governor limit for the total number of DML statements issued? Use the limits guide online to discover the answer. 150 DML
2. What will happen during the execution of this `for` loop if the number of records returned by the query in line 1 exceeds the number of DML statements allowed in a single transaction? Rollback direct

STAYING WITHIN GOVERNOR LIMITS WHEN USING DML (2)


```
List<Contact> contacts = new List<Contact>();
for(Contact aContact : [SELECT Id from Contact]) {
    //modify aContact
    contacts.add(aContact);
}
Database.update(contacts);
```

On crée une liste, que l'on fill, puis après l'avoir remplie on l'insert -> un unique DML.

1. What is the current governor limit for the heap size in synchronous Apex? 6 mo
2. What happens if the volume of data of `contacts` grows to be larger than the heap size limit?

STAYING WITHIN GOVERNOR LIMITS WHEN USING DML (3)


```
for (List<Contact> contacts : [SELECT Id FROM Contact]) {
    for (Contact aContact : contacts){
        //modify aContact
    }
    Database.update(contacts);
}
```

1. How does this code sample solve the issues we saw in the previous two code samples?

KEY TAKEAWAYS

- Apex's Data Manipulation Language allows you to save new and changed data to your org.
- The DML commands available include: insert, update, upsert, delete, and undelete.
- There are two ways to specify DML commands: the standalone command and the Database class method.
- You can choose partial processing with a statement such as: `Database.insert(sObject List, FALSE)`

Ne pas insérer un élément Null, cela génère une exception (de conception de code) si on tente un Database.insert(null)

MODULE 8:
TRIGGER ESSENTIALS

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

TRAILHEAD

MODULE OBJECTIVES

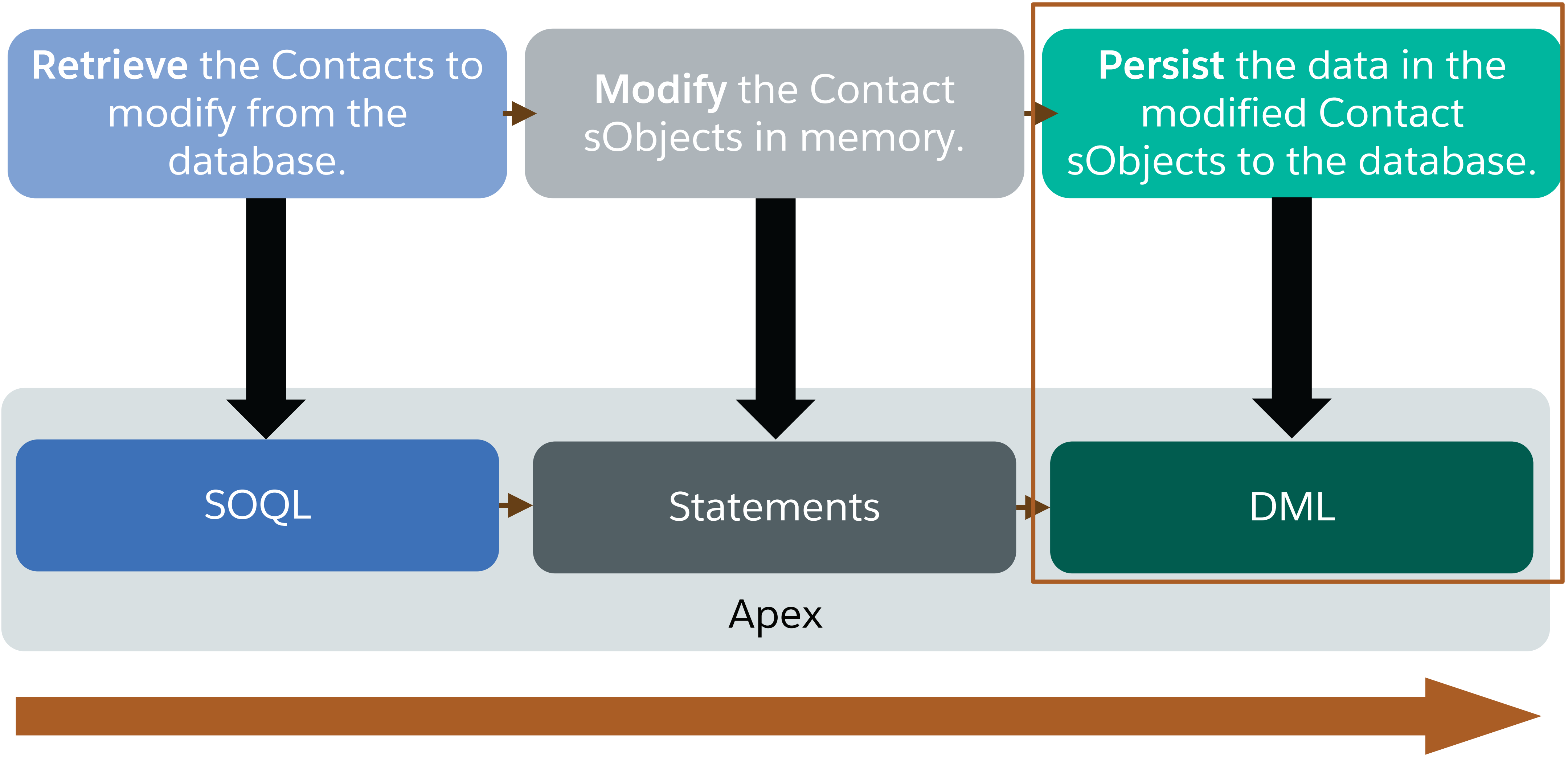
An instructor noticed that one of the courses she was scheduled to teach starts on a holiday. How can we ensure that Course Deliveries cannot be scheduled to start on a holiday?

To accomplish this, you need to:

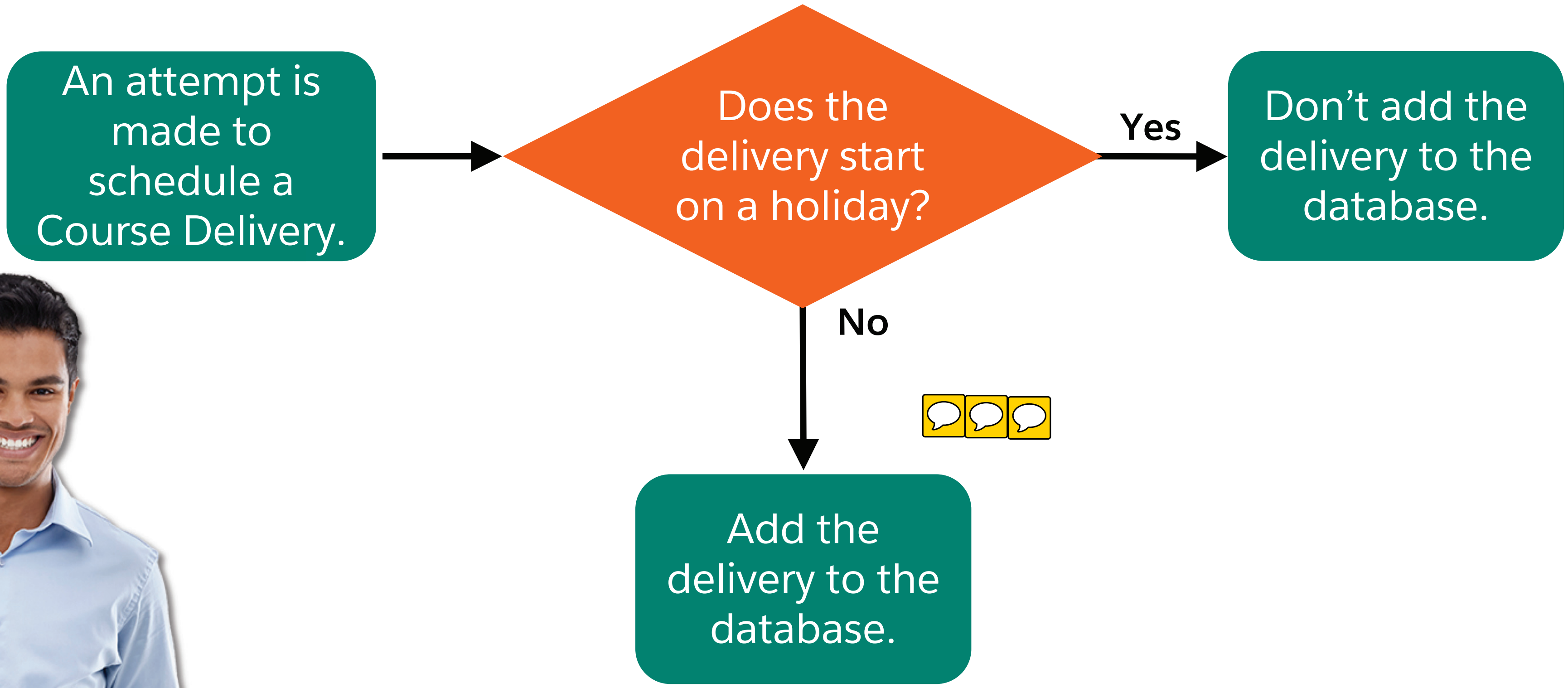
- Describe what a trigger is used for.
- Describe the syntax of a trigger definition.
- Use trigger context variables.

- **Automating Logic**
- Defining a Trigger
- Defining Trigger Logic

WORKING WITH DATA IN APEX: A COMMON PATTERN



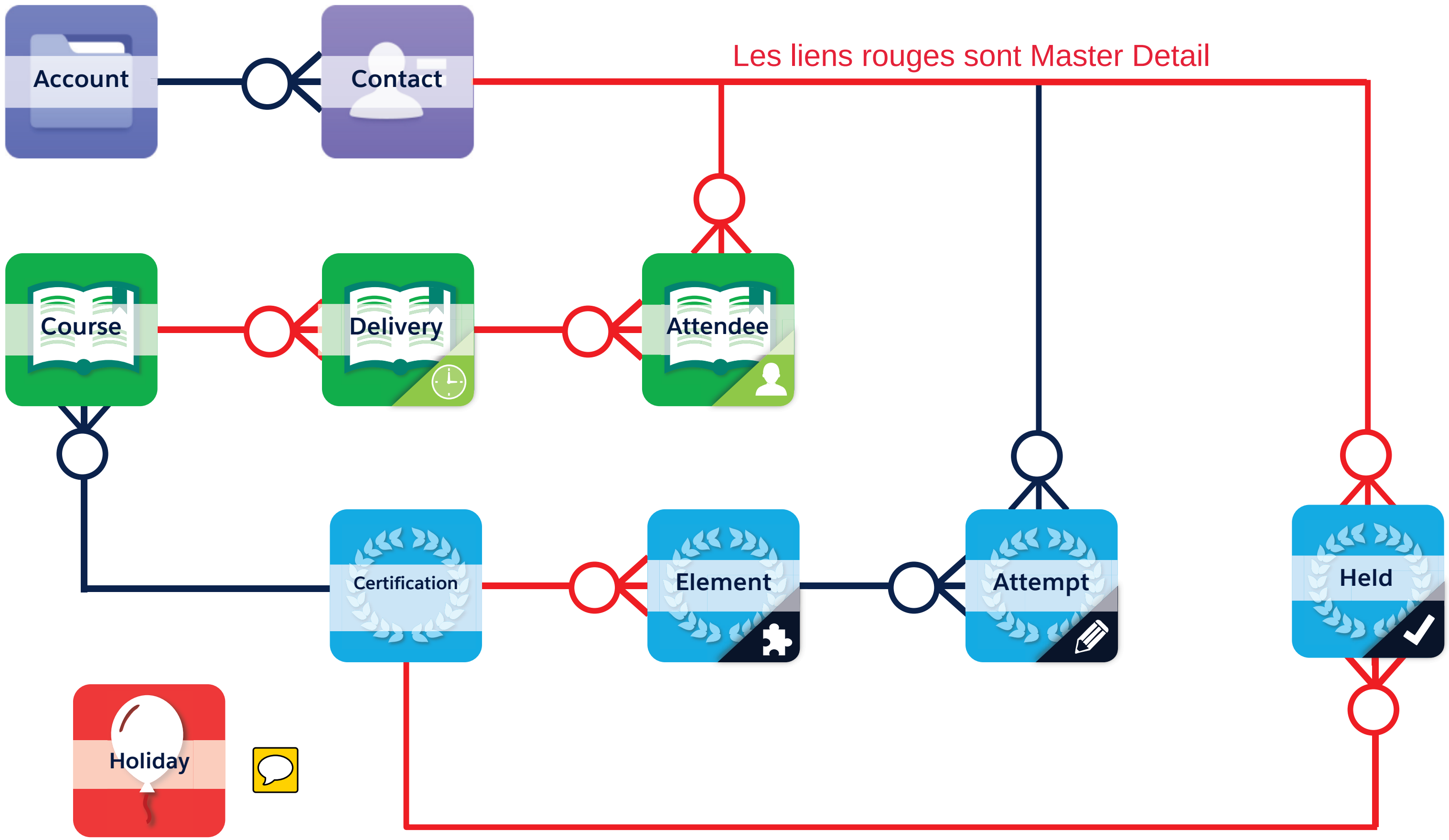
DETERMINING IF A COURSE DELIVERY STARTS ON A HOLIDAY



When a training admin tries to save a course delivery record, the system should ensure that the course delivery does not start on a holiday. Holidays are stored in the system-provided Holiday Object.

WHY CAN'T YOU BUILD THIS LOGIC DECLARATIVELY?

- **Cross-object formula field**
- **Workflow**
- **Validation rule**
- **Lightning process builder**



RESOURCE:

Use this comparison of declarative automation tools (scroll to bottom):
https://help.salesforce.com/apex/HTViewHelpDoc?id=process_which_tool.htm

A TRIGGER PROVIDES A CODE-BASED SOLUTION

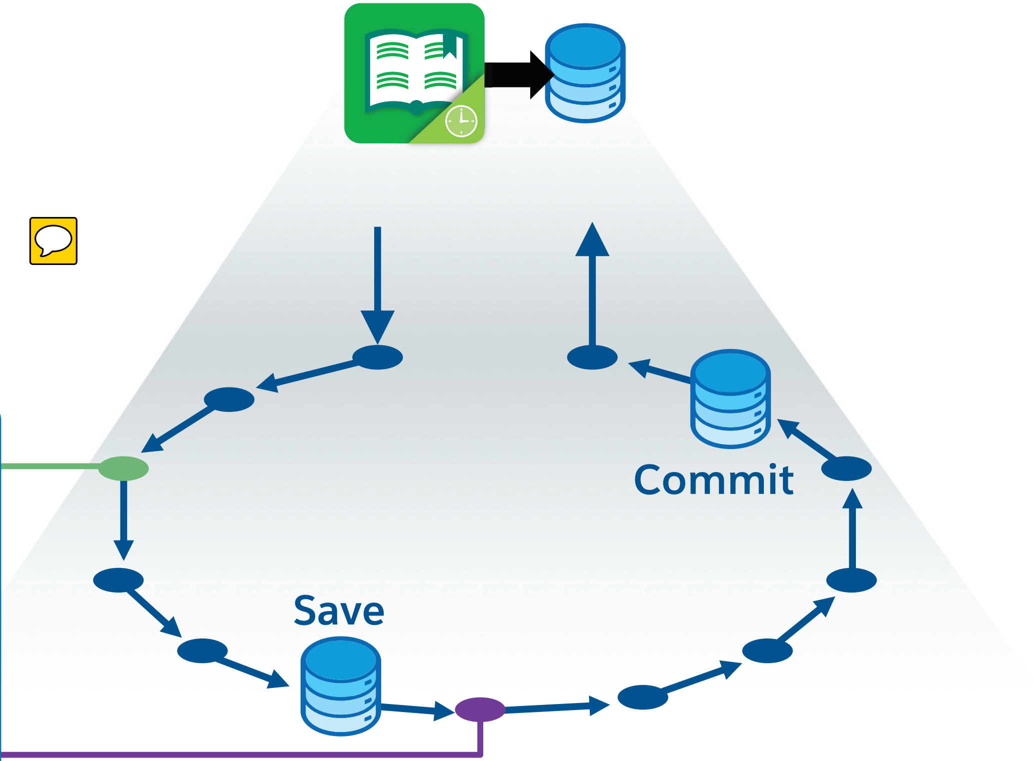
salesforce

DEFINITION:

Trigger: Apex code that is defined on a particular sObject that executes because a DML event has occurred on the corresponding Object.

TRIGGERS

Once the platform receives a request to perform a DML action, the platform executes the many steps of the "Save Order of Execution." Two of the steps execute triggers.



WHAT ARE THE TWO TYPES OF TRIGGERS USED FOR?

DEFINITION:

`before`: triggers are used to update record values.

`after`: triggers are used to access field values, such as Ids, that are set by the system and to effect changes in other records.

What type of trigger would you use to implement the following logic?

When a new Course Delivery has been scheduled, automatically post a Chatter message with a link to the Course Delivery record to the associated instructor.

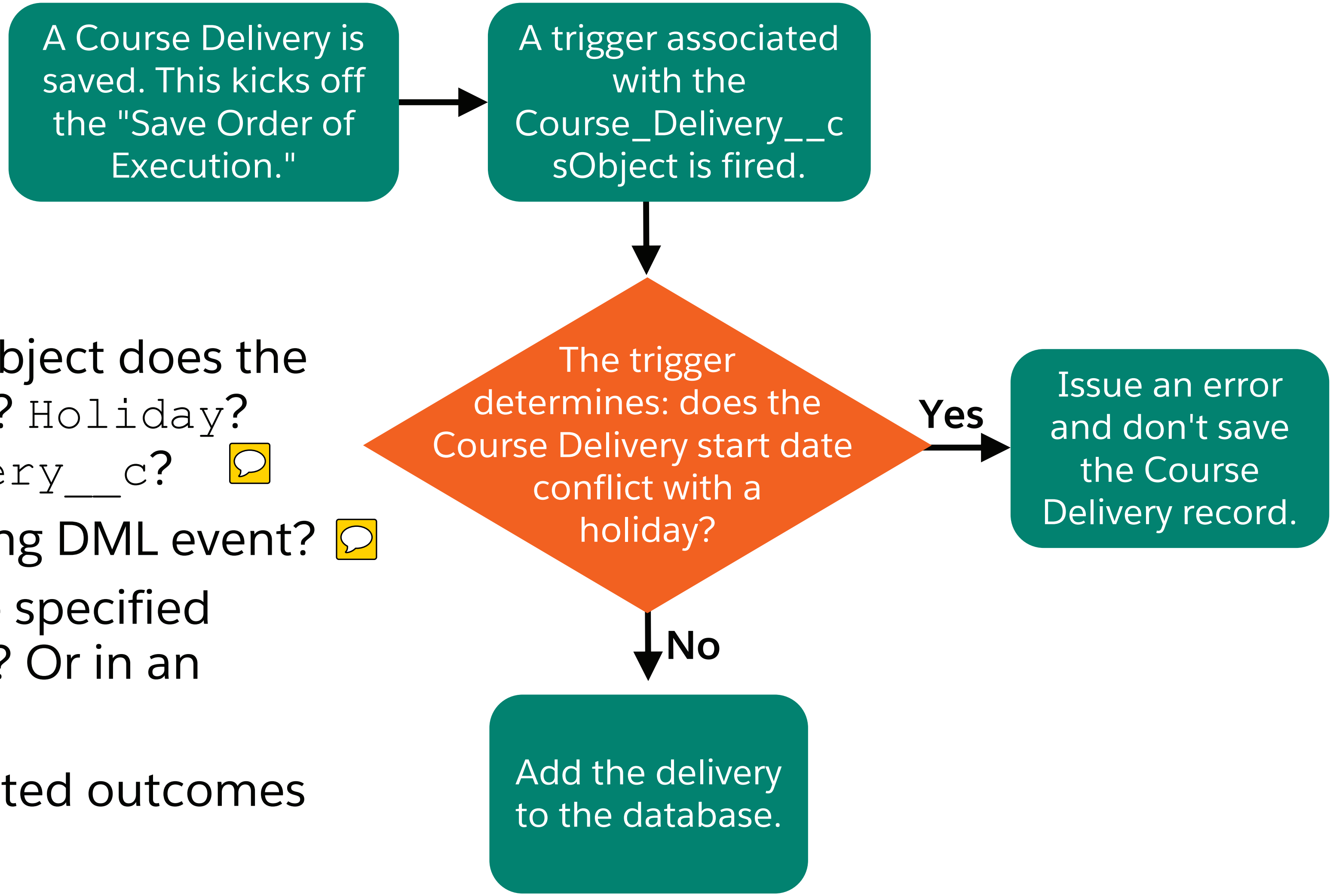
```
Trigger CourseDeliveryTrigger on Course_Delivery__c (before insert, before update) {
        //traitement;
}
```

MODULE AGENDA

MODULE 8: TRIGGER ESSENTIALS

- Automating Logic
- **Defining a Trigger**
- Defining Trigger Logic

FOUR KEY QUESTIONS TO ASK BEFORE WRITING A TRIGGER



1. On what type of sObject does the DML event happen? `Holiday`? Or `Course_Delivery__c`?
2. What is the triggering DML event?
3. Should this logic be specified in a `before` trigger? Or in an `after` trigger?
4. What are the expected outcomes of the trigger?

SYNTAX FOR DEFINING A TRIGGER

keyword

The name of the trigger.

keyword

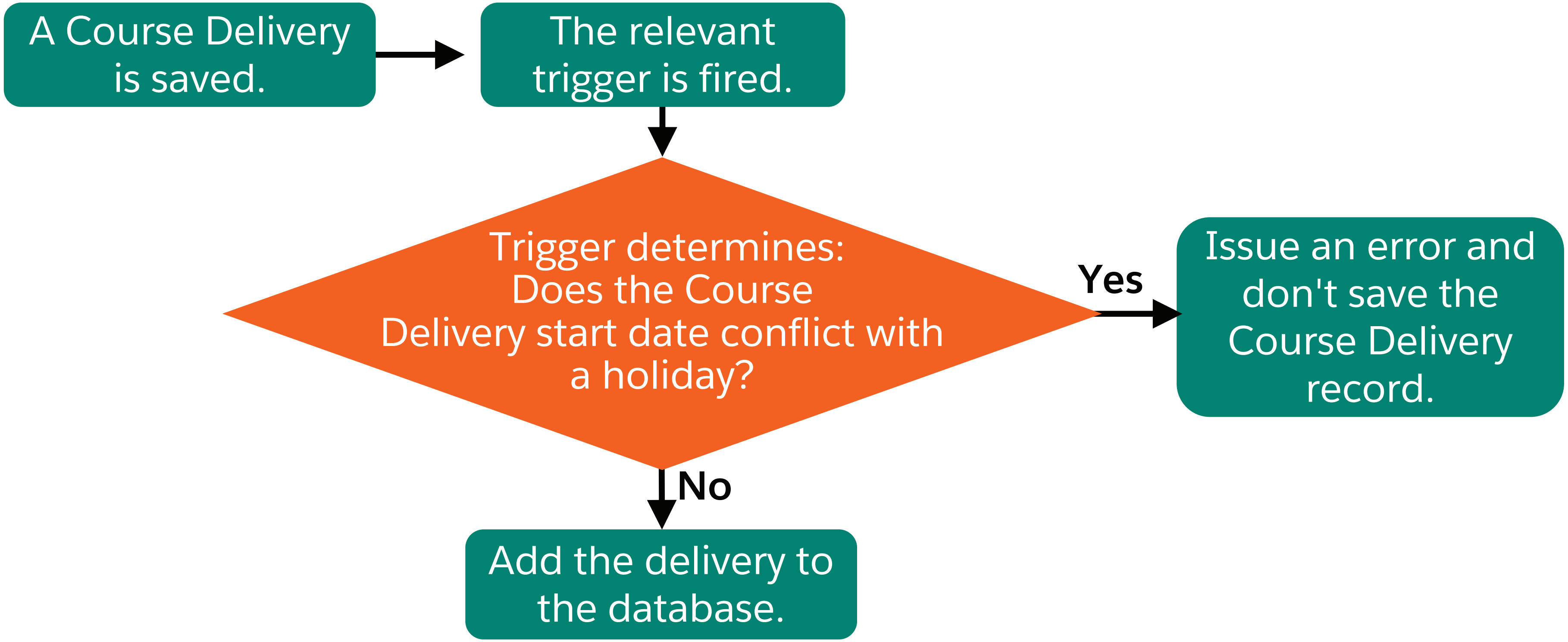
This trigger is a part of the Save Order of Execution for DML events that occur on this sObject.

```
trigger TriggerName on sObject (before insert, before
    update, before delete, after insert, after update, after delete,
    after undelete) {
    //Trigger logic …
}
```

Specify the DML events that fire this trigger.

Specify if this trigger contains logic for a `before` trigger and/or an `after` trigger.

PUT IT ALL TOGETHER: WRITE THE TRIGGER DEFINITION




```
Trigger CourseDeliveryTrigger on Course_Delivery__c (
    Before insert, before update) {
    // Business logic
}
```

YOUR TURN

8-1: DEFINE A TRIGGER

5 minutes

Goal:

Define a trigger on the `Course_Delivery__c` sObject.

Task:

Define a trigger.

MODULE 8: TRIGGER ESSENTIALS

- Automating Logic
- Defining a Trigger
- **Defining Trigger Logic**

TRIGGER LOGIC CAN ACCESS THE TRIGGER CONTEXT

salesforce

Trigger Context Variable	What does it contain?	Where is it available?
`isInsert`, `isBefore`, `is…` (etc.)	Returns true, if the DML operation (e.g., `isInsert`) or timing (e.g., `isBefore`) is accurate for the event.	All triggers
`new`	A list of the new versions of the sObjects.	Insert, update and undelete triggers
`newMap`	A map of the updated versions of the sObjects.	`before update`, `after insert`, `after update`, and `after undelete` triggers
`old`	A list of the previous versions of the sObjects.	Update and delete triggers
`oldMap`	A map of IDs to the previous versions of the sObjects.	Update and delete triggers
`isExecuting`	Returns true if the current context for the Apex code is a trigger, not a Visualforce page, a Web service, or an `executeanonymous` API call.	All triggers
`size`	Total number of sObjects in a trigger invocation.	All triggers

DEFINITION: A trigger has a run-time context which can be accessed using context variables of the `System.Trigger` class. The context contains information about the invoking DML event, the data available to the trigger, and more.

USING CONTEXT VARIABLES TO DETERMINE WHAT LOGIC EXECUTES


```
trigger MyTrigger on MyObject__c (before insert, before update, after update) {
    if (trigger.isBefore) {
        if (trigger.isInsert) {
            // Logic block 1
        }
        if (trigger.isUpdate) {
            // Logic block 2
        }
    } else {
        // Logic block 3
    }
}
```

WHAT IS IN Trigger.New AND Trigger.Old?


```
1  trigger CourseDeliveryTrigger on Course_Delivery__c (before insert, before update) {
2      System.debug('NEW' + trigger.new);
3      System.debug('OLD' + trigger.old);
4  }
```

1. Create a Course Delivery with a start date of February 2, 2020.

2. Insert the Course Delivery.

2A. What will Line 2 print?
2B. What will Line 3 print?

3. Update the inserted Course Delivery's start date to March 2, 2020.

3A. What will Line 2 print?
3B. What will Line 3 print?

WORKING WITH Trigger.new AND Trigger.oldMap

```
trigger CourseDeliveryTrigger on Course_Delivery__c (before
        insert,  before update, before delete, after insert,
        after update) {
        if (trigger.isAfter) {
                if (trigger.isUpdate) {
                        for (Course_Delivery__c cd :trigger.new) {
                                Date oldDate =
                                   trigger.oldMap.get(cd.id).Start_Date__c;
                                if (cd.Start_Date__c != oldDate) {
                                        // … Do some logic
                                }
                        }
                }
        }
}
```

Iterate over `trigger.new` to perform logic on each sObject.

Use `trigger.oldmap` to determine changes between old and new records.

TRIGGERS EXECUTE ON IMPLICITLY BATCHED DATA

How many times will this trigger run if the invoking DML action acted on a list of 200 Course Deliveries? 300?

En fait, le trigger coupe par blocks de 200, même si un seul DML de plus de 200 records.

```
trigger CourseDeliveryTrigger on Course_Delivery__c (before
      insert,  before update, before delete, after insert,
      after update) {
      if (trigger.isAfter) {
            if (trigger.isUpdate) {
                  for (Course_Delivery__c cd : trigger.new) {
                        Date oldDate =
                              trigger.oldMap.get(cd.id).Start_Date__c;
                        if (cd.Start_Date__c != oldDate) {
                              // … Do some logic
                        }
                  }
            }
      }
}
```

oldMap.get(cd.id) récupère l'objet (course_delivery__c) ayant l'id du trigger.new en boucle.
On peut donc interroger .start_date__c

The `trigger.new` list is implicitly batched and contains at most 200 sObjects per iteration of the trigger.

USING AddError TO PREVENT A DML ACTION IN A TRIGGER


```
trigger CourseDeliveryTrigger on Course_Delivery__c (before
      insert,  before update, before delete, after insert,
      after update) {

      for (Course_Delivery__c cd : trigger.new) {
          if (//… Some condition) {
              //prevent the invoking DML action from completing
              cd.addError('This sObject cannot be saved.');
          }
      }
}
```

`sObject.addError` will prevent completion of the DML action.

Cela arrête l'opération immédiatement.

JOIN ME

8-2: DEFINE THE TRIGGER'S BUSINESS LOGIC

15 minutes

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. Create a trigger.
3. Test the trigger’s logic.

KEY TAKEAWAYS

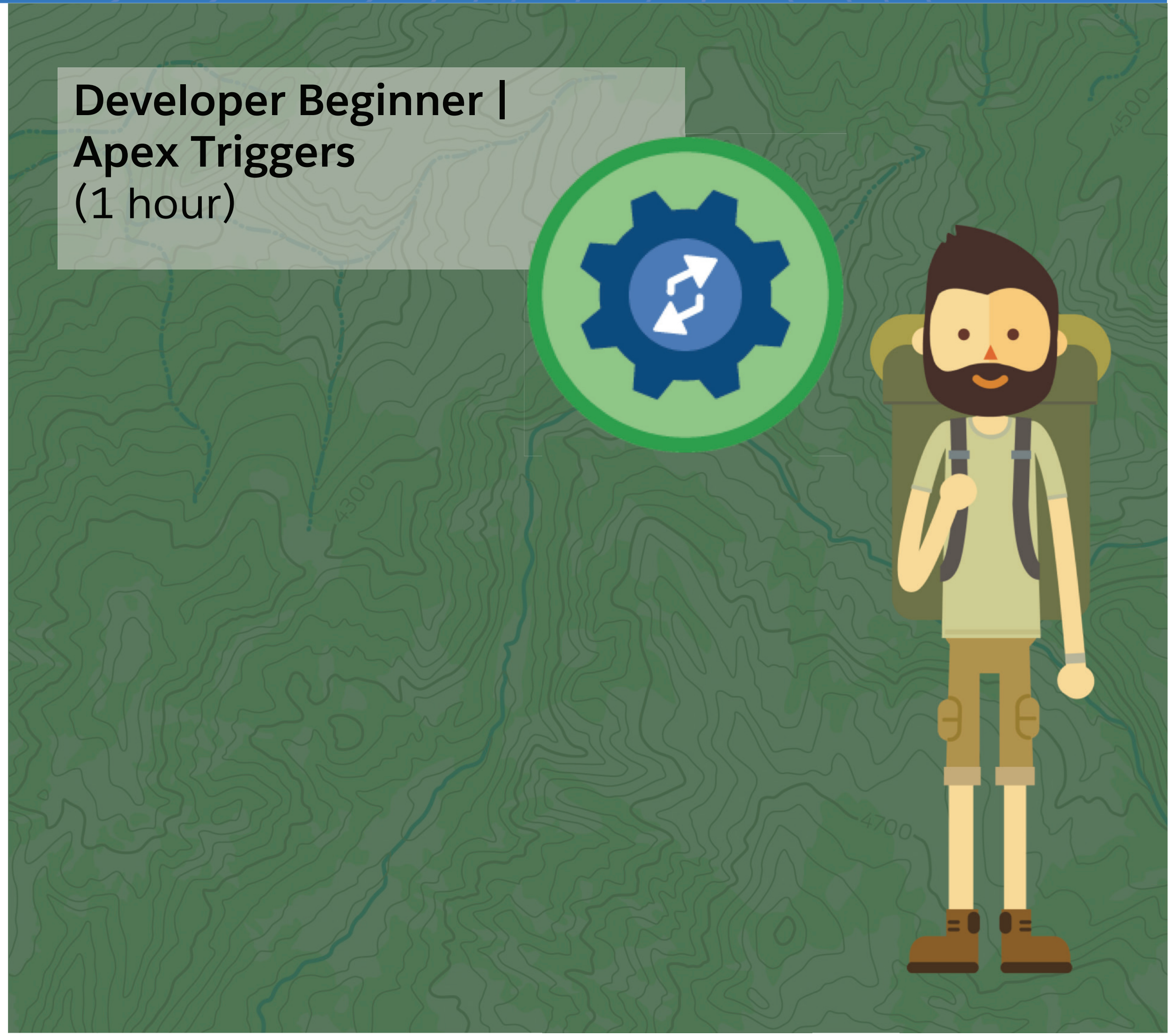
- A trigger can be used to automate business logic, after declarative options have been exhausted.
- `Before` triggers can be used to change field values or perform advanced validation prior to a record saving to the database.

En gros, on peut changer les données après le DML, et avant insertion définitive

- `After` triggers can be used to access a system-generated value, such as an Id, or perform additional DML on related records.
- The `System.Trigger` class has many Boolean variables that can help determine when/why logic in a trigger executes and what data is available to the trigger.
- A trigger has access to, and can sometimes modify, the data upon which the invoking DML action was called.

KNOWLEDGE CHECK

1. Which type of action fires a trigger? DML, sauf les Mass Transfer Records (un employé qui part, on transfère tous ses records)
2. In which type of trigger can the trigger context variable 'old' be used meaningfully? Update, delete
3. In which type of trigger can you edit `trigger.new`? Dans un before, pas dans un after ! Exception erreur de compilation si after.



Developer Beginner |
Apex Triggers
(1 hour)

MODULE 9:
APEX CLASS ESSENTIALS

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

MODULE OBJECTIVES

To ensure business logic in your trigger is easy to read and maintain, you need to move the logic into an Apex class.

To accomplish this, you need to:

- Describe how Apex classes are used.
- Define an Apex class.
- Determine what data an Apex class can access.

MODULE AGENDA

MODULE 9: APEX CLASS ESSENTIALS

- **Using an Apex Class**
- Defining an Apex Class
- Determining Data Access for an Apex Class

WHERE IS A CLASS DEFINED?

A package.

- Managed
- Unmanaged

Your org.

DEFINITION:

A **package** is a distributable container of application components.

APEX CLASS USE CASES

Encapsulating business logic invoked by a trigger:

```
1A public class sObjectTriggerHandler {
2A }
```

Encapsulating reusable test data: generation methods:

```
2A @isTest
2B public class TestDataFactory {
2C }
```

Controlling a Visualforce page:

```
3A public class ACustomController {
3B }
```

Custom Controller (Visual Force)

Testing: Classe de tests

```
4A @isTest
4B private class AClass_Test {
4C }
```

Generally, modeling data and actions:

```
5A public class GeneralClass {
5B      Boolean memberVariable;
5C }
```

Couche business

Implementing inheritance using an interface:

```
1A public interface Paginator {
1B }
```

```
2A public class AccountPaginator implements Paginator {
2B }
```

Implementing inheritance using a virtual class:

Virtual = Héritage ?

```
3A public virtual class CustomPaginator {
3B }
```

```
4A public class AccountPaginator extends CustomPaginator {
4B }
```

NOTE:

In Apex, a data type of one class can be cast to and from a data type of another class, but only if the classes are related through inheritance.

MODULE 9: APEX CLASS ESSENTIALS

- Using an Apex Class
- **Defining an Apex Class**
- Determining Data Access for an Apex Class

DEFINING AN APEX CLASS

Access modifier: who can see this class?

With/without sharing: which records can the class see?

```
public with sharing class MyClass {
    DataType memberVariable;

    DataType memberProperty { get; set; }

    public MyClass() {
        // … Constructor logic
    }

    public void memberMethod() {
        //… Method logic
    }
}
```

A class can contain 0+ member variables.

A class can contain 0+ properties.

A class can contain 0+ constructors.

A class can contain 0+ methods.

ACCESSING AN APEX CLASS OR CLASS MEMBER

Access Modifier Keyword	Applied to a Class	Applied to a Class Member
global Accessible en dehors de SalesForce	▪ Accessible to all Apex code everywhere ▪ Used to define code for asynchronous Apex and services (email, web)	
public Acessible à l'intérieur de l'application	Accessible within your application or namespace	
protected à oublier, cas d'héritage, très peu utilisé dans SF)	Not available	Accessible to any: ▪ Inner classes in the defining Apex class ▪ Classes that extend the defining Apex class
private	▪ Applied to inner classes to make them accessible locally ▪ Can be applied to test classes	▪ The default access modifier ▪ A private member is accessible only within the Apex class in which it is defined

DEFINITION:

Namespace prefixes are used in managed packages to differentiate custom object and field names from those in use by other organizations.

APPLYING `Static` AND `Final` TO A CLASS MEMBER

A class and its methods are implicitly `final` by default (no overridable).

```
public class MyClass {
    public final Integer FINAL_VAR1;
    public final Integer FINAL_VAR2 = 2;
    public final static Integer STATIC_FINAL_VAR3;
    public final static Integer STATIC_FINAL_VAR4 = 4;

    Static {
        STATIC_FINAL_VAR3 = 3;
    }

    Public MyClass() {
        FINAL_VAR1 = 1;
    }
}
```

A `static final` variable can be assigned at declaration or in static initialization.

A `final` class variable can be assigned only at declaration or in a constructor.

Are the access modifiers of the methods of the class correct for this scenario?

```
public with sharing class CourseDeliveryTriggerHandler {

    public static void logicBlock1() {
        // … Logic block 1
    }

    private static void logicBlock2() {
        // … Additional logic
    }
}
```

La classe appellée par le Trigger

En vrai, Private donc ça ne compile pas !

```
trigger CourseDeliveryTrigger on Course_Delivery__c (before insert,  before update) {
    if (trigger.isBefore) {
        if (trigger.isInsert) {
            CourseDeliveryTriggerHandler.logicBlock1();
        }
        if (trigger.isUpdate) {
            CourseDeliveryTriggerHandler.logicBlock1();
            CourseDeliveryTriggerHandler.logicBlock2();
        }
    }
}
```

Le trigger appelant la classe dans son traitement

YOUR TURN

9-1: DEFINE AN APEX CLASS

15 minutes

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.

MODULE AGENDA

MODULE 9: APEX CLASS ESSENTIALS

- Using an Apex Class
- Defining an Apex Class
- **Determining Data Access for an Apex Class**

REVIEWING DATA ACCESS

FLS (Field Level Security)

Profile CRED settings control access to the object, in this case Course Delivery.

Field-level security further defines access to fields.

Course Delivery Number	Course	Location	Start Date	... More fields
Delivery_00000	[101] AWCA Server	Tokyo, JP	2/15/2016	...
Delivery_00001	[101] AWCA Server	San Francisco, CA	6/72016	...
Delivery_00002	[101] AWCA Server	Paris, FR	3/22/2016	...

The Sharing Model determines row-level access.

1. What do Object CRUD and Field-level Security have in common?
2. What operations in Apex have to do with data access?

AN APEX CLASS…

… ignores Object CRED.

… ignores FLS.

Course Delivery Number	Course	Location	Start Date	… More fields
Delivery_00000	[101] AWCA Server	Tokyo, JP	2/15/2016	…
Delivery_00001	[101] AWCA Server	San Francisco, CA	6/72016	…
Delivery_00002	[101] AWCA Server	Paris, FR	3/22/2016	…

… can be programmed to respect or ignore the running user's record-level access during data operations (SOQL, DML, dot notation traversal).

USING THE With/Without Sharing KEYWORD PHRASE

```
1A public with sharing class RespectsSharing {
1B }
```

Respects the Sharing Model for the running user.

```
2A public without sharing class IgnoresSharing {
2B }
```

Ignores the Sharing Model for the running user.

A user who has Read access to 50 out of 110 Account records executes the methods to the right.

1. How many sObjects will be returned by `fetchAccounts()` in the top (A) example?
2. How many sObjects will be returned by `fetchAccounts()` in bottom (B) example?

```
public with sharing class QueryClass {
    public List<Account> fetchAccounts() {
        return [SELECT Id FROM Account];
    }
}
```

```
public without sharing class QueryClass {
    public List<Account> fetchAccounts() {
        return [SELECT Id FROM Account];
    }
}
```

retour de la liste direct

Accès à un champ normalement caché, si accès.

La profil perd son effet, le code prend le dessus sur les droits.

DOES THIS CLASS RESPECT THE RUNNING USER'S SHARING MODEL?


```
1A public class NoKeywordPhraseClass {
1B }
```

When this class is invoked by…	The Sharing Model is…
An anonymous block	Respected
A trigger	Ignored
Another class	• Respected, if the invoking class is "with sharing" • Ignored, otherwise

Les trigger font sauter les restrictions d'accès (à vérifier)

KEY TAKEAWAYS

- Business logic invoked by a trigger should be encapsulated in an Apex class. This is a best practise
- You can implement inheritance among classes using an interface or a virtual class. Et héritage implique possible casts.
- An access modifier determines the visibility of an Apex construct.
- The data that a class has access to is determined by the `with/without sharing` keyword phrase.

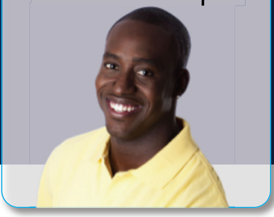
1. Which access modifier denotes a class that is only accessible within your application or namespace? Public
2. Which keyword phrase ensures an Apex class has access to all the records in the database? without sharing (valeur par défaut)
3. What can you use to implement inheritance among Apex classes? Extends
4. Which keyword indicates a variable in a class can only be assigned at declaration or in the class's constructors? Final static (final seul à ne pas faire)

MODULE 10:
THE SAVE ORDER OF EXECUTION AND APEX TRANSACTIONS

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

THE ORDER OF EXECUTION AND APEX TRANSACTIONS

Ryan Jackson
Lead Developer



Some of the triggers our team has written are suddenly not working correctly. Can you spend some time figuring out the issues?

Before you can begin, you need to:

- Describe key points in the Order of Execution.
- Describe how triggers fit into and can be impacted by the
- Order of Execution.
- Describe the lifecycle of an Apex Transaction.
- Describe the memory lifecycle for static variables.

MODULE AGENDA

MODULE 10: THE SAVE ORDER OF EXECUTION AND APEX TRANSACTIONS

- **Exploring the Save Order of Execution**
- Working with Apex Transactions

WATCH ME

10-1: EXPLORE THE IMPLICIT FIRING OF TRIGGERS

10 Minutes

Goal:

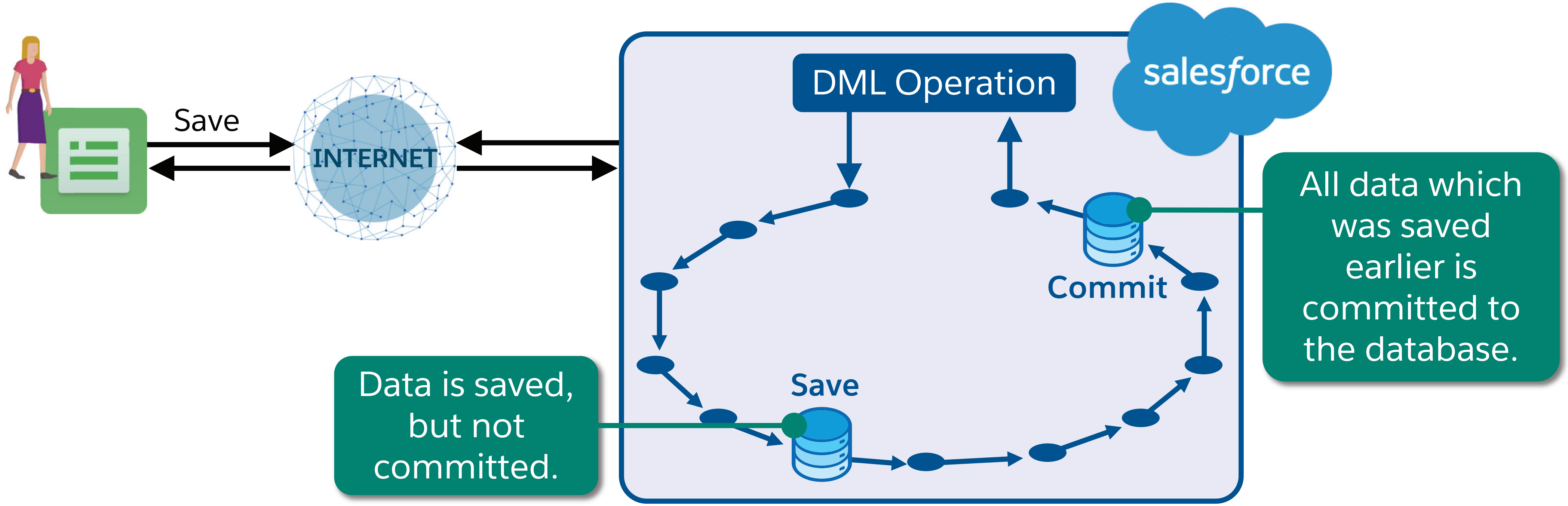
Determine the actions that occur when a Course record is saved.

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.

DEFINITION:

The **Save Order of Execution** describes the series of events that occur on the Force.com platform when a record is saved.



NOTE:

There is a similar series of events for delete and undelete operations.

WHAT BUSINESS PROCESSES ARE PART OF THE SAVE ORDER OF EXECUTION?

Which declarative features might affect the Save Order of Execution?

a) Workflow rules
b) Approval processes
c) Roll-up summaries
d) Formula fields

Which programmatic features might affect the Save Order of Execution?

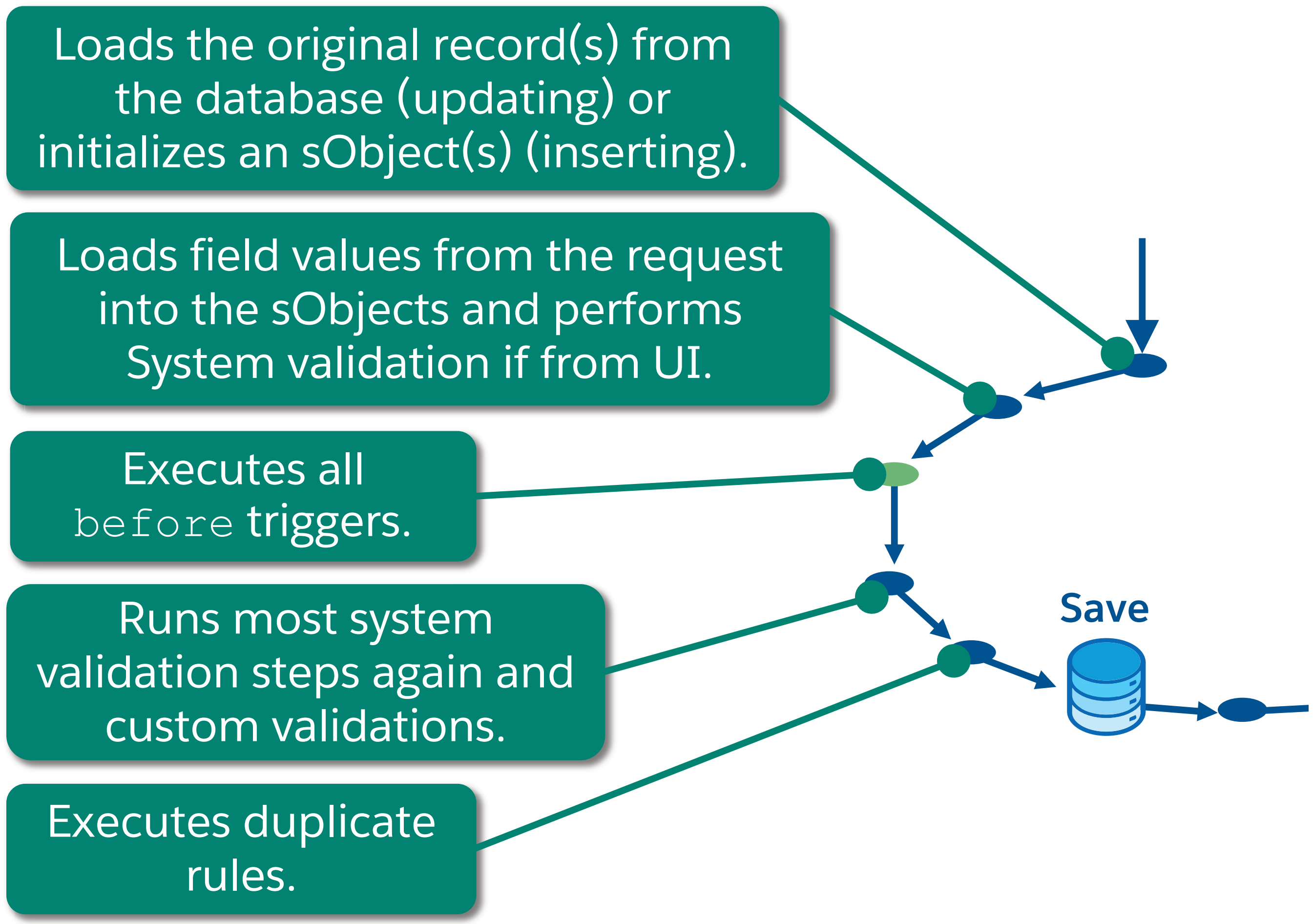
a) Apex code
b) Email sent from Apex code
c) Visualforce pages

WHAT HAPPENS BEFORE THE SAVE TO THE DATABASE?

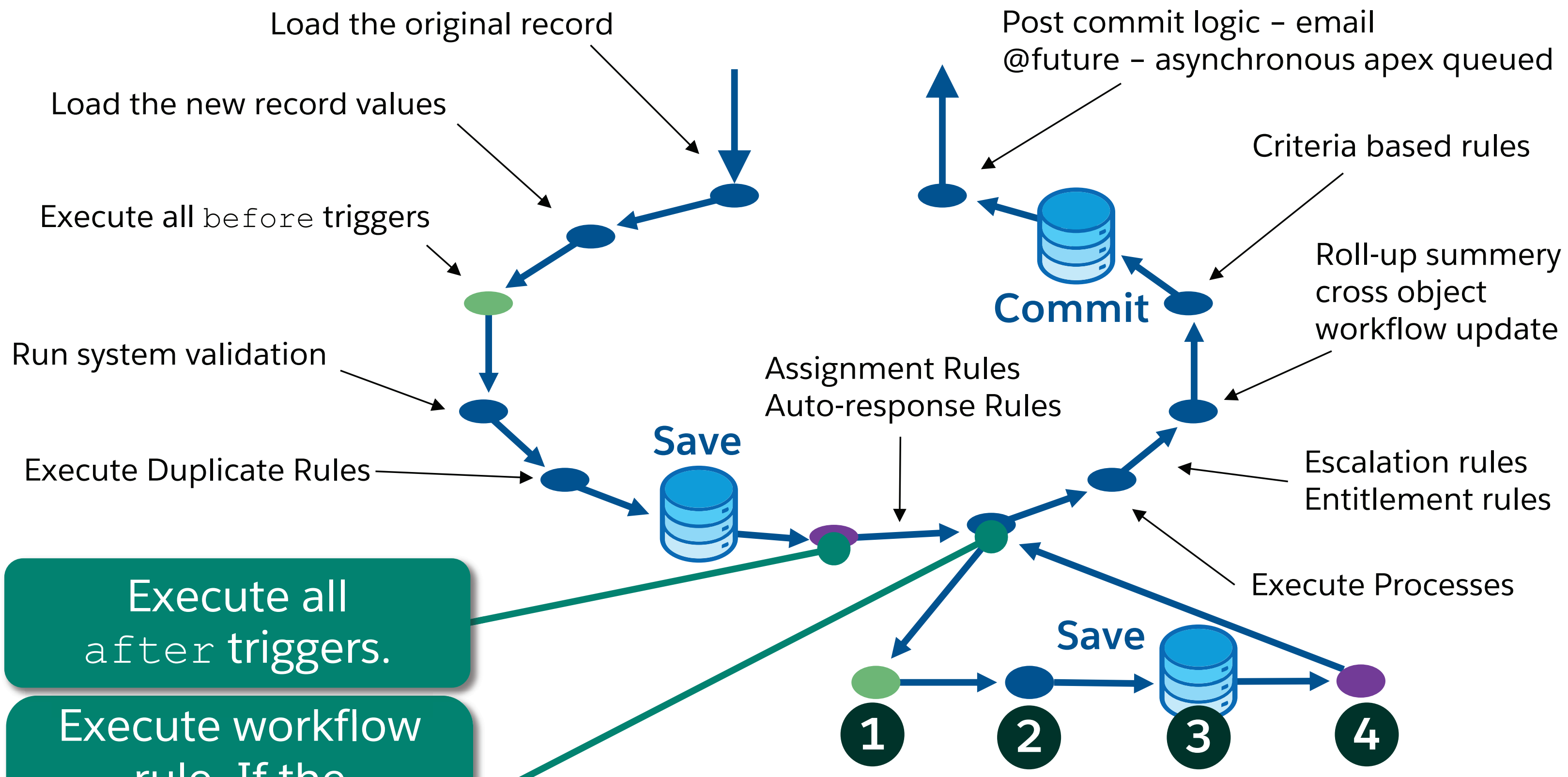
Consider the following scenario:

- The user saves a course delivery record.
- The record passes all system validations.
- A before trigger is executed.
- System validation fails.

What event is likely to have caused the data to become invalid?



WHAT HAPPENS AFTER THE SAVE TO THE DATABASE? PART 1: TRIGGERS AND WORKFLOW RULES



1 Execute all `before update` triggers.

2 Execute most System validations (no custom).

3 Save the updated record to the database.

4 Execute all `after update` triggers.

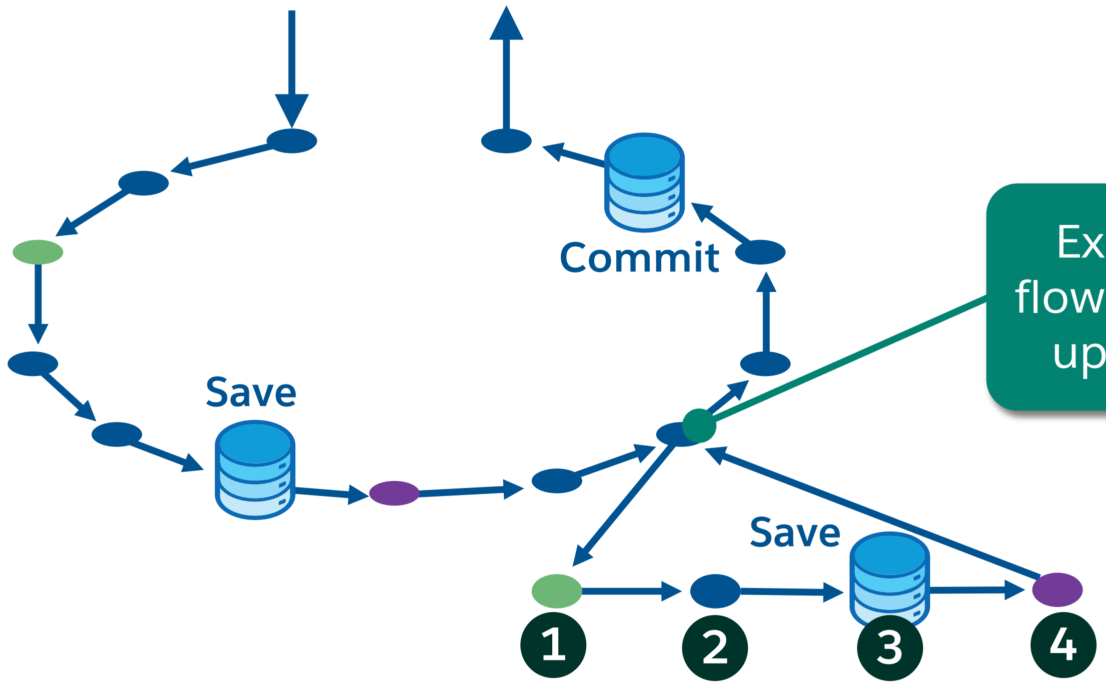
Execute all `after` triggers.

Execute workflow rule. If the workflow updates a field...

The execution of these `before` and `after` triggers can cause additional workflow rules to execute, causing these steps to begin again.

Custom validation rules, duplicate rules and escalation rules are not run again.

WHAT HAPPENS AFTER THE SAVE TO THE DATABASE? PART 2: TRIGGERS AND PROCESSES

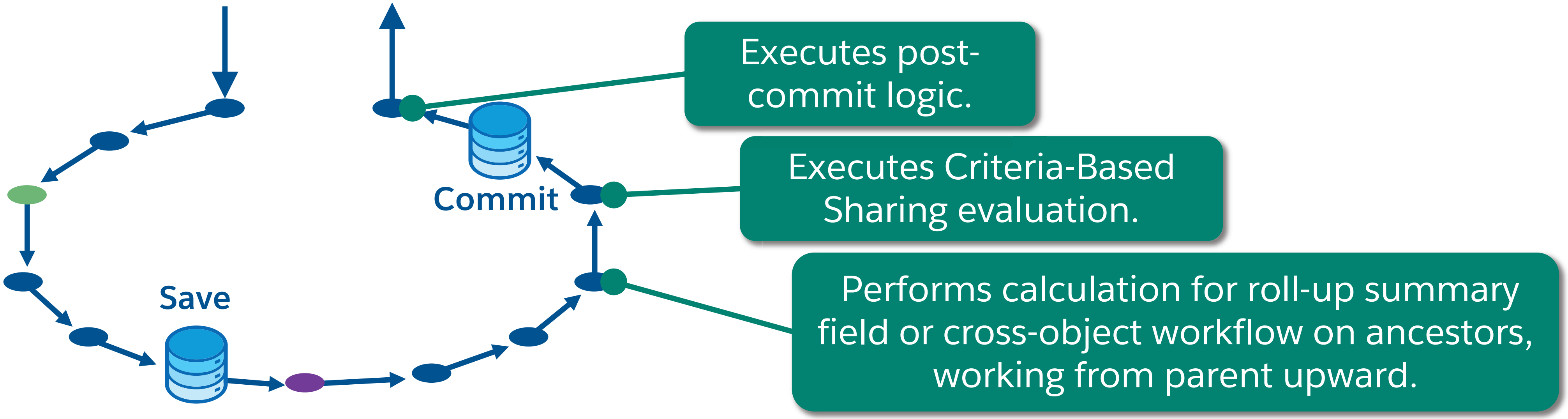


Execute process flow rule. If the flow updates a field...

1. Execute all `before update` triggers.
2. Execute most System validations (no custom).
3. Save the updated record to the database.
4. Execute all `after update` triggers.

The execution of these `before` and `after` triggers can cause processes to execute against the same record in the same transaction if the option "Reevaluate Records in the Process Builder" is chosen.

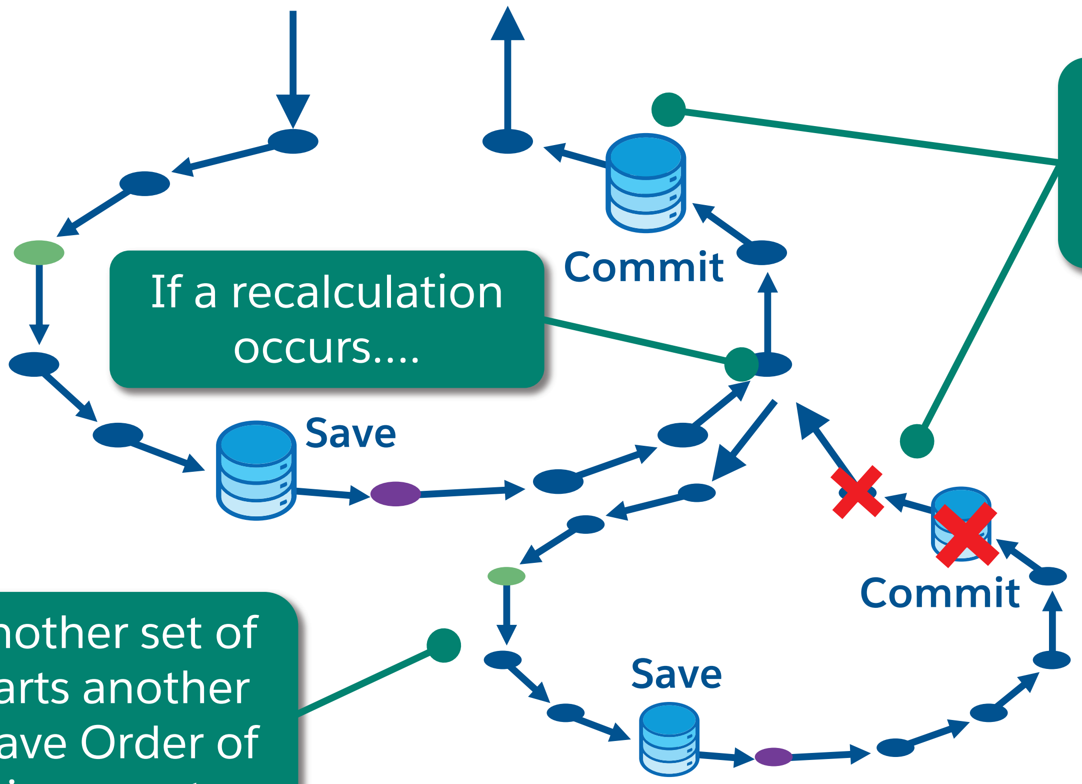
WHAT HAPPENS AFTER THE SAVE TO THE DATABASE? PART 3



What are the consequences of calculating a roll-up summary field?

WHAT HAPPENS WHEN ANOTHER DML SAVE OPERATION IS PERFORMED?

salesforce



Commit and post-commit events occur just once, at the end of the current process.

If a recalculation occurs....

Then another set of data starts another set of Save Order of Execution events.

If a trigger updates a record, which causes another trigger to execute and update the original record, you can potentially get into a recursive situation.

Situation récursive potentielle.

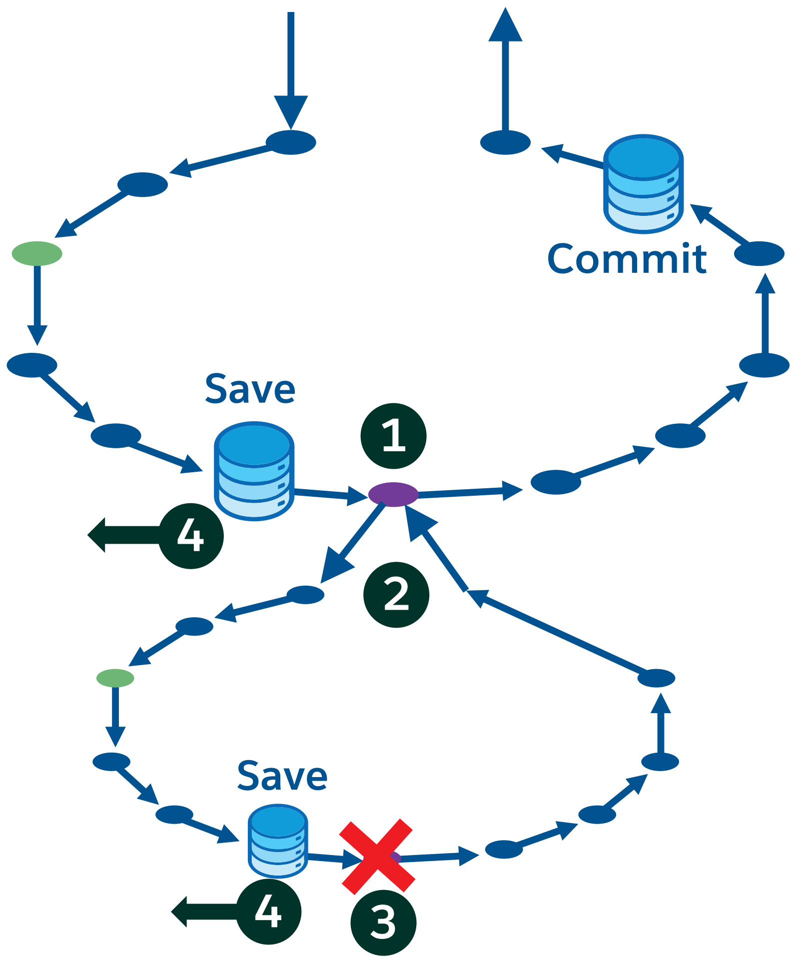
WHAT EVENTS IN THE SAVE ORDER OF EXECUTION CAUSE A NEW DML EVENT?

- Before triggers?
- System validation rules?
- Custom validation rules?
- Duplicate rules?
- After triggers?
- Workflows?
- Process flows?
- Calculations for roll-up summary fields?
- Cross-object workflows?
- Evaluation of criteria-based sharing?

WHAT MIGHT CAUSE THE COMMIT TO NOT HAPPEN?

- A system validation?
- A validation rule?
- A trigger?
- A workflow rule?
- A process/flow?

WHAT HAPPENS IF THERE IS A FAILURE?



1. An `after` trigger attempts to insert sObjects of another type.
2. The insert initiates a new Save Order of Execution loop.
3. An `after` trigger in the secondary loop fails (with no try-catch).
4. All changes to the database are rolled back and no further events in the Save Order of Execution are performed.

WHAT CONDITIONS WITHIN APEX CAN CAUSE A ROLLBACK?

In Apex, you want to insert a set of Course records and allow a partial save if some of the records are valid. What form of `insert` should you use?

Database.insert(;.. , false) ?

- An unhandled exception.
- When Apex adds an error to an sObject or field during an AllOrNone DML operation.

```
…
//within a trigger
//don't allow courses with students to be cancelled.
for (Course__c c : coursesWithStudents) {
    c.addError('Course has enrolled students');
}
…
```

YOUR TURN

10-2: VIEW THE EVENTS THAT OCCUR DURING A ROLLBACK

salesforce

10 Minutes

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.

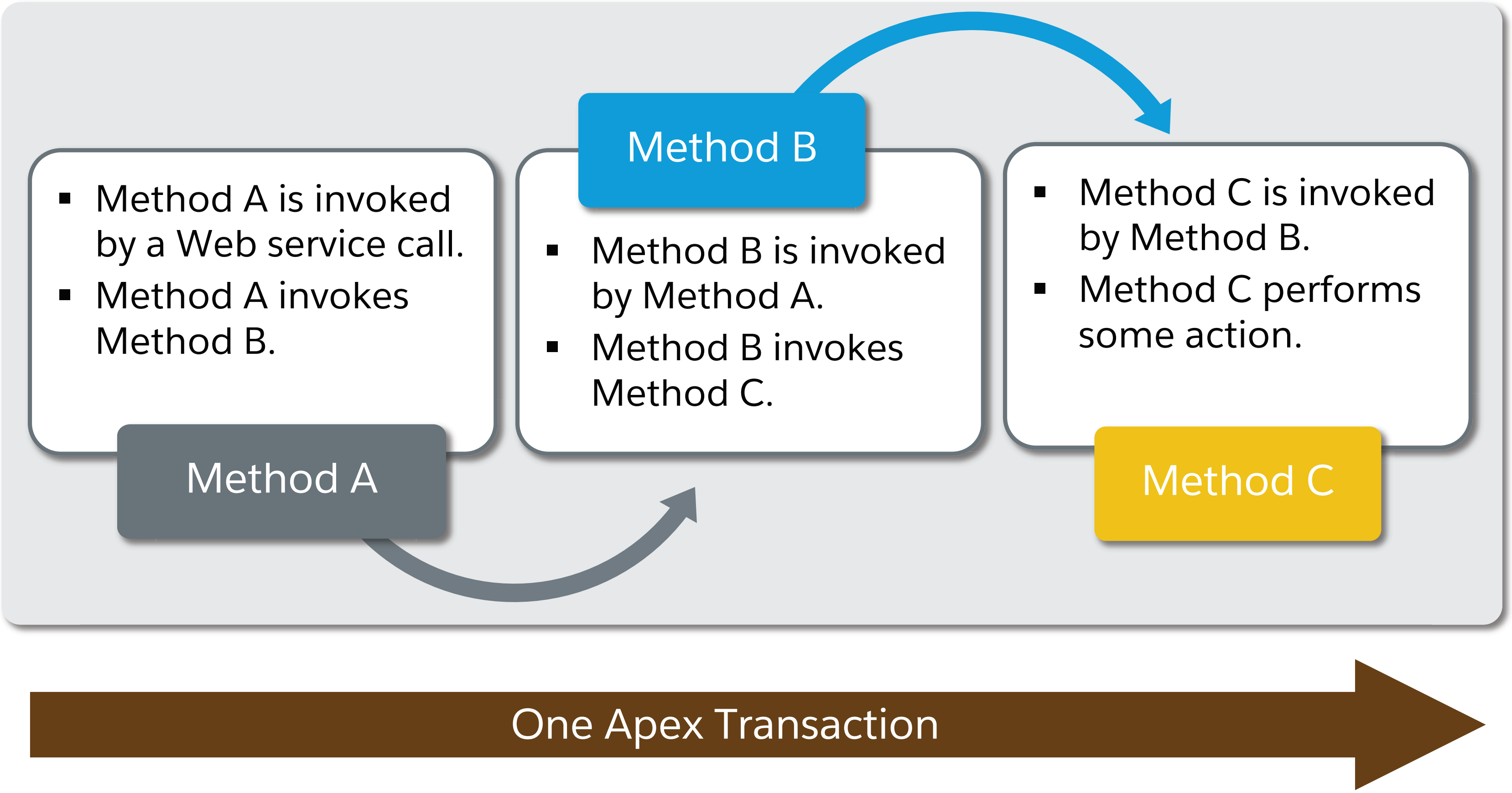
MODULE 10: THE SAVE ORDER OF EXECUTION AND APEX TRANSACTIONS

- Exploring the Save Order of Execution
- **Working with Apex Transactions**

WHAT IS AN APEX TRANSACTION?

DEFINITION:

An **Apex transaction** represents a set of operations that are executed as a single unit.



LIMITS!

- Most Apex limits are per Apex transaction.
- Limits may change with each release.

Code defensively!

```
Integer queryRowsRemaining = Limits.getLimitQueryRows() - Limits.getQueryRows();
List<Course_Attendee__c> courseAttendees = [SELECT ID, Name, Status__c
                                            FROM Course_Attendee__C
                                            LIMIT :queryRowsRemaining];
if (courseAttendees.size() == queryRowsRemaining) {
    displayErrorMessage('Refine your query criteria');
} else {…
```

ALERT: If you exceed a governor limit, your code will terminate with an unrecoverable exception.

APEX TRANSACTIONS THAT INVOKE TRIGGERS: PART 1

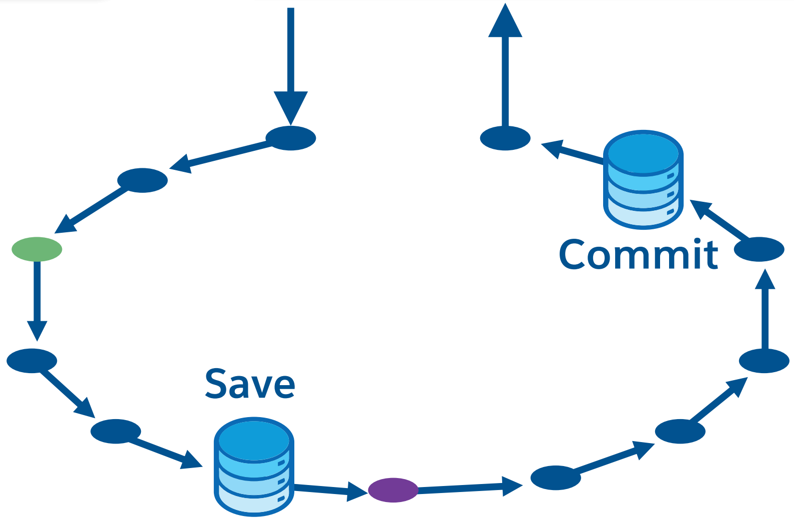
One Apex Transaction

Execute anonymous invokes `ManageContact.load();`

`ManageContact.load();` has statement to insert contacts.

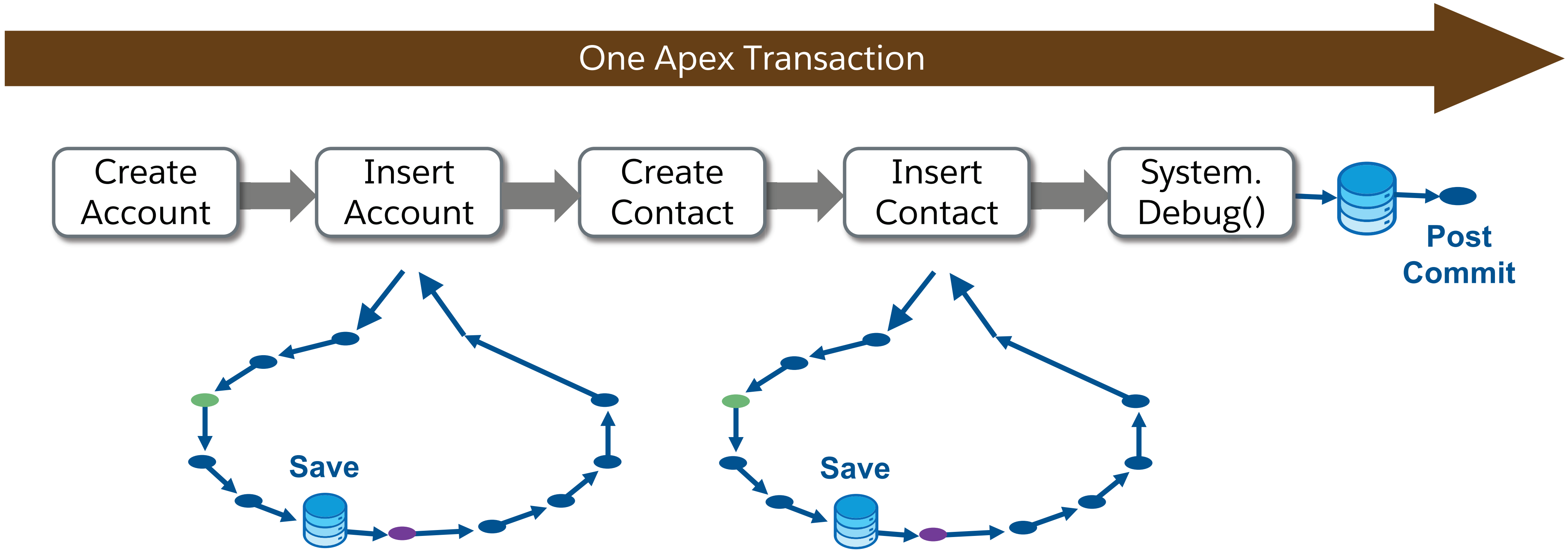
Insert statement causes steps in order of execution to begin.

L'Apex transaction, commence avant la trame ! (avant un insert etc)



The Apex transaction will include all Apex run during the steps of the order of execution, including any sub-processes.

APEX TRANSACTIONS THAT INVOKE TRIGGERS: PART 2



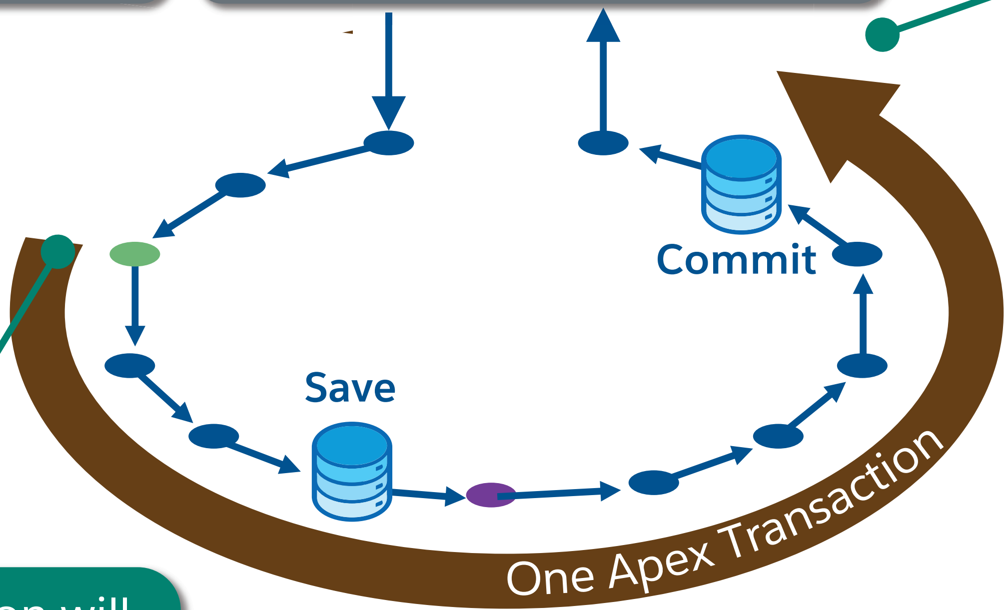

```
Account newAccount = new Account(Name = 'Acme');
insert newAccount;
Contact newContact = new Contact(LastName = 'Manning', AccountId = newAccount.Id);
insert newContact;
System.debug('About to commit...');
```

APEX TRANSACTIONS THAT BEGIN WITH TRIGGERS

User saves a record.

Insert statement causes order of execution to begin.

The Apex transaction ends after the commit is performed.



The Apex transaction will begin with the first Apex that *executes* during the steps of the Save Order of Execution.

WHAT WILL THIS APEX TRANSACTION INCLUDE?

Consider these events:

- A user saves an Account record.
- The `after` trigger on Account updates Contact records.
- The `before` trigger on Contact executes.
- A workflow rule on Contact fires and updates a field on the Contact records.
- The `before` trigger on Contact executes again.
- There are no errors or exceptions during the events in the order of execution.

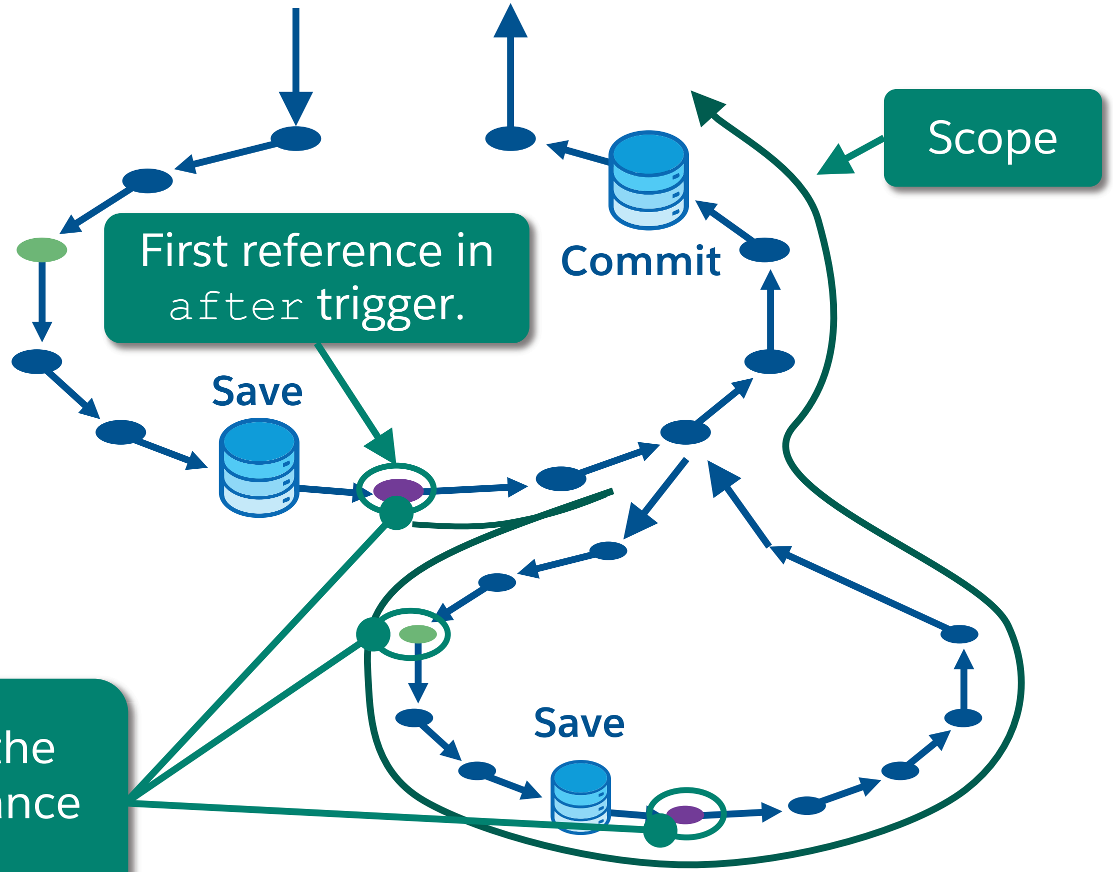
1. When does the Apex transaction begin?
2. Which triggers are considered part of the transaction?
3. When does the Apex transaction end?
4. When is the Account record committed to the database?
5. When are the Contact records committed to the database?

STATIC ATTRIBUTES AND APEX TRANSACTIONS

A static attribute within a class:

- Is loaded into the Apex transaction context when it is first referenced.
- Remains in scope until the Apex transaction completes.
- Is often used to identify and limit recursion in triggers.

Utilisation de variables statiques pour compter, par exemple :
Limit.getSOQLQuery() return le nombre de SOQL consommées à moment n.



Any references to the attribute within the Apex transaction refer to the same instance of the attribute.

Limit.getLimitSOQLQuery() renvoie le nombre total dispo au début (100 en synchrone)

On peut faire :
Limit.getLimitSOQLQuery()-Limit.getSOQLQuery() pour avoir le décompte du restant.

WATCH ME

10-3: SEE THE SAVE ORDER OF EXECUTION IN ACTION

salesforce

10 Minutes

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.

SUMMARY

- The save order of execution:
  - Includes the execution of triggers.
  - Retains an implicit savepoint at the beginning of a database transaction for potential rollback.
- An Apex transaction that involves DML operations will end after a commit to the database.
- A static variable within a class remains in scope for the duration of an Apex transaction.

All Apex code should consider the interaction between all events that are part of the order of execution.

Developer Beginner | Apex & .NET Basics
(1 hours, 15 minutes)

Developer Beginner | Developer Console Basics
(1 hour, 25 minutes)

MODULE 11:
TESTING ESSENTIALS

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

MODULE OBJECTIVES

Before you can deploy your code to production, you need to test the trigger that you wrote to ensure that a course is not scheduled on a holiday.

To accomplish this, you need to:

- Describe Apex's testing framework.
- Create test data.
- Write and run an Apex test.

MODULE 11: TESTING ESSENTIALS

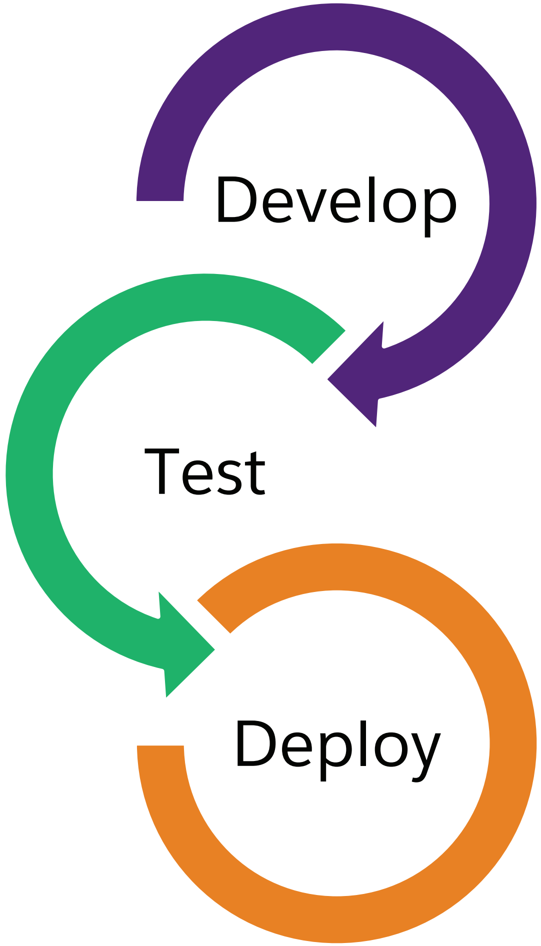
- **Describing Apex's Testing Framework**
- Creating Test Data
- Writing and Running an Apex Test

Story

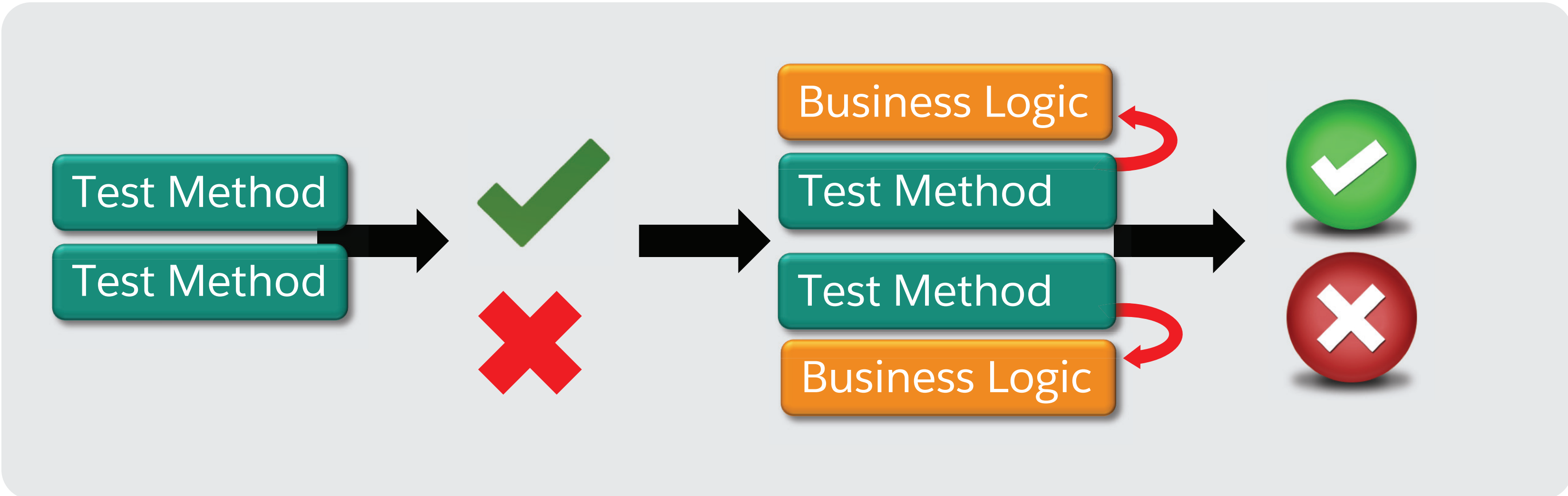
TESTING APEX CODE IS A PART OF THE SALESFORCE ALM

In our sandbox, I've created a number of triggers and classes for the certification application. Before I can deploy this code to production, I need to ensure:

- 75% of Apex code must be executed successfully by test methods.
- Every Apex trigger must have some coverage in test methods, even though every line of every trigger does not have to be executed by a test.
- Every Apex test method must execute without throwing any uncaught exceptions or exceeding governors.



THE PLATFORM INCLUDES A TESTING FRAMEWORK



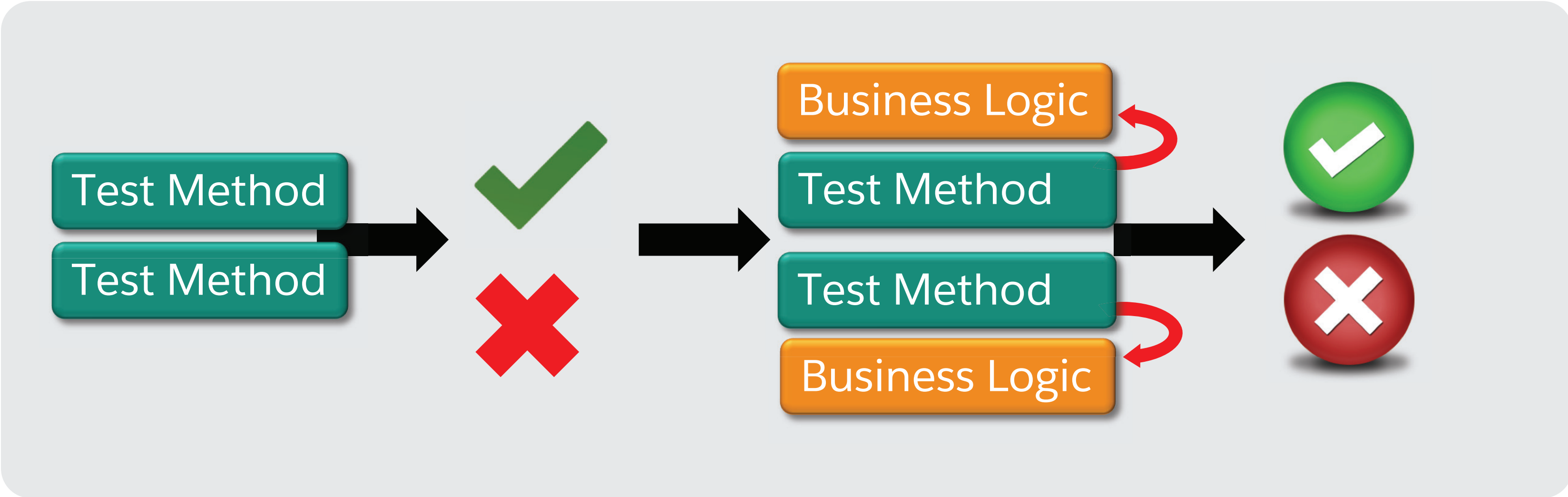
A developer writes test methods inside test classes.

Salesforce receives a specific command to run tests.

Salesforce runs test methods and any code they invoke.

Salesforce reports the success or failure of the tests, along with test coverage percentages.

WHAT IS AN APEX TEST METHOD?




```
@isTest
private class MyBusinessLogicClass_Test {
    private static testMethod void myBusinessLogicTest() {
        //… test Apex classes and triggers
    }
}
```

DEFINITION:

An **Apex test method** verifies whether a particular piece of Apex code is working as expected.

MODULE 11: TESTING ESSENTIALS

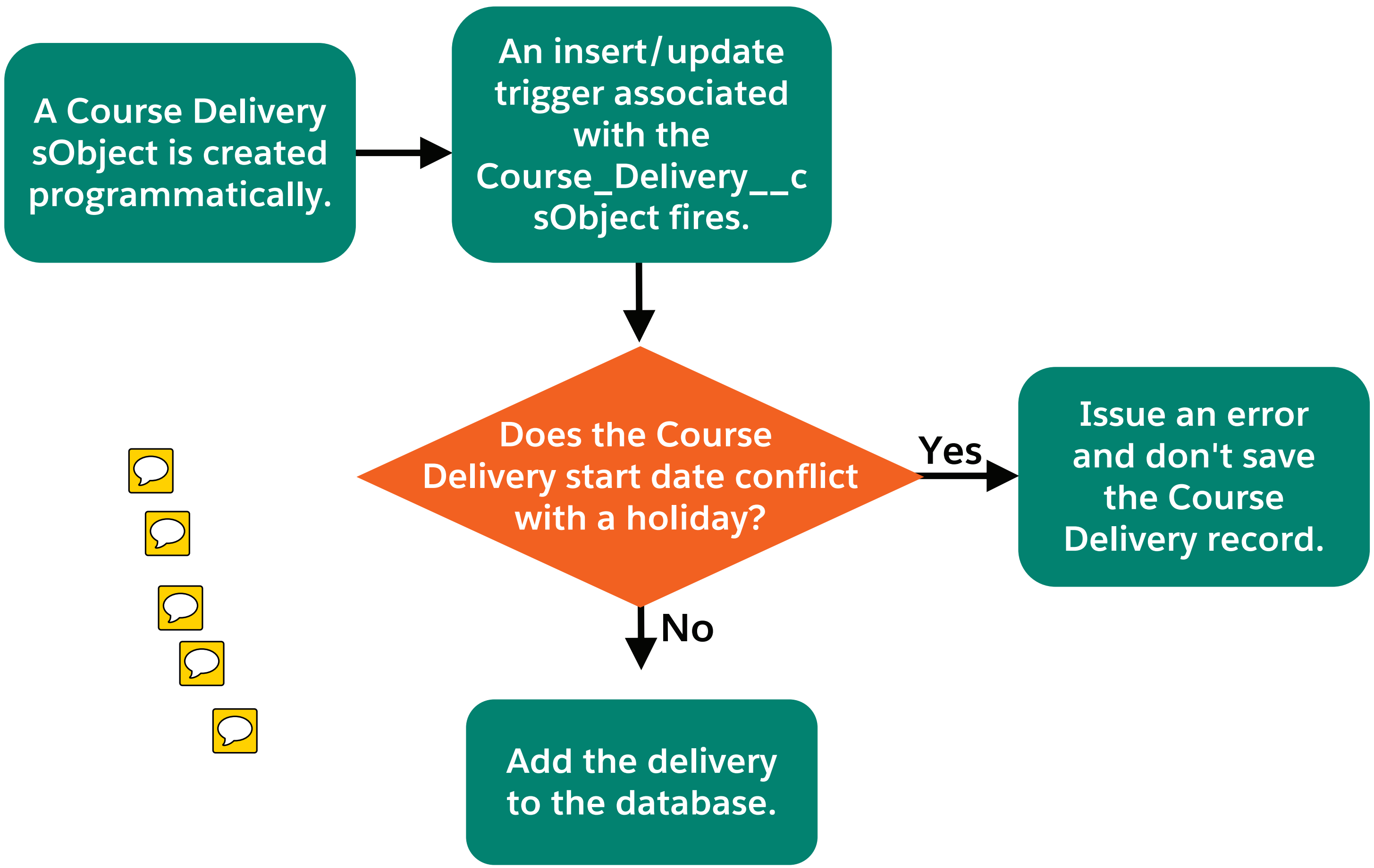
- Describing Apex's Testing Framework
- **Creating Test Data**
- Writing and Running an Apex Test

Trailhed - Dévl débutant
https://trailhead.salesforce.com/fr/trails/force_com_dev_beginner/modules/apex_testing/units/apex_testing_intro

What test data do you need:

1. In order to test both branches of this logic?
2. If the first green box is changed to: "Several new Course Deliveries are created programmatically"?
3. If the first green box is changed to: "A Course Delivery's start date is updated programmatically"?

On doit créer notre jeu de données, les tests ignorent les données existantes, base vierge.



LOADING TEST DATA DECLARATIVELY USING A STATIC RESOURCE

Loads test data from a Static Resource.

```
List<Holiday> holidays = Test.loadData(Holiday.sObjectType, 'Test_Holidays');
```

Test.loadData() pour charger direct par ex un CSV

Static Resource

Static Resource name = 'Test_Holidays'

```
Name,ActivityDate
Holiday 1,2016-01-01
Holiday 2,2016-07-04
Holiday 3,2016-11-11
```

First line must be column names.

CSV file

LOADING RELATED DATA USING A STATIC RESOURCE

salesforce

User-specified fake Id field and data.

Parent: Course	
Id	**Name**
1	Test Course 1
2	Test Course 2

Child: Course Delivery		
Course__c	**Id**	**Name**
1	101	Test Course Delivery 1
2	201	Test Course Delivery 2
2	301	Test Course Delivery 3

User-specified foreign-key field and data.

CREATING TEST DATA IN A TEST FACTORY CLASS VS. METHOD

- Test data "factories" can be made accessible to all test methods within an org or just to test methods in a particular test class.
- When localized, a test data factory method annotated with `@testSetup` executes before any other method in the test class and provides data to all the test methods in the test class.

When would you use the test data factory class over the method, and vice versa?

```
@isTest
public class MyTestDataFactory {
     public static void insertTestAccounts() {
          //… create and insert Accounts
     }
     public static void insertTestContacts() {
          //… create and insert Contacts
     }
     public static void insertTestHolidays() {
          //… create and insert Holidays
     }
}
```

```
@isTest
private class MyBusinessLogicClass_Test {
     @testSetup
     private static void myTestDataSetupMethod() {
          //… create and insert Accounts
          //… create and insert Contacts
          //… create and insert Holidays
          //… Etc…
     }
     private static testMethod void myTestMethod1() {
          //… Use the test data created above
     }
     private static testMethod void myTestMethod2() {
          //… Also uses the test data created above
     }
}
```

DOES TEST DATA PERMANENTLY CHANGE THE DATABASE?

```
1  @isTest
2  private class MyBusinessLogicClass_Test {
3      @testSetup
4      private static void myTestDataFactory() {
5          Course__c aCourse = new Course__c(Name = 'Course 1');
6          insert aCourse;
7          List<Course__c> courses = [SELECT Name FROM Course__c];
8          System.debug(courses.size());
9      }
10     private static testMethod void myTestMethod1() {
11         Course__c aCourse = new Course__c(Name = 'Course 2');
12         insert aCourse;
13         List<Course__c> courses = [SELECT Name FROM Course__c];
14         System.debug(courses.size());
15     }
16     private static testMethod void myTestMethod2() {
17         List<Course__c> courses = [SELECT Name FROM Course__c];
18         System.debug(courses.size());
19     }
20 }
```

1. What number will the debug statement on Line 8 print? 1
2. What number will the debug statement on Line 14 print? 2
3. What number will the debug statement on Line 18 print? 1
4. Outside of this test class, which of the inserted Courses exist?

Rolleback entre toutes les méthodes, mais une méthode qui en appelle d'autres, pas de rolleback jusqu'à la fin de la méthode appelante, ex :

USING LIVE DATA TO TEST

The class and method annotation `@isTest(seeAllData = true)` allows you to create test classes and test methods that have access to all data in the org.

Why do you think this is not a standard practice?

```
@isTest(seeAllData = true)
private class MyBusinessLogicClass_Test {
    private static testMethod void myBusinessLogicTest() {
        List<Holiday> holidays = [SELECT … ];
    }
}
```

JOIN ME

11-1: MAKE TEST DATA AVAILABLE TO TEST METHODS

10 minutes

Goal:

Create test data to test the Certification application.

Tasks:

1. Create test data using a Static Resource.
2. Create test data programmatically.

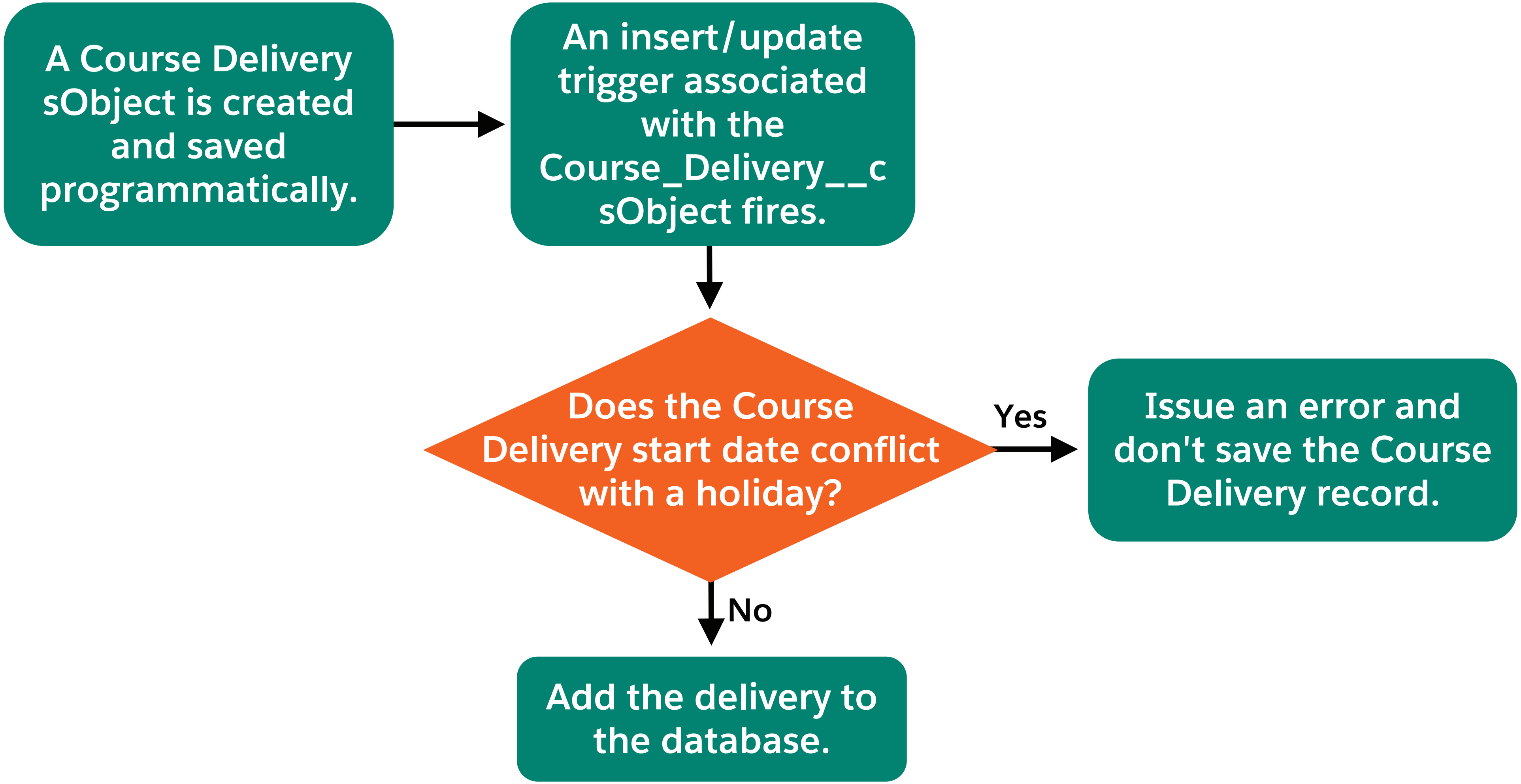
MODULE AGENDA

salesforce

MODULE 11: TESTING ESSENTIALS

- Describing Apex's Testing Framework
- Creating Test Data
- **Writing and Running an Apex Test**

WHAT EXPECTED OUTCOMES DO YOU NEED TO TEST?



AN APEX TEST METHOD INSIDE AN APEX TEST CLASS

Tells the platform the following class will only be used for testing.

Use `_Test` as the suffix for the name of a test class (recommended).

A test class should be `private`.

```
@isTest
private class MyBusinessLogicClass_Test {
    @testSetup
    private static void myTestDataFactory(){
        //… Create local test data
    }
    private static testMethod void myBusinessLogicTest(){
        //… test saved Apex code
    }
}
```

A test method must be static.

`testMethod` tells the testing framework that this method is a test. Not all methods in a test class need to be test methods.

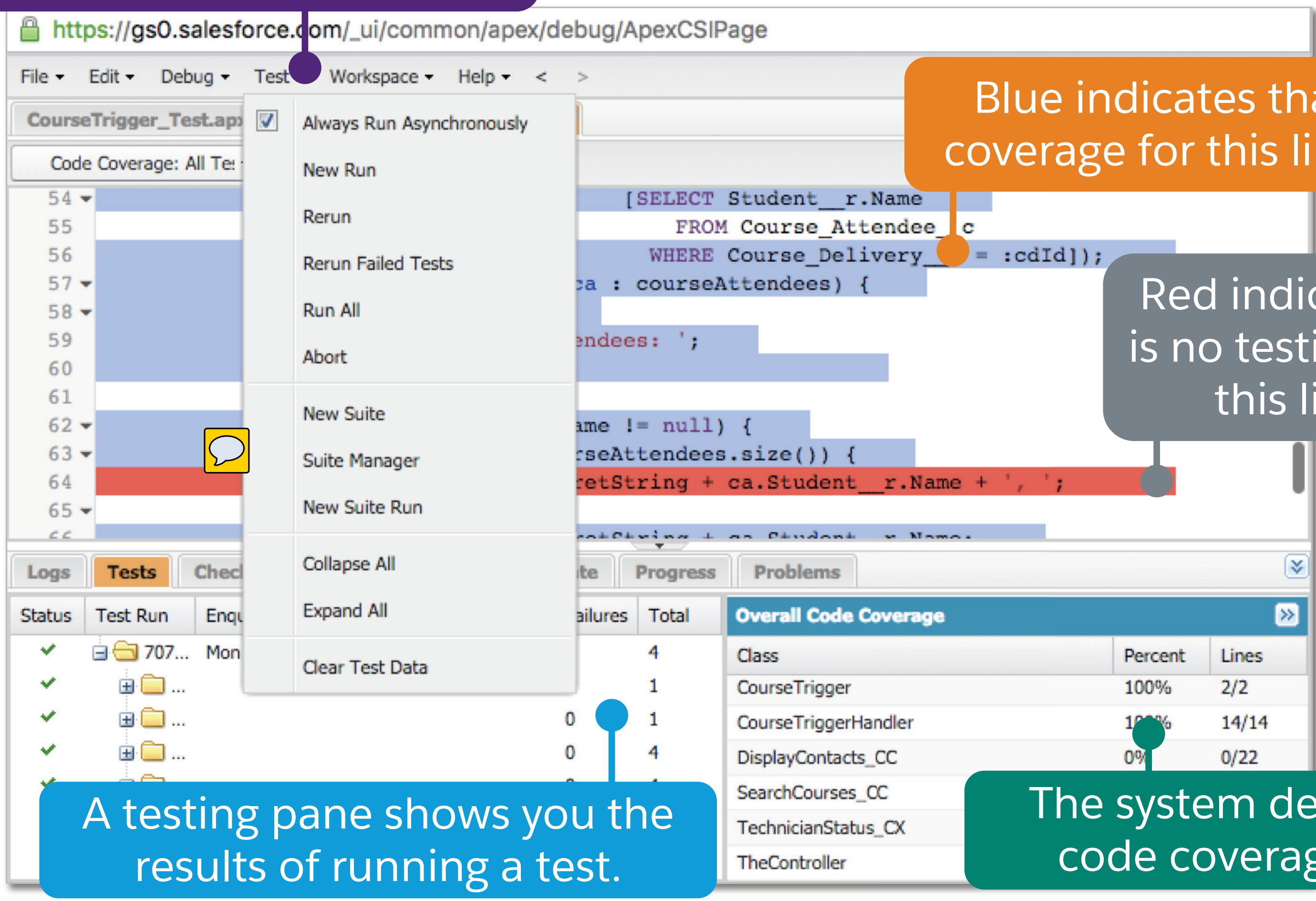
TESTING FOR EXPECTED OUTCOMES USING ASSERTIONS


```
@isTest
private class MyBusinessLogicClass_Test {
    private static testMethod void myBusinessLogicTest() {
        // create a list of 10 valid Courses
        // insert the list of 10 valid Courses
        List<Course__c> courses = [SELECT …]; // retrieve courses that were just inserted
        Integer numberCoursesInserted = courses.Size();

        //Use assert methods to verify that 10 courses were inserted
        System.Assert(numberCoursesInserted == 10);

        //An alternative
        //System.AssertEquals(10, numberCoursesInserted, 'ERROR: Expected 10 Courses');
    }
}
```

If this condition evaluates to false, a `System.AssertException` is raised.

RUNNING A TEST IN THE DEVELOPER CONSOLE

The test menu lets you choose which tests to run.



Blue indicates that there is coverage for this line of code.

Red indicates that there is no testing coverage for this line of code.

A testing pane shows you the results of running a test.

The system determines the code coverage per class.

JOIN ME

11-2: WRITE AND RUN AN APEX TEST

10 minutes

Goal:

Determine if the Certification application allows a Course Delivery to be created if it does not fall on a Holiday.

Tasks:

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

KEY TAKEAWAYS

- Salesforce provides a testing framework to make testing your Apex code easier and faster.
- `testMethod` specifies a method in a test class that will be used to test Apex code.
- Test data can be supplied declaratively and/or programmatically.
- An assertion validates an expected outcome about your application's status at a given time of execution.

1. What percentage of Apex code must be executed by test methods? 75 (mais 90% en vrai)
2. Which annotation for a method in a test class indicates that it will be run once before any of the class's test methods are executed? @testSetup
3. In a test method, an Integer variable named x has the value 10. When the statement `System.Assert(x == 10);` is executed in the method, what is the outcome?

Developer Beginner | Apex Testing
(45 minutes)

MODULE 12:
TESTING STRATEGIES

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

TRAILHEAD

Before moving code in a sandbox to production, you want to make sure you have 100% code coverage (75% is just the minimum) and that your tests ensure your Apex code meets your coding standards.

To accomplish this, you need to:

- Determine your code coverage percentages.
- Create tests using best practices.

MODULE AGENDA

MODULE 12: TESTING STRATEGIES

- **Understanding the Side Effects of Testing**
- Testing Using Best Practices

AVOIDING ISSUES CAUSED BY TESTS RUNNING IN PARALLEL

When tests are initiated from the Salesforce user interface (including the Developer Console), test classes are run in parallel. Individual test methods inside test classes are run serially, but not necessarily sequentially.

```
@isTest (seeAllData=true)
public class AccountClass_Test {
    private static testMethod void UpdateTest() {
        Account a = // Select a single account
        a.Name = 'Foo';
        update a;
        // test logic
    }
}
```

```
@isTest (seeAllData=true)
public class AccountClass_Test2 {
    private static testMethod void UpdateTest2() {
        Account a = // Select a single account
        a.Name = 'Boo';
        update a;
        // test logic
    }
}
```

1. If these tests are run simultaneously in the same sandbox, what issue might occur, assuming that `Account a` is populated with the same Account in both test methods?
2. How could you resolve this issue?

HOW IS CODE COVERAGE CALCULATED?

Running tests causes the code coverage percentages to be calculated.

```
Code coverage percentage =
(number of covered lines) / (number of covered lines + uncovered lines)
```

Only executable lines of code are included.

You create a test method that only enters the first branch of the conditional.

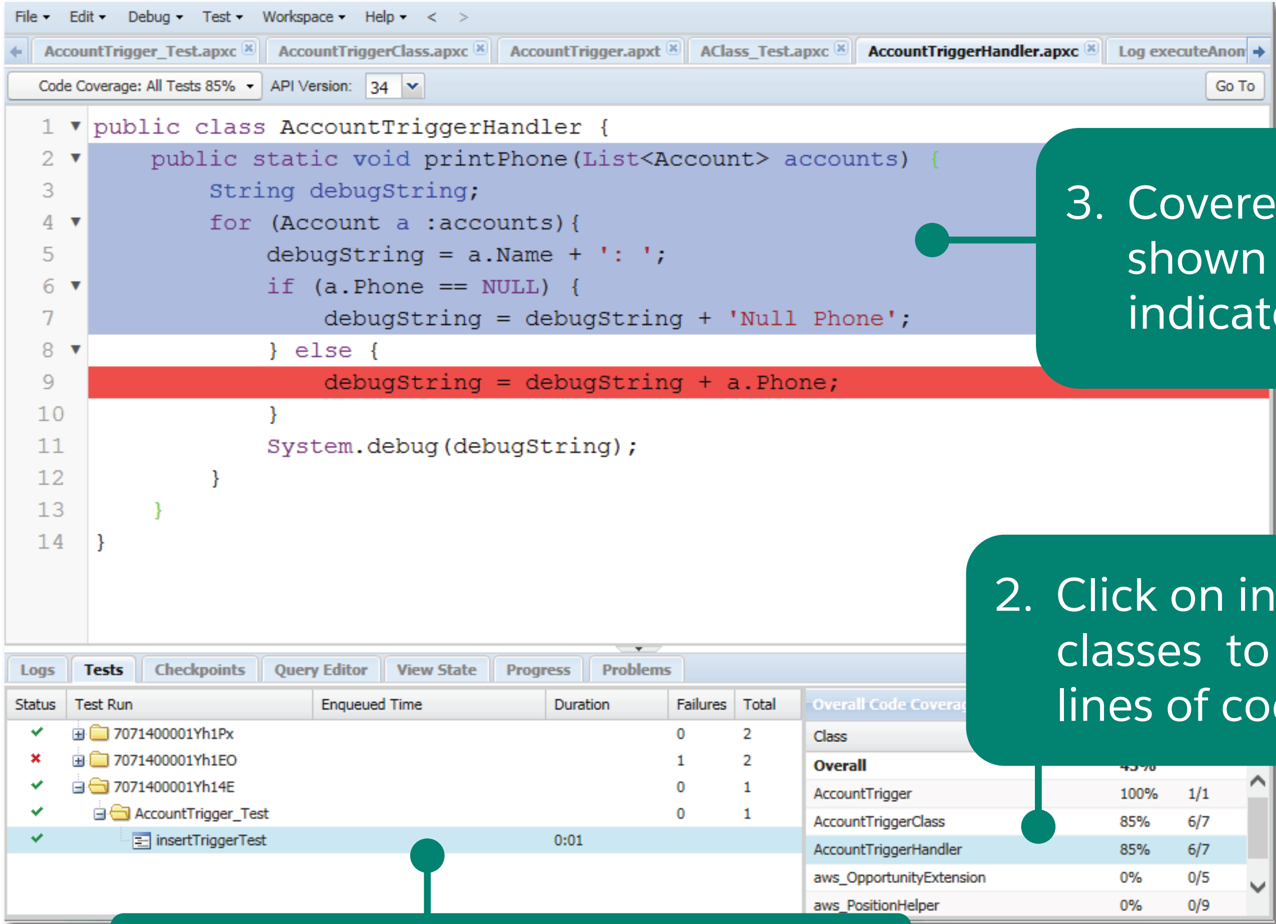
1. What is the total number of lines in the `AccountTriggerHandler` class that need to be covered to achieve 100% code coverage?
2. How many lines did your test cover?
3. What is the code coverage percentage for that test?

```
// Logic for the AccountTrigger (before inser
public class AccountTriggerHandler {
    public static void printPhone(List<Accoun
        String debugString;
         for (Account a :accounts){
              debugString = a.Name + ': ';
              if (a.Phone == NULL) {
                   debugString += 'Null Phone';
              } else {
                   debugString += a.Phone;
              }
              System.debug(debugString);
         }
    }
}
```

Not Counted | Counted

When you test a condition that executes only one branch of a conditional, you will only "get credit" for testing that branch, and not the entire conditional.

VIEWING TESTING RESULTS AND CODE COVERAGE



1. View a summary of test results.
2. Click on individual triggers and classes to view the blue/red lines of code coverage.
3. Covered lines are shown in blue. Red indicates no coverage.

STORED CODE COVERAGE CALCULATIONS


```
// Logic for the AccountTrigger (before insert)
public class AccountTriggerHandler {
    public static void printPhone(List<Account> accounts) {
        String debugString;
         for (Account a :accounts){
            debugString = a.Name + ': ';
            if (a.Phone == NULL) {
                 debugString += 'Null Phone';
            } else {
                debugString += a.Phone;
            }
            System.debug(debugString);
        }
    }
}
```

1. You created a test that only enters the first branch of the conditional.
2. You run the test.
3. Then you decide to remove the `else` condition in the `printPhone` method and re-save the class.

What do you think will happen to the stored code coverage calculation for this class? Remis à 0 dès qu'on touche à la classe,
Ou à la classe test.
Il faut rerun.

YOUR TURN

12-1: EXPLORE CODE COVERAGE

10 minutes

Goal:

Explore code coverage using the Developer Console.

Tasks:

1. Prepare the lab.
2. Run the test class.
3. View code coverage.

MODULE AGENDA

MODULE 12: TESTING STRATEGIES

- Understanding the Side Effects of Testing
- **Testing Using Best Practices**

TESTING DATA CONDITIONS: VALID, INVALID, AND BULK


```
// Logic for the AccountTrigger (before insert)
public class AccountTriggerHandler {
    public static void printPhone(List<Account> accounts) {
        String debugString;
         for (Account a :accounts){
             debugString = a.Name + ': ';
             if (a.Phone == NULL) {
                  debugString += 'Null Phone';
                  a.addError('Account has no phone');
             } else {
                 debugString += a.Phone;
             }
             System.debug(debugString);
         }
    }
}
```

For the `AccountTriggerHandler` class:

1. How would you construct a test of valid data?
2. How would you construct a test of invalid data?
3. How would you construct a test of bulk data? Why is it important to bulk test?

Pour être exhaustif, il faut absolument tester le bulk, donc tester avec 200 valeurs (et pas que quelques unes).

TESTING LIMITS

`startTest` and `stopTest` methods:

- Are used, in conjunction with methods of the Limits class, to validate how close your business logic code is to exceeding governor limits.
- Demarcate the boundaries of your actual test code within a test method.
- Provide a new context (i.e., an entirely new set of governor limits).

```
1  @isTest
2  private class AClass_Test {
3      private static testMethod void myClassTestUser() {
4          Account anAccount = new Account (name = 'Salesforce');
5          insert anAccount;
6          System.debug(Limits.getDMLStatements());
7          Test.startTest(); // Starts a brand new set of limits
8          System.debug(Limits.getDMLStatements());
9          Test.stopTest();  // Ends the set of limits started on line 7
10         System.debug(Limits.getDMLStatements());
11     }
12 }
```

1. What will line 6 print in the debug log?
2. What will line 8 print in the debug log?
3. What will line 10 print in the debug log?

System.runAs(user U) : spécifier l'utilisateur

TESTING with Sharing USING System.RunAs(User u)

System.runAs(User u) enables you to write test methods that allow you to specify the user context so that the user's record sharing is enforced.

```
@isTest
private class MyClass_Test {
    @testSetup
    private static void testDataSetup() {
        Profile p = // Write a query to select a profile
        User u = new User(Lastname = 'Foo', ProfileId = p.id);
        insert u;
    }

    private static testMethod void myClassTestUser() {
        User testUser = // Find Foo, who has set up above
        System.runAs(testUser) {
        // Now, the rest of test runs as testUser.
        // Otherwise the test would run as System.
       }
    }
)
```

SETTING UP DATA FOR System.RunAs(User u)

A standard user should only see their own Certification Held records. Training Admins should be able to see all Certification Held records.

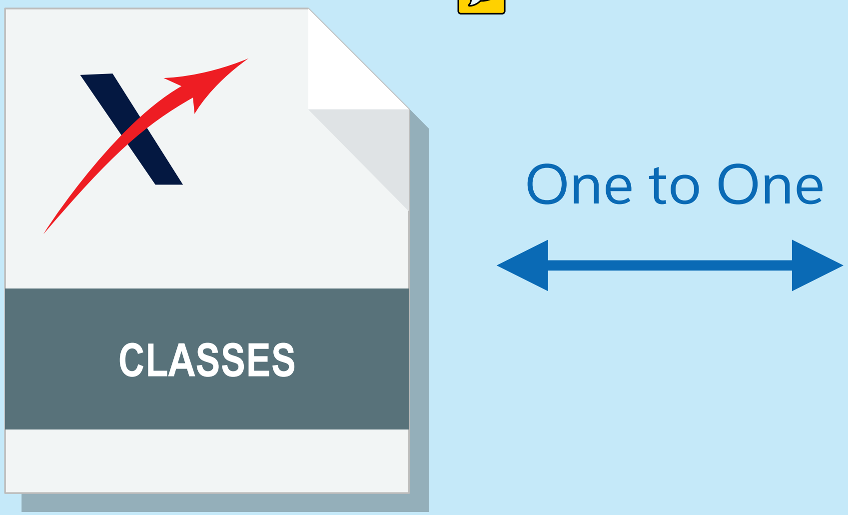
```
// The class that needs to be tested
public with sharing
     class CertficationHeldClass {
     public static Integer countDailyCertHeld() {
          // Returns the count of
          // Certification Held records
          // created today.
     }
}
```

```
@isTest
private class CertificationHeldClass_Test {
     @testSetup
     static void setupUsers() {
          // … Create and insert a standard user 'su1'
          // … insert 2 Cert Held records for that user
          // … Create and insert standard user 'su2'
          // … Create a Training Admin user 'ta1'
     }

     private static testMethod void testTrainingAdmin() {
          // TODO 1
     }

     private static testMethod void testStandardUser2() {
          // TODO 2
     }
}
```

1. Write the pseudocode to test `countDailyCertHeld()` with the Training Admin context.
2. Write the pseudocode to test `countDailyCertHeld()` with the Standard User 2's context.

Une méthode par profil !
Ce qui est une bonne pratique.

BREAKING UP YOUR TEST CLASSES



ENSURING TEST DATA IS CREATED PROPERLY WHEN DEPLOYED


```
1  @isTest
2  private class MyClass_Test {
3      @testSetup
4      private static void testDataSetup () {
5          Test.loadData(Holiday.sObjectType, 'Test_Holidays');
6          User aUser = new User(ProfileId='00e30000001tbKQ');
7          // Set up u's required fields
8          insert aUser;
9      }
10 )
```

This `testSetup` method ran properly in the sandbox where it was originally created.

1. What needs to happen in the production org to ensure Line 5 properly executes in production?
2. What might happen when line 8 executes in production? How can you overcome this issue?

KEY TAKEAWAYS

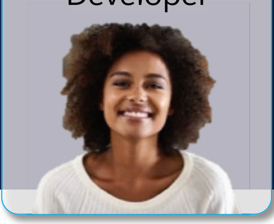
- Tests have side effects, including causing code coverage percentages to be calculated.
- Stored calculations can be refreshed by re-running tests.
- Many types of code, such as blank lines and debug statements, do not count toward your code percentage targets.
- To achieve 100% coverage of a conditional statement, you must test each of its conditions.
- Don't hardcode Ids in your test data.
- Test your code fully by testing a variety of data conditions, including bulk data.
- `System.runAs()` can help you test your sharing model.
- The `Test` class has methods that can be used to demarcate the boundaries of a new context in a test method. This is useful for testing governor limits.

MODULE 13:
STRATEGIES FOR DESIGNING EFFICIENT APEX SOLUTIONS

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

Cassie Evans

Developer



Some of the legacy triggers in our org are failing, even though we have test cases. We are also hitting a lot of limits. Can you work on fixing them?

Before you can begin, you need to:

- Describe practices for writing code that is easy to maintain and extend.
- Write triggers and classes that assume batches of data as input.
- Write code that works efficiently with the database, both in querying and using DML.

MODULE 13: STRATEGIES FOR DESIGNING EFFICIENT APEX SOLUTIONS

- **Working Efficiently with the Database**
- Designing Triggers
- Designing Classes

KEY QUERY LIMITS

- Total number of SOQL queries issued. 100
- Total number of records retrieved by SOQL queries. 50.000
- Total heap size. 6 mega

ANTI-PATTERN: QUERYING WITHIN A LOOP

What limit might your code exceed if you query within a loop?

```
public static void provideAccessLMS (List<Course_Attendee__c> courseAttendees,
                                     Map<Id,Course_Attendee__c> oldMap) {
    String emails = '';
    Integer numberOfNewEnrollees = 0;
    for (Course_Attendee__c cA: courseAttendees) {
        if (cA.status__c == 'Enrolled')  {
            List<Course_Attendee__c> attendeeInfo = [SELECT Course_Delivery__r.Name,
                                                            Student__r.email, Status__c
                                                     FROM Course_Attendee__c
                                                     WHERE Id = :ca.Id];
            for (Course_Attendee__c cAwithInfo: attendeeInfo) {
                emails += '\n' + cAwithInfo.Course_Delivery__r.Name + ': '
                          + cAwithInfo.Student__r.email;
                numberOfNewEnrollees++;
            }
        }
    }

    …
}
```

Querying within a loop.

PRINCIPLE: USE COLLECTIONS TO MINIMIZE THE NUMBER OF QUERIES


```
public class CourseAttendeeTriggerHandler {
    public static void provideAccessLMS (List<Course_Attendee__c> courseAttendees,
                                         Map<Id,Course_Attendee__c> oldMap) {
        Set<Id> courseAttendeesToChangeAccess = new Set<Id>();
        for (Course_Attendee__c cA : courseAttendees) {
            if (cA.status__c == 'Enrolled') {
                if (oldMap == null || !oldMap.containsKey(cA.Id)
                  || oldMap.get(cA.id).status__c != cA.status__c) {
                    courseAttendeesToChangeAccess.add(cA.Id);
                }
              }
        }
        if (courseAttendeesToChangeAccess.size() > 0) {
            String emails = '';
            Integer numberOfNewEnrollees=0;
            for (List<Course_Attendee__c> caList : [SELECT Course_Delivery__r.Name, Student__r.email
                                                    FROM Course_Attendee__c
                                                    WHERE Id in :courseAttendeesToChangeAccess
                                                    ORDER BY Course_Delivery__r.Name]) {
…
```

Gather values in a collection to perform a single query.

Reduces total number of SOQL queries issued and total number of records retrieved by SOQL queries.

BE AWARE: PARENT-CHILD QUERIES ARE COUNTED DIFFERENTLY


```
[SELECT Id, Name, (SELECT Name FROM Courses__r) FROM Certification__c]
```

Row counts from sub-queries contribute to the overall row count.

Traverser au maximum les liens, une unique requête !

PRINCIPLE: BRING BACK WHAT YOU NEED IN A SINGLE QUERY


```
public class CourseAttendeeTriggerHandler {
    public static void provideAccessLMS (List<Course_Attendee__c> courseAttendees,
                                         Map<Id,Course_Attendee__c> oldMap) {
        Set<Id> courseAttendeesToChangeAccess = new Set<Id>();
        for (Course_Attendee__c cA : courseAttendees) {
            if (cA.status__c == 'Enrolled') {
                if (oldMap == null || !oldMap.containsKey(cA.Id)
                  || oldMap.get(cA.id).status__c != cA.status__c) {
                    courseAttendeesToChangeAccess.add(cA.Id);
                }
            }
        }
        if (courseAttendeesToChangeAccess.size() > 0) {
            String emails = '';
            Integer numberOfNewEnrollees=0;
            for (List<Course_Attendee__c> caList : [SELECT Course_Delivery__r.Name, Student__r.email
                                                    FROM Course_Attendee__c
                                                    WHERE Id in :courseAttendeesToChangeAccess
                                                    ORDER BY Course_Delivery__r.Name]) {
…
```

Bring back fields from related records, rather than querying separately.

Reduces total number of SOQL queries issued.

PRINCIPLE: REDUCING THE DATA SET


```
public class CourseAttendeeTriggerHandler {
    public static void provideAccessLMS (List<Course_Attendee__c> courseAttendees,
                                         Map<Id,Course_Attendee__c> oldMap) {
        Set<Id> courseAttendeesToChangeAccess = new Set<Id>();
        for (Course_Attendee__c cA : courseAttendees) {
            if (cA.status__c == 'Enrolled') {
                if (oldMap == null || !oldMap.containsKey(cA.Id)
                  || oldMap.get(cA.id).status__c != cA.status__c) {
                    courseAttendeesToChangeAccess.add(cA.Id);
                }
            }
        }
        if (courseAttendeesToChangeAccess.size() > 0) {
            String emails = '';
            Integer numberOfNewEnrollees=0;
            for (List<Course_Attendee__c> caList : [SELECT Course_Delivery__r.Name, Student__r.email
                                                    FROM Course_Attendee__c
                                                    WHERE Id in :courseAttendeesToChangeAccess
                                                    ORDER BY Course_Delivery__r.Name]) {
…
```

Use `Trigger.oldMap` to identify and discard records where there is no modification in the field of interest.

Apply filters when possible.

Filtrer au max, ne pas récupérer ce dont on n'a pas besoin.

Reduces total number of records retrieved by SOQL queries and heap size.

PRINCIPLE: USE SOQL FOR LOOPS TO REDUCE HEAP SIZE

FOR LOOP

```
…
if (courseAttendeesToChangeAccess .size() > 0) {
    String emails = '';
    Integer numberOfNewEnrollees=0;
    for (List<Course_Attendee__c> caList : [SELECT Course_Delivery__r.Name, Student__r.email
                                              FROM Course_Attendee__c
                                              WHERE Id in : courseAttendeesToChangeAccess
                                              ORDER BY Course_Delivery__r.Name]) {
        for (Course_Attendee__c ca: caList) {
            emails += '\n' + ca.Course_Delivery__r.Name + ': ' + ca.Student__r.email;
            numberOfNewEnrollees++;
        }
    }
…
```

Records automatically batched for efficient processing.

Reduces heap size.

TIP FOR IMPROVING SOQL PERFORMANCE

Using indexed fields when possible may improve performance.

```
SELECT Course_Delivery__r.Name, Student__r.email
FROM Course_Attendee__c
WHERE Id in :idsToChangeAccess
ORDER BY Course_Delivery__r.Name
```

Ids and fields marked as External ID are indexed.

Les champs indexés permettent de requêter plus vite.

Account Fields

Help for this Page

This page allows you to specify the fields that can appear on the Account page. You can create up to 500 Account custom fields.

Note that deleting a custom field will delete any filters that use the custom field. It may also change the result of Assignment or Escalation Rules that rely on the custom field data.

Set History Tracking

Account Standard Fields

Account Standard Fields Help

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
Edit	Account Currency	CurrencyIsoCode	Picklist		
	Account Name	Name	Name		✓
Edit	Account Number	AccountNumber	Text(40)		
Edit	Account Owner	Owner	Lookup(User)		✓
Edit	Account Record Type	RecordType	Record Type		✓

YOUR TURN

13-1: REFACTOR A TRIGGER TO AVOID SOQL LIMITS

15 Minutes

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.

KEY DML LIMITS

- Total number of DML statements issued. 150
- Total number of records processed as a result of DML statements. 10.000
- Total heap size.

ANTI-PATTERN: DML STATEMENTS


```
public with sharing class CourseDeliveryTriggerHandler {
    public static void cancelEnrollees(List<Course_Delivery__c> triggerNew,
                                       Map <Id,Course_Delivery__c> oldMap) {
        Set<Id> cancelledCourseDeliveries = new Set<Id>();
        for (Course_Delivery__c cD : triggerNew) {
            if (cD.Status__c == 'Cancelled' && oldMap.get(cD.id).Status__c != 'Cancelled') {
                cancelledCourseDeliveries.add(cD.id);
            }
        }
        for (Course_Attendee__c cA :[SELECT id
                                     FROM Course_Attendee__c
                                     WHERE Course_Delivery__c in :cancelledCourseDeliveries
                                      AND Status__c = 'Enrolled']) {
            cA.Status__c = 'Course Cancelled';
            try {
                update cA;
            } catch (System.DmlException ex) {
                System.debug(ex);
             }
        }
    }
}
```

One DML statement per sObject.

PRINCIPLE: BATCH DATA FOR DML STATEMENTS


```
public with sharing class CourseDeliveryTriggerHandler {
    public static void cancelEnrollees(List<Course_Delivery__c> triggerNew,
                                       Map <Id, Course_Delivery__c> oldMap) {
        Set<Id> cancelledCourseDeliveries = new Set<Id>();
        for (Course_Delivery__c cD : triggerNew) {
            if (cD.Status__c == 'Cancelled' && oldMap.get(cD.id).Status__c! = 'Cancelled') {
                cancelledCourseDeliveries.add(cD.id);
            }
        }
        List<Course_Attendee__c> courseAttendeesToUpdate = new List<Course_Attendee__c>();
        for (Course_Attendee__c cA :[SELECT id
                                     FROM Course_Attendee__c
                                     WHERE Course_Delivery__c in :cancelledCourseDeliveries
                                        AND Status__c =
            cA.Status__c = 'Course Cancelled';
            courseAttendeesToUpdate.add(cA);
        }
        if (courseAttendeesToUpdate.size() > 0) {
            try {
                update courseAttendeesToUpdate;
            }  catch (System.DmlException ex) {
                System.debug(ex);
            }
 }}} //combined for on screen
```

Use a collection to gather data on which DML will be performed.

Update several values at once.

Reduces total number of DML statements issued and heap size.

PRINCIPLE: USE SOQL FOR LOOPS TO MAKE BATCHES OF 200


```
//More before this…
for (List<Course_Attendee__c> cAList :[SELECT id FROM Course_Attendee__c
                                        WHERE Course_Delivery__c in :cancelledCourseDeliveries
                                        AND Status__c = 'Enrolled']) {
    for (Course_Attendee__c cA :caList) {
        cA.Status__c = 'Course Cancelled';
    }
    try {
        update cAList;
    }  catch (System.DmlException ex) {
        System.debug(ex);
    }
}
```

Inputs to loop are batched from SOQL for loop.

DML statement for the batch.

YOUR TURN

13-2: REFACTOR A TRIGGER TO AVOID DML LIMITS

15 Minutes

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.

MODULE 13: STRATEGIES FOR DESIGNING EFFICIENT APEX SOLUTIONS

- Working Efficiently with the Database
- **Designing Triggers**
- Designing Classes

MULTIPLE TRIGGERS ON THE SAME OBJECT

What are the pros and cons of having multiple triggers on the same object?

```
trigger CourseAttendeeValidateTrigger on Course_Attendee__c (before insert) {
    CourseAttendeeValidateTriggerHandler.validateEnrollment(Trigger.new);
}
```

```
trigger CourseAttendeeNotificationTrigger on Course_Attendee__c (after insert, after update) {
    CourseAttendeeNotificationTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);
}
```

```
trigger CourseAttendeeLMSTrigger on Course_Attendee__c (after insert, after update) {
    CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);
}
```

la bonne pratique est de faire un trigger par événement : le Before Insert ne traitera que le Before Insert, et tous.

PRINCIPLE: ONE TRIGGER PER OBJECT


```
trigger CourseAttendeeTrigger on Course_Attendee__c (before insert, after insert, after update) {
    if (Trigger.isBefore && Trigger.isInsert) {
        CourseAttendeeTriggerHandler.validateEnrollment(Trigger.new);
    } else if ( Trigger.isAfter ) {
        if ( Trigger.isInsert || Trigger.IsUpdate ) {
            CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);
            CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);
        }
    }
}
```

1 Should your trigger list all potential contexts?

2 What form of conditional statement should you have?

Faire un événement par trigger (Abdel-Nor)

Mettre des noms techniques pour les triggers, ex : AccountBeforeInsert (before insert) {}
Ex : ManageHoliday.CheckHoliday (List)

Mettre un ByPass !

EXAMPLE OF CONDITIONAL PATTERN FOR EASY MODIFICATION


```
trigger CourseAttendeeTrigger on Course_Attendee__c (before insert, after insert, after update) {
    if (Trigger.isBefore) {
        if (Trigger.isInsert) {
            CourseAttendeeTriggerHandler.validateEnrollment(Trigger.new);
        }
    } else if (Trigger.isAfter) {
         if (Trigger.isInsert) {
             CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);
             CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);
         } else if(Trigger.IsUpdate) {
             CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);
             CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);
         }
    }
}
```

Easy to add additional logic and order trigger events.

HOW SHOULD YOU MODULARIZE YOUR CODE?

What concerns would you have around adding business logic to this trigger?

```
trigger CourseAttendeeTrigger on Course_Attendee__c (before insert, after insert, after update) {
    if (Trigger.isAfter) {
        if (Trigger.isInsert) {
            Set<Id> newToChangeAccess = new Set<Id>();
            for (Course_Attendee__c cA : Trigger.new) {
                if (cA.Status__c == 'Enrolled') {
                    if (Trigger.oldMap == null || !Trigger.oldMap.containsKey(cA.Id)
                        || Trigger.oldMap.get(cA.id).Status__c != cA.Status__c) {
                         newToChangeAccess.add(cA.Id);
                    }
                }
            }
            if (newToChangeAccess.size() > 0) {
                String emails = '';
                Integer numberOfNewEnrollees = 0;
                for (List<Course_Attendee__c> caList : [SELECT Course_Delivery__r.Name, Student__r.email
                                                          FROM Course_Attendee__c
                                                          WHERE Id in :newToChangeAccess
                                                          ORDER BY Course_Delivery__r.Name]) {
                                                              for (Course_Attendee__c ca: caList) {
                 …
```

PRINCIPLE: KEEP BUSINESS LOGIC OUT OF YOUR TRIGGER

La logique est dans du ClassTriggerHandler, classe statique avec méthodes statiques.

The trigger should at most contain the traffic cop logic.

```
trigger CourseAttendeeTrigger on Course_Attendee__c (before insert, after insert, after update) {
    if (Trigger.isBefore) {
        if (Trigger.isInsert) {
            CourseAttendeeTriggerHandler.validateEnrollment(Trigger.new);
        }
    } else if (Trigger.isAfter) {
         if (Trigger.isInsert) {
             CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);
             CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);
         } else if(Trigger.IsUpdate) {
             CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);
             CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);
         }
    }
}
```

SHOULD YOU EXPLICITLY PASS TRIGGER CONTEXT DATA?

```
trigger CourseAttendeeTrigger on Course_Attendee__c (before insert, after insert, after update) {
    if (Trigger.isBefore) {
        if (Trigger.isInsert) {
            CourseAttendeeTriggerHandler.validateEnrollment(Trigger.new);
        }
    } else if (Trigger.isAfter) {
         if (Trigger.isInsert) {
             CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);
             CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);
         } else if(Trigger.IsUpdate) {
             CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);
             CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);
         }
    }
}
```

Trigger context variables can be implicitly available to methods called using handler classes, but the code will need to cast them to their type.

MODULE AGENDA

MODULE 13: STRATEGIES FOR DESIGNING EFFICIENT APEX SOLUTIONS

- Working Efficiently with the Database
- Designing Triggers
- **Designing Classes**

PRINCIPLE: MODULARIZE BUSINESS LOGIC

Write individual methods for each piece of business logic.

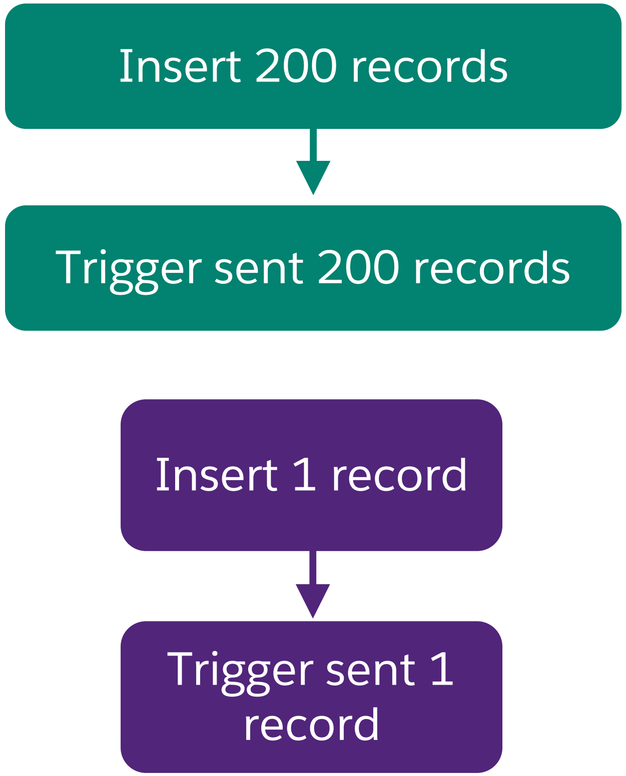
```
public class CertificationAttemptTriggerHandler {
    public static void grantInstructorSharingAccess(List<Certification_Attempt__c> triggerNew,
                                 Map<Id, Certification_Attempt__c> oldMap, Boolean isInsert, Boolean isUpdate) {

        //Method does work…
        …
    }

    public static void validateCertificationAttempt(List<Certification_Attempt__c> triggerNew) {

      //Method does more work
      …
    }

    //Additional methods here…
    …
}
```

Define `static` methods when possible.

- Consider breaking business logic down further in order to enhance readability and make unit testing easier.
- Write utility methods to capture functionality used in more than one piece of business logic.

CONSIDERATIONS AROUND TRIGGER BATCHES

- How many records should your trigger design anticipate?
- Is the order in which records are sent through a trigger predictable?
- Are the batches processed one at a time or in parallel?
- How does this apply to classes?



KEY TAKEAWAYS

- Use collections to:
  - Gather data before querying.
  - Gather data before performing a DML operation.
- Use the one trigger per object principle.
- Keep business logic out of triggers.
- Always design triggers and class methods to handle bulk inputs.
- SOQL for loops can help your code avoid exceeding limits.

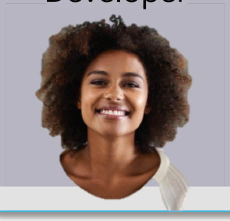
- Avoid:
  - Queries inside of loops.
  - DML inside of loops.

MODULE 14:
TRIGGER DESIGN STRATEGY

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

Cassie Evans

Developer



We need an automated way to determine whether a candidate qualifies to hold a certification upon completion of a certification element. Can you work on that?

Before you can begin, you need to:

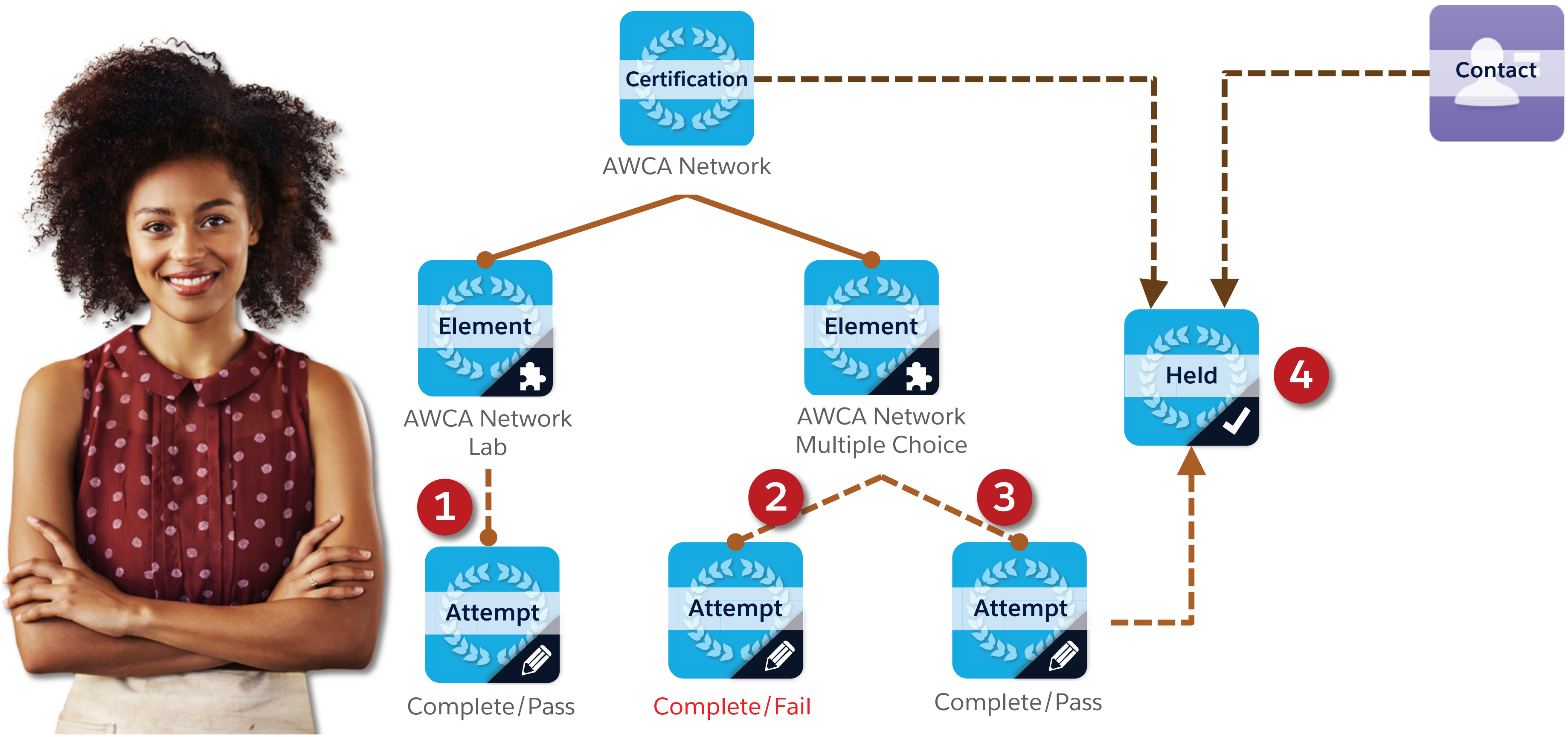
- List declarative mechanisms you can use to implement complex business logic, for what types of problems they are best used, and their limitations.
- Describe ways in which you can use declarative functionality to improve your programmatic solutions.

MODULE 14: TRIGGER DESIGN STRATEGY

- **Analyzing the Problem**
- Creating a Solution

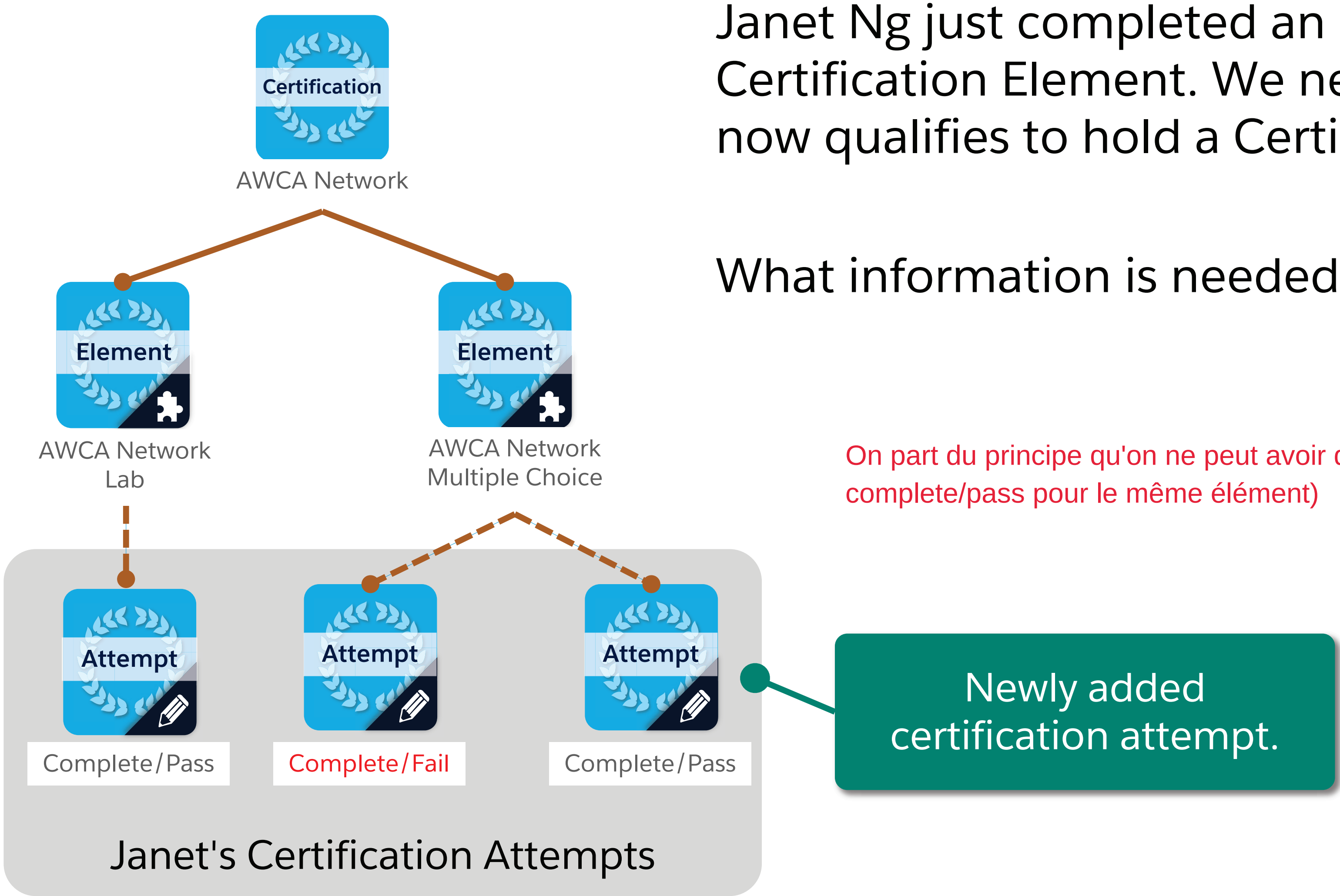
AUTOMATICALLY CREATING CERTIFICATION HELD RECORDS

We need a process that automatically creates a Certification Held record. When a candidate passes a Certification Attempt, indicating they have completed a Certification Element, we need to determine if they qualify for the related Certification.



Cassie Evans
Developer

WHAT CRITERIA WILL DETERMINE IF A CERTIFICATION HELD RECORD SHOULD BE CREATED?



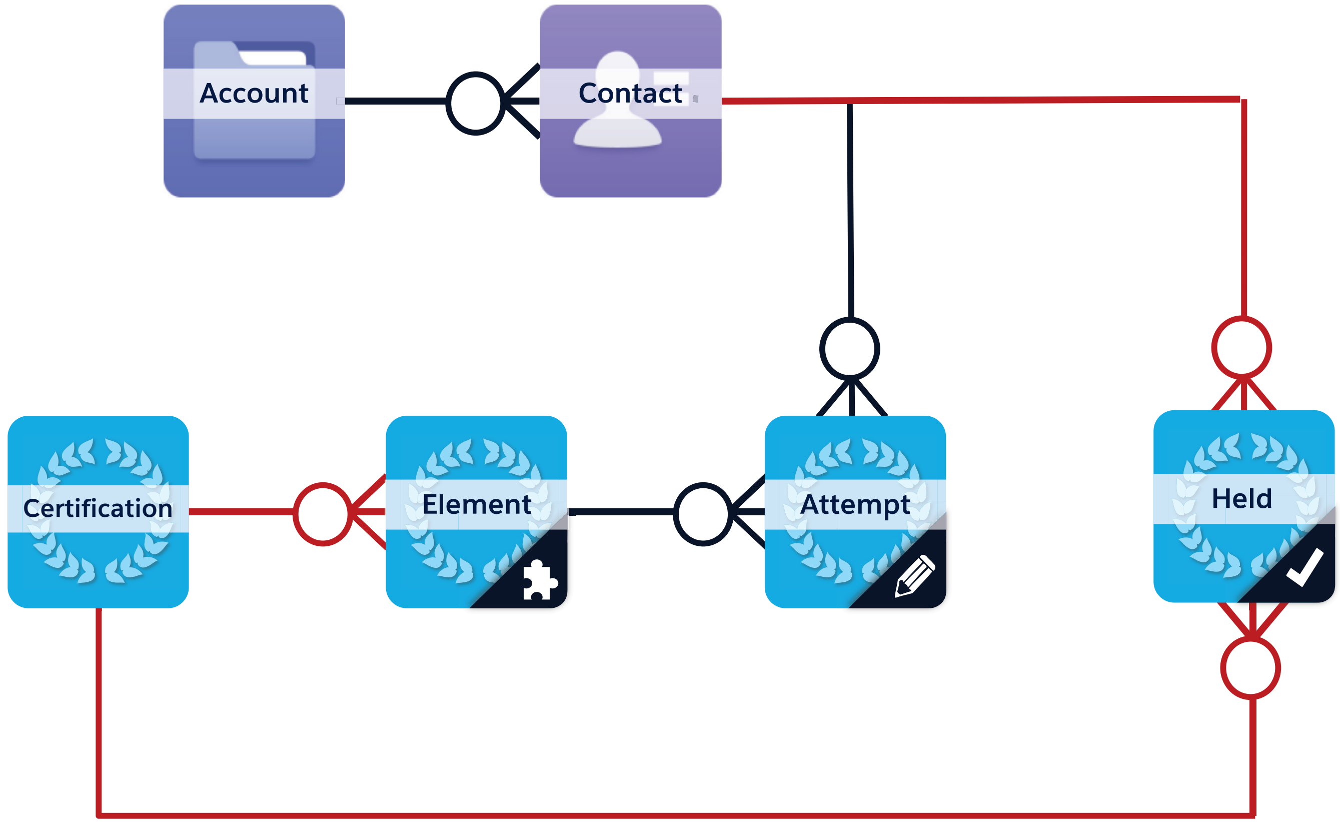
Janet Ng just completed an attempt of a Certification Element. We need to determine if she now qualifies to hold a Certification.

What information is needed to solve this problem?

On part du principe qu'on ne peut avoir de doublons d'attempt complete/pass (pas de complete/pass pour le même élément)

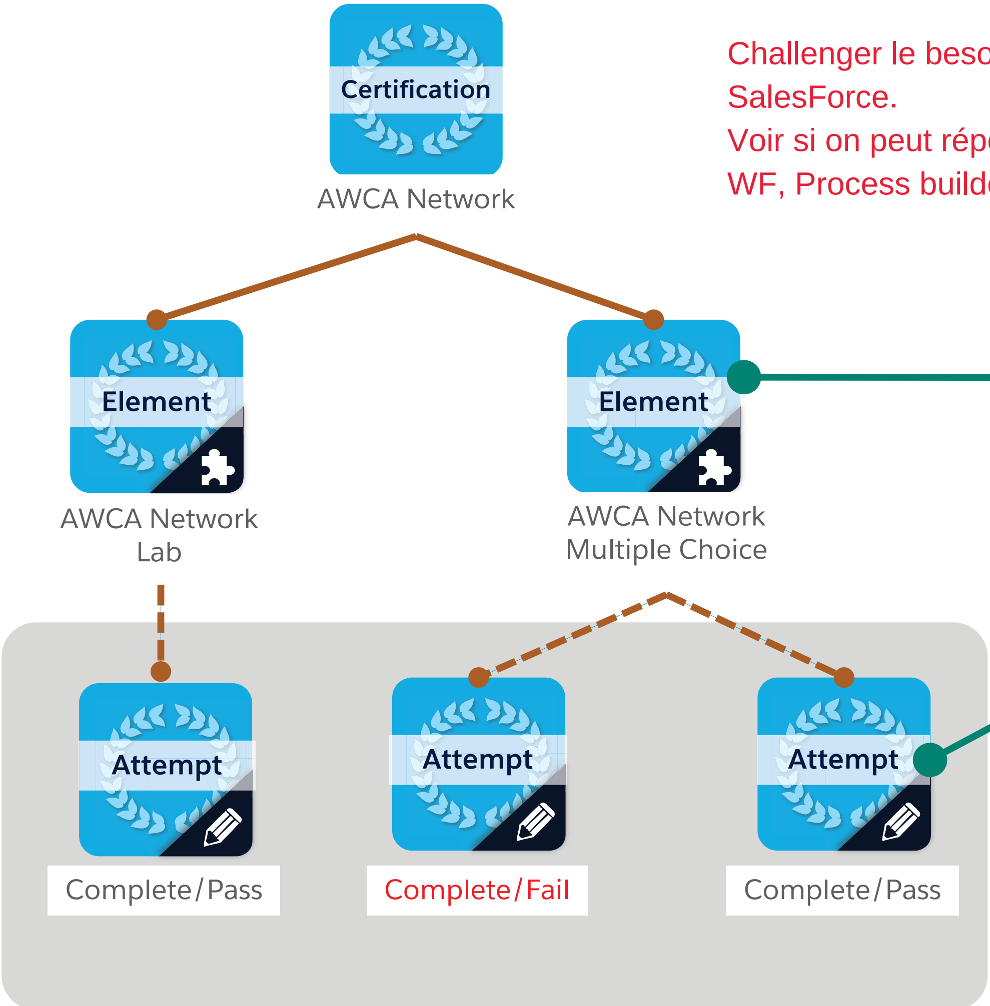
HOW DO YOU GET THE NEEDED DATA?

- Changes to which object are potentially firing this automation? Attempt
- How do you find the related certification? Relations
- How do you find the elements associated with that certification?
- How do you determine whether the candidate has passed those elements?



Matching elements to certification attempts is not trivial.
Does working with batched data make it easier or harder?

CAN THE SOLUTION USE SOMETHING BESIDES MATCHING?



Challenger le besoin : demander au client si.. pour cadrer avec les moyens de SalesForce.
Voir si on peut répondre au besoin de façon standard, quels sont les outils : WF, Process builder, VisualFlow, rollup, champ formule, validation rule VR

2 Elements for Certification AWCA Network.

Candidate has 2 Passed Certification Attempts for Certification AWCA Network.

Since the counts match, the candidate gets a Certification Held record!

WHAT TOOL SHOULD BE USED TO CREATE A SOLUTION?

360 salesforce

	Workflow	Process Builder	Trigger	Formula Fields	Roll-up Summaries
Find the related certification.	Yes	Yes	Yes	Yes	No
Find the elements associated with that certification.	No	Difficult	Yes	No	Yes
Determine whether the candidate has passed those elements.	No	Difficult	Yes	No	No
Create a distantly related object.	No	Yes	Yes	No	No

MODULE 14: TRIGGER DESIGN STRATEGY

- Analyzing the Problem
- **Creating a Solution**

WHAT TYPE OF TRIGGER IS NEEDED?

sObject type:

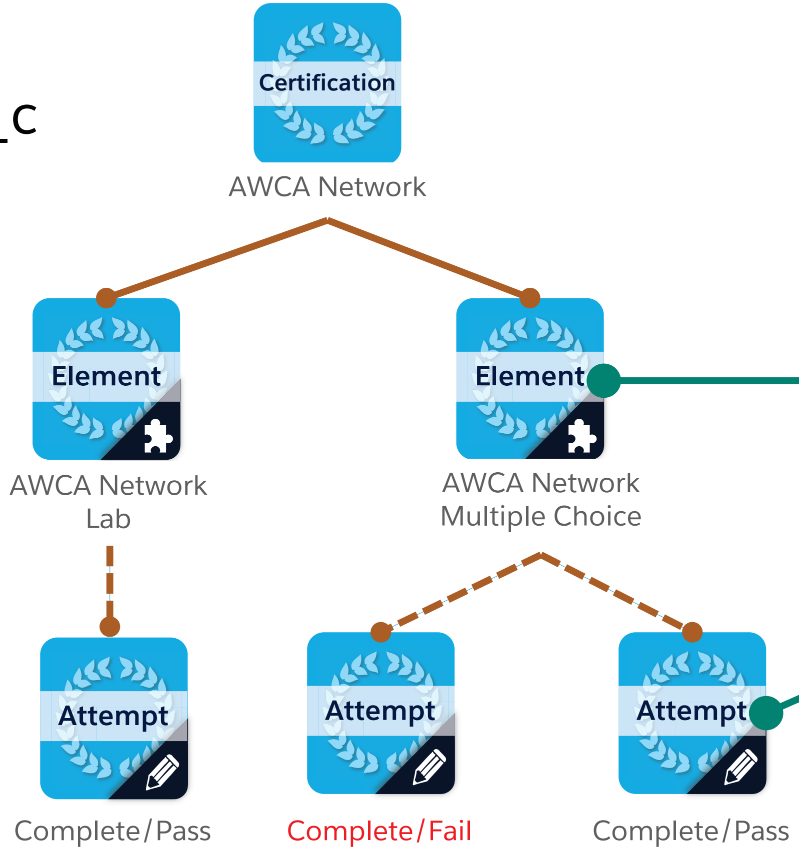
Certification_Attempt__c

Trigger operation(s)?

insert, update

before or after?

after



2 Elements for Certification AWCA Network.

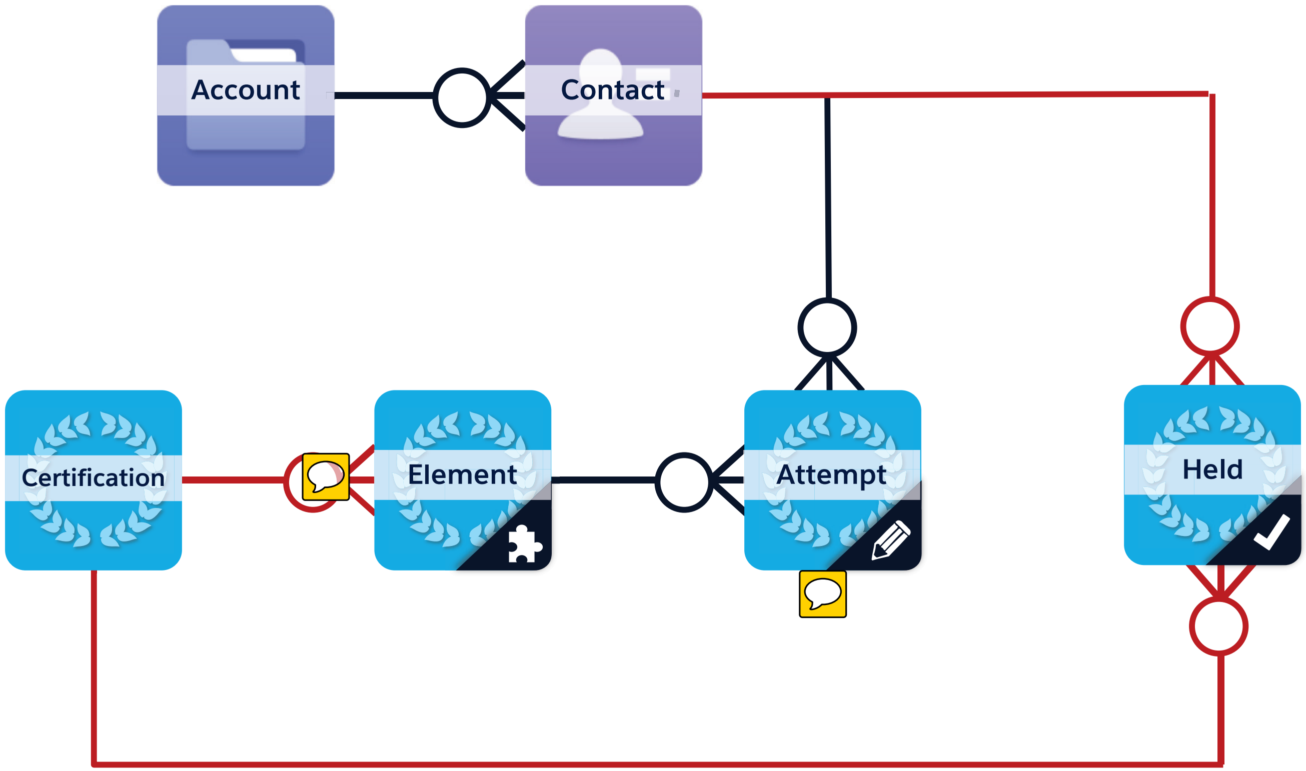
Candidate has 2 Passed Certification Attempts for Certification AWCA Network.

Since the **counts** match, the candidate gets a Certification Held record!

HOW MANY QUERIES ARE NEEDED FOR THE SOLUTION?

Queries (for bulk):

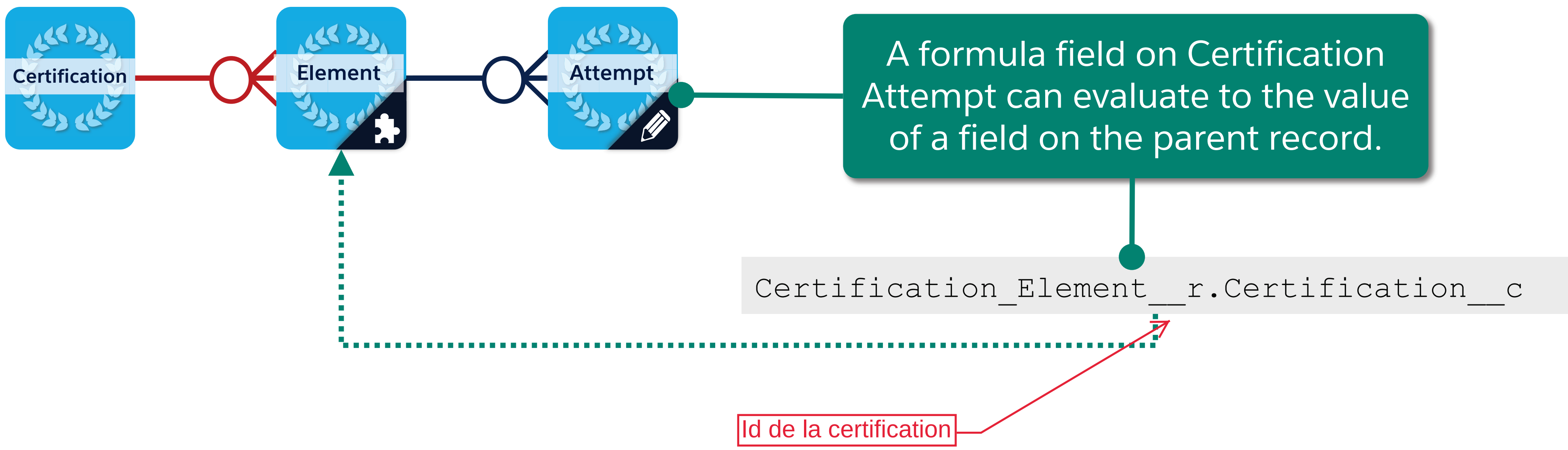
- Find the related certifications.
- Find the number of elements associated with each certification.
- Find the attempts each candidate has passed for each relevant certification.



HOW CAN YOU USE DECLARATIVE FEATURES TO REDUCE THE NUMBER OF QUERIES?

How do you find the related certification?

FOrmule de Certification Attempt sur le master -> Certifiation Element



YOUR TURN

14-1: CREATE A FORMULA FIELD TO ELIMINATE A QUERY

salesforce

5 Minutes

Goal:

Create a formula field so that the certification Id is available on the Certification Attempt object.

Task:

Create a hidden formula field on the Certification Attempt object.

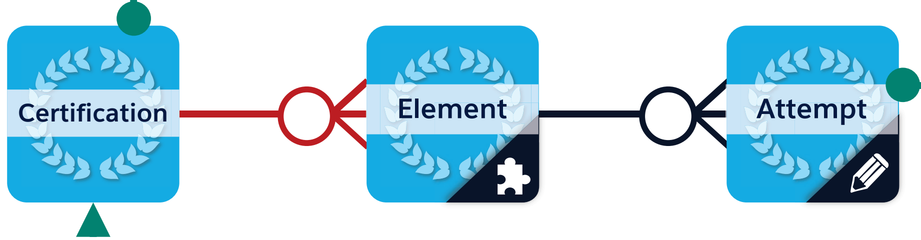
HOW CAN YOU USE DECLARATIVE FEATURES TO REDUCE THE NUMBER OF QUERIES? (CONT.)

How do you find the number of elements associated with that certification?

Use a roll-up summary field to track the number of elements of the certification.

How can you make that value easily accessible from a Certification Attempt?

A formula field on Certification Attempt can evaluate to a field on the grandparent record.



YOUR TURN

14-2: CREATE FIELDS FOR COUNTING CERTIFICATION ELEMENTS

5 Minutes

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.

WHAT QUERIES DO WE NEED NOW?

salesforce

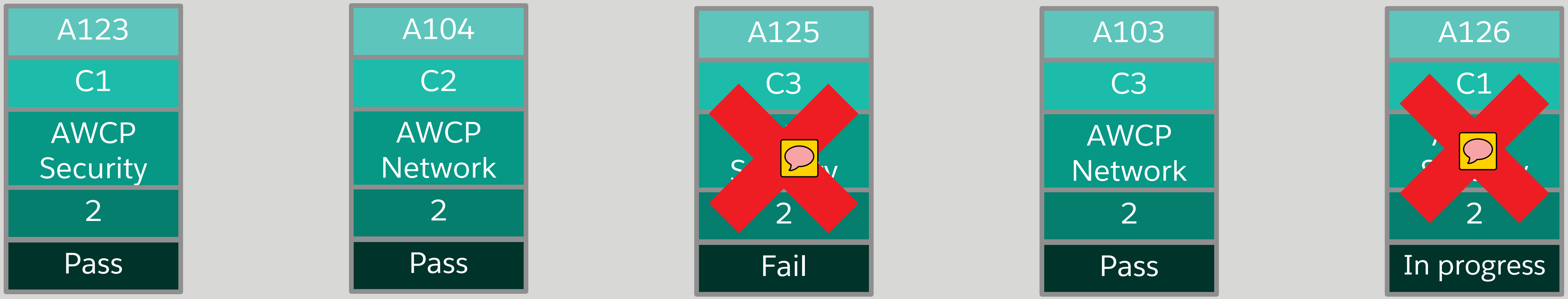
Original Queries:

- ~~Find the related certifications~~.
- ~~Find the number of elements associated with each certification~~.
- Find the attempts each relevant candidate has passed for each relevant certification.

WHICH RECORDS IN `Trigger.new` SHOULD BE CONSIDERED?

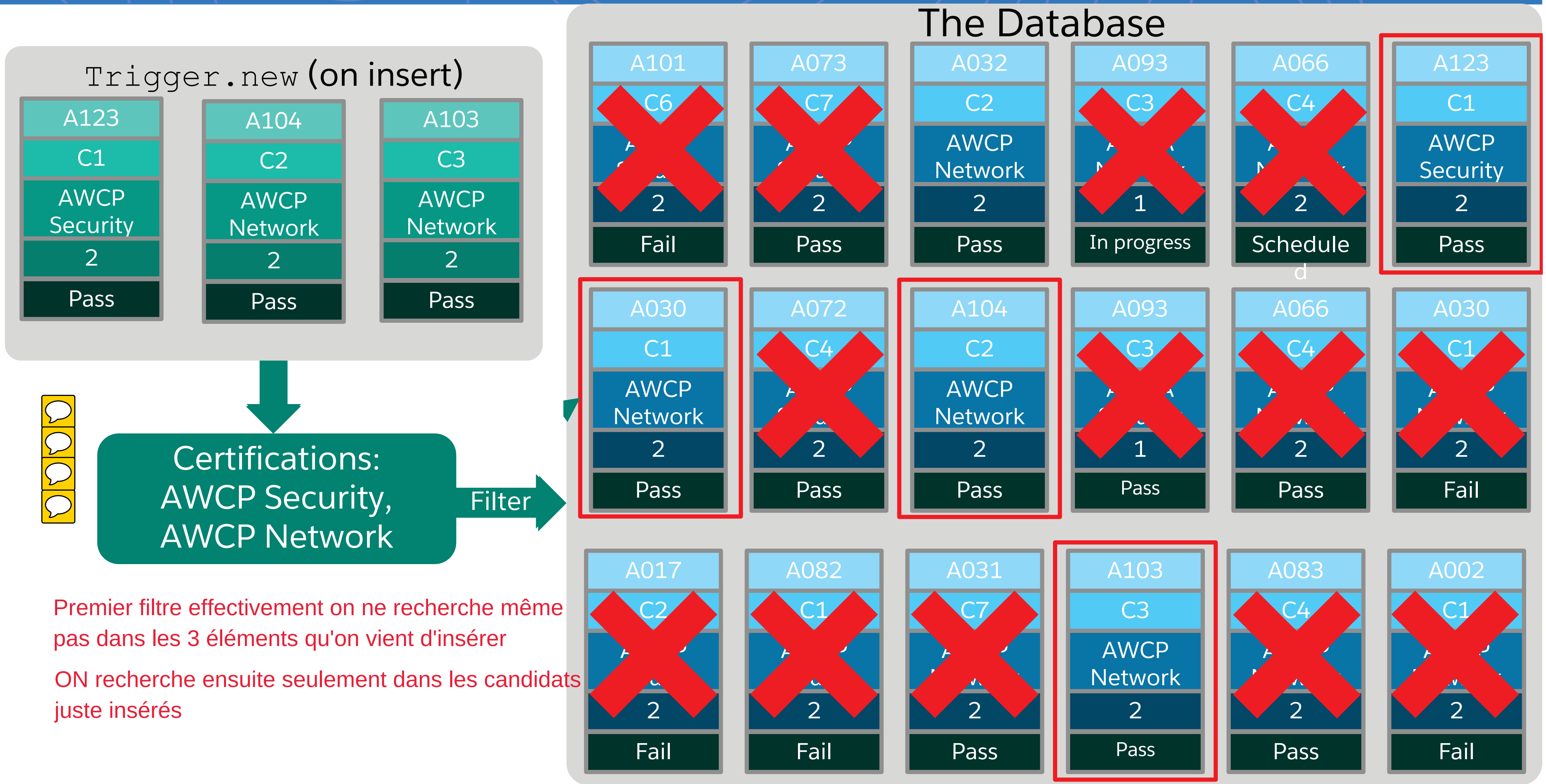
Consider only the records that have (just gotten) the status 'Complete/Pass.'

`Trigger.new` (on insert)



If this trigger fired on update, how would you determine if a record had just received the status "Complete/Pass"?

USING FILTERS IN QUERIES TO MINIMIZE ROW COUNT



Premier filtre effectivement on ne recherche même pas dans les 3 éléments qu'on vient d'insérer

ON recherche ensuite seulement dans les candidats juste insérés

YOUR TURN

14-3: CREATE COLLECTIONS TO FILTER THE QUERY

salesforce

20 Minutes

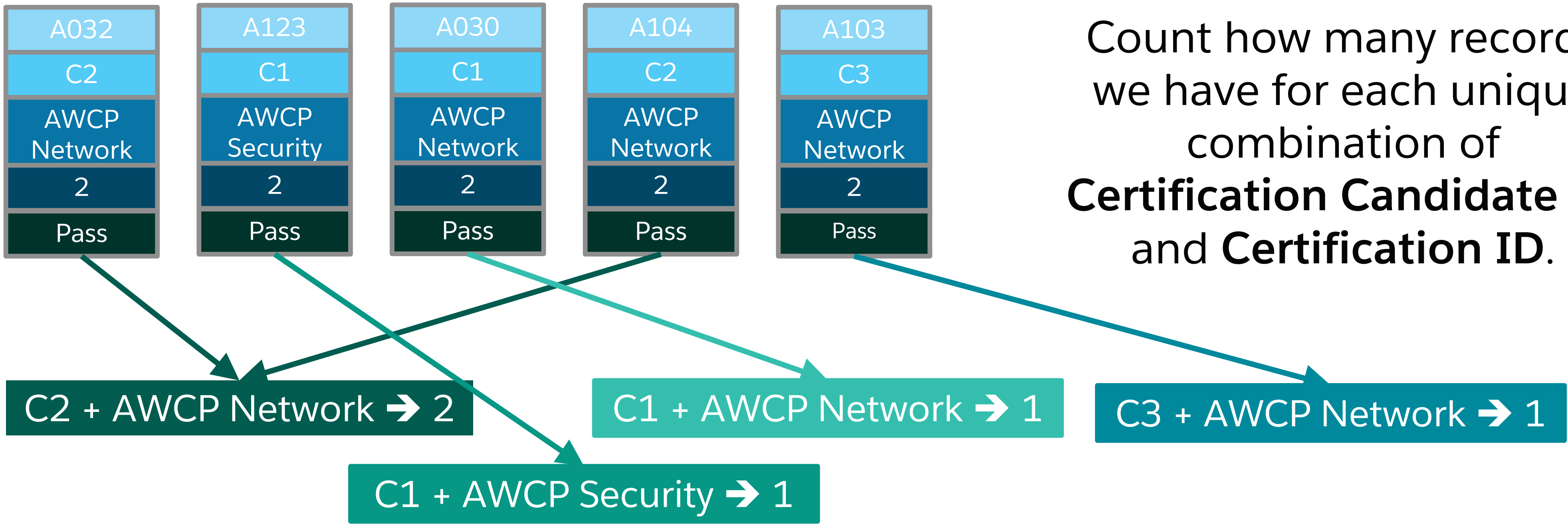
Goal:

Create the query that will bring back Certification Attempt records.

Tasks:

1. Create the collections needed to filter the query.
2. Write a SOQL for loop that performs the query.
3. Change code to call the logic for the new class handler method.
4. Test your new trigger logic.

HOW IS THE COUNT FOR PASSED CERTIFICATION ATTEMPTS ACCOMPLISHED?



Count how many records we have for each unique combination of **Certification Candidate ID** and **Certification ID**.

What type of collection can be used to store these results?

Key	Value
C2 + AWCP Network	2
C1 + AWCP Security	1
C1 + AWCP Network	1
C3 + AWCP Network	1

YOUR TURN

14-4: USE A MAP TO AGGREGATE RESULTS

salesforce

20 minutes

Goal:

Aggregate the results of the query for use in determining if a candidate has passed all elements of a certification.

Tasks:

1. Add logic to construct and use a Map to aggregate query results.
2. Test your new trigger logic.

WHAT ARE THE FINAL STEPS IN DETERMINING IF A CERTIFICATION HELD RECORD SHOULD BE CREATED?

`Trigger.new` (on insert)

A123	A104	A103
C1	C2	C3
AWCP Security	AWCP Network	AWCP Network
2	2	2
Pass	Pass	Pass

Need 2 records for the key `C1 + AWCP Security` to create a Certification Held record.

Will certification candidate C2 hold a new certification?

Will certification candidate C3 hold a new certification?

Map

Key	Value
`C2 + AWCP Network`	2
`C1 + AWCP Security`	1
`C1 + AWCP Network`	1
`C3 + AWCP Network`	1

1 record found.

No record created.

YOUR TURN

14-5: CREATE CERTIFICATION HELD RECORDS

15 minutes

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.

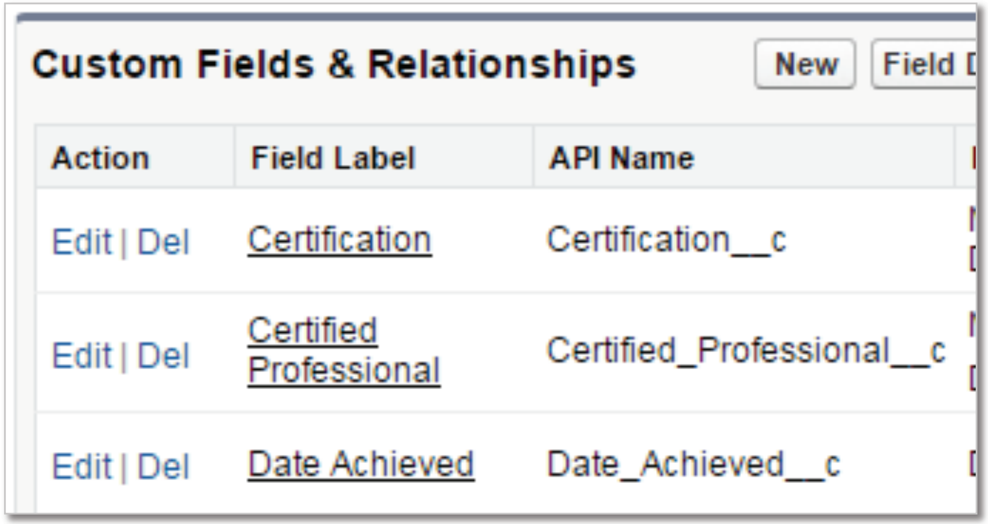
HOW CAN YOU AVOID CREATING DUPLICATE CERTIFICATION HELD RECORDS?

How can we assure that duplicate Certification Held records are not created, whether through a trigger or through the user interface?

What qualifies a Certification Held record as a duplicate?

Can you create a key that can be used to avoid duplicate records?

- Duplicate rule?
- Workflow?
- Trigger?



Custom Fields & Relationships New Field

Action	Field Label	API Name
Edit | Del	Certification	Certification__c
Edit | Del	Certified Professional	Certified_Professional__c
Edit | Del	Date Achieved	Date_Achieved__c

Solution:

A new text field marked as unique composed of the two values.

YOUR TURN

14-6: USE A WORKFLOW TO AVOID CREATION OF DUPLICATE RECORDS (OPTIONAL)

10 Minutes

Goal:

Create a workflow rule that will prevent duplicate Certification Held records from being created.

Task:

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.

KEY TAKEAWAYS

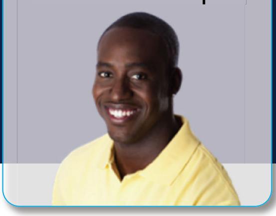
When creating a solution you should consider:

- Declarative business logic options.
- How declarative features, such as formula fields and roll-up summary fields, can help to simplify and improve the performance of your code.

MODULE 15:
CREATING VISUALFORCE PAGES

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

Ryan Jackson
Lead Developer



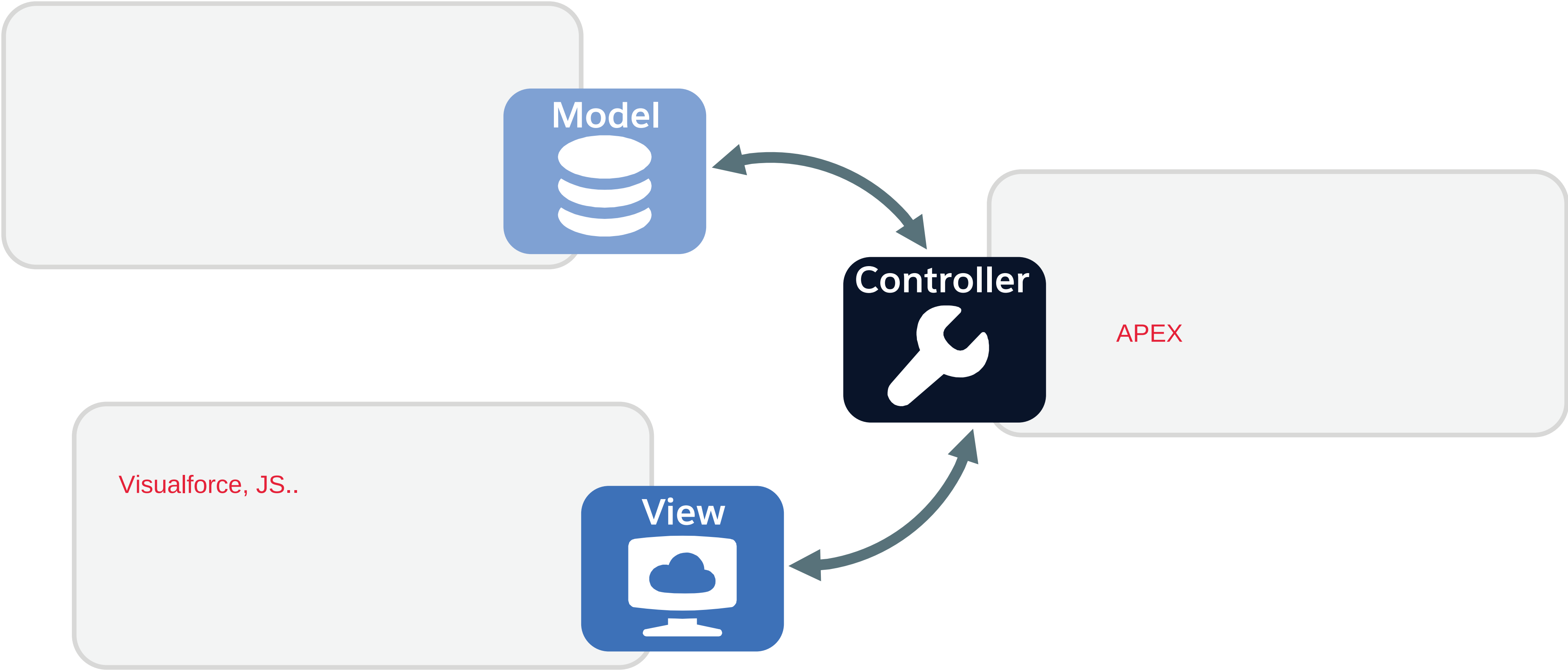
We need to create a page that allows the operations team to print a course completion certificate for students.

To accomplish this, you need to:

- Create a Visualforce page.
- Reference a standard controller.
- Launch a Visualforce page using a custom button.
- Display data from a record in a Visualforce page.

- **Understanding Visualforce**
- Creating a Visualforce Page
- Displaying Record Data and Launching a Visualforce Page

MAPPING SALESFORCE ELEMENTS TO THE MVC



WHAT IS VISUALFORCE? - MARKUP

DEFINITION:

Visualforce markup language: A tag-based markup language, similar to HTML.

```
<apex:page controller="MyController_CC" tabStyle="Account">
    <apex:form >
        <apex:pageBlock id="contactList" title="Verify Accounts">
            <apex:panelGrid cellspacing="4" columns="6" >
            …
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

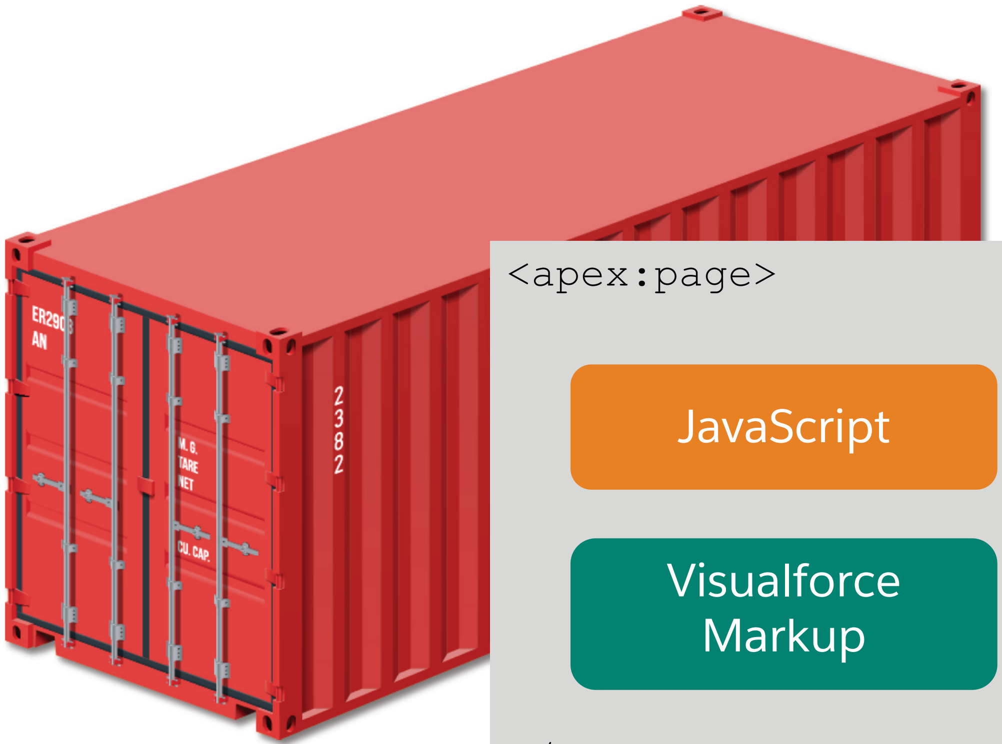
DEFINITION:

Visualforce controllers: Standard controllers provided by Salesforce or custom controllers or controller extensions written in Apex

```
public class MyController_CC {
    private final Account account;
    public MyController_CC() {
       account = [SELECT Id, Name, Site FROM Account WHERE Id =
                   :ApexPages.currentPage().getParameters().get('id')];
   }
    public PageReference save() {
       update account;
       return null;
    }
}
```

WHAT IS A VISUALFORCE PAGE?

A Visualforce page is a container.



<apex:page>

JavaScript

HTML

Visualforce Markup

CSS

</apex:page>

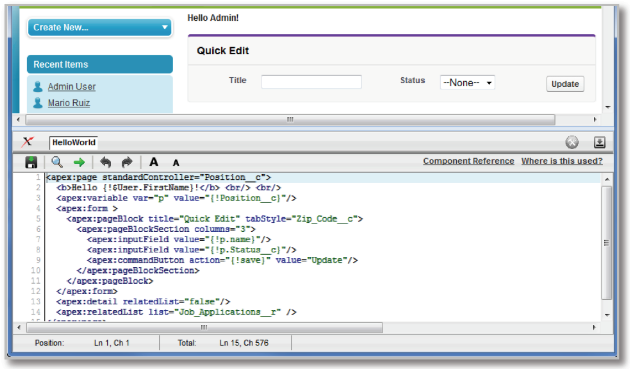
VISUALFORCE PAGES AND THE MVC

Out of the Box (classic SalesForce)		Using Visualforce
Standard and Custom Objects	**Model**	Standard and Custom Objects
Page Layouts	**View**	Visualforce Pages
Standard Controller	**Controller**	Standard and Custom Controllers

MODULE 15: CREATING VISUALFORCE PAGES

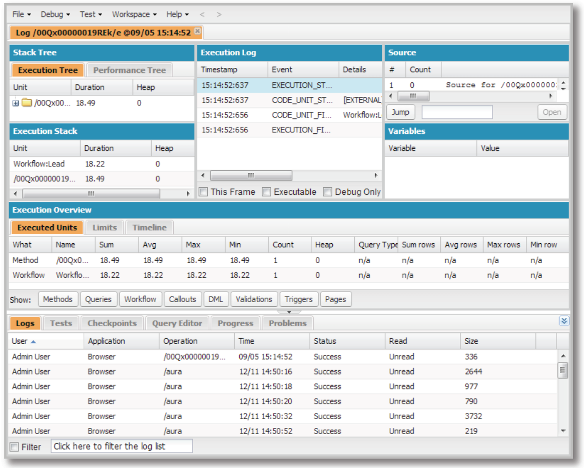
- Understanding Visualforce
- **Creating a Visualforce Page**
- Displaying Record Data and Launching a Visualforce Page

TOOLS FOR CREATING A VISUALFORCE PAGE



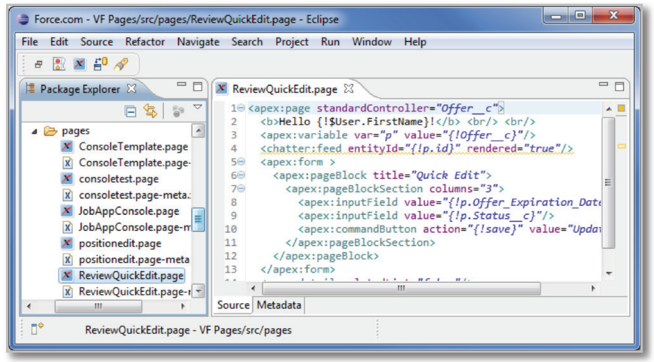
Inline Editor

Turn on Development Mode to access the inline editor for Visualforce pages and controllers.



Developer Console

A built-in browser-based IDE that is part of your Salesforce org.



Force.com IDE

A plug-in for Eclipse

NOTE: There are also a number of other third-party tools for authoring Visualforce.

ELEMENTS OF A VISUALFORCE PAGE: VISUALFORCE MARKUP

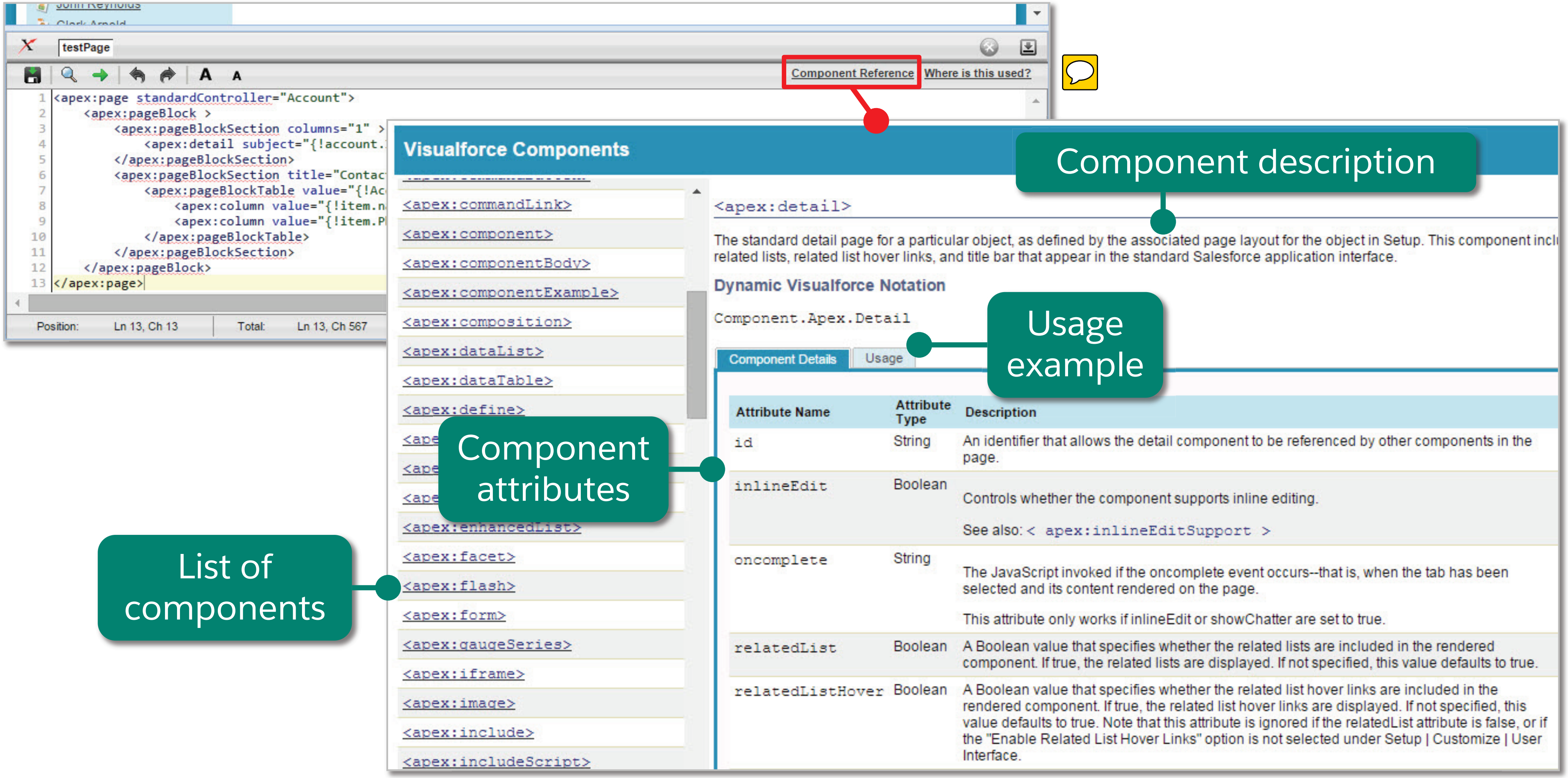
Identifies Account's standard controller

Includes a custom style sheet (stored as a static resource) in the page.

Includes JavaScript (stored as a static resource) in the page.

```
<apex:page standardController="Account"
        standardStylesheets="false" showHeader="false">

    <apex:stylesheet value="{!$Resource.customCSS }"/>

        <apex:includeScript value="{!$Resource.jQuery }"/>

        <apex:dataTable value="{!account.contacts}" var="c"
            headerClass="tablehead" rowClasses="odd,even" >
            <apex:column>
                <apex:facet name="header">First Name</apex:facet>
                <apex:outputText value="{!c.FirstName}"/>
            </apex:column>
            <apex:column>
                <apex:facet name="header">Last Name</apex:facet>
                <apex:outputField value="{!c.LastName}"/>
            </apex:column>
         </apex:dataTable>
</apex:page>
```

Displays data about the contacts.


testPage
Component Reference
Where is this used?
<apex:page standardController="Account">
<apex:pageBlock >
<apex:pageBlockSection columns="1" >
<apex:detail subject="{!account.
</apex:pageBlockSection>
<apex:pageBlockSection title="Contac
<apex:pageBlockTable value="{!Ac
<apex:column value="{!item.n
<apex:column value="{!item.P
</apex:pageBlockTable>
</apex:pageBlockSection>
</apex:pageBlock>
</apex:page>
Position: Ln 13, Ch 13 Total: Ln 13, Ch 567
Visualforce Components
<apex:commandLink>
<apex:component>
<apex:componentBody>
<apex:componentExample>
<apex:composition>
<apex:dataList>
<apex:dataTable>
<apex:define>
<apex:enhancedList>
<apex:facet>
<apex:flash>
<apex:form>
<apex:gaugeSeries>
<apex:iframe>
<apex:image>
<apex:include>
<apex:includeScript>
<apex:detail>
The standard detail page for a particular object, as defined by the associated page layout for the object in Setup. This component incl
related lists, related list hover links, and title bar that appear in the standard Salesforce application interface.
Dynamic Visualforce Notation
Component.Apex.Detail
Component Details
Usage
Attribute Name | Attribute Type | Description
id | String | An identifier that allows the detail component to be referenced by other components in the page.
inlineEdit | Boolean | Controls whether the component supports inline editing. See also: < apex:inlineEditSupport >
oncomplete | String | The JavaScript invoked if the oncomplete event occurs--that is, when the tab has been selected and its content rendered on the page. This attribute only works if inlineEdit or showChatter are set to true.
relatedList | Boolean | A Boolean value that specifies whether the related lists are included in the rendered component. If true, the related lists are displayed. If not specified, this value defaults to true.
relatedListHover | Boolean | A Boolean value that specifies whether the related list hover links are included in the rendered component. If true, the related list hover links are displayed. If not specified, this value defaults to true. Note that this attribute is ignored if the relatedList attribute is false, or if the "Enable Related List Hover Links" option is not selected under Setup | Customize | User Interface.
Component description
Usage example
Component attributes
List of components

EXPRESSION SYNTAX

Tags use the same expression syntax as formula fields, validation rules, etc. All content in `{!...}` is evaluated as an expression.

Object Name

Field Name

```
<apex:inputField value="{!Course__c.Duration__c}"/>
```

Contains the expression

Global variables, whose names start with `$`, can be accessed using the same syntax. For example:

- `{!$User.`*`fieldName`*`}`
- `{!$Page.`*`otherVisualforcePage`*`}`
- `{!$Resource.`*`staticResource`*`}`

15-1: CREATE A SIMPLE VISUALFORCE PAGE

10 minutes

Goal:

Create a simple Visualforce page that displays your name.

Tasks:

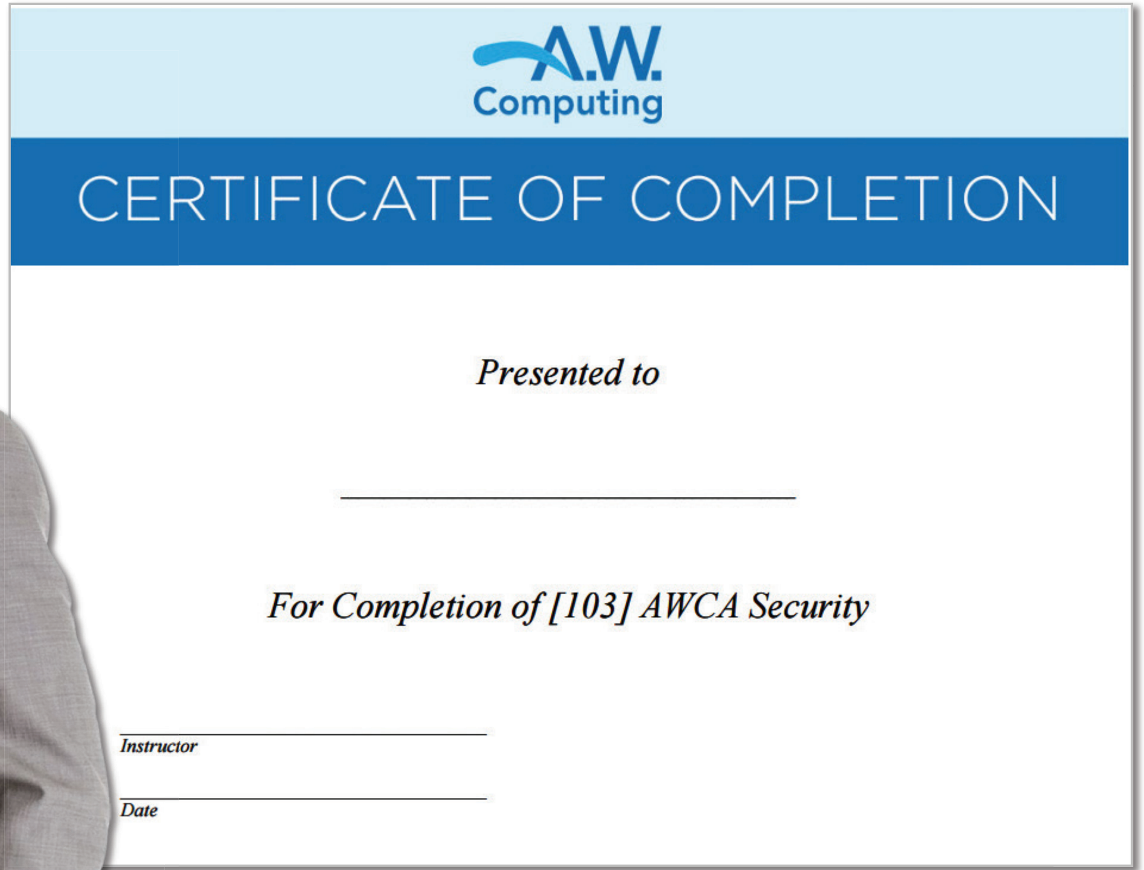
1. Create a Visualforce page using the inline editor.
2. Add static text to the page.
3. Add a reference to the global user variable to display your name.

MODULE AGENDA

- Understanding Visualforce
- Creating a Visualforce Page
- **Displaying Record Data and Launching a Visualforce Page**

CREATING A COURSE COMPLETION CERTIFICATE

Now you'll need to pull in some data from the Course object to build this course completion certificate and decide how to launch the page.


A.W. Computing

CERTIFICATE OF COMPLETION

Presented to

____

For Completion of [103] AWCA Security

____
Instructor

____
Date

Ryan Jackson
Lead Developer

- The standardController attribute specifies the object that is used to control the behavior of this page, and the style of the tab that will display the page.

```
<apex:page standardController="Account">
```

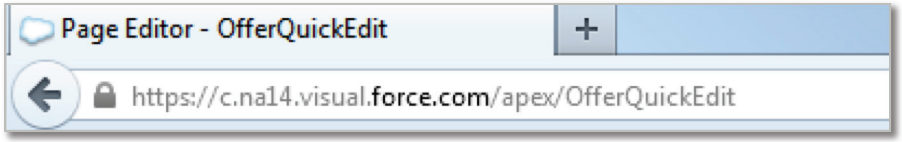
- The Id parameter on the URL binds the page to a single record, giving it data context.

```
https://na1.salesforce.com/apex/myPage?id=0013000000gzexd
```

NOTE: The record Id being passed in must be for a record of the same object as that specified by the standardController attribute.

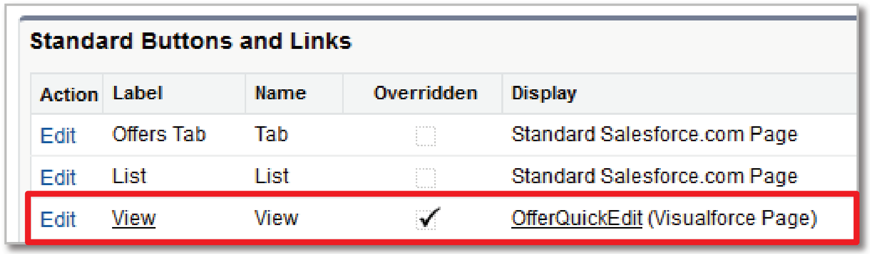
LAUNCHING VISUALFORCE PAGES

- Provide the URL for the page.

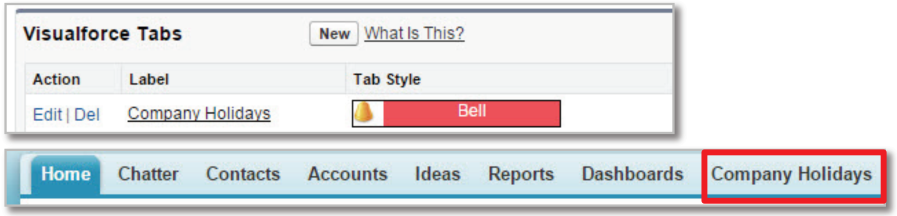


- Create a custom button or link.

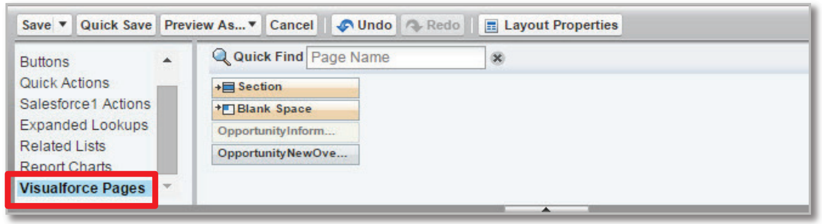
- Override one of the standard actions.



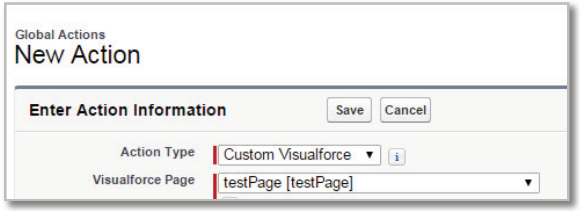
- Create a custom Visualforce tab.



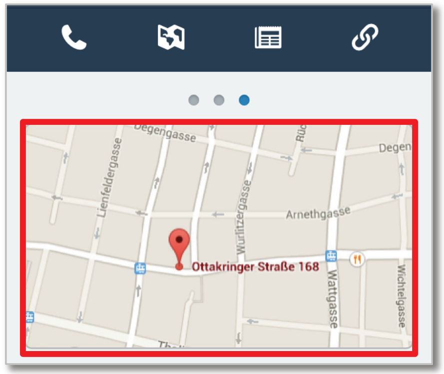
- Include inline in a page layout.



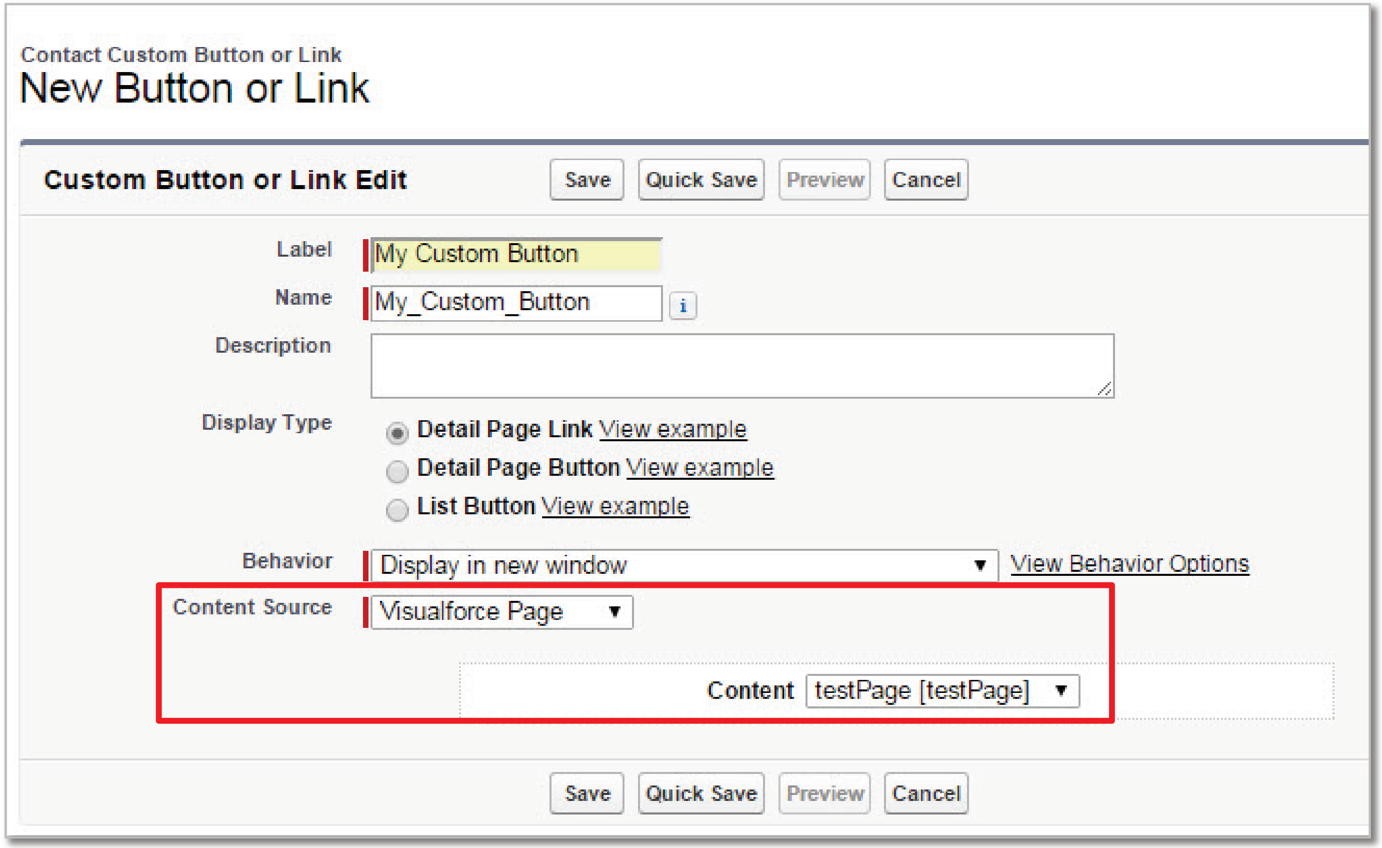
- Create a custom action.



- Create a mobile card.



LAUNCHING A VISUALFORCE PAGE WITH A CUSTOM BUTTON



NOTE:

You must add your custom button to a Page Layout for users to see it.

15-2: DISPLAY DATA IN A VISUALFORCE PAGE

15 minutes

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 the Developer Console.
3. Create a custom button to launch the new Visualforce page.
4. Test the new page.

KEY TAKEAWAYS

- Visualforce allows you to customize the View and Controller layers of Salesforce.
- Visualforce consists of a tag-based markup language and standard and custom controllers.
- Standard controllers provide access to an sObject's data and allow you to perform common database operations on that data.
- A Visualforce page can be launched in a number of ways, including by creating a custom button, overriding a standard action, or creating a custom Visualforce tab.

Developer Beginner | Visualforce Basics
(2 hours, 25 minutes)

Developer Beginner | Quick Start: Visualforce
(15 minutes)

MODULE 16:
EXPLORING THE VIEW AND CONTROLLER LAYERS OF VISUALFORCE

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

TRAILHEAD

EXPLORING THE VIEW AND CONTROLLER LAYERS OF VISUALFORCE

We need you to create a page that shows a technician's status at a glance, including certifications held, courses taken, and any certifications in progress. This page should allow the user to easily navigate to related records and edit the current record.

To accomplish this, you need to:

- Create a Visualforce page.
- Display related data.
- Invoke standard controller actions.

MODULE 16: EXPLORING THE VIEW AND CONTROLLER LAYERS OF VISUALFORCE

- **Accessing Data on Related Records**
- Exploring Visualforce Tags and Built-in Styling

TRAVERSING AN sObject RELATIONSHIP IN AN EXPRESSION

As with API and Apex queries, you can use expression syntax to retrieve data from objects related to the current object:

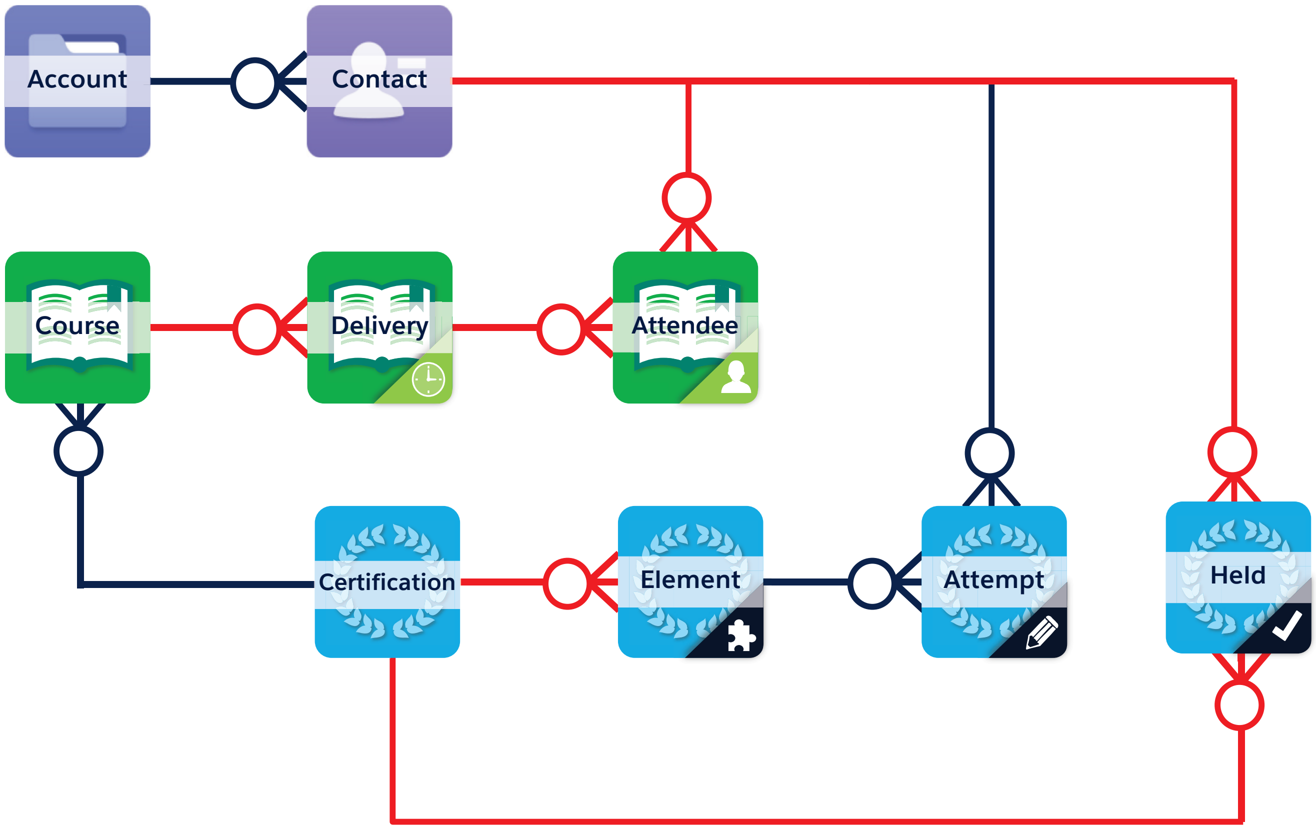
- You can traverse up five levels of child-to-parent relationships.
  – Example:

```
<apex:page standardController="Contact">
        <apex:outputField value="{!contact.Account.Owner.FirstName}">
</apex:page>
```

- You can traverse down one level of parent-to-child relationships to return an array of all child rows for that parent.
  – Example:

```
<apex:page standardController="Account">
    <apex:pageBlock>
        <apex:pageBlockTable value="{!account.Contacts}">
 ...
</apex:page>
```

Can you use a standard controller? Which one?



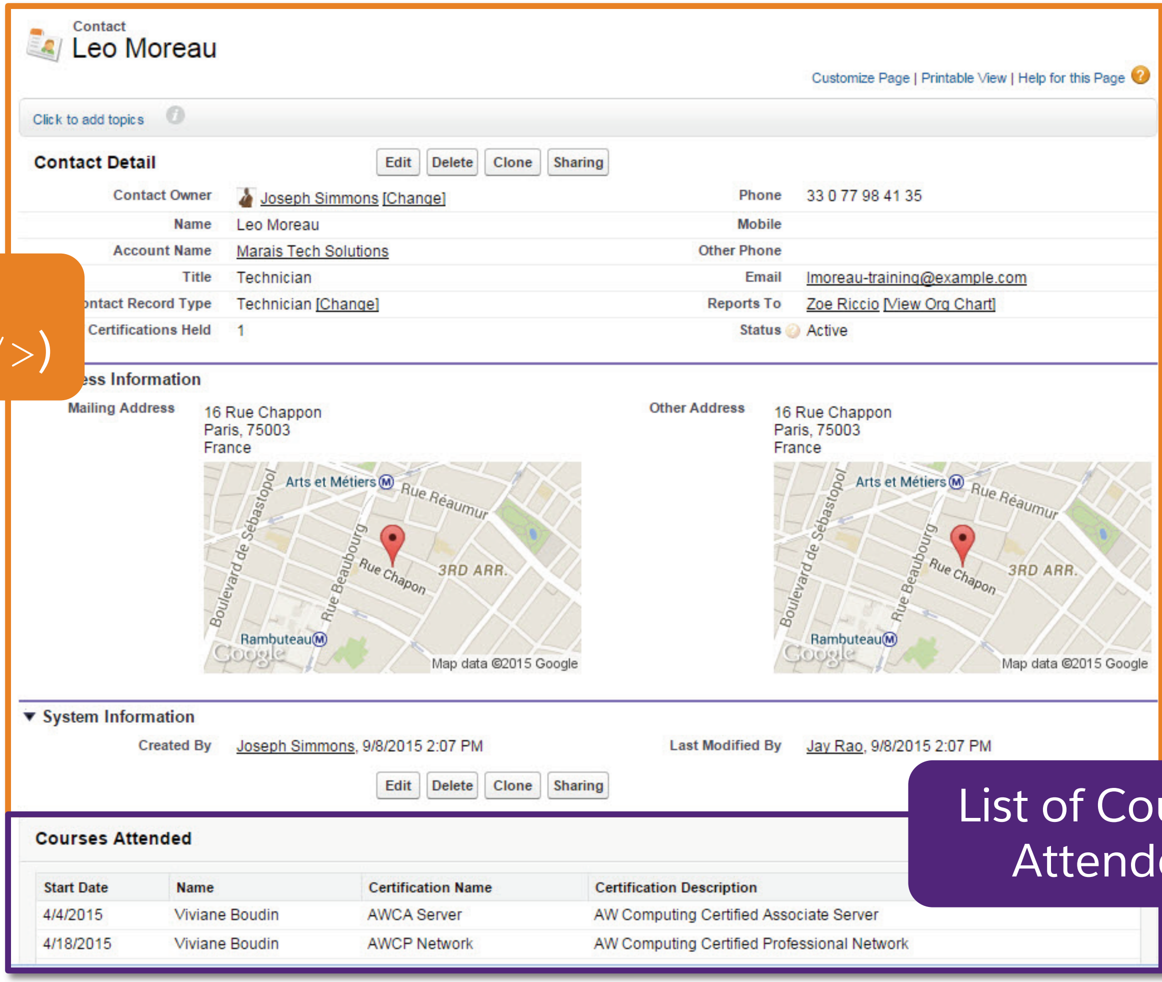
Page One: A class roster displaying the course name, date, and students.

Page Two: Display all of the Certification Attempts related to a Certification on the Certification Record.

Page Three: A technician status page showing Technician Name, Employer Name, Certifications Held, and Courses Attended.

You need to create a technician status page that looks like this:

Ryan Jackson
Lead Developer



Record Detail
(<apex:detail/>)

List of Courses Attended

HOW CAN YOU DISPLAY CHILD RECORDS IN A VISUALFORCE PAGE?

The name of the variable that represents one element of the collection.

Identifies the collection to be displayed.

```
<apex:page standardController="Account">
    <apex:pageBlock>
        <apex:pageBlockTable var="item" value="{!account.Contacts}">
            <apex:column value="{!item.name}"/>
        </apex:pageBlockTable>
    </apex:pageBlock>
</apex:page>
```

Name
Pat Stumuller
Jane Argano
Clark Arnold
John Reynolds
Marian Lane

These tags work with collections:

<apex:pageBlockTable> <apex:dataTable> <apex:dataList> <apex:repeat>

HOW SHOULD WE LAUNCH THIS PAGE?

1. Custom button or link.
2. Override a standard action.
3. Custom Visualforce tab.
4. Inline in a page layout.

JOIN ME

16-1: CREATE A SIMPLE TECHNICIAN STATUS PAGE

salesforce

15 minutes

Goal:

Create a simple technician status page to display technician name, account name, and courses attended. 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. Complete the TODO sections to display the necessary data.
4. Create a custom button on the Technician Page Layout to launch the page.
5. Test your new page.

MODULE AGENDA

salesforce

MODULE 16: EXPLORING THE VIEW AND CONTROLLER LAYERS OF VISUALFORCE

- Accessing Data on Related Records
- **Exploring Visualforce Tags and Built-in Styling**

Story

REFINE THE TECHNICIAN STATUS PAGE

Refine the Technician Status page by displaying only the fields you need, then add links to allow users to navigate to each related course delivery record.

Technician Status Edit Technician Record

Technician Name	Leo Moreau	Email	lmoreau-training@example.com
Phone	33 0 77 98 41 35	Account	Marais Tech Solutions

Course Listing

Date	Name	Certification Name	Certification Description
Sat Apr 04 00:00:00 GMT 2015	Viviane Boudin	AWCA Server	AW Computing Certified Associate Server
Sat Apr 18 00:00:00 GMT 2015	Viviane Boudin	AWCP Network	AW Computing Certified Professional Network
Sat Jun 13 00:00:00 GMT 2015	Viviane Boudin	AWCA Server	AW Computing Certified Associate Server

Edit Technician Record

Ryan Jackson
Lead Developer

COARSE- VS. FINE-GRAINED VISUALFORCE TAGS

Coarse-grained tags bring in a large chunk of Salesforce UI. This includes:

- <apex:detail>
- <apex:enhancedList>
- <apex:listView>
- <apex:relatedList>

Fine-grained tags bring in smaller pieces, giving you more flexibility. This includes:

- <apex:outputField>
- <apex:inputField>
- <apex:pageBlock>
- And many more

HOW CAN YOU CALL ACTION METHODS?

You can call an action in the action parameter of a variety of Visualforce tags, using the {!...} notation. For example, this tag:

```
<apex:commandButton action="{!edit}"value="My Edit Button"/>
```

Creates this button: My Edit Button which calls the edit method on a standard controller.

Action-Aware Tags	Use Case
<apex:commandButton>	Creates a button that calls an action.
<apex:commandLink>	Creates a link that calls an action.
<apex:actionPoller>	Periodically calls an action.
<apex:actionSupport>	Makes an event (such as "onclick," "onmouseover," etc.) on another (named) component call an action.
<apex:actionFunction>	Defines a new JavaScript function that calls an action.
<apex:page>	Calls an action when the page is loaded.

The standard controller action methods include: save, quicksave, edit, delete, cancel, and list.

NAVIGATING WITH OUTPUT LINKS

The `value` attribute specifies the destination of the link.

The Link will resolve to the page layout for this record.

Option 1

```
<apex:outputLink value="/{!item.id}">Link</apex:outputLink>
```

Option 2 (recommended)

```
<apex:outputLink value="{!URLFOR($Action.Course__c.View, item.Id)}">Link</apex:outputLink>
```

Converts the specified action and target record into a URL.

Uses the global $Action variable to access the View action on the Course controller.

Specifies which record to use when calling the action.

The global $Action variable lets you call **any** action on that controller, including the New action, Edit action, List action, and any other actions specific to that object.

YOUR TURN

16-2: REFINE YOUR PAGE AND ADD NAVIGATIONAL LINKS

10 minutes

Goal:

Refine your Visualforce page to display only the necessary fields, then add navigational links so users can easily edit related records.

Tasks:

1. Overwrite the existing TechnicianStatus Visualforce page to only display the necessary fields.
2. Complete the TODO sections to refine your page and add navigational links.
3. Test the new page.

KEY TAKEAWAYS

- You can use expression syntax to retrieve data from objects related to the current object.
- These Visualforce tags work with collections:
  <apex:pageBlockTable>
  <apex:dataTable>
  <apex:dataList>
  <apex:repeat>
- You can use action-aware tags to call methods from a controller.

MODULE 17:

WORKING WITH CUSTOM CONTROLLERS AND CONTROLLER EXTENSIONS

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

EXTENDING STANDARD CONTROLLERS

Cassie Evans

Developer

We need you to create some Visualforce pages that require functionality that is not available with a standard controller.

To accomplish this, you need to:

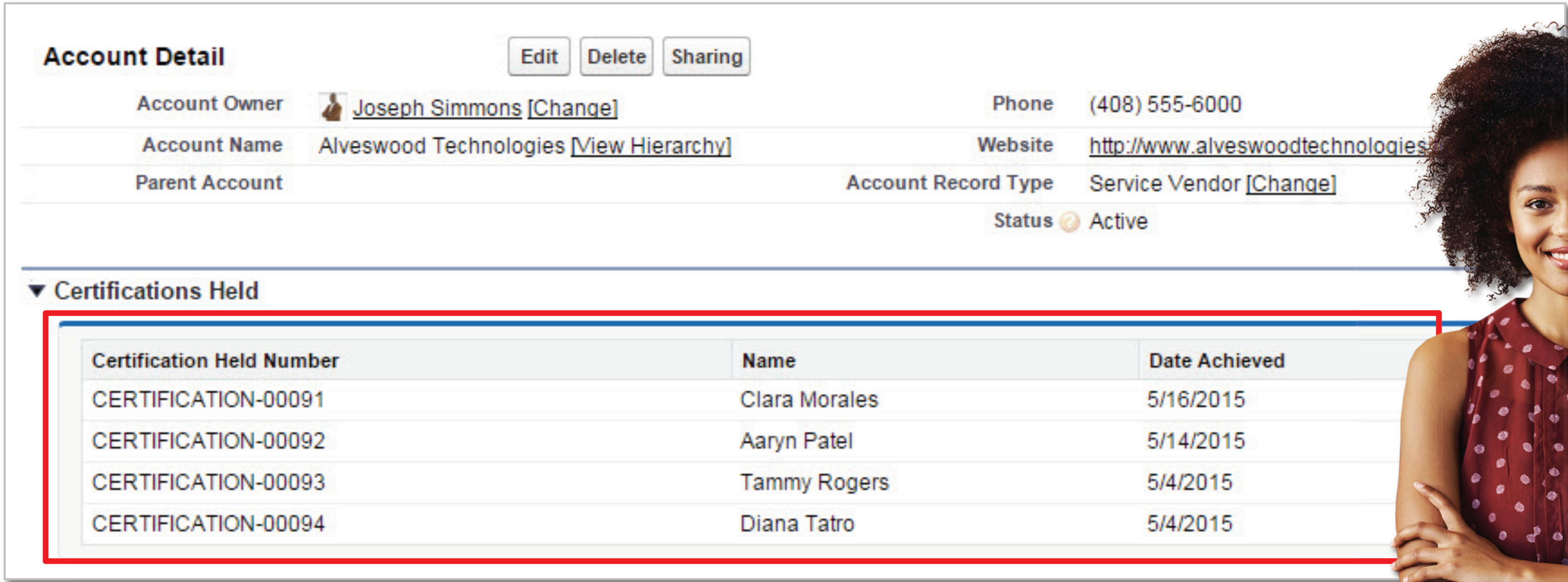
- Create controller extensions.
- Create a custom controller.
- Work with properties.
- Use Page References.
- Invoke custom methods in Visualforce pages.

- **Referencing Custom Controllers and Controller Extensions**
- Working with Getters, Setters, and Properties
- Working with Action Methods
- Handling Basic Errors

Story

REFERENCING CUSTOM CONTROLLER CODE

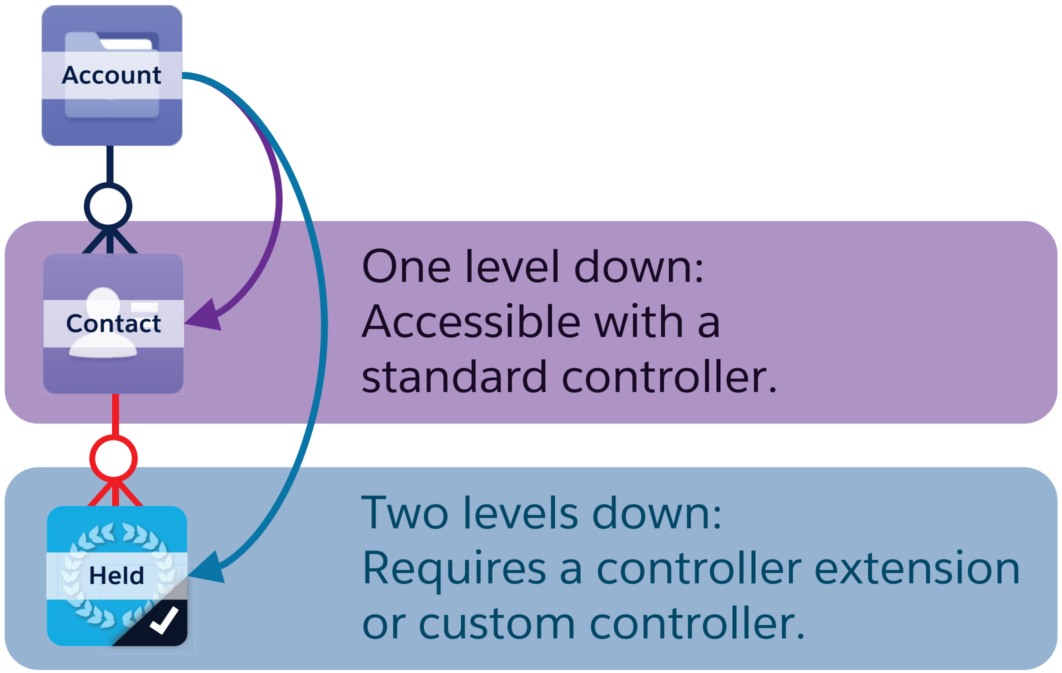
We need to create a Visualforce page to display all the certifications held by technicians associated with an account on the account page layout. I've already written the controller extension you'll need to get the data; you just need to reference it in the page.



Cassie Evans
Developer

WHY CAN'T YOU USE A STANDARD CONTROLLER?

You need a custom controller or controller extension because you must display data that is not accessible with a standard controller.



Additional reasons to use a custom controller or controller extension:

- Create custom behaviors
- Override existing functionality
- Customize the navigation

WHAT CAN CONTROLLER EXTENSIONS AND CUSTOM CONTROLLERS PROVIDE?

Getter methods that allow the view to retrieve data from the controller

Setter methods that allow the view to set data in the controller

Properties that can be used to get and set or store values

Action methods to perform logic or navigation

```
public class TheController {
    String searchText;

    public String getSearchText() {
        return searchText;
    }

    public void setSearchText(String s) {
        searchText = s;
    }

    public List<Lead> results { get; set;}

    public PageReference doSearch() {
        results = (List<Lead>)[ FIND :searchText
                                RETURNING Lead(Name, Email) ][0];
        return null;
    }
}
```

WHY MUST A PAGE USE A CONTROLLER EXTENSION?

There are two reasons you must choose a controller extension over a custom controller:

1. You want to use functionality already existing in the standard or custom controller you are using for the page.
2. You need to use your page with declarative Salesforce features that depend on a standard controller, such as creating a custom button or including your page within a page layout.

NOTE:

A custom controller extension can be used with either a standard controller OR a custom controller class.

CHOOSING A CUSTOM CONTROLLER VS. A CONTROLLER EXTENSION

Should you use a custom controller or a controller extension?

1. A quick-create page, allowing the user to create an account, contact, and opportunity, all from one page.
2. A page with custom functionality that can be launched using a button on a page layout.
3. A multi-page wizard, which guides a user through the process of designing a sales plan for a new customer.
4. A page that, on saving a new certification record, automatically navigates the user to a page where they can create a new related certification element.

ALERT:

A custom controller must explicitly define **all** data and action methods that will be used on the page, including methods already found in standard controllers if you need them.

INVOKING CONTROLLER EXTENSIONS AND CUSTOM CONTROLLERS IN VISUALFORCE PAGES

salesforce

- Controller Extensions are invoked using the extensions attribute of the opening `<apex:page>` tag:

```
<apex:page standardController="controllerName" extensions="ExtensionName, OtherExtension">
```

- Custom controllers are invoked using the controller attribute of the opening `<apex:page>` tag:

```
<apex:page controller="customControllerName" >
```

NOTE:

A single page can include only one main controller (standard or custom), but it may reference several controller extensions.

REFERENCING A PROPERTY VALUE IN A VISUALFORCE PAGE

Cassie has created a read-only property to retrieve a list of Certifications Held related to the current account. You'll need to reference the property in your page:

```
public with sharing class AccountDisplayCertsHeld_CX {

    public List<Certification_Held__c> results {
        get {
            if (results == null) {
              results = [SELECT Id, Name, Date_Achieved__c,
                 Certified_Professional__r.Name
                         FROM Certification_Held__c
                         WHERE Certified_Professional__r.Account.Id = : account.Id];
            }
            return results;
         }
        private set;
    }

    private final Account account;

    // Constructor used to get the Account record
    public AccountDisplayCertsHeld_CX(ApexPages.StandardController stdController) {
        this.account = (Account)stdController.getRecord();
    }
}
```

In a Visualforce page, you can reference the list of results using expression syntax: `{!results}`

YOUR TURN

17-1: REFERENCE A CONTROLLER EXTENSION IN A VISUALFORCE PAGE

salesforce

15 minutes

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. Upload the pre-existing controller extension in the org.
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.

MODULE 17: WORKING WITH CUSTOM CONTROLLERS AND CONTROLLER EXTENSIONS

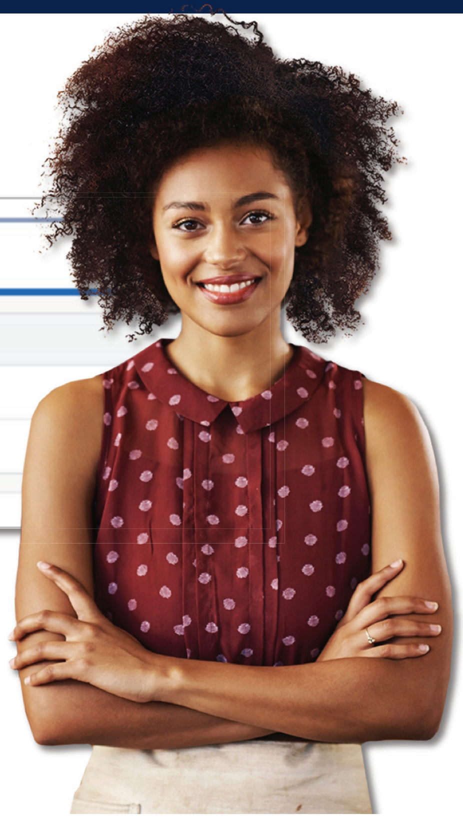
- Referencing Custom Controllers and Controller Extensions
- **Working with Getters, Setters, and Properties**
- Working with Action Methods
- Handling Basic Errors

WRITING A READ-ONLY PROPERTY

We need to create a second Visualforce page to display currently in-progress certification attempts associated with an account embedded on the page layout. You'll need to create another Visualforce page and a controller extension to access the data.

Certification Attempts

Certification Attempt Number	Name	Certification Element Name
ATTEMPT-00138	Joshua Wagner	AWCA Network Multiple Choice
ATTEMPT-00139	Grant Wei	AWCA Server Multiple Choice



Cassie Evans
Developer

CONTROLLER EXTENSION CONSTRUCTORS

A controller extension constructor takes an argument of type `ApexPages.StandardController` or a custom controller class.

- The extension can use this variable to call `getId()` or `getRecord()` to access the sObject created by the primary controller.

```
public class myControllerExtension {
    private final Account acct;
    public myControllerExtension(ApexPages.StandardController stdController) {
        this.acct = (Account)stdController.getRecord();
    }
    public String getGreeting() {
        return 'Hello ' + acct.name + ' (' + acct.id + ')';
    }
}
```

penser à caster

NOTE: Other than the constructor, custom controllers can contain all the same elements seen in controller extensions.

WRITING A GETTER METHOD

Getter methods return object data from a controller to a Visualforce page.

A controller or extension `with sharing` respects the sharing model set up in the org. A controller or extension `without sharing` bypasses that model.

On force pour que l'utilisateur ne puisse pas faire de bêtises

```
public with sharing class MyController {
    public String getName() {
        return 'MyController';
    }
    public Account getAccount() {
        return [SELECT Id, Name from Account
          WHERE Id = :ApexPages.currentPage().getParameters().get('Id')];
  }
}
```

NOTE:

Every value calculated by a controller and displayed in a page must have a corresponding getter method or property.

REFERENCING A GETTER METHOD

In a Visualforce page, data from a getter method can be accessed using the `{!DataName}` expression syntax.

```
public with sharing class MyController {
    public String getName() {
        return 'MyController';
    }
}
```

```
<apex:page controller="MyController">
    This page is using the {!name} controller!
</apex:page>
```

WRITING A SETTER METHOD

Setter methods pass user-specified values from a page to a controller.

```
public with sharing class MySearchController {
    String privateSearchText;
    List<Lead> results;
    public String getSearchText() {
        return privateSearchText;
    }
    public void setSearchText(String s) {
        privateSearchText = s;
    }
}
```

Setter methods always take the form `set`*`DataName`*`()`.

NOTE:

Every time an action method is invoked, all setter methods are automatically executed before the action method.

AUTOMATIC SET BEHAVIOR

Visualforce calls setter on primitives only. Controllers have getters returning sObjects but do not need setters for non-primitive values.

```
public class myControllerExtension {
    private final Account acct;
    public myControllerExtension(ApexPages.StandardController
              stdController) {
        this.acct = (Account)stdController.getRecord();
    }
    public String getGreeting() {
        return 'Hello ' + acct.name + ' (' + acct.id + ')';
    }
}
```

This stores the account object in the controller.

USING PROPERTIES FOR GETTING AND SETTING

- Apex properties are similar to variables and can be used in place of getter and setter methods.
- Property definitions can include a `get` block, a `set` block, or both.
- A property might look like this:

```
public class BasicProperty {
    public integer prop {
        get { return prop; }
        set { prop = value; }
    }
}
```

- If no additional functionality is required, you can use automatic properties to make coding easier:

```
public class AutomaticProperty {
    public integer MyReadOnlyProp { get; }
    public double MyProperty { get; private set; }
}
```

NOTE:

Properties with a private setter or without a setter are read-only.

YOUR TURN

17-2: CREATE A SIMPLE READ-ONLY PROPERTY

salesforce

15 minutes

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.

MODULE AGENDA

salesforce

MODULE 17: WORKING WITH CUSTOM CONTROLLERS AND CONTROLLER EXTENSIONS

- Referencing Custom Controllers and Controller Extensions
- Working with Getters, Setters, and Properties
- **Working with Action Methods**
- Handling Basic Errors

Story

CREATING THE COURSE DELIVERY SEARCH PAGE

When I'm signing technicians up for training, I'd like to be able to search for several courses at once and see all of their upcoming course deliveries, so I can put together a schedule. Ideally, the search page would look like this:

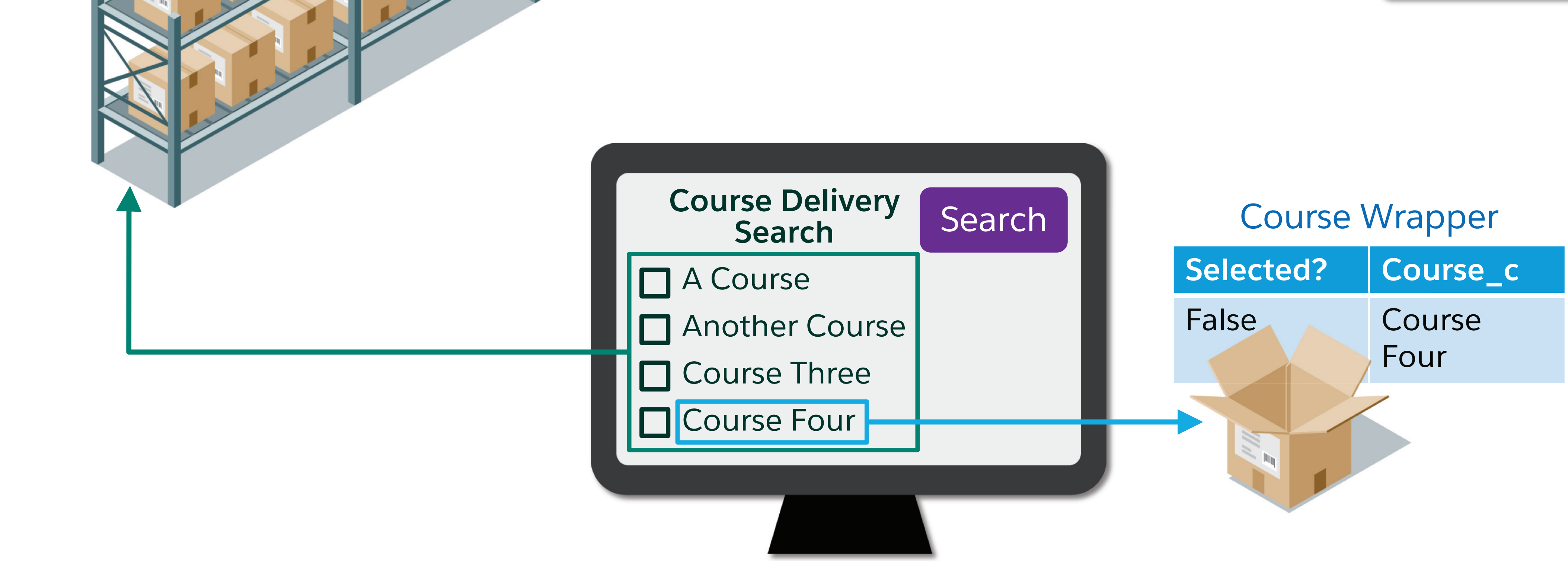
Course Search [See Upcoming Course Deliveries]

Select	Course Name	Certification Name	Duration	Status
☑	[201] AWCP Server	AWCP Server	5	Active
☐	[102] AWCA Network	AWCA Network	4	Active
☐	[103] AWCA Security	AWCA Security	4	Active
☐	[302] AWCM Network	AWCM Network	5	Active
☐	[203] AWCP Security	AWCP Security	5	Active
☐	[101] AWCA Server	AWCA Server	4	Active
☑	[301] AWCM Server	AWCM Server	5	Active
☑	[303] AWCM Security	AWCM Security	5	Active
☐	[202] AWCP Network	AWCP Network	5	Active
☐	[401] Data Recovery		3	Active
☐	[402] Managing Network Load		3	Active

[See Upcoming Course Deliveries]

Nicki Sanchez
Training Coordinator

WHAT DO YOU NEED TO IMPLEMENT THE SEARCH PAGE?



WHAT IS A WRAPPER CLASS?

DEFINITION:

A **Wrapper class** is commonly used to extend the properties of an sObject. It is an Apex class with a public property of the type we want to wrap and, optionally, some other properties.

```
public class CourseWrapper {
    public Course__c course {get; set;}
    public Boolean checked {get; set;}

    public CourseWrapper(Course__c c) {
        course = c;
        checked = false;
    }
}
```

An individual Course__c sObject

The boolean indicating whether the checkbox is checked

The constructor, which sets the value of course to the Course__c we pass in, then sets checked to false

YOUR TURN

17-3: CREATING A READ/WRITE PROPERTY IN A CUSTOM CONTROLLER

25 minutes

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.

IMPLEMENTING THE SEARCH BUTTON

When a user clicks the Search button, your controller should create a new list, this time containing only the selected Courses. You will use this list to search for future course deliveries.

All Courses

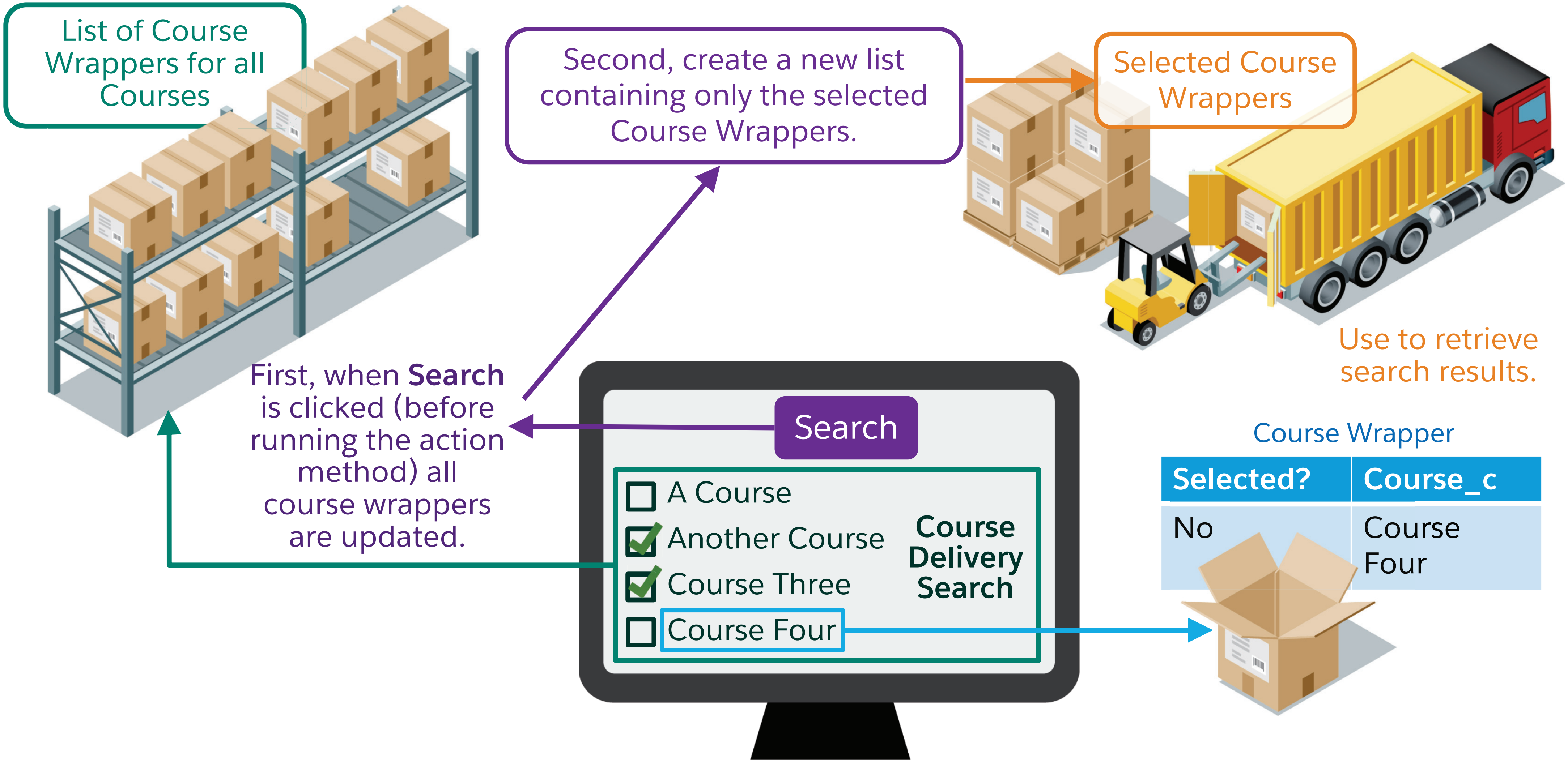
Selected?	Course__c
False	Course One
True	Course Two
True	Course Three
False	Course Four

Selected Courses

Selected	Course__c
True	Course Two
True	Course Three

Cassie Evans
Developer

HOW WILL THE SEARCH PAGE WORK?



Selected?	Course_c
No	Course Four

Action methods in controllers and controller extensions are bound to the `action` attribute in action-aware Visualforce tags. You use them just like you'd use a standard controller method:

```
<apex:commandButton action="{!save}" value="My Save Button" >
```

```
<apex:commandButton action="{!myCustomAction}" value="My Custom Action Button" >
```

YOUR TURN

17-4: IMPLEMENT THE SEARCH BUTTON

salesforce

10 minutes

Goal:

Create an action method to create a list of just the selected courses.
Your method should fire when the user clicks the **Search** button.

Tasks:

1. Write the code necessary to create a map of selected courses.
2. Test the code changes.

NAVIGATING AND DISPLAYING SEARCH RESULTS

After the user clicks **Search**, your page should deliver them to a page that displays all of the results of their search like this:

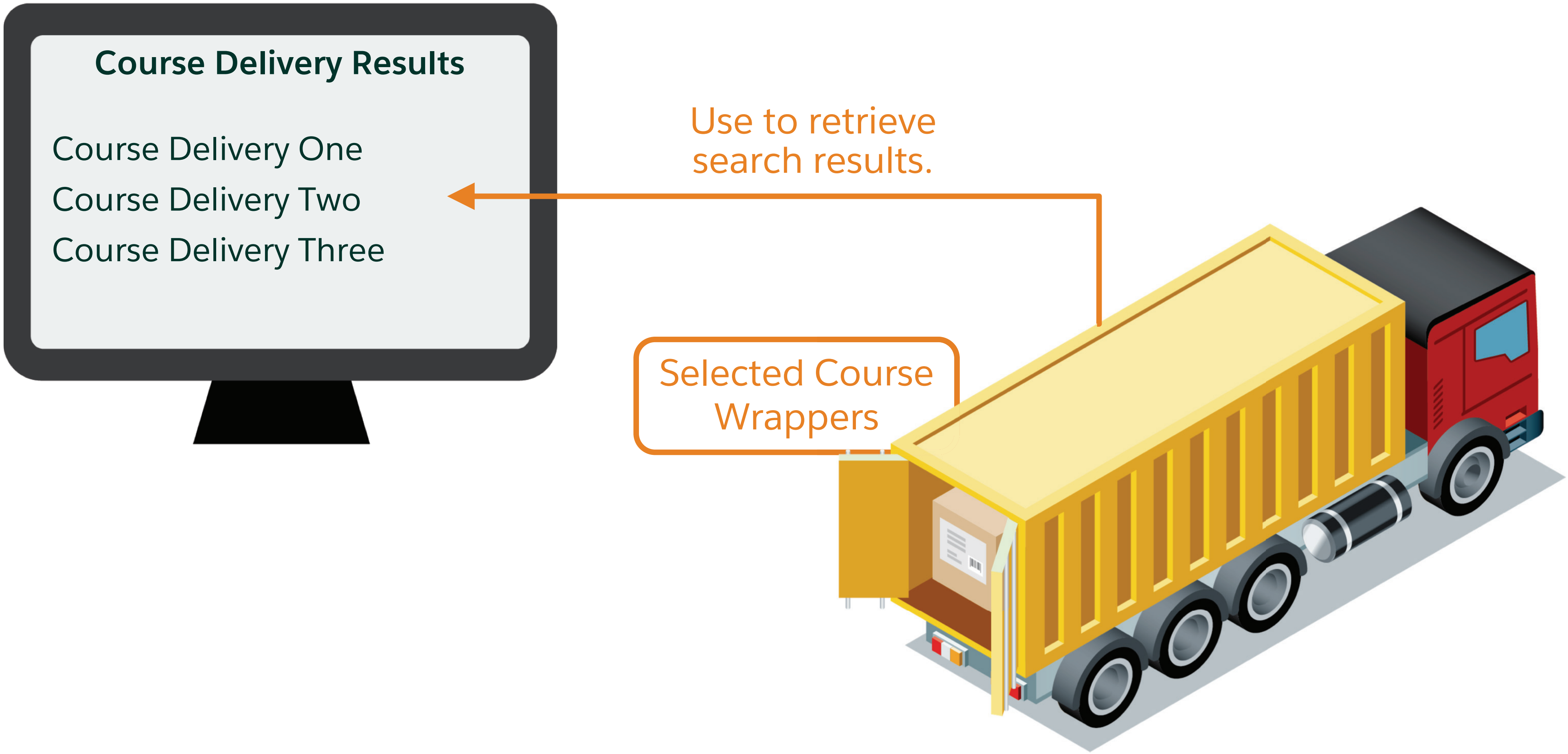
Cassie Evans
Developer

Upcoming Course Deliveries New Search

Course Name	Course Delivery Number	Instructor Name	Location	Start Date	Status
[201] AWCP Server	DELIVERY-00028	Raymond Montoya	San Francisco, US	8/9/2015	Scheduled
[301] AWCM Server	DELIVERY-00030	Sasha Vincent	Chicago, US	8/30/2015	Scheduled
[303] AWCM Security	DELIVERY-00031	Heidi Rosen	London, GB	9/13/2015	Scheduled

New Search

HOW WILL THE SEARCH PAGE WORK?



WORKING WITH PageReferences

DEFINITION: The Apex **PageReference** object represents a UI page. It consists of a URL and a set of query parameter name/value pairs, among other values.

A `PageReference` object is used to:

- View or set URL query string parameter name/value pairs for a page.
- Navigate the user to a different page as the result of an action method.

Au chargement d'une page, si get/set, et constructeur et bouton submit :
Viewstate, puis appel du constructeur.
Ensuite, appel de tous les getter de la page !! Même si pas utilisés.
On écrit alors dans le viewstate (le viewstate maintient l'état des variables entre le controller et la page)

INSTANTIATING PAGES USING PageReferences

Depending on the type of page you plan to return, the instantiation of a PageReference varies.

Destination Page	Instantiation
Current Page	`PageReference pageRef = ApexPages.currentPage();`
Visualforce Page	`PageReference pageRef = Page.visualforcePageName;`
A (possibly non-Visualforce) Salesforce Page	`PageReference pageRef = new PageReference('partialURL');`
Non-Salesforce Website	`PageReference pageRef = new PageReference('fullURL');`

WORKING WITH URL PARAMETERS

Use the `getParameters()` and `get()` methods to obtain the parameters passed to the executing page.

```
1A // Get the Id that was passed to the current page
2A PageReference pageRef = ApexPages.currentPage();
3A id recordId = pageRef.getParameters().get('id');
```

Use the `getParameters()` and `put()` methods to add parameters to the page that you want to navigate to.

```
1B // When you want to pass parameters to another page...
2B PageReference pageRef2 = Page.Page2;
3B pageRef2.getParameters().put('id', recordId);
```

Parameters are processed as key-value pairs in a map.

Use the following methods to retrieve additional information from the `PageReference` object:

- `getContent()`
- `getHeaders()`
- `getParameters()`
- `getRedirect()`
- `getUrl()`

NAVIGATING TO A NEW PAGE USING A PAGEREFERENCE

To navigate to a new page as part of an action method, return a PageReference:

```
public class MySecondController {
    Account account;
    public Account getAccount() {
        if (account == null) account = new Account();
        return account;
    }
    public PageReference save() {
        insert account;
        PageReference acctPage = new ApexPages.StandardController(account).view();
        acctPage.setRedirect(true);
        return acctPage;
    }
}
```

YOUR TURN

17-5: REDIRECTING TO A RESULTS PAGE

25 minutes

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.

MODULE AGENDA

salesforce

MODULE 17: WORKING WITH CUSTOM CONTROLLERS AND CONTROLLER EXTENSIONS

- Referencing Custom Controllers and Controller Extensions
- Working with Getters, Setters, and Properties
- Working with Action Methods
- **Handling Basic Errors**

WRITING CONTROLLER CODE TO HANDLE ERRORS

salesforce

```
public PageReference save() {
    try {
        update record;
    }
    catch(System.DMLException ex) {
        ApexPages.addMessages(ex);
        return null;
    }

    return new PageReference('/' + record.Id);
}
```

To handle any exception that might be thrown, use try and catch blocks.

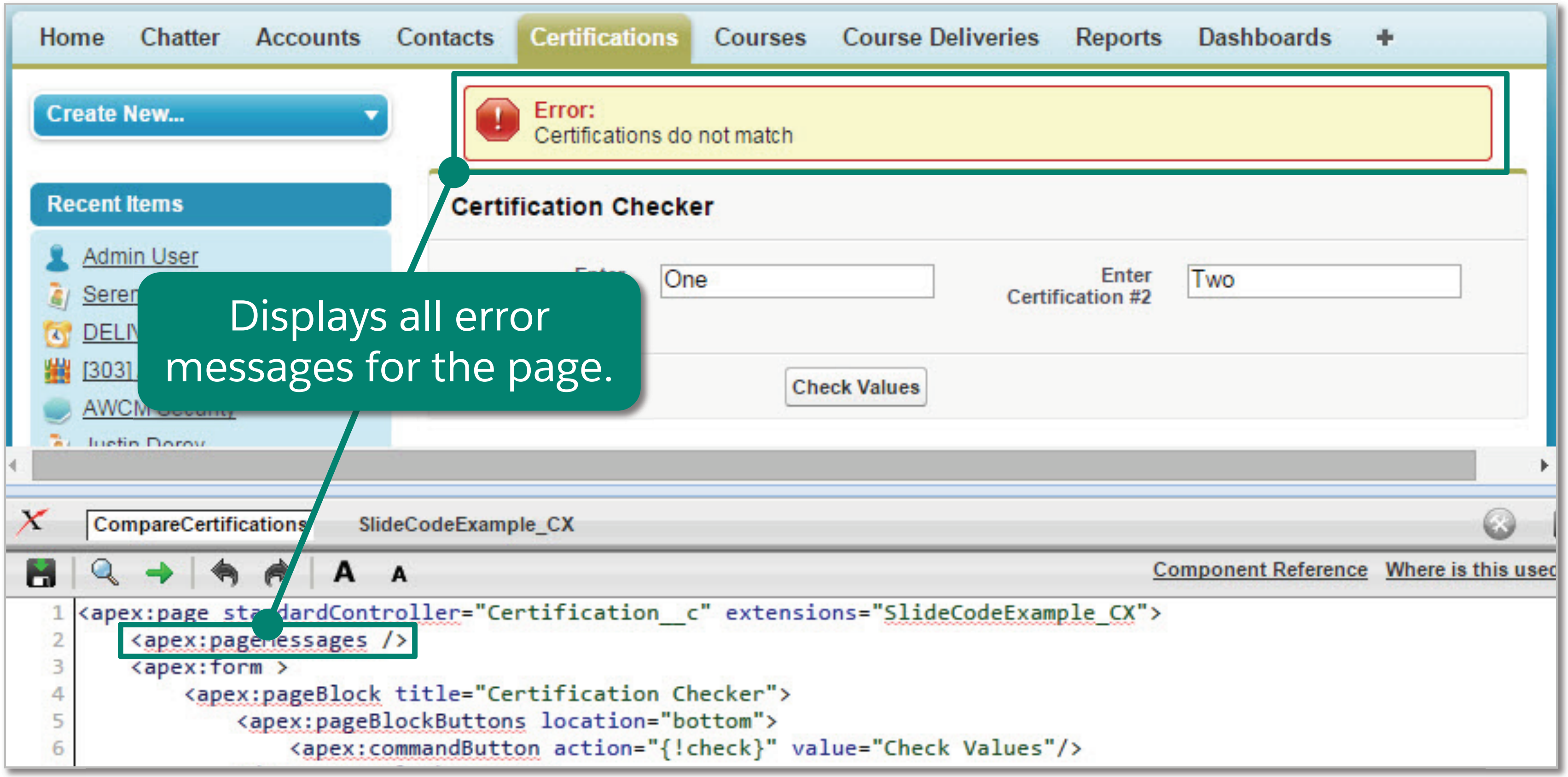
```
public void check() {
    if ( cert1 == cert2 ) {
        ApexPages.addMessage(new
         ApexPages.message(ApexPages.Severity.INFO,
         'YEAH! Certifications match'));
    } else {
        ApexPages.addMessage(new
         ApexPages.message(ApexPages.Severity.ERROR,
         'Certifications do not match'));
    }
}
```

For other error conditions, write custom logic.

NOTE:

Use the `ApexPages.addMessages` method to add messages to the page.

DISPLAYING ERROR MESSAGES IN VISUALFORCE PAGES



NOTE:

The `<apex:pageMessage>` tag places messages on a Visualforce page with specific styling. When you use the `<apex:form>` tag, use `<apex:pageMessages>`.

YOUR TURN

17-6: HANDLE BASIC SAVE ERRORS IN YOUR METHOD

10 minutes

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 courses.
2. Add the `<apex:pageMessages/>` tag to your Visualforce page.
3. Test the code changes.

KEY TAKEAWAYS

- When a standard controller does not provide the functionality you need, you can create a custom controller or controller extension.
- You must choose a controller extension instead of a custom controller if you need to use features or functionality associated with a standard controller.
- Getter methods allow the view to retrieve data using the controller.
- Setter methods allow the view to set data in the controller.
- Properties can be used in place of getter and setter methods.
- Action methods can return a PageReference to perform navigation and/or implement custom logic.
- The `<apex:pageMessages />` tag allows you to display error messages.

KNOWLEDGE CHECK

salesforce

1. A developer is creating a Visualforce page that will use the standard Account controller. Which page requirements mean the developer must also include a controller extension?
2.
```
public List<Certification_Attempt__c> certAttempts {
  public get {
    if (certAttempts == null) {
      certAttempts = [SELECT Id, Name FROM Certification_Attempt__c LIMIT 10];
    }
    return certAttempts;
  }
  private set;
}
```
   Given the code above, what is true?
3. What do the `with sharing` keywords indicate when included in a Visualforce controller extension declaration?
4. You need to navigate to another Visualforce page. How should you instantiate your PageReference?

MODULE 18:
WORKING WITH LIST CONTROLLERS AND SOSL QUERIES

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

TRAILHEAD

Cassie Evans

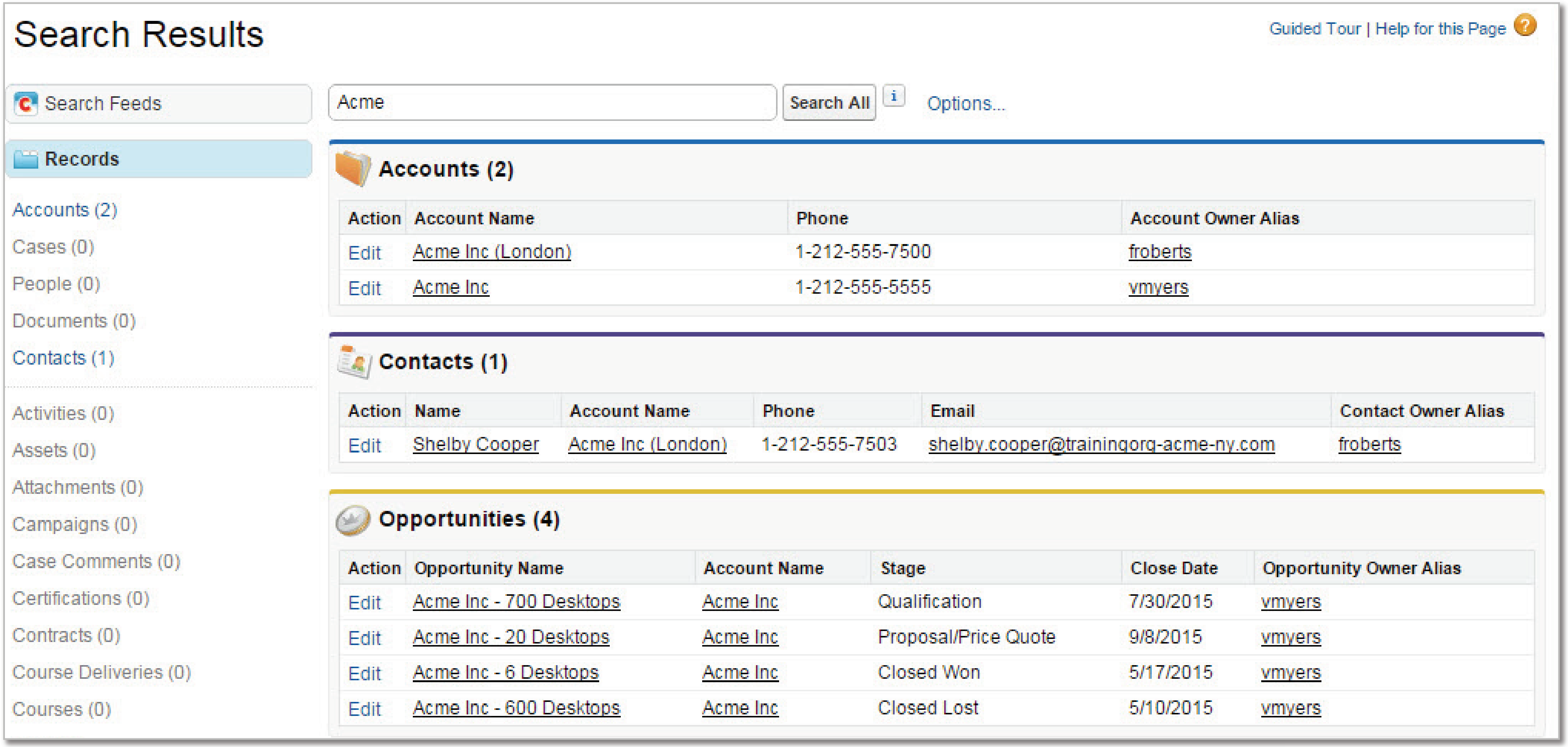
Developer

We need you to create a page that allows users to search all text fields for the Contact object.

To accomplish this, you need to:

- Understand standard list controllers.
- Create a SOSL query.
- Create a custom list controller.

When do you need to implement a custom search page?



Search Results

Guided Tour | Help for this Page

Search Feeds

Acme | Search All | Options...

Records

- Accounts (2)
- Cases (0)
- People (0)
- Documents (0)
- Contacts (1)
- Activities (0)
- Assets (0)
- Attachments (0)
- Campaigns (0)
- Case Comments (0)
- Certifications (0)
- Contracts (0)
- Course Deliveries (0)
- Courses (0)

Accounts (2)

Action	Account Name	Phone	Account Owner Alias
Edit	Acme Inc (London)	1-212-555-7500	froberts
Edit	Acme Inc	1-212-555-5555	vmyers

Contacts (1)

Action	Name	Account Name	Phone	Email	Contact Owner Alias
Edit	Shelby Cooper	Acme Inc (London)	1-212-555-7503	shelby.cooper@trainingorg-acme-ny.com	froberts

Opportunities (4)

Action	Opportunity Name	Account Name	Stage	Close Date	Opportunity Owner Alias
Edit	Acme Inc - 700 Desktops	Acme Inc	Qualification	7/30/2015	vmyers
Edit	Acme Inc - 20 Desktops	Acme Inc	Proposal/Price Quote	9/8/2015	vmyers
Edit	Acme Inc - 6 Desktops	Acme Inc	Closed Won	5/17/2015	vmyers
Edit	Acme Inc - 600 Desktops	Acme Inc	Closed Lost	5/10/2015	vmyers

MODULE 18: WORKING WITH LIST CONTROLLERS AND SOSL QUERIES

- **Working with Standard List Controllers**
- Writing a Simple SOSL Query
- Creating a Custom List Controller

STANDARD LIST CONTROLLERS

DEFINITION:

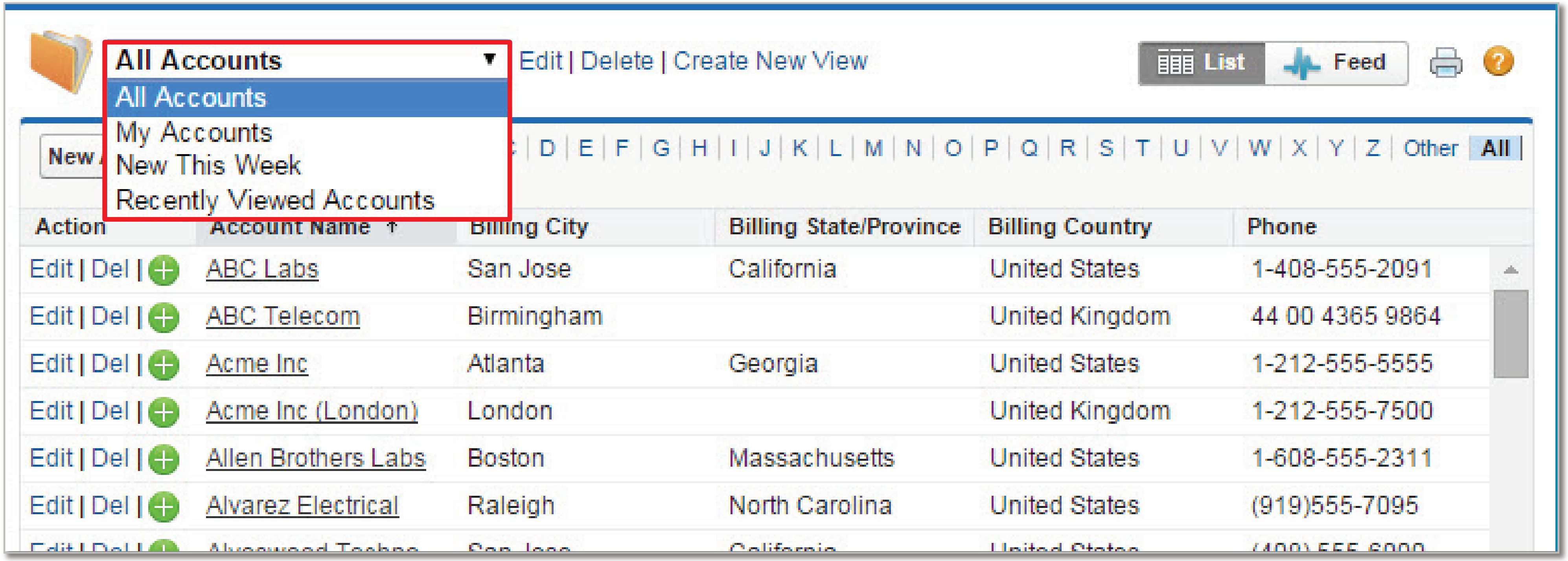
Standard list controllers allow you to create Visualforce pages that display and act on a set of records, such as list pages, related lists, and mass action pages.

Including the `recordSetVar` attribute selects the standard list controller for the account object instead of the regular account controller.

```
<apex:page standardController="Account" recordSetVar="accounts" >
```

The value of the recordSetVar attribute becomes the name of the property that enables page access to the resulting list of records.

A list view allows you to quickly view a specific set of records.



By default, Visualforce pages built using a standard list controller display records using the user's most recently accessed list view.

STANDARD LIST CONTROLLERS EXAMPLE

What will this code display?

```
<apex:page standardController="Account" recordSetVar="accounts" >
    <apex:pageBlock >
        <apex:pageBlockTable value="{!accounts}" var="acc">
            <apex:column value="{!acc.name}"/>
        </apex:pageBlockTable>
    </apex:pageBlock>
</apex:page>
```

You can create a drop-down list to allow the user to filter by a specific list view.

Creates a list of options that allows users to select a value.

Sets the `filterId` attribute on the standard list controller to the selected value.

```
<apex:pageBlock title=”Accounts" id="accountList">

    <apex:selectList value="{!filterId}" size="1">
        <apex:selectOptions value="{!listViewOptions}"/>
        <apex:actionSupport event="onchange" reRender="accountList"/>
    </apex:selectList>
    ...
</apex:pageBlock>
```

Creates the options in the selectList above.

Sets the currently-configured set of list views as the options in the list.

USING THE <apex:actionSupport> TAG

DEFINITION:

The `<apex:actionSupport>` component adds AJAX support to another component. This allows the component to be refreshed asynchronously by the server when a particular event occurs, such as a button click or mouse-over.

You can use this tag to refresh a portion of the page when the user selects a list view.

```
<apex:pageBlock title="Contacts List" id="contacts list">

    <apex:selectList value="{!filterId}" size="1">
        <apex:selectOptions value="{!listViewOptions}"/>
        <apex:actionSupport event="onchange" reRender="contacts list"/>
    </apex:selectList>
    ...
</apex:pageBlock>
```

The `event` attribute specifies the DOM event that generates the AJAX request.

The `reRender` attribute specifies the Id of the component to re-render.

PAGINATION WITH STANDARD LIST CONTROLLERS

In addition to save, quicksave, list, and cancel, standard list controllers provide additional pagination methods:

Method	Description
first()	Displays the first page of records in the set.
last()	Displays the last page of records in the set.
next()	Displays the next page of records in the set.
previous()	Displays the previous page of records in the set.
getHasNext()	Indicates whether there are more records after the current page set.
getHasPrevious()	Indicates whether there are more records before the current page set.

NOTE:

By default, standard list controllers show 20 records in each page.

USING PAGINATION METHODS


```
<apex:pageBlock id="accountList" title="Accounts">
    <apex:pageMes

        <apex:pageBlockButtons >
            <apex:commandButton status="notifyUser" reRender="accountList" value="|<"
                       title="First" action="{!first}" disabled="{!NOT(HasPrevious)}" />
            <apex:commandButton status="notifyUser" reRender="accountList" value="<"
                       title="Previous" action="{!previous}" disabled="{!!HasPrevious}" />
            <apex:commandButton status="notifyUser" reRender="accountList" value=">"
                       title="Next" action="{!next}" disabled="{!!HasNext}" />
            <apex:commandButton status="notifyUser" reRender="accountList" value=">|"
                       title="Last" action="{!last}" disabled="{!!HasNext}" />
            <apex:outputText >
                {!(pageNumber * pageSize) + 1 - pageSize}
                       through
                {!IF((pageNumber * pageSize) > resultSize, resultSize,(pageNumber *
                       pageSize))}
                       of {!resultSize} records
            </apex:outputText>
        </apex:pageBlockButtons>
        ...
```

Calls the standard `first()` method.

Disables the button if there are no previous records.

Indicates which records are on the current page.

JOIN ME

18-1: CREATE A PAGE TO DISPLAY A LIST OF RECORDS

salesforce

15 minutes

Goal:

Create a Visualforce page that displays all Account records and uses pagination and filtered list views.

Tasks:

1. Create a page to display all Account records.
2. Add pagination and filtered list views to the page.
3. Test your new page.

MODULE AGENDA

MODULE 18: WORKING WITH LIST CONTROLLERS AND SOSL QUERIES

- Working with Standard List Controllers
- **Writing a Simple SOSL Query**
- Creating a Custom List Controller

ANATOMY OF A SIMPLE SOSL FIND STATEMENT

DEFINITION:

SOSL (Salesforce Object Search Language) allows developers to search text, email, and phone fields in multiple objects simultaneously.

Data type that holds the search results.

```
List<List<sObject>> acmes =
      [ FIND 'Acme' IN ALL FIELDS RETURNING Account, Opportunity ];
```

What string are we searching for?

Which type of field should be searched?

Which type of data should be returned?

THE FIND CLAUSE: USING WILDCARDS IN SOSL SEARCH TERMS


```
[ FIND 'Acme' IN ALL FIELDS RETURNING Account, Opportunity ];
```

In addition to searching for simple strings, you can use wildcards at the middle or end of your search term.

Asterisk

Search Term	Return Values
'A*e'	Acme Acre Ace Antigone

Question Mark

Search Term	Return Values
'Ac?e'	Acme Acre

Search Term	Return Values
"Acme" OR "Global Media"	Acme Global Media

THE IN CLAUSE: SPECIFYING A SEARCH GROUP

```
[ FIND 'Acme' IN ALL FIELDS RETURNING Account, Opportunity ];
```

The `IN` *`SearchGroup`* clause tells Salesforce which fields to search. The options for *`SearchGroup`* are:

ALL FIELDS	**NAME FIELDS**	**EMAIL FIELDS**	**PHONE FIELDS**
Includes all searchable text fields.	Includes only fields where `nameField = true`.	Includes all email fields.	Includes all phone number fields.

THE RETURNING CLAUSE


```
[ FIND 'Acme' IN ALL FIELDS RETURNING Account, Opportunity ];
```

Choose which types of objects and which fields are returned using `RETURNING`.

- Return Ids from a single, specified object.

```
1A List<List<sObject>> acmes1 = [ Find 'Acme' RETURNING Account ];
```

- Return Ids from multiple, specified objects.

```
1B List<List<sObject>> acmes2 = [ FIND 'Acme' RETURNING Account, Opportunity ];
```

- Return specified fields from a single, specified object.

```
1C List<List<sObject>> acmes3 = [ FIND 'Acme' RETURNING Account(Id, Name, Phone) ];
```

- Return records that meet specific criteria in a `WHERE` clause.

```
1D List<List<sObject>> acmes4 = [ FIND 'Acme' RETURNING Opportunity(Name, Amount WHERE Amount>500) ];
```

WHERE CAN YOU USE SOSL?

You can execute SOSL searches inside:

- APIs.
- Apex statements, using:
  - Bracket notation.

```
[ Find 'Acme' RETURNING Account ]
```

  - `Search.query()`.

```
Search.query('Find {Acme} Returning Account')
```

COMPARING SOSL AND SOQL

salesforce

Use SOSL When:

- You don’t know which object or field the data resides in.
- You want to retrieve multiple (possibly unrelated) objects and fields efficiently.

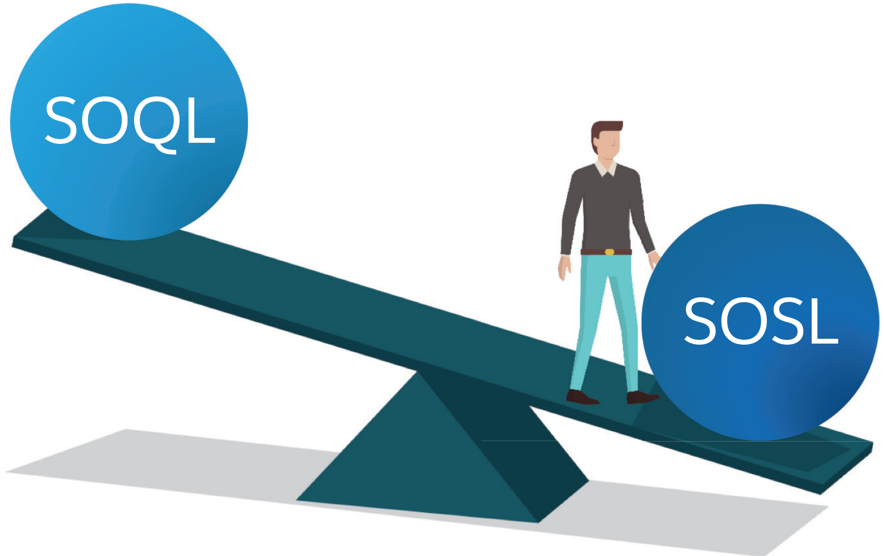
Attention, SOSL pas utilisable dans le CONTEXTE trigger !

Use SOQL When:

- You know where the data resides.
- You want data from one object or multiple related objects (using relationship queries).
- You want to count the number of records meeting criteria.
- You want data from number, date, or checkbox fields.

Which option should you choose: SOSL or SOQL?

1. Retrieve the names and Ids of all records containing the word "laptops." SOSL car on ne sait pas ou sont les données
2. Retrieve the future course deliveries of a given course, in order of the date of delivery. SOQL
3. Count the number of opportunity records with "Acme" in the name field. SOQL



WHAT DOES A SOSL SEARCH RETURN?

SOSL has only one return data type: a List of Lists of sObjects.

```
List<List<sObject>> acmes = [ Find 'Acme' RETURNING Account(Name), Opportunity(Name) ];
```

The list contains one list for each sObject specified in the RETURNING clause.

SOSL Return Structure	Returned Records: Record Name
acmes[0] (Accounts)	acmes[0][0]: ‘Acme‘
acmes[1] (Opportunities)	acmes[1][0]: ‘Acme – 1,200 Widgets‘
	acmes[1][1]: ‘Acme – 200 Widgets‘
	acmes[1][2]: ‘Acme – 600 Widgets‘

Each sObject list contains those sObects found meeting the search criteria.

USE MULTIPLE FOR LOOPS TO PROCESS A <List<List<sObject>>


```
List<List<sObject>> acmes =
      [ Find 'Acme'
        RETURNING Account(Name), Opportunity(Name) ];

//The first sObject is the List of Accounts
List<Account> acmeAccounts = acmes[0];
for (Account acmeAccount : acmeAccounts) {
    System.debug('Account: ' + acmeAccount.Name);
}

//The second sObject is the List of Opportunities
List<Opportunity> acmeOpportunities = acmes[1];
for (Opportunity acmeOpportunity : acmeOpportunities) {
    System.debug('Opportunity: ' + acmeOpportunity.Name);
}
```

Find all Account and Opportunity records containing 'Acme.'

Parse the Accounts that have been returned.

Parse the Opportunities that have been returned.

Execution Log

Timestamp	Event	Details
14:34:49:113	USER_DEBUG	[8]|DEBUG|Account: Acme Inc (London)
14:34:49:113	USER_DEBUG	[8]|DEBUG|Account: Acme Inc
14:34:49:114	USER_DEBUG	[14]|DEBUG|Opportunity: Acme Inc - 6 Desktops
14:34:49:114	USER_DEBUG	[14]|DEBUG|Opportunity: Acme Inc - 20 Desktops
14:34:49:114	USER_DEBUG	[14]|DEBUG|Opportunity: Acme Inc - 600 Desktops
14:34:49:114	USER_DEBUG	[14]|DEBUG|Opportunity: Acme Inc - 700 Desktops

VARIABLE AND EXPRESSION BINDING IN SOSL SEARCHES

You can bind expressions and variables in SOSL searches.

Bound expression

```
List<List<sObject>> acmes1 = [ Find :('Ac' + 'me') RETURNING Account ];
```

Bound variable

```
String company = 'Acme';
List<List<sObject>> acmes2 = [ Find :company RETURNING Account ];
```

GOVERNOR LIMITS FOR SOSL SEARCHES IN APEX

salesforce

Governor limits for SOSL searches include:

- Total number of SOSL searches issued.
- Total number of records retrieved by a single SOSL search.
- Total size of the heap.

RESOURCE:

You can find more information about current governor limits by searching for "Apex Governor Limits" in Help and Training.

YOUR TURN

18-2: INTEGRATE SOSL SEARCH IN A VISUALFORCE PAGE

15 minutes

Goal:

Search for text across records of various sObject types.

Tasks:

1. Construct a simple SOSL search.
2. Create a code block to cycle through the results.

MODULE 18: WORKING WITH LIST CONTROLLERS AND SOSL QUERIES

- Working with Standard List Controllers
- Writing a Simple SOSL Query
- **Creating a Custom List Controller**

We need you to create a page that lets users search all text fields on the contact object, and displays the results in a table. It should look like this:

Search Contacts

Search Text: Acme* [Search] |< < > >|

Name	Account Name	Phone	Email
Shelby Cooper	Acme Inc (London)	1-212-555-7503	shelby.cooper@trainingorg-acme-ny.com
Valerie Wilson	Acme Inc	1-212-555-5006	valerie.wilson@trainingorg-acmeinc.com
Leanne Tomlin	Acme Inc	1-212-555-5002	leanne.tomlin@trainingorg-acmeinc.com
Sandra Deely	Acme Inc	1-212-555-5003	sandra.deely@trainingorg-acmeinc.com
Carson Connelly	Acme Inc	1-212-555-5004	carson.connelly@trainingorg-acmeinc.com
Dave Carroll	Acme Inc	1-212-555-5005	dave.carroll@trainingorg-acmeinc.com

Cassie Evans
Developer

CREATING A CUSTOM LIST CONTROLLER

DEFINITION:

ApexPages.StandardSetController objects allow you to create list controllers similar to, or as extensions of, the pre-built Visualforce list controllers provided by Salesforce.

```
public class DisplayCertifications_CC {
    // Standard Set Controller
    public ApexPages.StandardSetController setCon {
        get {
            if(setCon == null) {
                setCon = new ApexPages.StandardSetController(Database.getQueryLocator(
                        [SELECT Name, Status__c FROM Certification__c]));
            }
            return setCon; }
            set;
    }
    // Return a list of records
    public List<Certification__c> getCertifications() {
        return (List<Certification__c>) setCon.getRecords();
    }
}
```

Instantiate the standardSetController to use built-in set controller methods.

Include your own custom methods.

INSTANTIATING THE StandardSetController CLASS

Create a new instance of the `StandardSetController` class based on:

- A pre-defined list of records.

```
1A  List<account> accountList = [SELECT Name FROM Account LIMIT 20];
2A  ApexPages.StandardSetController ssc = new ApexPages.StandardSetController(accountList);
```

- The results of a SOQL query.

```
1B  ApexPages.StandardSetController ssc = new
2B  ApexPages.StandardSetController(Database.getQueryLocator(
        'SELECT Name FROM Account LIMIT 20'));
```

PASSING IN USER INPUT


```
public with sharing class DisplayContacts_CC {
    public String searchText {get; set;}
        ...
    public void Search() {
        if (String.isNotBlank(searchText) && searchtext.length() > 1) {
            // Search for Contacts
            searchText = searchText + '*';
            List<sObject> contacts = [FIND :searchText IN ALL FIELDS
                                      RETURNING Contact (Id, Name, Email, Phone)][0];
            setCon = new ApexPages.StandardSetController(contacts);
        ...
```

Use a property bound to an `inputField` to get search text from the user.

Use the property to perform a SOSL query and instantiate the `standardSetController`.

YOUR TURN

18-3: CREATE A SIMPLE SEARCH PAGE

salesforce

15 minutes

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.

KEY TAKEAWAYS

- Standard list controllers allow you to create Visualforce pages that display and act on a set of records.
- Standard list controllers provide additional pagination methods that regular controllers do not.
- You can filter the records a list controller displays by setting the `filterId`.
- SOSL allows developers to search text, email, and phone fields in multiple objects simultaneously.
- SOSL queries return a List of Lists of sObjects, and can be processed using multiple for loops.
- You can create a custom list controller based on a SOQL query or custom list of records.

KNOWLEDGE CHECK

1. Which fields can you search using SOSL?
2. What can be used as the basis for a custom list controller?
3. What can a SOSL search return?
4. How can you invoke a standard list controller?

MODULE 19:
VISUALFORCE DEVELOPMENT CONSIDERATIONS

Salesforce provides powerful OOTB (out of the box) tools for application development. Before developing custom solutions using Visualforce, you need to understand the considerations and best practices.

To accomplish this, you need to:

- Determine whether a declarative solution exists for your requirements.
- Describe common governor limit issues and security concerns.
- Describe Visualforce best practices.

MODULE AGENDA

MODULE 19: VISUALFORCE DEVELOPMENT CONSIDERATIONS

- **When to Use Visualforce**
- Visualforce and Governor Limits
- Security Considerations for Visualforce
- Developing Pages for Mobile Devices
- JavaScript in Visualforce

A PAGE LAYOUT VS. A VISUALFORCE PAGE

Page Layout vs.	Visualforce Page
Interface generated automatically.	Developer-generated interface.
Maintained by administrator.	Maintained by developer.
Automatically get new features.	Does not automatically get new features.
Only accessible within the org.	Can be accessed outside of Salesforce.
Limited control of interface.	Full control of interface and behavior.

NOTE:

Only use Visualforce if standard behavior needs to be replaced or enhanced, if custom behavior is desired, or if the desired page look and feel cannot be achieved with a standard page layout.

19-1: DETERMINE WHETHER A DECLARATIVE SOLUTION EXISTS

5 Minutes

Goal:

Determine which scenarios are best solved using Visualforce.

Task:

Review each scenario, and determine if Visualforce would be the best solution.

MODULE AGENDA

MODULE 19: VISUALFORCE DEVELOPMENT CONSIDERATIONS

- When to Use Visualforce
- **Visualforce and Governor Limits**
- Security Considerations for Visualforce
- Developing Pages for Mobile Devices
- JavaScript in Visualforce

WHAT GOVERNOR LIMITS SHOULD BE CONSIDERED?

A Visualforce page and its associated custom controllers and controller extensions are subject to limits.

Limits apply to the:

- Maximum view state size in a Visualforce page.
- Maximum number of items for iteration components (e.g., `<apex:pageBlockTable>` and `<apex:repeat>`).
- Maximum number of rows retrieved by queries for a single Visualforce page request.

NOTE:

Refer to the Visualforce Limits document online for further information.

WHAT IS VIEW STATE?

Visualforce pages that contain an <apex:form> component also contain an encrypted, hidden form field that encapsulates the view state of the page.

- This field preserves page, field, and controller values between round trips to the server.
- When the HTML markup for the page is rendered, the current state of the page and values that must be retained during postback are serialized into base64-encoded strings and stored in the view state hidden field.

TRANSIENT VARIABLES EXAMPLE

Declaring variables as `transient` reduces view state size.

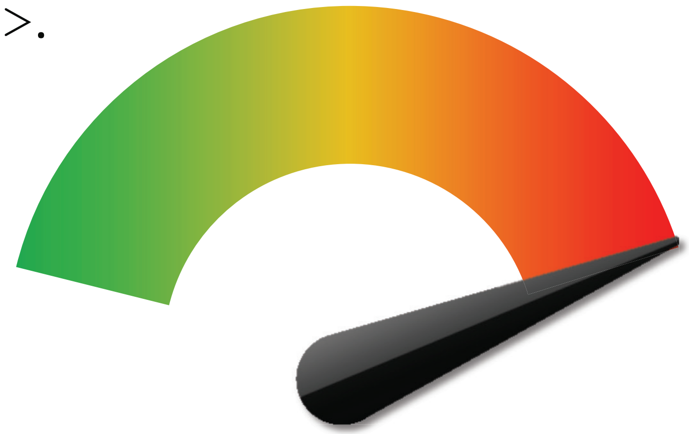
```
<apex:page controller="ExampleController">
    T1: {!t1} <br/>
    T2: {!t2} <br/>
    <apex:form>
        <apex:commandLink value="refresh"/>
    </apex:form>
</apex:page>
```

```
public class ExampleController {
    DateTime t1;
    transient DateTime t2;
    public String getT1() {
        if (t1 == null) t1 = System.now();
        return '' + t1;
    }
    public String getT2() {
        if (t2 == null) t2 = System.now();
        return '' + t2;
    }
}
```

Use the `transient` keyword to declare instance variables that should not be included in the view state of a Visualforce page.

MINIMIZING THE RISK OF HITTING GOVERNOR LIMITS

- Use the transient keyword for your instance variables (not properties) to prevent them from being included in the view state.
- Use filters in queries to limit the number of records displayed.
- Cache property values – only query for data if the property is null.
- Avoid nesting components such as <apex:repeat>, <apex:outputPanel>, and <apex:outputField>.



MODULE AGENDA

MODULE 19: VISUALFORCE DEVELOPMENT CONSIDERATIONS

- When to Use Visualforce
- Visualforce and Governor Limits
- **Security Considerations for Visualforce**
- Developing Pages for Mobile Devices
- JavaScript in Visualforce

SECURITY FOR APEX CLASSES AND CONTROLLERS

- Custom controllers, and their extensions, do not respect record-level security or profile permissions, including FLS.
- It is possible to use a profile to restrict access to a class. Note that:
  - Permissions are checked at the top level only.
  - Users with the "Author Apex" permission can access all code.

NOTE:

Use the `with sharing` keyword to enforce restrictions.

Standard controllers execute in user mode. Extension controllers do not.

DEFINITION:

SOQL injection involves modifying a SOQL statement to trick the application into performing unintended commands.

An injection can happen when user-supplied input is not validated before using it in a dynamic SOQL query. Visualforce provides tools to mitigate this common security risk. Consider this code that creates a dynamic query:

```
1A  String qryString = 'SELECT Id FROM Contact WHERE ' + '(Name like \'%' + name + '%\')';
2A  queryResult = Database.query(qryString);
```

When a user supplies the value `Bob` as input, the resultant query is:

```
1B  SELECT Id FROM Contact WHERE (Name like '%Bob%')
```

However, an unexpected value such as `test% ') OR (Name LIKE '` results in:

```
1C  SELECT Id FROM Contact WHERE (Name LIKE '%test%') OR (Name LIKE '%')
```

which unintentionally displays all Contacts.

SOQL INJECTION DEFENSES

- Use static queries and binding variables if possible.

  The previous code can be rewritten like this:

```
String queryName = '%' + name + '%';
queryResult = [SELECT Id FROM Contact WHERE Name like :queryName];
```

- If dynamic SOQL is required, use methods such as `String.escapeSingleQuotes` to sanitize user-supplied input.

19-2: DEFEND AGAINST SOQL INJECTION

10 Minutes

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.

The CertificationHeldQuickInfo VF page will display the name of the Certification and the Certified Professional when a Certification Number of a Certification Held record is supplied. It will also display the total number of professionals for that Certification.

MODULE AGENDA

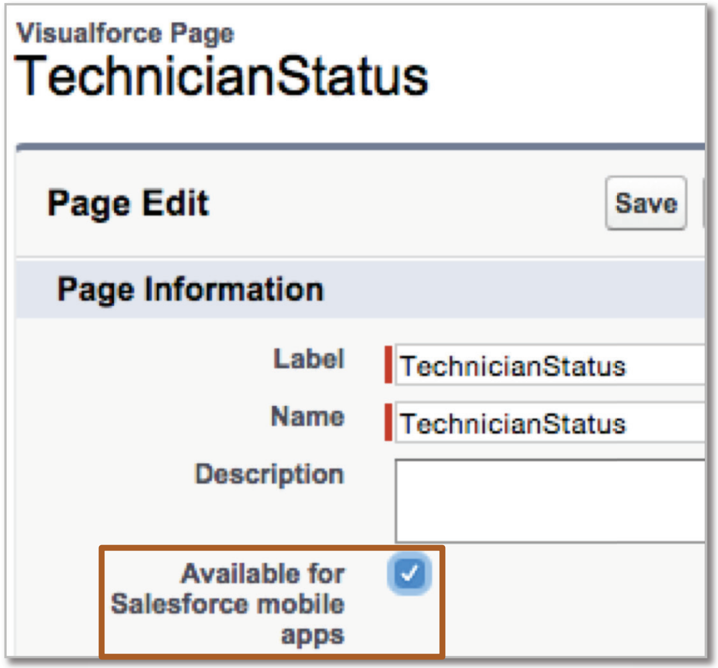
MODULE 19: VISUALFORCE DEVELOPMENT CONSIDERATIONS

- When to Use Visualforce
- Visualforce and Governor Limits
- Security Considerations for Visualforce
- **Developing Pages for Mobile Devices**
- JavaScript in Visualforce

UI CONSIDERATIONS FOR MOBILE

- Visualforce pages need to be optimized for mobile form-factors.
- Pages designed for Salesforce1 should tick the **Available for Salesforce mobile apps** checkbox.

Dos	Don'ts
Responsive Design	Desktop Optimized Design
HTML markup	Standard VF tags
Keep it simple	Support every bell and whistle
Optimize for touch	Optimize for click



WHAT ARE LIGHTNING COMPONENTS?

DEFINITION:

The **Lightning Component** framework is a UI framework for developing dynamic web apps for mobile and desktop devices.

Lightning Components:

- Are built using the Lightning Component framework.
- Are native to the platform.
- Promote rapid, modular development.
- Use an event-driven architecture.
- Encapsulate HTML, JavaScript, and CSS.

Salesforce1 is an app composed of Lightning Components.

RESOURCE:

Search for `Creating Lightning Components` in Help and Training for more detailed information on Lightning Components.

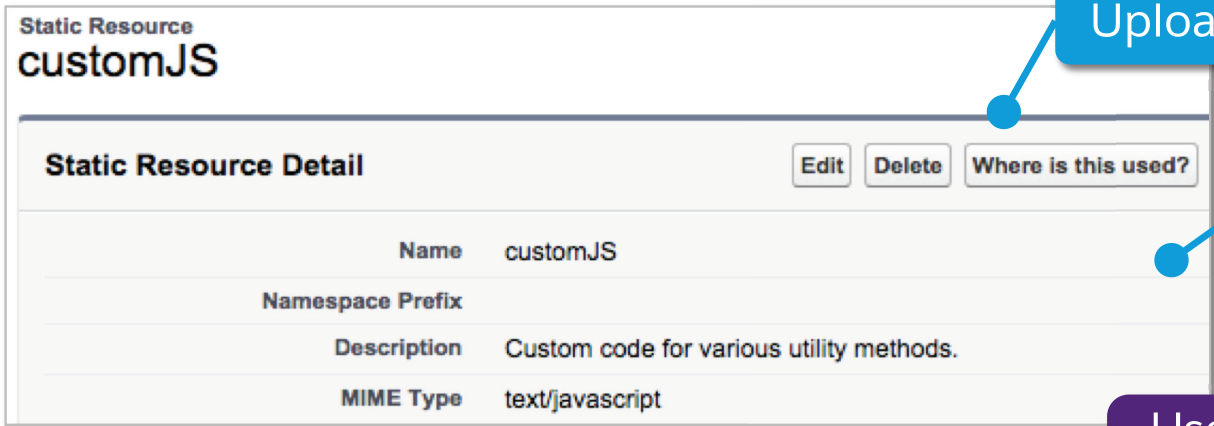
MODULE AGENDA

MODULE 19: VISUALFORCE DEVELOPMENT CONSIDERATIONS

- When to Use Visualforce
- Visualforce and Governor Limits
- Security Considerations for Visualforce
- Developing Pages for Mobile Devices
- **JavaScript in Visualforce**

CREATING CUSTOM BUTTONS THAT USE JAVASCRIPT

- Inline JavaScript can be included in Visualforce pages.
- Referencing Static Resources is useful when using JavaScript libraries, or for re-using JavaScript logic across multiple pages.



Upload the JavaScript files as Static Resources.

```
function changeFont(input, textid){
...}
```

Use `<apex:includeScript>` and `$Resource` to reference the JavaScript file.

```
<apex:includeScript value="{!$Resource.customJS}" />
<input id="checkbox" type="checkbox" onclick="changeFont(this,'{!$Component.thePanel}');"/>
```

Call the JavaScript functions.

YOUR TURN

19-3: CREATE A CUSTOM BUTTON THAT USES JAVASCRIPT (OPTIONAL)

salesforce

10 Minutes

Goal:

Modify the TechnicianStatus Visualforce page to include a custom button.

Tasks:

1. Upload the JavaScript as a static resource.
2. Open the TechnicianStatus Visualforce page and complete the TODOs.
3. Test the button.

The TechnicianStatus page should include a Cancel button to return the user to the Contact detail page. The only way to do so currently is to click the browser back button.

Jason Beck

Beginning Developer

KEY TAKEAWAYS

- Only use Visualforce if standard behavior needs to be overridden or the look and feel of the page needs to change.
- A Visualforce page and its associated custom controllers and controller extensions are subject to limits.
- Use the `transient` keyword to reduce the view state, and hence the payload.
- Always filter your queries.
- Always try to use “with sharing” for classes.
- Use Lightning Components to develop pages for mobile devices.

MODULE 20: TESTING VISUALFORCE CONTROLLERS

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE

You must write tests for the controller extension used by our updated technician status page.

To accomplish this, you need to:

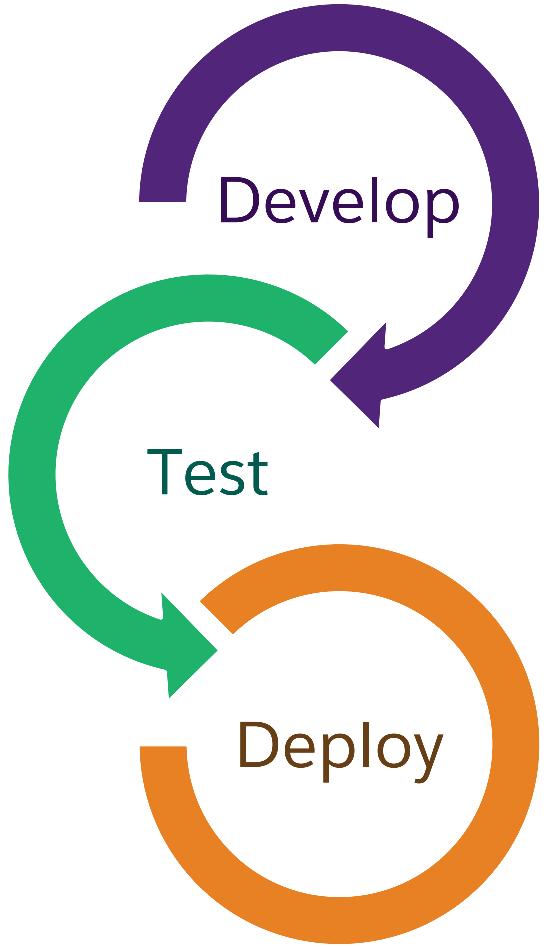
- Understand how a Visualforce controller interacts with the view.
- Understand testing controller constructors.
- Understand testing action methods, getters, setters, and properties.

- **Understanding Visualforce Controller Testing**
- Testing a Visualforce Controller Constructor
- Testing Action Methods
- Testing Getters, Setters, and Properties

VISUALFORCE CONTROLLERS ARE LIKE OTHER APEX CLASSES

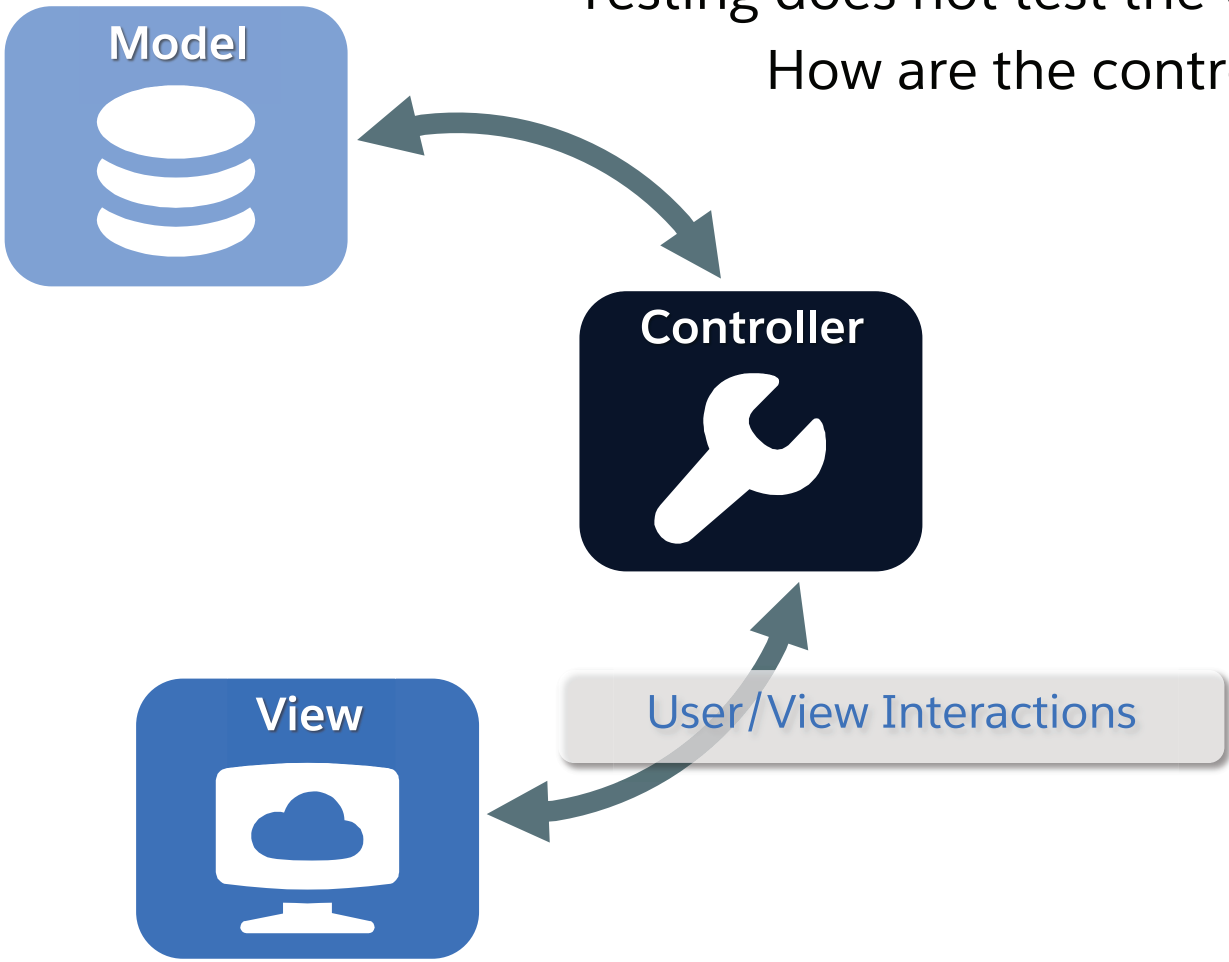
Just like the Apex triggers and classes we tested earlier, we need to ensure:

- 75% of Apex code must be executed successfully by test methods.
- Every Apex test method must execute without throwing any uncaught exceptions or exceeding governors.

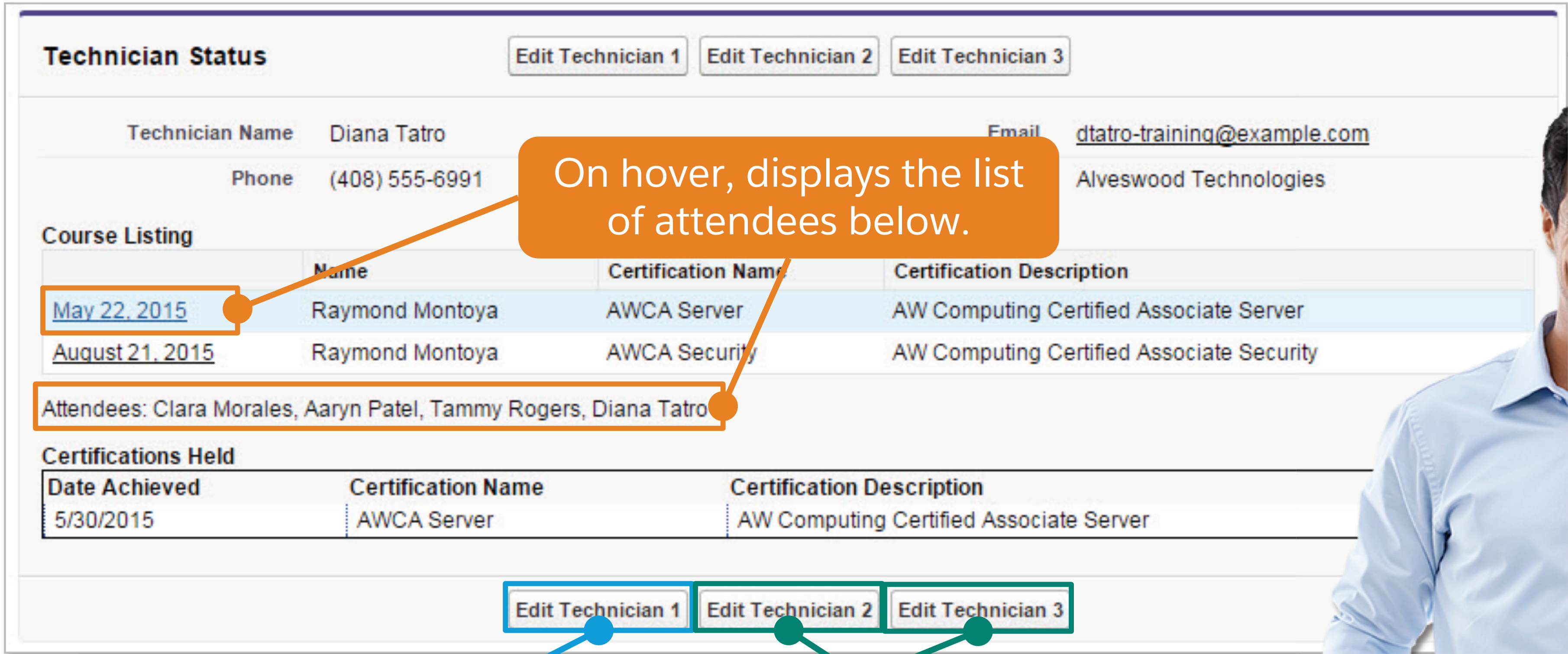


WHAT'S DIFFERENT ABOUT TESTING CONTROLLERS?

Testing does not test the view; it only tests the controller.
How are the controller methods invoked in a test, absent the view?



UNDERSTANDING THE TECHNICIAN STATUS PAGE AND EXTENSION



On hover, displays the list of attendees below.

Calls the standard edit action on the technician record.

Each calls a different custom method to redirect to the edit page.

MODULE AGENDA

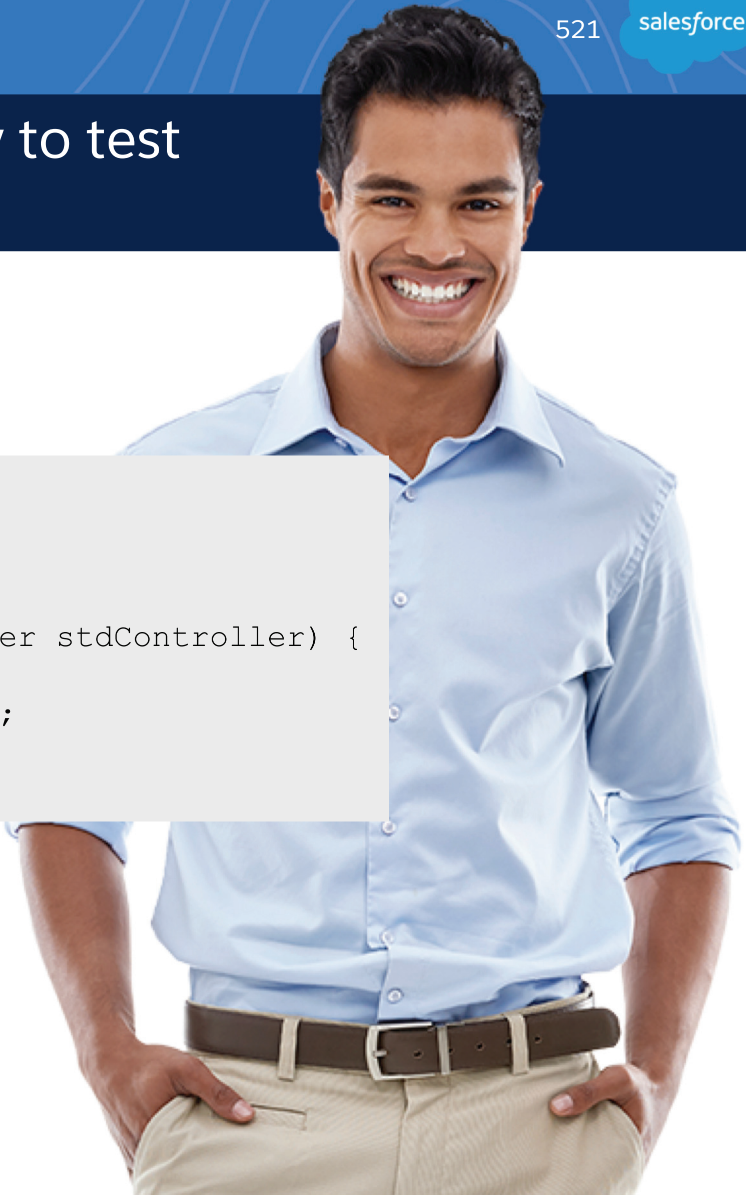
MODULE 20: TESTING VISUALFORCE CONTROLLERS

- Understanding Visualforce Controller Testing
- **Testing a Visualforce Controller Constructor**
- Testing Action Methods
- Testing Getters, Setters, and Properties

Story

TESTING THE CONSTRUCTOR

We'll start by writing the test methods necessary to test the controller extension's constructor.



```
public class TechnicianStatus_CX {
    private final Contact contact;
    private ApexPages.StandardController sController;

    public TechnicianStatus_CX(ApexPages.StandardController stdController) {
        sController = stdController;
        this.contact = (Contact)stdController.getRecord();
    }
}
```

TESTING CONTROLLER CONSTRUCTORS: SETTING currentPage()

Constructors vary from controller to controller and extension to extension. You should test everything the controller or extension is doing:

```
public class MyController {
    private PageReference sourcePage;
    public MyController() {
        this.sourcePage = ApexPages.currentPage();
    }
}
```

The constructor sets the current page as the source page.

```
// Test code
PageReference testPage = Page.MyPage;
Test.setCurrentPage(testPage);
MyController testCtrl = new MyController();
```

Use `Test.setCurrentPage()` to assign a mock Visualforce page for `ApexPages.currentPage()`.

TESTING CONTROLLER CONSTRUCTORS: SETTING PAGE PARAMETERS

Populate HTTP query string parameters so that they can be found by the controller.

```
public class MyController {
    private Id curId;
    public MyController() {
        this.curId = ApexPages.currentPage().getParameters().get('id');
    }
}
```

The constructor stores the current Id parameter as `curId` to use it later.

```
// Test code
Account testAccount = new Account();
… // populate testAccount with good mock data
insert testAccount;
PageReference testPage = Page.MyPage;
testPage.getParameters().put('id', testAccount.id);
Test.setCurrentPage(testPage);
MyController testCtrl = new MyController();
```

Create a test account and use its Id to populate the Id parameter on your test page so you will have good mock data available.

TESTING CONTROLLER EXTENSIONS: MOCKING THE CONTROLLER

If your controller extension makes use of methods or data from the controller it extends, you must mock the controller.

Instantiate the controller your extension is extending using your test record.

```
// Test code
Account testAccount = new Account();
… // populate with test data
ApexPages.StandardController stdCtrl
             = new ApexPages.StandardController(testAccount);
MyControllerExt testExt = new MyControllerExt(stdCtrl);
… // call setters, action, and getter methods
```

Then pass that controller to your extension.

YOUR TURN

20-1: WRITE THE TEST METHODS FOR THE CONSTRUCTOR

salesforce

20 minutes

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.

MODULE AGENDA

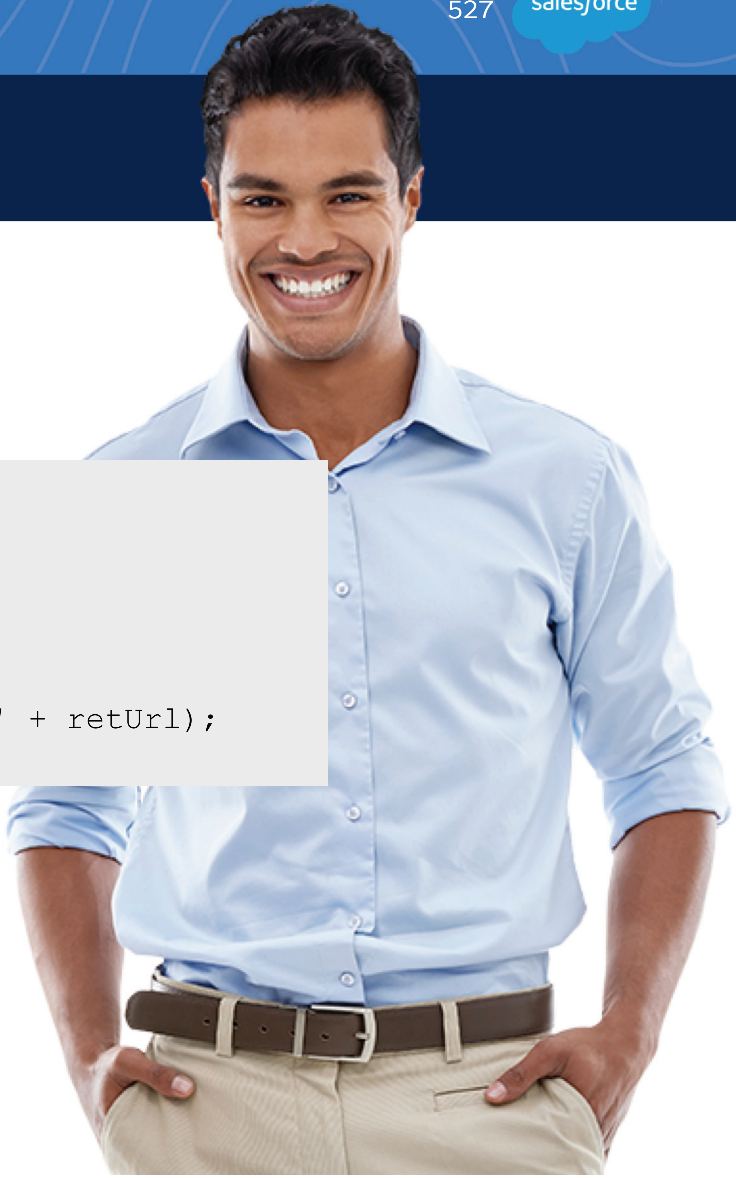
MODULE 20: TESTING VISUALFORCE CONTROLLERS

- Understanding Visualforce Controller Testing
- Testing a Visualforce Controller Constructor
- **Testing Action Methods**
- Testing Getters, Setters, and Properties

Story

TESTING ACTION METHODS

Now you need to test the action methods in this controller extension.



```
public PageReference editContact2() {
    return sController.Edit();
}

public PageReference editContact3() {
    String retUrl = '/apex/TechnicianStatus?id=' + contact.Id;
    return new PageReference('/' + contact.Id + '/e' + '?retURL=' + retUrl);
}
```

COMPONENTS OF AN ACTION METHOD

Many action methods do two things (and you should test them both)!

1

Do something:
- Perform logic
- Perform DML
- Perform a calculation

2

Navigate somewhere:
- A Visualforce page
- A standard page
- The current page

TESTING AN ACTION METHOD

Mimic Visualforce page interactions by explicitly calling methods.

- Button clicks (e.g., `{!save}`) call action methods.
- Verify what the action "does."

```
// Test code
MyController testCtrl = new MyController();
 testCtrl.setName('John');
 testCtrl.setAccount(new Account());
 PageReference successPage = testCtrl.save();
```

VERIFY WHERE THE ACTION "GOES"

You can verify navigation by comparing PageReference URLs:

```
public class MyController {
    …
    public PageReference save() {
        … // do something, check results
        if (success) return Page.Success;
        else return Page.Failure;
    }
}
```

```
// Test Code
Account testAccount = new Account();
// populate testAccount with good mock data
MyController testCtrl = new MyController();
System.assertEquals(testCtrl.save().getURL(), Page.Success.getURL());
… // populate testAccount with bad mock data
System.assertEquals(testCtrl.save().getURL(), Page.Failure.getURL());
```

YOUR TURN

20-2: WRITE UNIT TESTS FOR ACTION METHODS

salesforce

20 minutes

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.

MODULE AGENDA

MODULE 20: TESTING VISUALFORCE CONTROLLERS

- Understanding Visualforce Controller Testing
- Testing a Visualforce Controller Constructor
- Testing Action Methods
- **Testing Getters, Setters, and Properties**

TESTING GETTER AND SETTER METHODS

Form fields (e.g., `{!obj.field}`) call getters and setters.

```
public with sharing class MySearchController {
  String searchText;
  public String getSearchText() {
      return searchText;
  }
  public void setSearchText(String s) {
      searchText = s;
  }
}
```

```
String testString = 'Test String';
Test.startTest();
controller.setSearchText(testString);
Test.stopTest();
System.assertEquals(controller.getSearchText(), testString);
```

Use the setter method to set the value equal to the testString variable.

Compare the testString variable to the results of the getter method.

TESTING GETTERS: A MORE COMPLEX EXAMPLE

Depending on your code, you may wish to check whether your method returns:

- Specific results.
- Results containing a specific value.
- A valid result (`!null`).

```
// List of Course Attendees associated with the Course Delivery Date that the user hovers over
public List<Course_Attendee__c> getAttendeeList() {
    String cdId = apexPages.currentPage().getParameters().get('courseDeliveryId');
    return [SELECT Student__r.Name FROM Course_Attendee__c WHERE Course_Delivery__c = :cdId];
}
```

```
Test.startTest();
// Mimics the user getting the attendeeList for a specific Course Delivery
List<Course_Attendee__c> attendeeList = controllerExt.getAttendeeList();
Test.stopTest();
System.assert(attendeeList.size() > 0);
```

TESTING PROPERTIES

- Calling a setter method in a test class

```
controller.setMethodName(value);
```

- Setting a property in a test class

```
controller.propertyName = value;
```

YOUR TURN

20-3: WRITE UNIT TESTS FOR GETTERS AND SETTERS

20 minutes

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.

KEY TAKEAWAYS

- Just like the Apex triggers and classes, you must ensure that Visualforce controllers meet code coverage requirements.
- When testing Visualforce controllers, you must mimic the behaviors of the user and the view.
- The major categories of Visualforce controller testing are:
  - The constructor
  - Action methods
  - Getters, setters, and properties

Thank you for attending!

Your satisfaction is very important to us. Click on the **Course Survey** link located on the Home Page of your training org to give us your feedback.

Get certified! Go to www.webassessor.com/salesforce to register.

Take the next step! Register for the next course! Go to www.salesforce.com/training.

TRAINING AND CERTIFICATION RESOURCES

Training:
www.salesforce.com/training

Salesforce Certification:
certification.salesforce.com

Follow us on Twitter:
@Trailhead

Trailhead:
trailhead.salesforce.com