



## *Student Guide*

# Programmatic Development Using Apex and Visualforce

DEV450-S18-V1-SG





## 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	<b>Trigger Essentials</b> <ul style="list-style-type: none"><li>▪ Automating Logic</li><li>▪ Defining a Trigger<ul style="list-style-type: none"><li><i>Your Turn 8-1 (5 min): Define a Trigger</i></li></ul></li><li>▪ Defining Trigger Logic<ul style="list-style-type: none"><li><i>Join Me 8-2 (15 min): Define the Trigger's Business Logic</i></li></ul></li></ul>
70 minutes	<b>Apex Class Essentials</b> <ul style="list-style-type: none"><li>▪ Using an Apex Class</li><li>▪ Defining an Apex Class<ul style="list-style-type: none"><li><i>Your Turn 9-1 (15 min): Define an Apex Class</i></li></ul></li><li>▪ Determining Data Access for an Apex Class</li></ul>
80 minutes	<b>The Save Order of Execution and Apex Transactions</b> <ul style="list-style-type: none"><li>▪ Exploring the Save Order of Execution<ul style="list-style-type: none"><li><i>Watch Me 10-1 (10 min): Explore the Implicit Firing of Triggers</i></li><li><i>Your Turn 10-2 (10 min): View the Events that Occur During a Rollback</i></li></ul></li><li>▪ Working with Apex Transactions<ul style="list-style-type: none"><li><i>Watch Me 10-3 (10 min): See the Save Order of Execution in Action</i></li></ul></li></ul>
75 minutes	<b>Testing Essentials</b> <ul style="list-style-type: none"><li>▪ Describing Apex's Testing Framework</li><li>▪ Creating Test Data<ul style="list-style-type: none"><li><i>Your Turn 11-1 (10 min): Make Test Data Available to Test Methods</i></li></ul></li><li>▪ Writing and Running an Apex Test<ul style="list-style-type: none"><li><i>Your Turn 11-2 (10 min): Write and Run an Apex Test</i></li></ul></li></ul>
50 minutes	<b>Testing Strategies</b> <ul style="list-style-type: none"><li>▪ Understanding the Side Effects of Testing<ul style="list-style-type: none"><li><i>Your Turn 12-1 (10 min): Explore Code Coverage</i></li></ul></li><li>▪ Testing Using Best Practices</li></ul>
95 minutes	<b>Strategies for Designing Efficient Apex Solutions</b> <ul style="list-style-type: none"><li>▪ Working Efficiently with the Database<ul style="list-style-type: none"><li><i>Your Turn 13-1 (15 min): Refactor a Trigger to Avoid SOQL Limits</i></li><li><i>Your Turn 13-2 (15 min): Refactor a Trigger to Avoid DML Limits</i></li></ul></li></ul>



## AGENDA

### DEV450: Programmatic Development Using Apex and Visualforce

---

## Day Four

---

95 minutes	<b>Strategies for Designing Efficient Apex Solutions (cont.)</b> <ul style="list-style-type: none"><li>▪ Designing Triggers</li><li>▪ Designing Classes</li></ul>
120 minutes	<b>Trigger Design Strategy</b> <ul style="list-style-type: none"><li>▪ Analyzing the Problem</li><li>▪ Creating a Solution</li></ul> <p><i>Your Turn 14-1 (5 min): Create a Formula Field to Eliminate a Query</i></p> <p><i>Your Turn 14-2 (5 min): Create Fields for Counting Certifications Elements</i></p> <p><i>Your Turn 14-3 (15 min): Create Collections to Filter the Query</i></p> <p><i>Your Turn 14-4 (20 min): Use a Map to Aggregate Results</i></p> <p><i>Your Turn 14-5 (15 min): Create Certification Held Records</i></p> <p><i>Your Turn 14-6 (10 min): Use a Workflow to Avoid Creation of Duplicate Records (optional)</i></p>
55 minutes	<b>Creating Visualforce Pages</b> <ul style="list-style-type: none"><li>▪ Understanding Visualforce</li><li>▪ Creating a Visualforce Page</li><li>▪ Displaying Record Data and Launching a Visualforce Page</li></ul> <p><i>Join Me 15-1 (10 min): Create a Simple Visualforce Page</i></p> <p><i>Your Turn 15-2 (15 min): Display Data in a Visualforce Page</i></p>
50 minutes	<b>Exploring the View and Controller Layers</b> <ul style="list-style-type: none"><li>▪ Accessing Data on Related Records</li><li>▪ Exploring Visualforce Tags and Built-in Styling</li></ul> <p><i>Join Me 16-1 (15 min): Create a Simple Technician Status Page</i></p> <p><i>Your Turn 16-2 (10 min): Refine Your Page and Add Navigational Links</i></p>
140 minutes	<b>Working with Custom Controllers and Controller Extensions</b> <ul style="list-style-type: none"><li>▪ Referencing a Custom Controller</li><li>▪ Working with Getters, Setters, and Properties</li></ul> <p><i>Your Turn 17-1 (5 min): Reference a Controller Extension in a Visualforce Page</i></p> <p><i>Your Turn 17-2 (10 min): Create a Simple Read-Only Property</i></p>



## AGENDA

### DEV450: Programmatic Development Using Apex and Visualforce

---

## Day Five

---

140 minutes	<b>Working with Custom Controllers and Controller Extensions (cont.)</b> <ul style="list-style-type: none"><li>▪ Working with Action Methods<ul style="list-style-type: none"><li><i>Your Turn 17-3 (15 min): Writing a Read/Write Property in a Custom Controller</i></li><li><i>Your Turn 17-4 (10 min): Implementing the Search Button</i></li><li><i>Your Turn 17-5 (10 min): Redirecting to a Results Page</i></li></ul></li><li>▪ Handling Basic Errors<ul style="list-style-type: none"><li><i>Your Turn 17-6 (10 min): Handle Basic Save Errors in Your Method</i></li></ul></li></ul>
120 minutes	<b>Working with Lists Controllers and SOSL Queries</b> <ul style="list-style-type: none"><li>▪ Working with Standard List Controllers<ul style="list-style-type: none"><li><i>Join Me 18-1 (15 min): Create a Page to Display a List of Records</i></li></ul></li><li>▪ Writing a Simple SOSL Query<ul style="list-style-type: none"><li><i>Your Turn 18-2 (15 min): Integrate SOSL Search in a Visualforce Page</i></li></ul></li><li>▪ Creating a Custom List Controller<ul style="list-style-type: none"><li><i>Your Turn 18-3 (15 min): Create a Simple Search Page</i></li></ul></li></ul>
60 minutes	<b>Visualforce Development Considerations</b> <ul style="list-style-type: none"><li>▪ When to Use Visualforce<ul style="list-style-type: none"><li><i>Your Turn 19-1 (5 min): Determine Whether a Declarative Solution Exists</i></li></ul></li><li>▪ Visualforce and Governor Limits</li><li>▪ Security Considerations for Visualforce<ul style="list-style-type: none"><li><i>Your Turn 19-2 (10 min): Defend Against SOQL Injection</i></li></ul></li><li>▪ Developing Pages for Mobile Devices</li><li>▪ JavaScript in Visualforce<ul style="list-style-type: none"><li><i>Your Turn 19-3 (10 min): Create a Custom Button that Uses JavaScript</i></li></ul></li></ul>
100 minutes	<b>Testing Visualforce Controllers</b> <ul style="list-style-type: none"><li>▪ Understanding Visualforce Controller Testing<ul style="list-style-type: none"><li><i>Your Turn 20-1 (20 min): Write the Test Methods for the Constructor</i></li></ul></li><li>▪ Testing a Visualforce Controller Constructor</li><li>▪ Testing Action Methods<ul style="list-style-type: none"><li><i>Your Turn 20-2 (20 min): Write Unit Tests for Action Methods</i></li></ul></li><li>▪ Testing Getters, Setters, and Properties<ul style="list-style-type: none"><li><i>Your Turn 20-3 (20 min): Write Unit Tests for Getters and Setters</i></li></ul></li></ul>
15 minutes	<b>Wrap Up</b>



salesforce



# PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE



[www.salesforce.com/training](http://www.salesforce.com/training)  
© Copyright 2018 salesforce.com, inc.  
All rights reserved. Various trademarks  
held by their respective owners.



© Copyright 2000-2018 salesforce.com, inc. All rights reserved.  
Various trademarks held by their respective owners.

This document contains proprietary information of salesforce.com, inc., it is provided under a license agreement containing restrictions on use, duplication and disclosure and is also protected by copyright law. Permission is granted to customers of salesforce.com, inc. to use and **modify** this document for their internal business purposes only. Resale of this document or its contents is prohibited.

The information in this document is subject to change without notice. Should you find any problems or errors, please log a case from the Support link on the Salesforce home page. Salesforce.com, inc. does not warrant that this document is error-free.

# FORWARD LOOKING STATEMENTS

3

salesforce

## 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](http://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.



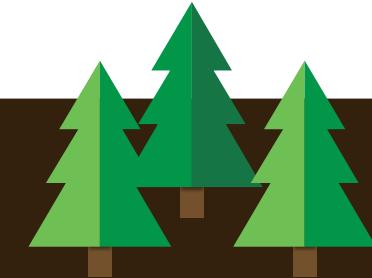
## Logistics

- 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

5

salesforce

[Module 1: Welcome to AW Computing](#)

[Module 2: Building Objects and Fields](#)

[Module 3: Working Effectively with Objects and Fields](#)

[Module 4: Programming with Apex](#)

[Module 5: Use SOQL to Query Your Org's Data](#)

[Module 6: Use SOQL to Query Parent-Child Relationships](#)

[Module 7: DML Essentials](#)

[Module 8: Trigger Essentials](#)

[Module 9: Apex Class Essentials](#)

[Module 10: The Save Order of Execution and Apex Transactions](#)

[Module 11: Testing Essentials](#)

[Module 12: Testing Strategies](#)

[Module 13: Strategies for Designing Efficient Apex Solutions](#)

[Module 14: Trigger Design Strategy](#)

[Module 15: Creating Visualforce Pages](#)

[Module 16: Exploring the View and Controller Layers of Visualforce](#)

[Module 17: Working with Custom Controllers and Controller Extensions](#)

[Module 18: Working with List Controllers and SOSL Queries](#)

[Module 19: Visualforce Development Considerations](#)

[Module 20: Testing Visualforce Controllers](#)

# MODULE 1: WELCOME TO AW COMPUTING

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE



Programme de révision :

<http://www.salesforceben.com/platform-developer-certification-guide-tips/>



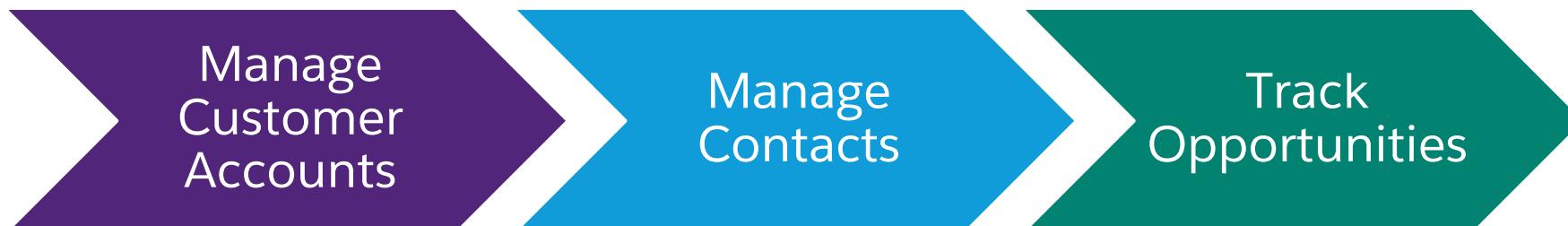
1 Market St.  
San Francisco, CA 94105  
United States  
  
+1.415.901.7901  
[www.aw-computing.com](http://www.aw-computing.com)

**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.”



Track account information.

Manage Contacts

Track key contacts for each account.

Track Opportunities

Move deals through the sales cycle.



Deals Won



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.”





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

Track vendor information.

### Manage Technician Contacts

Track technicians for each vendor.

### Manage Courses and Deliveries

Manage training delivery schedule and enroll technicians in courses.

### Manage Certifications and Attempts

Track where technicians are in the certification process.



Technicians Certified

# THE SALESFORCE USER INTERFACE

11

salesforce

The screenshot shows the Salesforce Accounts page with several UI elements highlighted:

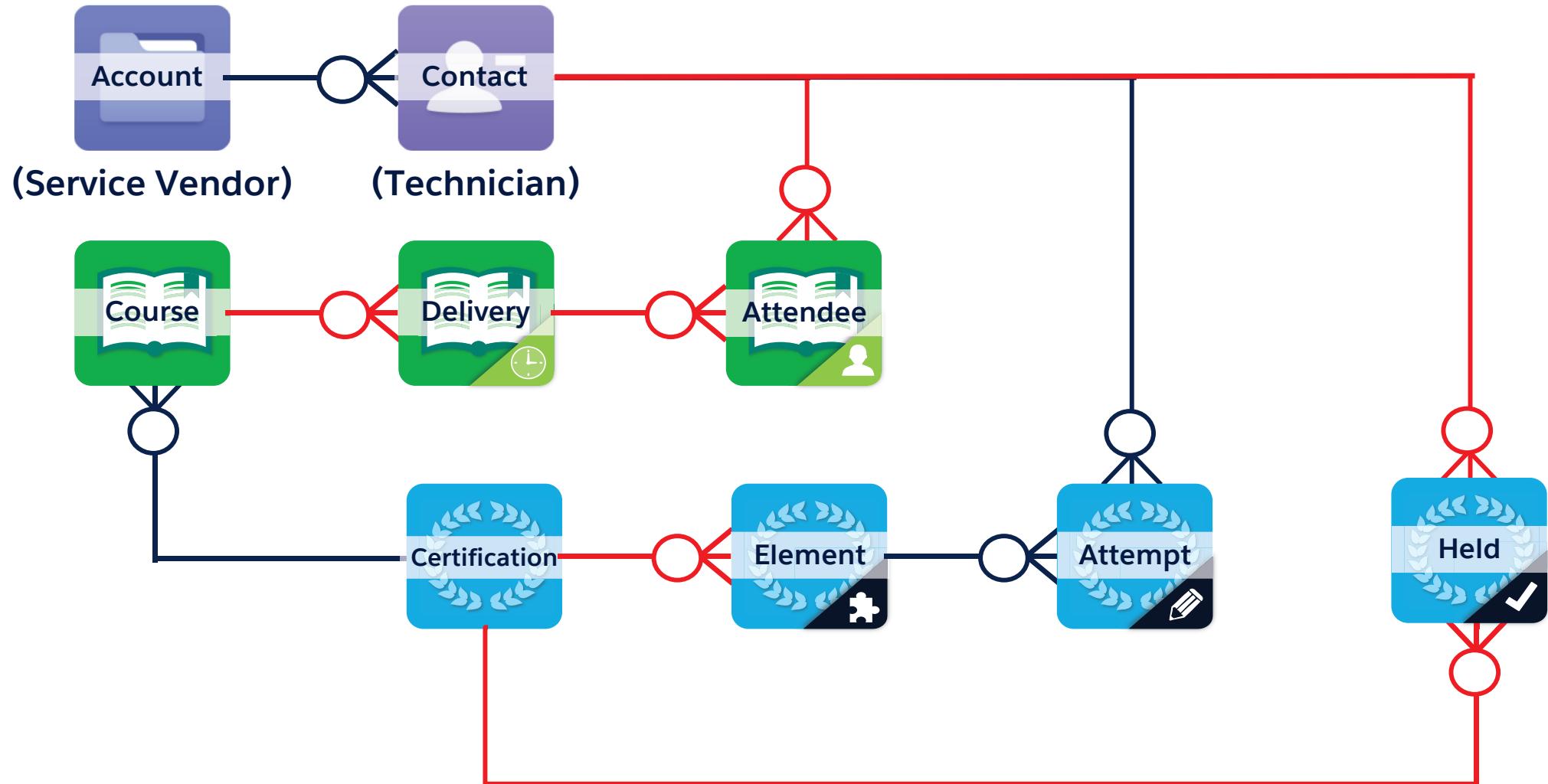
- Global Search**: A callout pointing to the search bar in the top navigation bar.
- "Your Name" Menu**: A callout pointing to the user profile dropdown in the top navigation bar.
- Force.com App Picker**: A callout pointing to the dropdown menu in the top navigation bar.
- Tabs**: A callout pointing to the Accounts tab in the top navigation bar.
- List Views**: A callout pointing to the dropdown menu for list views in the center of the page.
- Reports**: A callout pointing to the Reports section on the left.
- Tools**: A callout pointing to the Tools section on the right.

Key UI elements visible on the page:

- Top navigation bar with tabs: Home, Chatter, Profile, Groups, Files, Leads, Accounts, Contacts, Opportunities, Rec.
- Search bar: Search... Search
- User profile: Admin User
- Setup and Help & Training links
- Sales dropdown
- Create New... button
- Recent Items: No records to display
- Recycle Bin
- Quick Create: \*Account Name, Phone, Website, Save
- View dropdown: All Accounts, All Accounts (selected), My Accounts, New This Week, Recently Viewed Accounts
- Go! button
- Edit | Create New View
- New button
- Recently Viewed dropdown
- Reports section: Active Accounts, Accounts with last activity > 30 days, Account Owners, Contact Role Report, Account History Report, Partner Accounts
- Tools section: Import My Accounts & Contacts, Import My Organization's Accounts & Contacts, Mass Delete Accounts, Transfer Accounts, Merge Accounts, Sales Methodologies

# UNDERSTANDING THE CERTIFICATION APPLICATION

12



**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.

**Ryan Jackson**  
Lead Salesforce  
Programmatic  
Developer



**Cassie Evans**  
Salesforce  
Programmatic  
Developer



**Jason Beck**  
Salesforce  
Programmatic  
Developer





# DEFINING YOUR PROJECTS

15

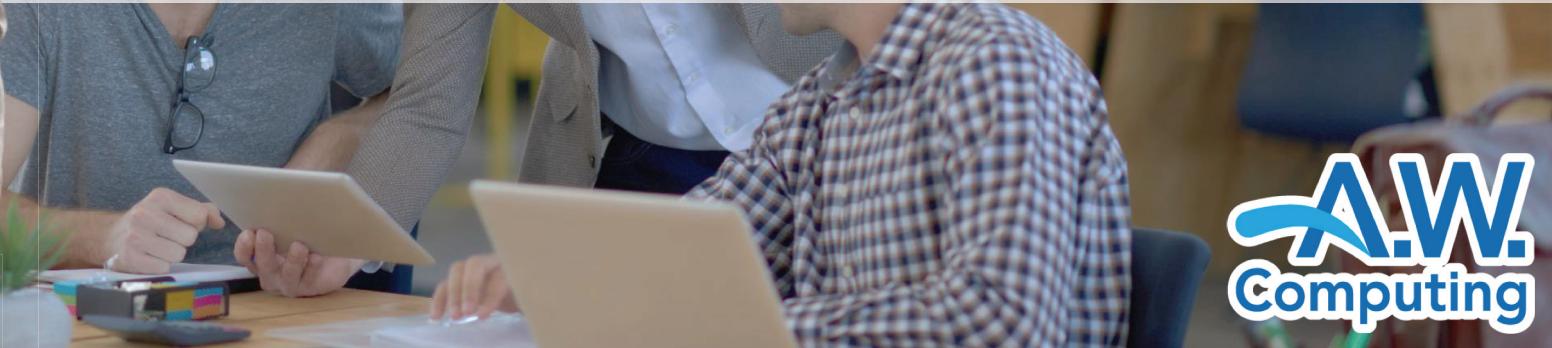
salesforce



**Ryan Jackson**  
Lead 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.

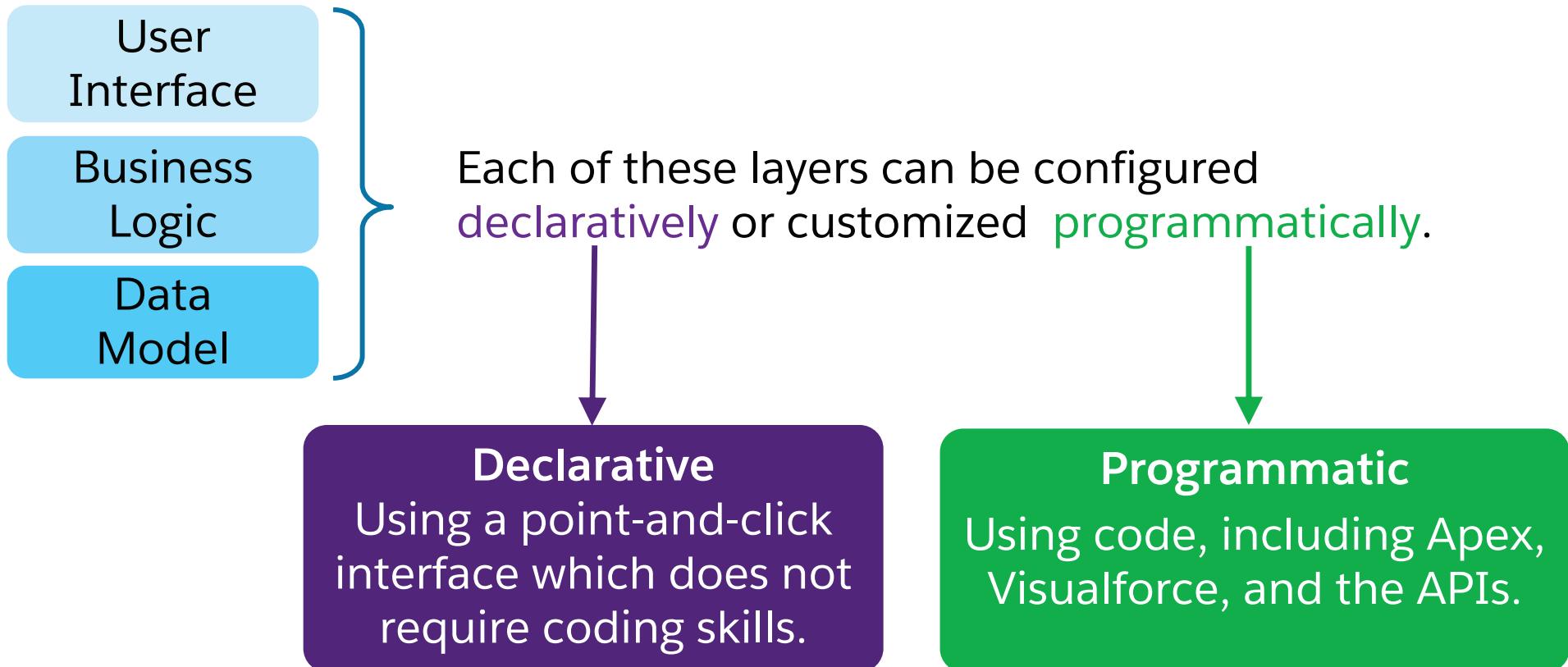


**A.W.**  
Computing

# HOW CAN SALESFORCE BE CUSTOMIZED?

16

salesforce



**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?

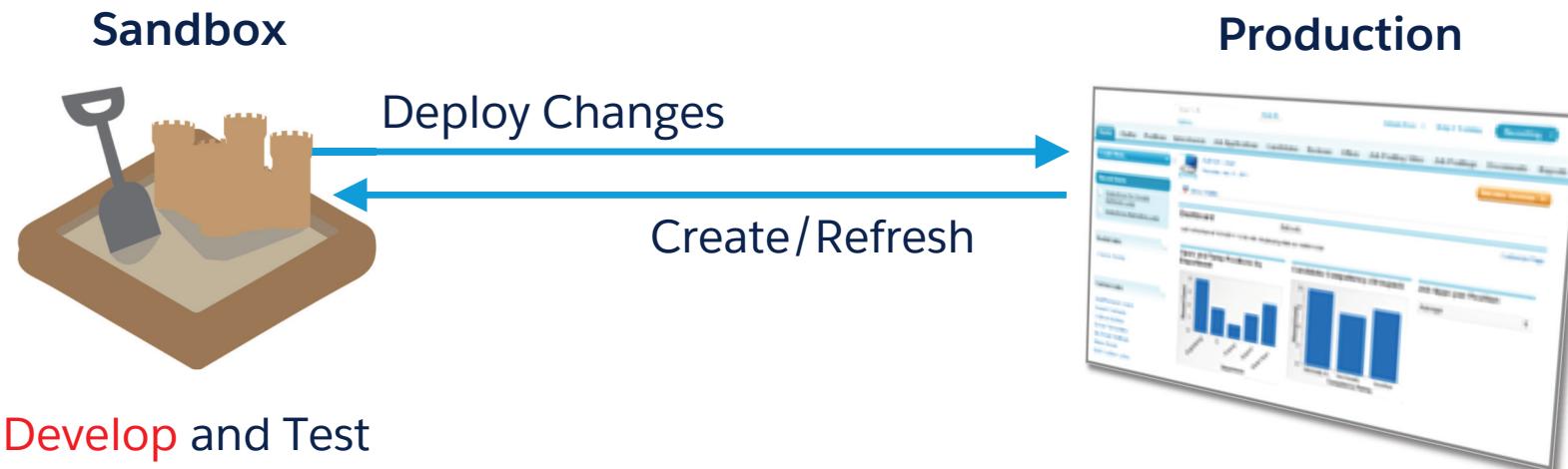
18

salesforce

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.

**Goal:**

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

**Task:**

Create a full sandbox named Dev.

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

## 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



Jason Beck  
Developer



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

24

salesforce

### 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?

25

salesforce

User Interface

View

Business Logic

Controller

Data Model

Model

# WHAT IS AN OBJECT ON THE FORCE.COM PLATFORM?

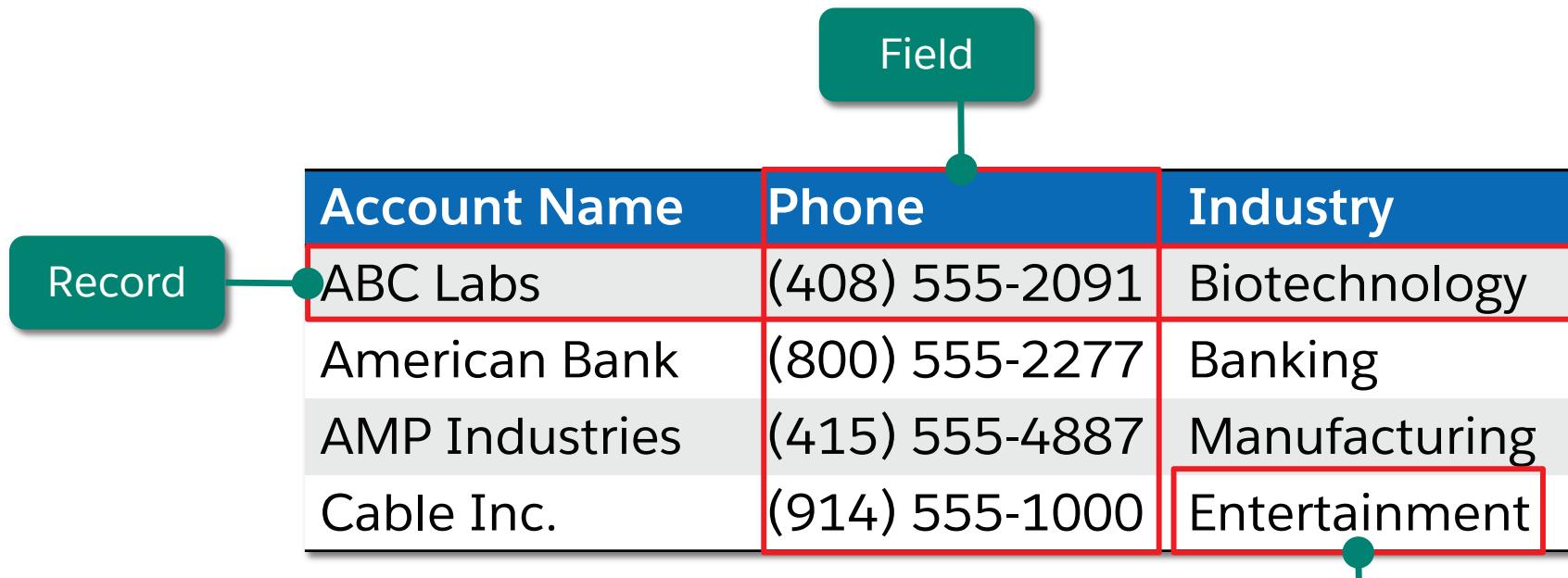
26

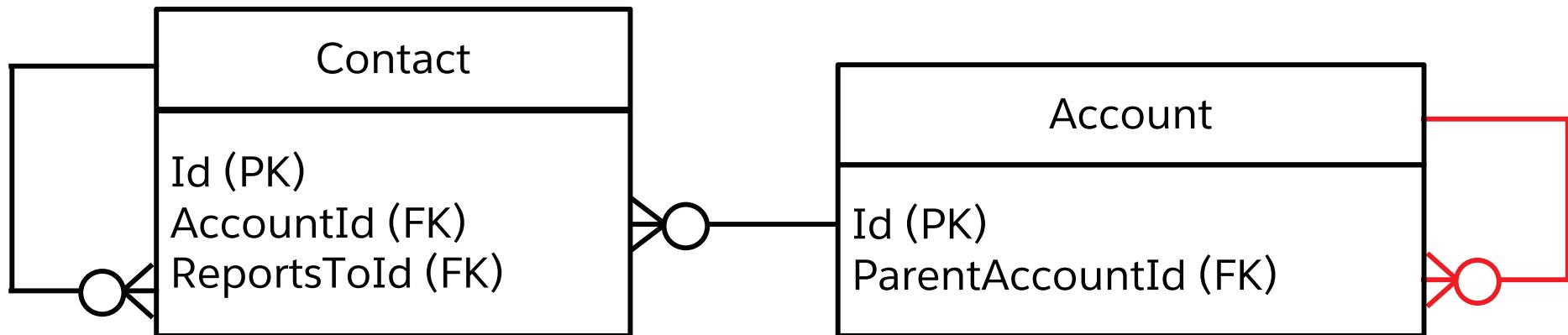
salesforce

DEFINITION:



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





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

28

salesforce

Schema Builder

Close Auto-Layout View Options ▾

Elements Objects

Select objects to display on the builder.

Select from All Objects ▾

Quick Find...

Select All | Clear All

Account

Activity

Asset

Campaign

CampaignMember

Case

Certification\_\_c

Certification\_Attempt\_\_c

Certification\_Element\_\_c

Certification\_Held\_\_c

Contact

Contract

Course\_\_c

Course\_Attendee\_\_c

Course\_Delivery\_\_c

Customer\_Story\_\_c

DuplicateRecordItem

DuplicateRecordSet

Event

Idea

Lead

Opportunity

Contact

Number\_of\_Certifications\_Held\_c  
Roll-Up Summary (COUNT Certification Held)

Account *Lookup(Account)*

AssistantName *Text(40)*

AssistantPhone *Phone*

Birthdate *Date*

CurrencyIsoCode *Picklist*

Owner *Lookup(User)*

RecordType *Record Type*

CreatedBy *Lookup(User)*

Jigsaw *Text(20)*

Department *Text(80)*

Description *Long Text Area(32000)*

DoNotCall *Checkbox*

Email *Email*

HasOptedOutOfEmail *Checkbox*

Fax *Fax*

HasOptedOutOfFax *Checkbox*

HomePhone *Phone*

LastModifiedBy *Lookup(User)*

LastCURequestDate *Date/Time*

LastCUUpdateDate *Date/Time*

LeadSource *Picklist*

Show More Fields

Account

CurrencyIsoCode *Picklist*

Name *Name*

AccountNumber *Text(40)*

Owner *Lookup(User)*

RecordType *Record Type*

Site *Text(80)*

AccountSource *Picklist*

AnnualRevenue *Currency(18, 0)*

BillingAddress *Address*

CreatedBy *Lookup(User)*

Jigsaw *Text(20)*

Description *Long Text Area(32000)*

NumberOfEmployees *Number(8, 0)*

Fax *Fax*

Industry *Picklist*

LastModifiedBy *Lookup(User)*

Ownership *Picklist*

Parent *Hierarchy*

Phone *Phone*

Rating *Picklist*

ShippingAddress *Address*

Sic *Text(20)*

Show More Fields

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 AGENDA

29

salesforce

## 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

30

salesforce

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



**Jason Beck**  
Developer

Custom Object  
**Customer Story**

[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	<input checked="" type="checkbox"/>
Object Name	Customer_Story	Track Activities	<input type="checkbox"/>
API Name	Customer_Story__c	Allow in Chatter Groups	<input type="checkbox"/>
		Allow Sharing	<input checked="" type="checkbox"/>
		Allow Bulk API Access	<input checked="" type="checkbox"/>
		Allow Streaming API Access	<input checked="" type="checkbox"/>
		Track Field History	<input type="checkbox"/>
		Deployment Status	Deployed
		Help Settings	Standard salesforce.com Help Window
Created By	<a href="#">Admin User</a> , 6/11/2015 2:23 PM	Modified By	<a href="#">Admin User</a> , 6/11/2015 2:23 PM

**Standard Fields** [Standard Fields Help](#)

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
<a href="#">Edit</a>	<a href="#">Created By</a>	CreatedBy	Lookup(User)		
<a href="#">Edit</a>	<a href="#">Currency</a>	CurrencyIsoCode	Picklist		
<a href="#">Edit</a>	<a href="#">Customer Story Name</a>	Name	Text(80)		<input checked="" type="checkbox"/>
<a href="#">Edit</a>	<a href="#">Last Modified By</a>	LastModifiedBy	Lookup(User)		
<a href="#">Edit</a>	<a href="#">Owner</a>	Owner	Lookup(User,Queue)		<input checked="" type="checkbox"/>

# WHAT IS A STANDARD OBJECT?

31

salesforce

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	<a href="#">Account Currency</a>	CurrencyIsoCode	Picklist
	<a href="#">Account Name</a>	Name	Name
Edit	<a href="#">Account Number</a>	AccountNumber	Text(40)
Edit	<a href="#">Account Owner</a>	Owner	Lookup(User)
Edit	<a href="#">Account Record Type</a>	RecordType	Record Type
Edit	<a href="#">Account Site</a>	Site	Text(80)
Replace   Edit	<a href="#">Account Source</a>	AccountSource	Picklist
Edit	<a href="#">Annual Revenue</a>	AnnualRevenue	Currency(18, 0)
	<a href="#">Billing Address</a>	BillingAddress	Address
	<a href="#">Created By</a>	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?

32

salesforce

DEFINITION:



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

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		<a href="#">Edit</a>	<a href="#">Delete</a>
Singular Label	Customer Story	Description	Used to track customer success stories.
Plural Label	Customer Stories	Enable Reports	<input checked="" type="checkbox"/>
Object Name	Customer_Story	Track Activities	<input type="checkbox"/>
API Name	Customer_Story_c	Allow in Chatter Groups	<input type="checkbox"/>

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

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	<input type="checkbox"/>
Create	<input type="checkbox"/>
Edit	<input type="checkbox"/>
Delete	<input type="checkbox"/>
View All	<input type="checkbox"/>
Modify All	<input type="checkbox"/>

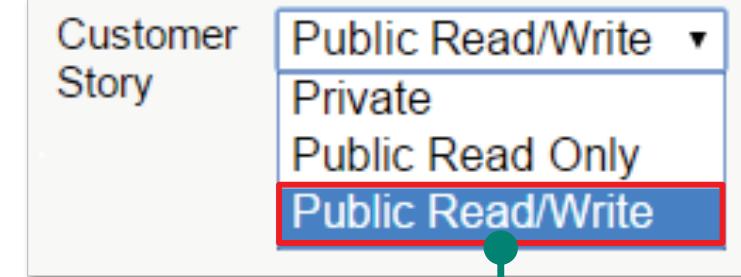
Object permissions determine whether users can view, create, edit, or delete records in an object.

**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:

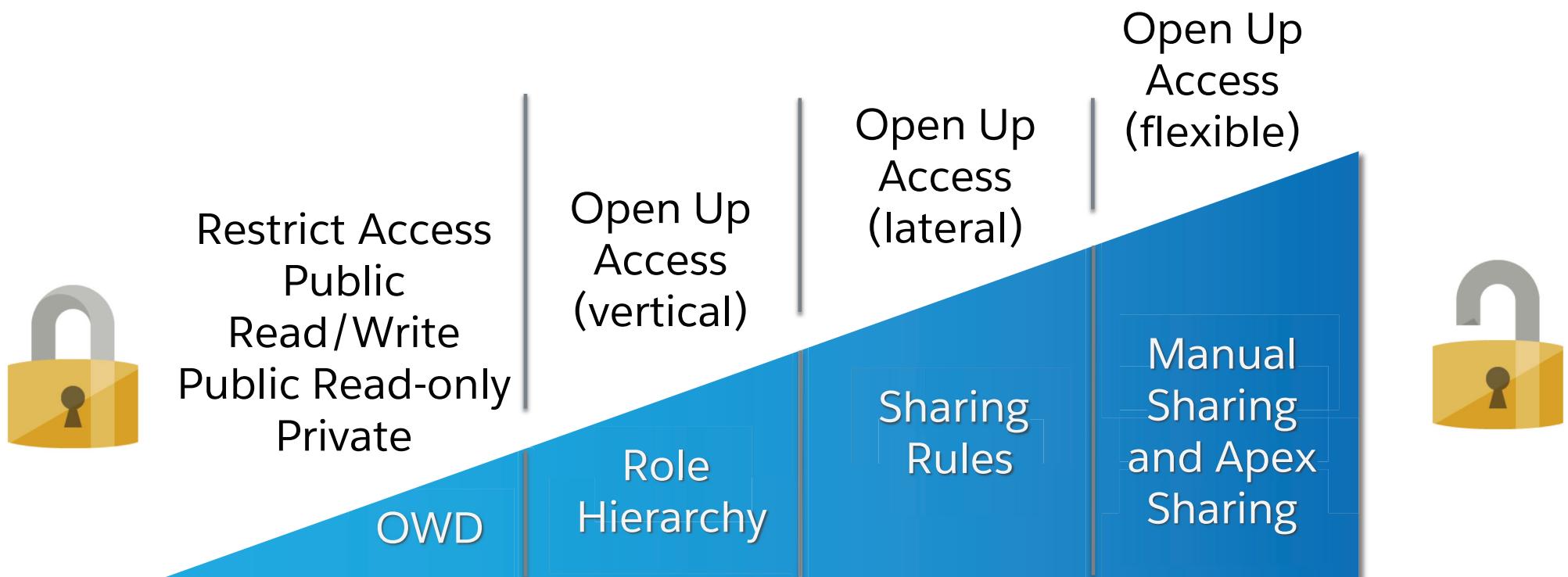
- View and edit the record.
- Transfer the record to a different owner.
- Delete the record.
- Share the record.

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



Default value for new custom objects.

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.



**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

37

salesforce

## MODULE 2: BUILDING OBJECTS AND FIELDS

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





# CAPTURING INFORMATION

38

salesforce

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
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">Installation Time (days)</a>	Installation_Time__c	Number(3, 0)			<a href="#">Admin User, 6/22/2015 5:54 AM</a>
<a href="#">Edit</a>   <a href="#">Del</a>   <a href="#">Replace</a>	<a href="#">Products</a>	Products__c	Picklist (Multi-Select)			<a href="#">Admin User, 6/22/2015 5:39 AM</a>
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">Story Description</a>	Story_Description__c	Text(255)			<a href="#">Admin User, 6/23/2015 5:12 AM</a>

# WHAT IS A STANDARD FIELD?

39

salesforce

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	<input type="checkbox"/>
Edit	Medium	<input checked="" type="checkbox"/>
Edit   Del	Low	<input type="checkbox"/>

- Format of auto number fields.

Display Format	<input type="text" value="00000000"/>
Example:	A-00000
What Is This?	<a href="#">What Is This?</a>
Next Number	<input type="text" value="1004"/>

Field Information	
Field Label	Contact Name
Data Type	Lookup(Contact)
Help Text	
Field Name	Contact
Child Relationship Name	<a href="#">Cases</a>
Lookup Filter	
Filter Criteria	Contact Name: Account Name ID EQUALS Case: Account Name ID
Filter Type	<b>Required.</b> 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	<input checked="" type="checkbox"/>



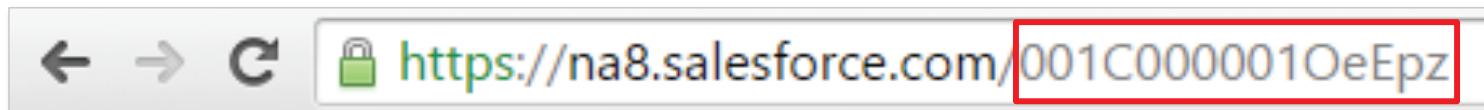
NOTE: You cannot delete standard fields, but you can hide them from users.

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

## Standard Fields

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

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



Id à 15 chiffres.

# WHAT IS A CUSTOM FIELD?

41

salesforce

DEFINITION:



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

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

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

## 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

43

salesforce

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".

Name used in code and by the API.

Field Information		Object Name	<a href="#">Customer Story</a>
Field Label	Installation Time (days)	Data Type	Number
Field Name	Installation_Time		
API Name	Installation_Time_c		
Description	Installation time		
Help Text	Number of days from purchase to installation complete		
Created By	<a href="#">Admin User</a> , 6/22/2015 5:54 AM	Modified By	<a href="#">Admin User</a> , 6/22/2015 5:54 AM

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

# OTHER FIELD ATTRIBUTES

44

salesforce

Description (administrative purposes).

Text displayed when users hover over the Info icon.

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

Field Information	
Field Label	Installation Time (days)
Field Name	Installation_Time
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
General Options	
Required	Require a value in order to save.
Unique	Enforce uniqueness across records.
External ID	Indicates that the field is a key from an external system (indexed).
Default Value	Value used to pre-populate field data.
Number Options	
Length	3
Decimal Places	0



NOTE:

The field attributes vary depending on the field type.

**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

46

salesforce

## MODULE 2: BUILDING OBJECTS AND FIELDS

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



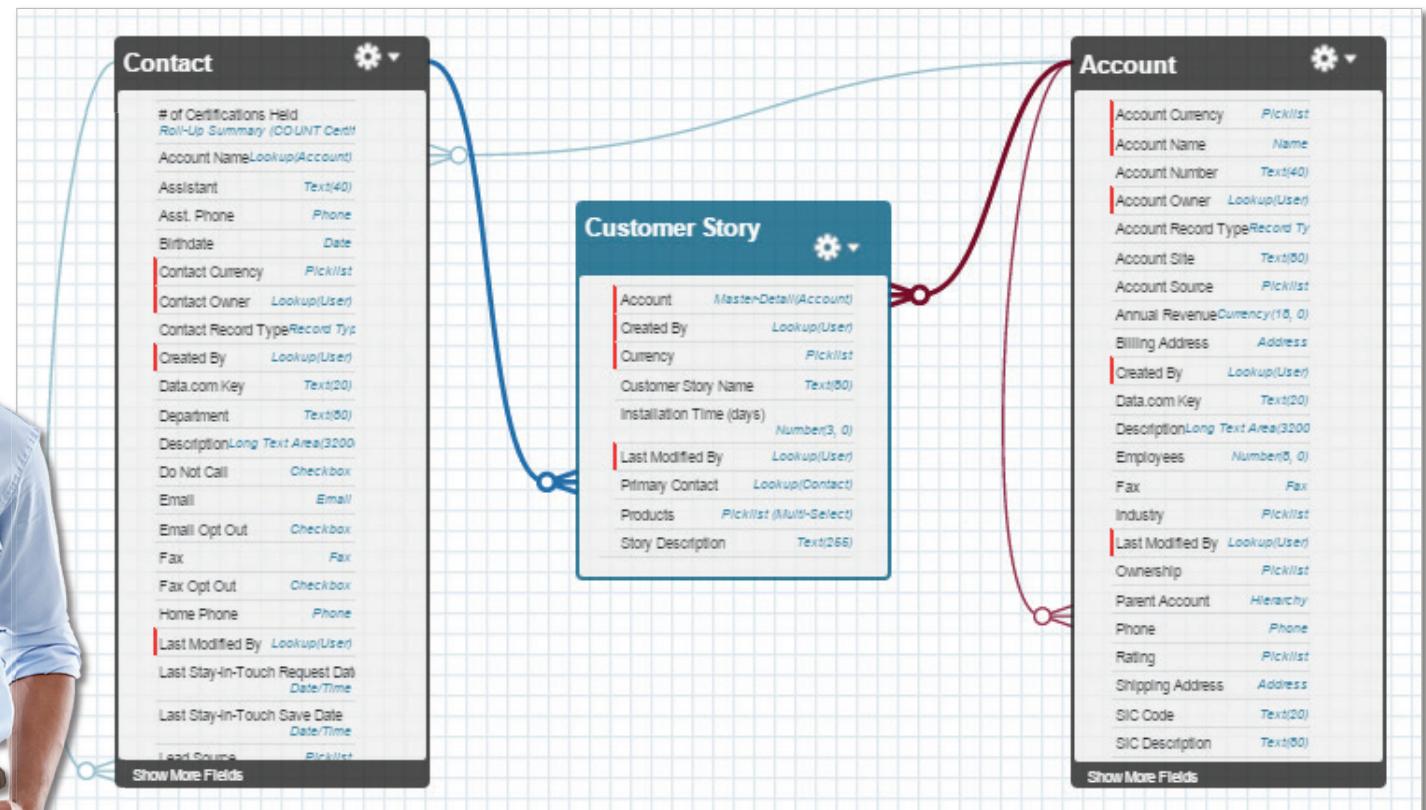


# RELATING OBJECTS TO ONE ANOTHER

47

salesforce

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



**Jason Beck**  
Developer

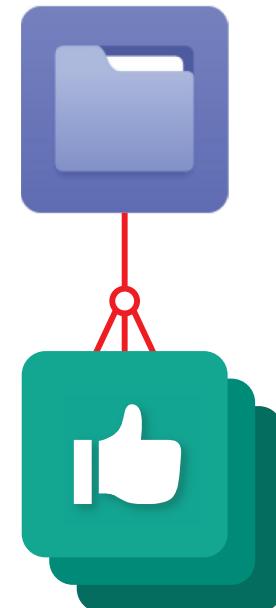
You can create a one-to-many relationship between two objects.

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

These are the types of relationship fields:

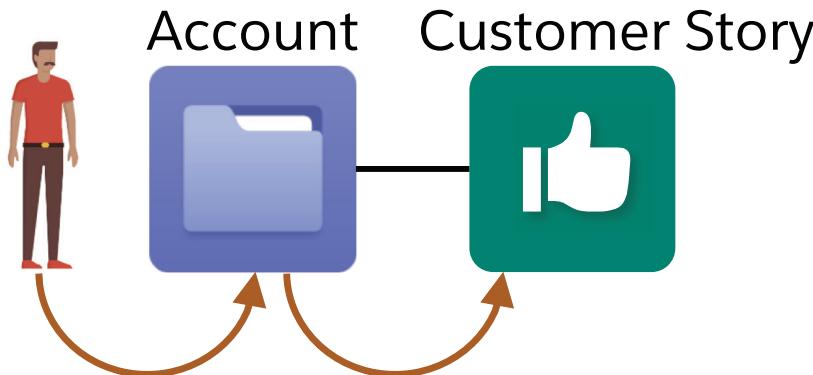
- Master-Detail
- Lookup

Account



Customer  
Story

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



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

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

Customer Story Detail		Edit	Delete	Clone
Customer Story Name	Test Account Customer Story	Account	Test Account	
Story Description	Major win at a new customer against top competitor.	Primary Contact	Kate Hanson	

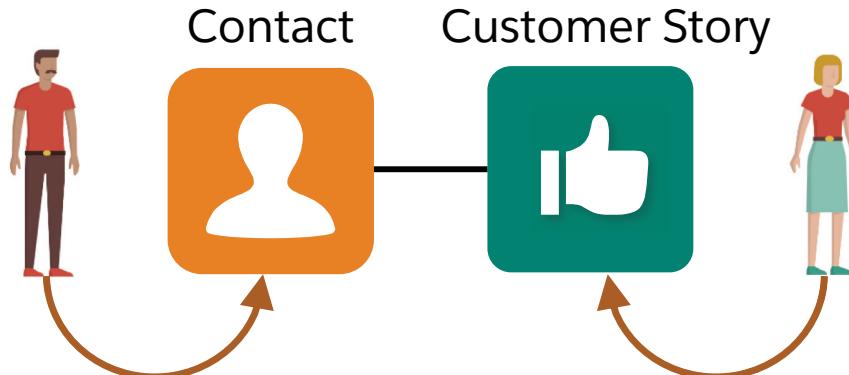
- You can add a lookup filter.
- You can choose whether or not the detail record can be reparented.



NOTE:

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

- The child record and parent record have independent sharing.



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

<b>Required</b>	<input type="checkbox"/> Always require a value in this field in order to save a record
<b>What to do if the lookup record is deleted?</b>	<input checked="" type="radio"/> Clear the value of this field. You can't choose this option if you make this field required. <input type="radio"/> Don't allow deletion of the lookup record that's part of a lookup relationship. <input type="radio"/> Delete this record also.

- You can add a lookup filter.

**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 Relationships

Parent field on child can be required or optional.

No impact on security and access.

If parent field is *required*, you cannot delete parent when referenced by child.

If parent field is *optional*, choose one of three delete behaviors.

Lookup field on page layout depends on required/optional choice.

## Master-Detail Relationships

Parent field on child is required.

Access to parent determines access to children.

Deleting parent automatically deletes children.

Lookup field on page layout is required.

# WHAT IS A JUNCTION OBJECT?

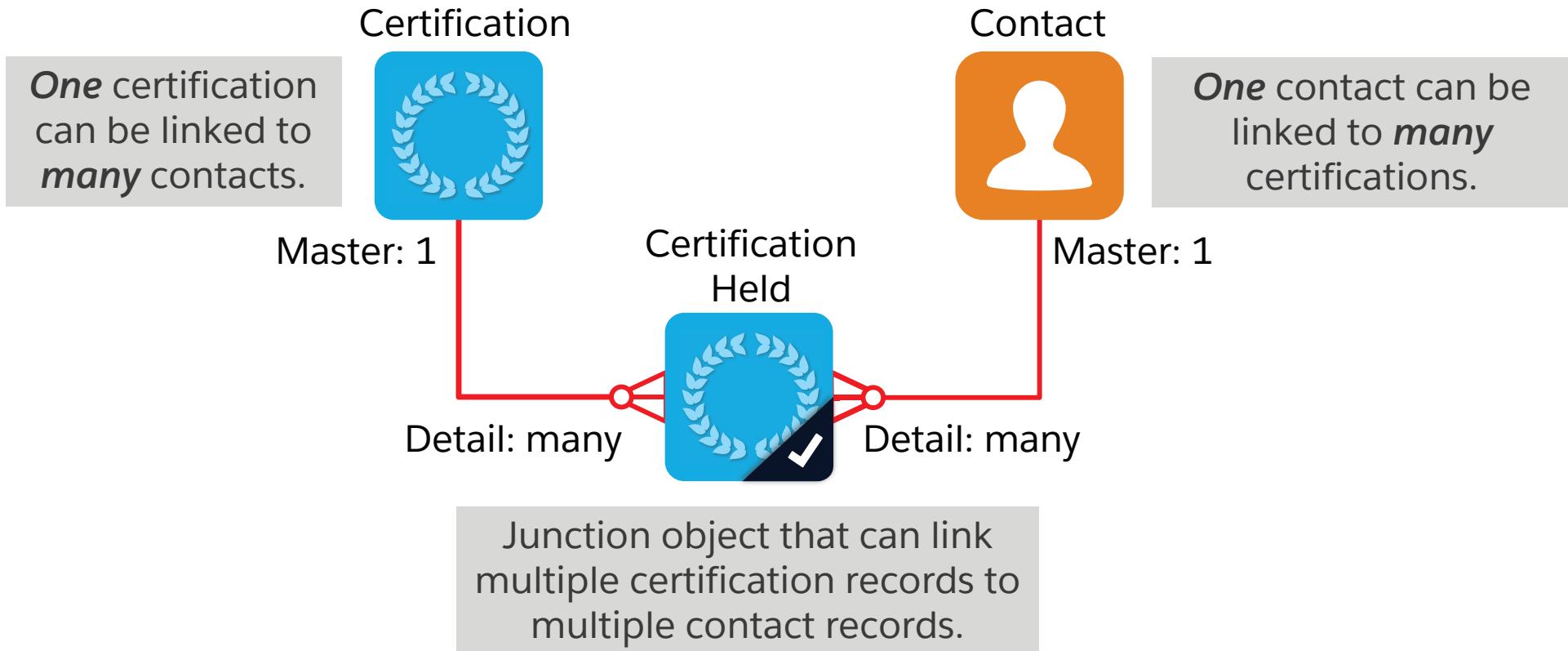
53

salesforce

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.



# METADATA VS. DATA

54

salesforce

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

Custom Fields & Relationships			
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

Data is the value of fields in the records.

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

Data

# WHAT DO YOU GET WITH AN OBJECT?

55

salesforce

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.



## KNOWLEDGE CHECK

57

salesforce

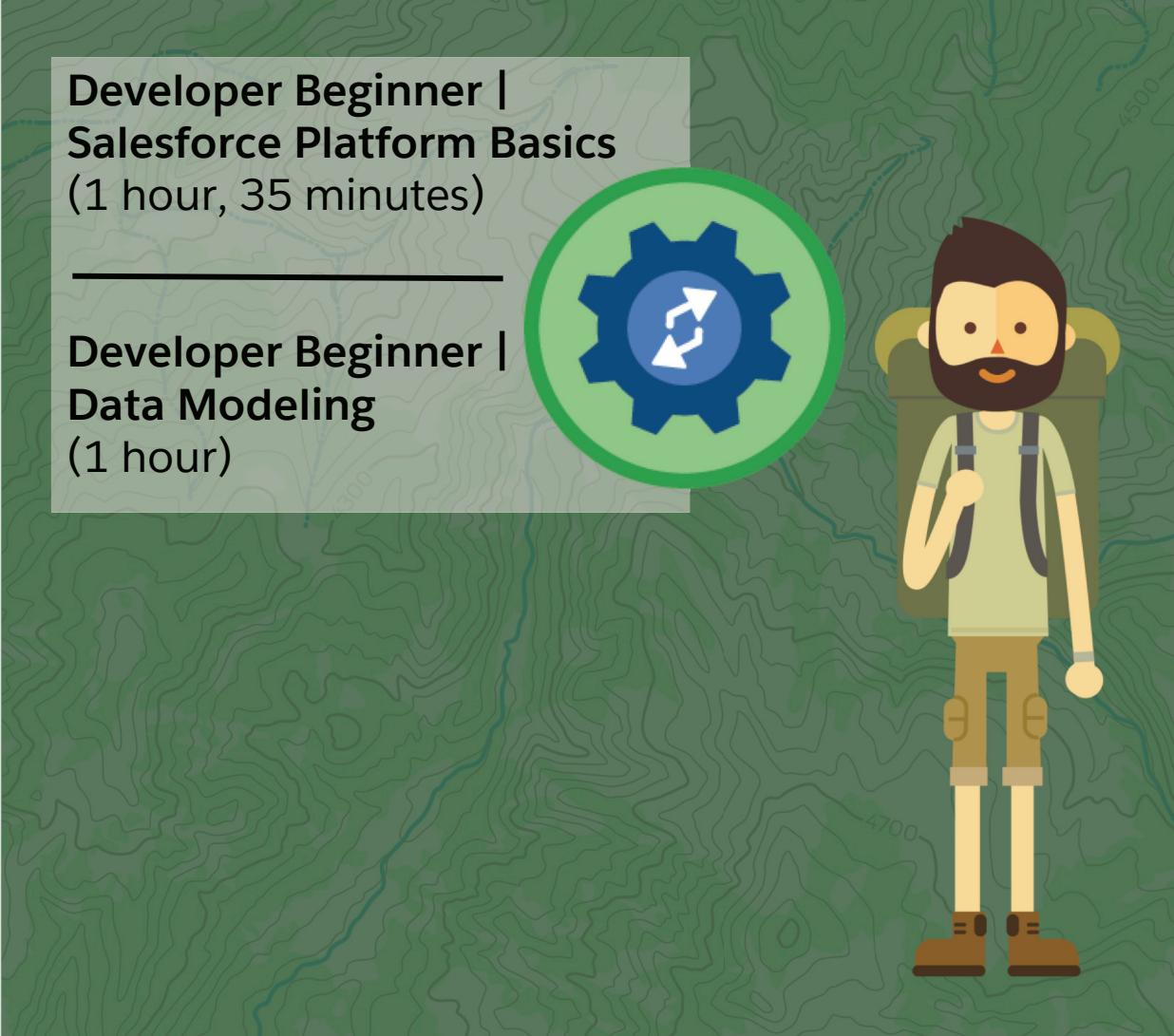
1. How can a developer customize an object on the Force.com platform? **Schema builder or object manager**
2. How can a developer reference a field programmatically? **Le nom API ( obj\_\_c)**
3. What is a consideration for creating a master-detail relationship between two objects? **Notion de visibilité,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



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.**

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 AGENDA

62

salesforce

### 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 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	<a href="#">Nicki Sanchez [Change]</a>
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	<a href="#">Sasha Vincent</a>
Course	<a href="#">[401] Data Recovery</a>	Start Date	5/13/2015
Region	NAMER	Status	Delivered

# WHAT IS A FORMULA FIELD?

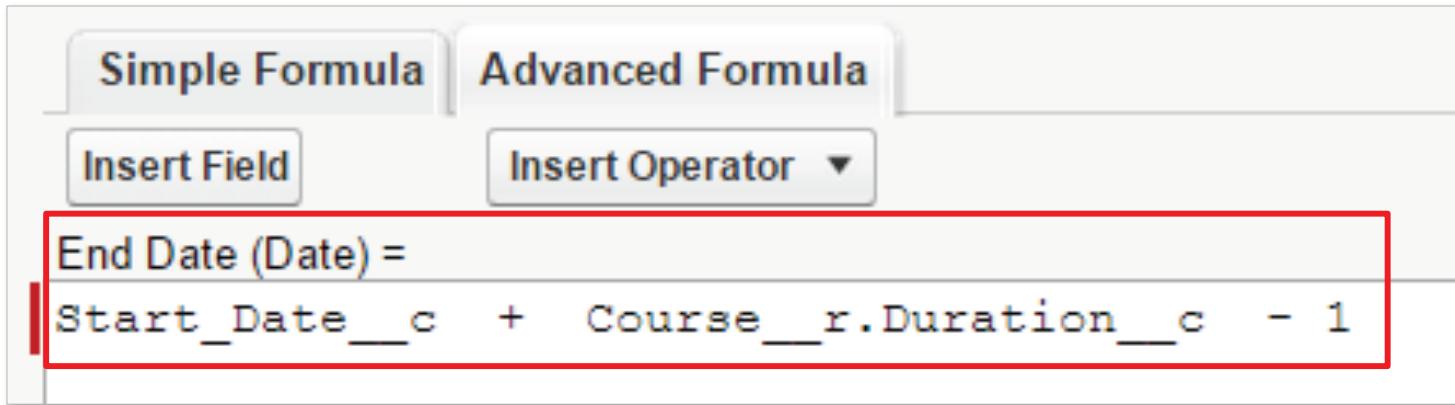
64

salesforce

DEFINITION:



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

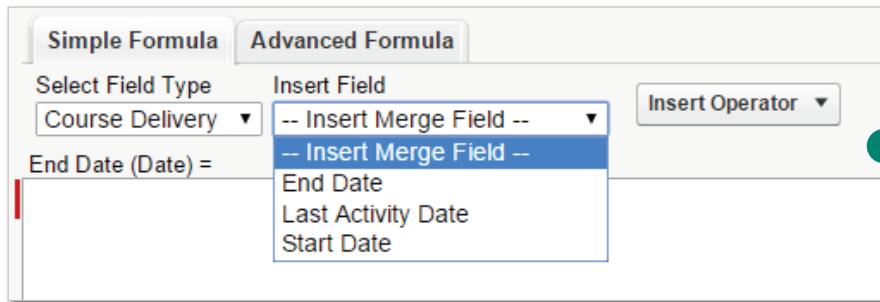


The image shows a screenshot of the Salesforce formula editor. At the top, there are two tabs: "Simple Formula" (which is selected) and "Advanced Formula". Below the tabs are two buttons: "Insert Field" and "Insert Operator". The formula field itself is labeled "End Date (Date) =". The formula is: `Start_Date__c + Course__r.Duration__c - 1`. The entire formula input area is highlighted with a red box.

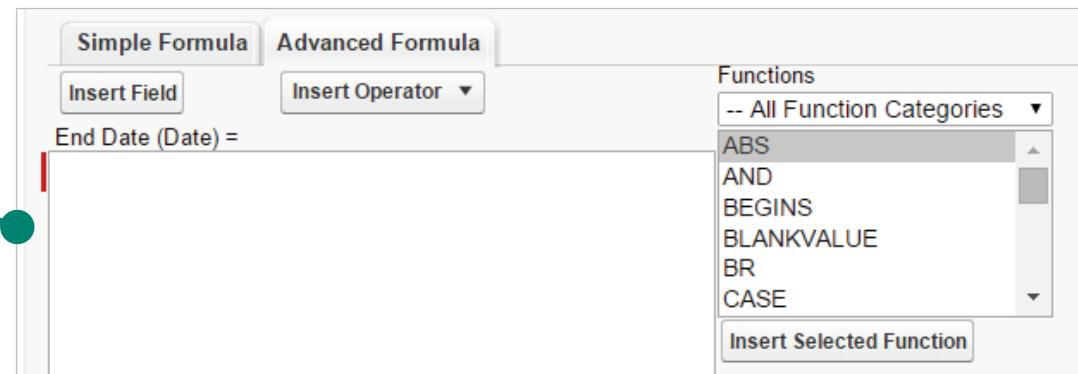
# DEFINING A FORMULA

65

salesforce



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.



NOTE:

Schema Builder only supports the Simple Formula editor.

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

**Insert Field**

Select a field, then click Insert. Labels followed by a ">" indicate that there are more fields available.

Course Delivery >  
\$Api >  
\$Organization >  
\$Profile >  
\$System >  
\$User >  
\$UserRole >

Course >  
Course Delivery Number  
Created By >  
Created By ID  
Created Date  
Currency  
End Date  
Instructor

# of Courses Delivered >  
Certification  
Certification >  
Course Name  
Created By >  
Created By ID  
Created Date  
Currency  
Duration

You have selected:  
**Course\_\_r.Duration\_\_c**  
Type: Number  
API Name: Course\_\_r.Duration\_\_c

Insert

Fields on Child      Fields on Parent

**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 AGENDA

68

salesforce

### 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

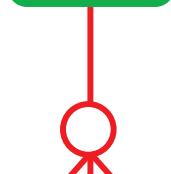


Cassie Evans

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
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">DELIVERY-00003</a>	APAC	Singapore, SG	2/4/2015	Cancelled	<a href="#">Kim Tran</a>
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">DELIVERY-00004</a>	EMEA	London, GB	1/28/2015	Delivered	<a href="#">Heidi Rosen</a>
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">DELIVERY-00005</a>	NAMER	Chicago, US	12/31/2014	Delivered	<a href="#">Sasha Vincent</a>
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">DELIVERY-00018</a>	APAC	Singapore, SG	4/22/2015	Delivered	<a href="#">Kim Tran</a>
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">DELIVERY-00024</a>	NAMER	Toronto, CA	6/10/2015	Cancelled	<a href="#">Patrick Hughes</a>
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">DELIVERY-00026</a>	EMEA	Berlin, DE	7/1/2015	Scheduled	<a href="#">Eugene Peters</a>

# WHAT IS A ROLL-UP SUMMARY FIELD?

70

salesforce

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.

Select Object to Summarize

Master Object: Course  
Summarized Object: Course Deliveries

■ = Required Information

Select Roll-Up Type

COUNT  
 SUM  
 MIN  
 MAX

Field to Aggregate: --None--

Filter Criteria

All records should be included in the calculation  
 Only records meeting certain criteria should be included in the calculation

Field	Operator	Value	AND
Status	equals	Cancelled	AND
--None--	--None--		AND
--None--	--None--		AND
--None--	--None--		AND

RESOURCE:



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

**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.

**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

73

salesforce

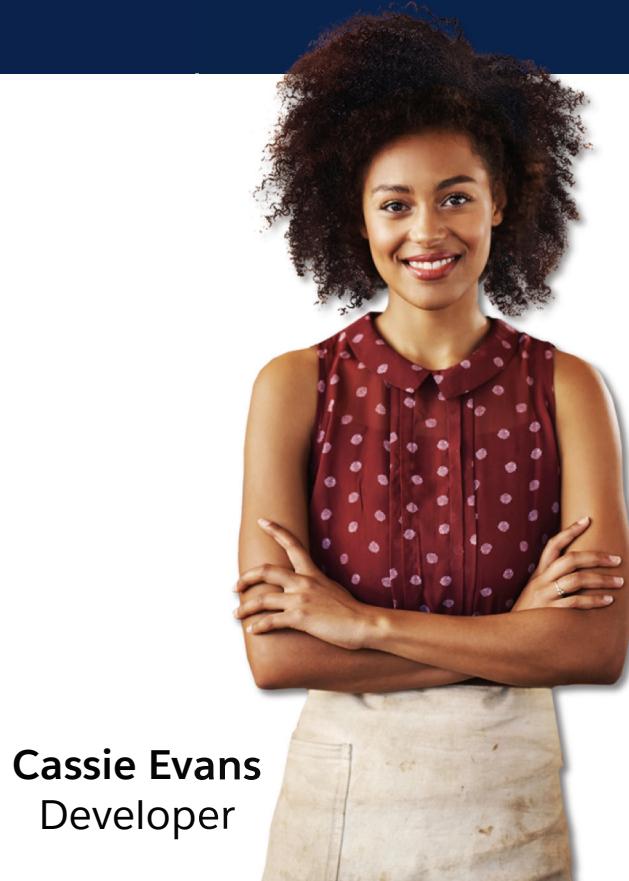
## MODULE 3: WORKING EFFECTIVELY WITH OBJECTS AND FIELDS

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



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?

75

salesforce

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.

The image shows two screenshots of Salesforce page layouts. The top screenshot is for an 'Account Detail' record. It displays fields for Account Owner (Matt Wilson), Account Name (ABC Labs), Parent Account, Account Record Type (Customer), and standard metadata like Website and Support Level. It also shows sections for 'Additional Information', 'Address Information', and 'System Information', along with a 'Custom Links' section containing links to Google Search and Google News. The bottom screenshot is for a 'Contacts' list. It shows a table with columns for Action, Contact Name, Title, Email, and Phone. The contacts listed are Arnold Adams, Bertha Boxer, Francis Buchner, and Sophie Kostos, each with their respective titles, email addresses, and phone numbers. The top screenshot is annotated with four callout boxes: 1) 'Add, remove, and move fields. Make fields read-only or required.' points to the Account Detail fields. 2) 'Create and move sections.' points to the section headers. 3) 'Display custom links.' points to the 'Custom Links' section. 4) 'Add, remove, and move related lists. Change which columns are displayed.' points to the Contacts list table.

# WHAT IS A RECORD TYPE?

76

salesforce

DEFINITION:



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



Customer

Contacts [4] Opportunities [2] Cases [2] Open Activities [0] Activity History [0] Notes & Attachments [0]

**Account Detail** Edit Delete Sharing

Account Owner	Matt Wilson [Change]	Phone	1-408-555-2091
Account Name	ABC Labs [View Hierarchy]	Website	<a href="http://www.ABCLabs.training">http://www.ABCLabs.training</a>
Parent Account		Support Level	Silver
Account Record Type	Customer [Change]		

▼ Additional Information

Industry	Biotechnology	Annual Revenue	USD 7,500,000.00
Employees	120		



Service Vendor

Contacts [5+] Open Activities [0] Activity History [0] Notes & Attachments [0]

**Account Detail** Edit Delete Sharing

Account Owner	Joseph Simmons [Change]	Phone	(416) 555-8000
Account Name	Mimico Systems [View Hierarchy]	Website	<a href="http://www.mimicosystems.training">http://www.mimicosystems.training</a>
Parent Account		Support Level	Standard Vendor
Account Record Type	Service Vendor [Change]	Status	Active

▼ Additional Information

Employees	220	Annual Revenue	USD 11,000,000.00
-----------	-----	----------------	-------------------

# SPECIFYING PICKLIST VALUES

77

salesforce

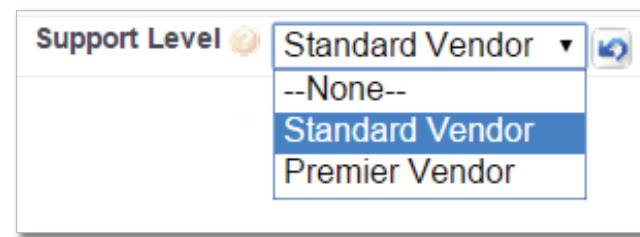
You can specify different picklist values for each record type.



Customer



Service  
Vendor



# ASSIGNING RECORD TYPES AND PAGE LAYOUTS

78

salesforce

Profile  
Training User

Find Settings... | Clone | Delete | Edit Properties

Profile Overview > Object Settings Accounts

Accounts | Save | Cancel

Tab Settings  
Default On

Account: Record Types and Page Layout Assignments

Record Types	Page Layout Assignment	Assigned Record Types	Default Record Type
--Master--	Account Layout	<input type="checkbox"/>	<input checked="" type="radio"/>
Customer	Customer Account Layout	<input type="checkbox"/>	<input checked="" type="radio"/>
Service Vendor	Service Vendor Account Layout Account Layout Customer Account Layout Service Vendor Layout	<input checked="" type="checkbox"/>	<input type="radio"/>

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.

**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

80

salesforce

### 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**



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?

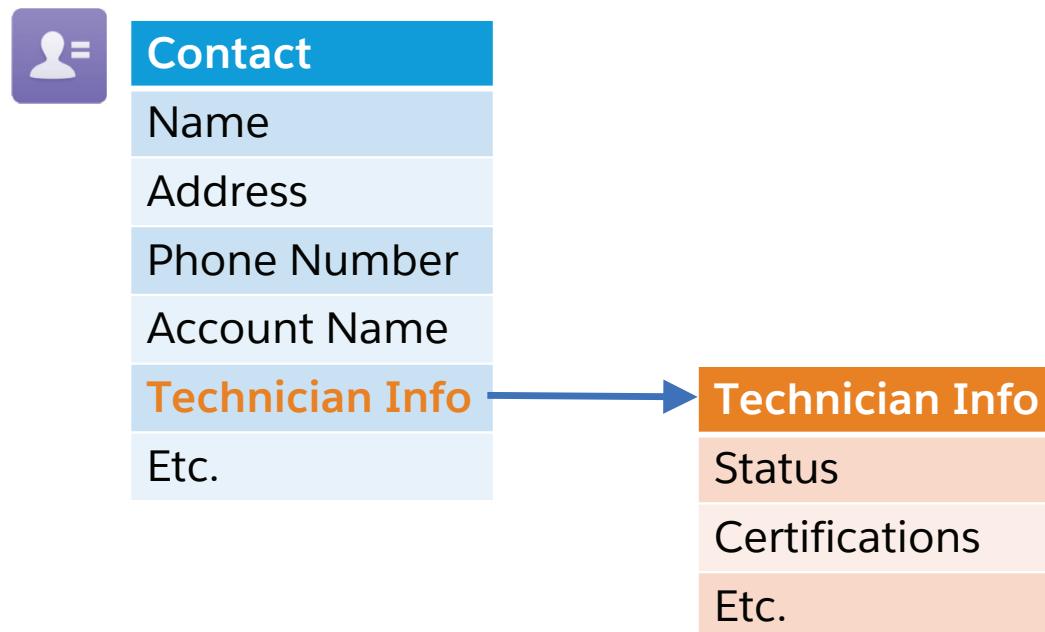
	Option A	Option B
Object	Use the standard Contact object	Create a custom object called Technician
Custom Fields	Only the Status field	All fields except Name
Relationships	None	A relationship from Technician to Account
Page Layouts & Record Types	2 page layouts 2 record types	None

Licences communiques 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.

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?



## 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	Location
APAC	Tokyo, JP
EMEA	Singapore, SG
NAMER	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

84

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

85

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?



**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



Ryan Jackson

Lead Developer



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 AGENDA

89

salesforce

### 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

90

salesforce

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.

**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

92

salesforce

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?

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

# WHAT IS APEX?

93

salesforce

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.

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.



**Goal:**

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

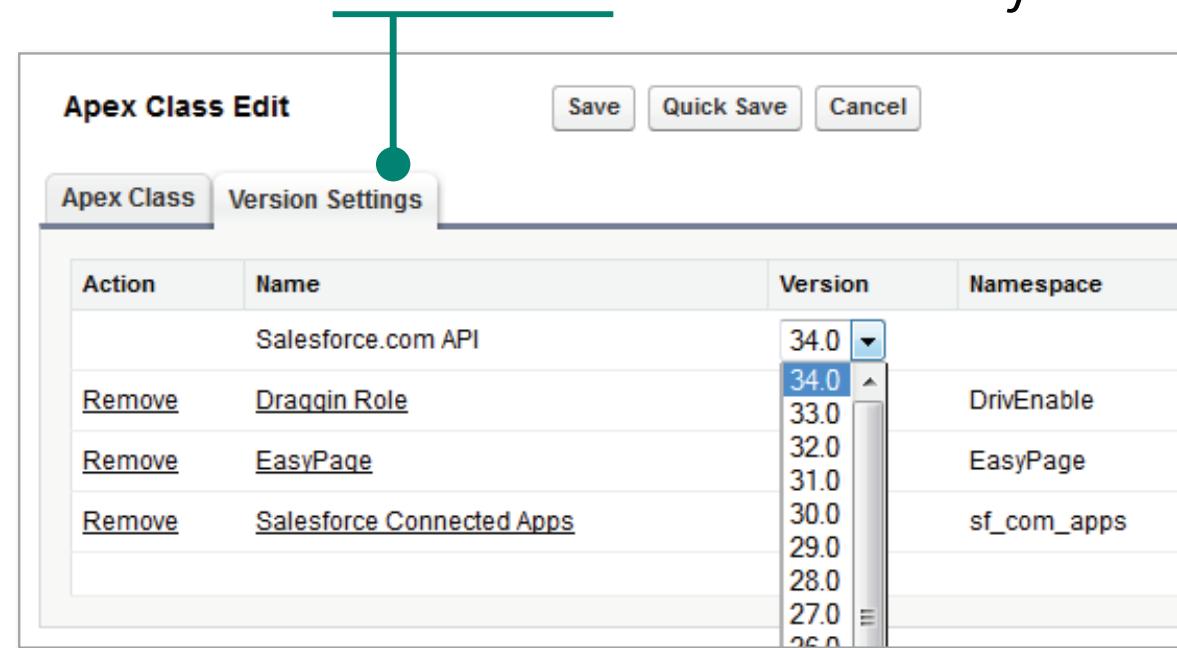
**Tasks:**

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.

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#?

```
1 public class ContactManager {  
2     public static ID addContact(String lastNameToInsert, String firstNameToInsert) {  
3         Contact contactToAdd = new Contact(firstName=firstNameToInsert,  
                                         lastName=lastNameToInsert);  
4         insert contactToAdd;  
5         return contactToAdd.Id;  
6     }  
7 }
```

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.

**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.

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?



- 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.

**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



Blob	Decimal	Long
Boolean	Double	String
Date	ID	Time
Datetime	Integer	

```
1 //Sample String method
2 Boolean validString = userString.isAlphanumeric();
3
4 //Sample Datetime method
5 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.

- 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());

```

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));

```

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



# MODULE AGENDA

105

salesforce

## 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.

## A Single Apex Transaction

- Entry point from execute anonymous.
- Calls Method 1.

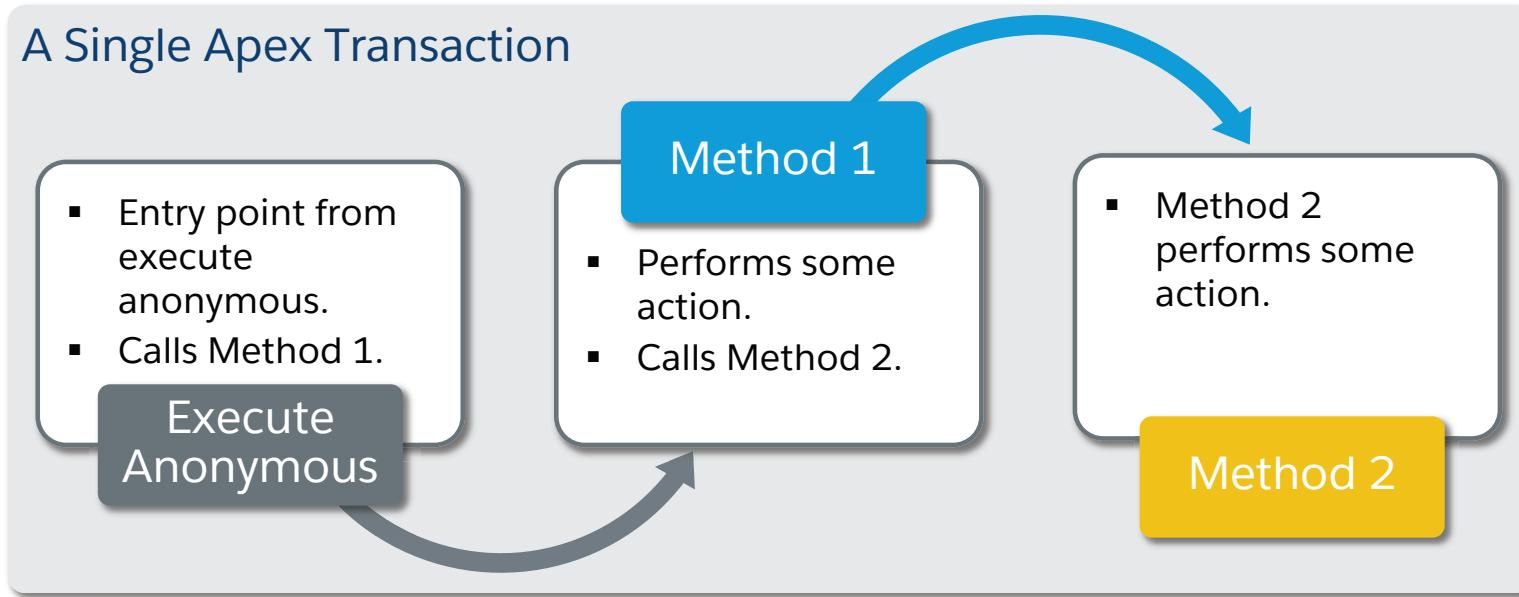
Execute  
Anonymous

### Method 1

- Performs some action.
- Calls Method 2.

- Method 2 performs some action.

Method 2



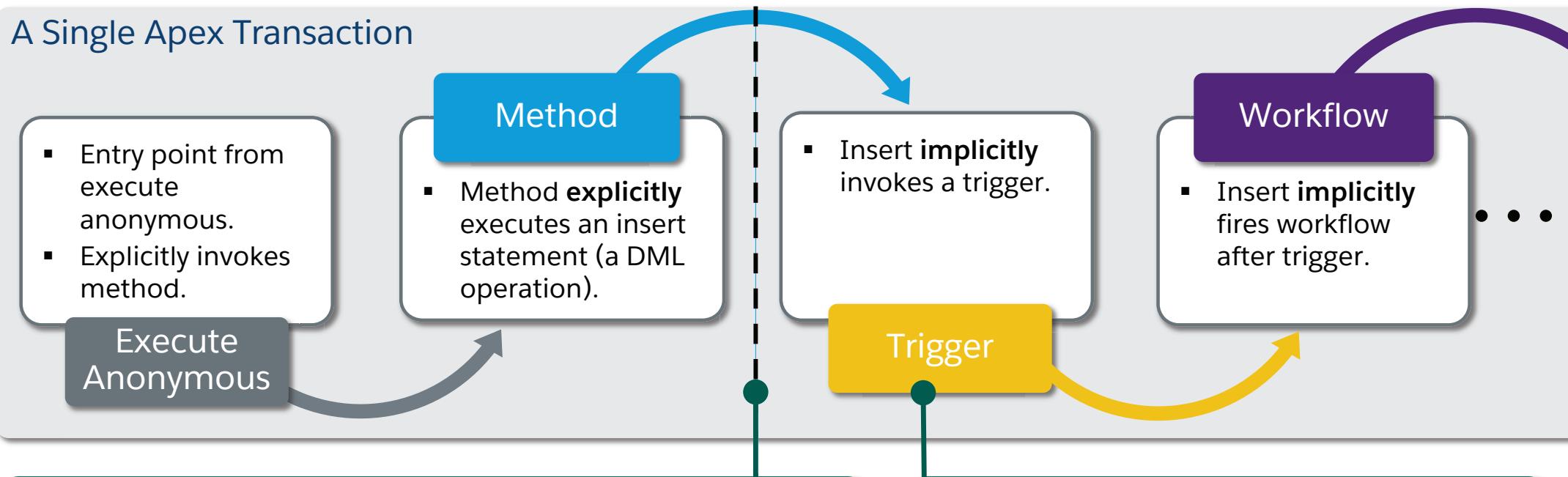
# APEX TRANSACTIONS THAT OPERATE ON THE DATABASE

107

salesforce

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.

## A Single Apex Transaction



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.

**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?

109

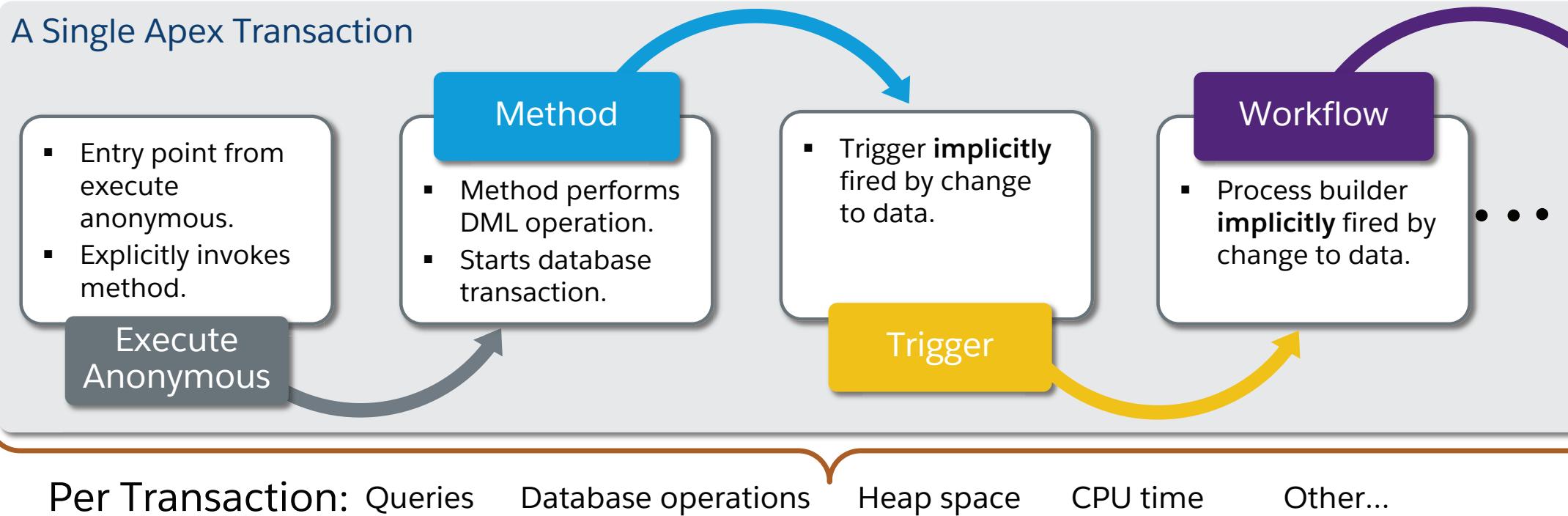
salesforce

DEFINITION:



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

## A Single Apex Transaction



ALERT:



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



## 4-7: PROFILE LIMITS USING THE DEVELOPER CONSOLE

110

salesforce

5 minutes

### Goal:

Investigate limits using the Developer Console.

### Task:

Review the limit profiling information in the log.



# MODULE AGENDA

111

salesforce

## MODULE 4: PROGRAMMING WITH APEX

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



# WORKING WITH DATA IN APEX: A COMMON PATTERN

112

salesforce

**Retrieve** the Contacts to modify from the database.

**Modify** the Contact sObjects in memory.

**Persist** the data in the modified Contact sObjects to the database.

SOQL

Statements

DML

Apex



# WHAT IS AN sObject?

113

salesforce

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.

```
1A Account acctToAdd = new Account(name = 'Feldman Associates');  
2A insert acctToAdd;
```

```
1B public void updateLocation(List<Course_Delivery__c> deliveriesToCheck) {  
2B     List<Course_Delivery__c> deliveriesToUpdate = new List<Course_Delivery__c>();  
3B     for (Course_Delivery__c delivery: deliveriesToCheck) {  
4B         if (delivery.location__c == 'Tokyo') {  
5B             delivery.region__c = 'JAPA';  
6B             deliveriesToUpdate.add(delivery);  
7B         }  
8B     }  
9B     update deliveriesToUpdate;  
10B }
```

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

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?

115

salesforce

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

```
1 public void updateLocation(List<Course_Delivery__c> deliveriesToCheck) {  
2     List<Course_Delivery__c> deliveriesToUpdate = new List<Course_Delivery__c>();  
3     for (Course_Delivery__c delivery: deliveriesToCheck) {  
4         if (delivery.location__c == 'Tokyo') {  
5             delivery.region__c = 'JAPA';  
6             deliveriesToUpdate.add(delivery);  
7         }  
8     }  
9     update deliveriesToUpdate;  
10 }
```

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.

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

```
1  public void updateLocation(List<Course_Delivery__c> deliveriesToCheck) {  
2      List<Course_Delivery__c> deliveriesToUpdate = new List<Course_Delivery__c>();  
3      for (Course_Delivery__c delivery: deliveriesToCheck) {  
4          if (delivery.location__c == 'Tokyo') {  
5              delivery.region__c = 'JAPA';  
6              deliveriesToUpdate.add(delivery);  
7          }  
8      }  
9      update(deliveriesToUpdate);  
10 }
```

Custom Fields & Relationships		
Action	Field Label	API Name
Edit   Del	<u>Course</u>	Course__c
Edit   Del	<u>Instructor</u>	Instructor__c
Edit   Del   Replace	<u>Location</u>	Location__c

# HOW IS AN sObject NAMED IN APEX?

117

salesforce

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



Contact	
Number_of_Certifications_Held_c	<i>Roll-Up Summary (COUNT Certification_t)</i>
Account	<i>Lookup(Account)</i>
AssistantName	<i>Text(40)</i>
AssistantPhone	<i>Phone</i>

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

Certification_Held_c	
Certification_c	<i>Master-Detail(Certification)</i>
Name	<i>Auto Number</i>
Certified_Professional_c	<i>Master-Detail(Contact)</i>
CreatedBy	<i>Lookup(User)</i>
CurrencyIsoCode	<i>Picklist</i>
Date_Achieved_c	<i>Date</i>
LastModifiedBy	<i>Lookup(User)</i>

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

## 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

119

salesforce

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.

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

# USING APEX TO GET DATA FROM THE DATABASE

120

salesforce

```
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.

**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

122

salesforce

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.

```
1 //Populated from the database.  
2 List<Contact> contactsToModel = [SELECT id, AccountId, Account.Name  
                                     FROM contact];  
3 Contact contactToModel = contactsToModel[0];  
4 //Have the new contact associated with the same account as contactToModel using  
5 //the ID AccountId field.  
6 Contact ContactToAdd = new Contact(AccountID=contactToModel.AccountId);  
7  
8 //Output the name of the Account just added to the new contact using the Account field  
9 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.)

123

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?

124

salesforce

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?

125

salesforce

For each blank, what type must be specified?

```
1 if (oldCertAttempts.length() > 0) {  
2     Certification_AccessAttempt__c certificationAttempt = oldCertAttempts[0];  
3     1 var1 = certificationAttempt.Certification_Element__c;  
4 2 : un Contact var2 = certificationAttempt.Certification_Candidate__r;  
5     3 var3 = certificationAttempt.Certification_Candidate__r.Account;  
6 }
```

**1** **2 : un Contact** **3 : Un account**

# REFERENCING A RELATIONSHIP FIELD ON A PARENT sObject

126



## 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.

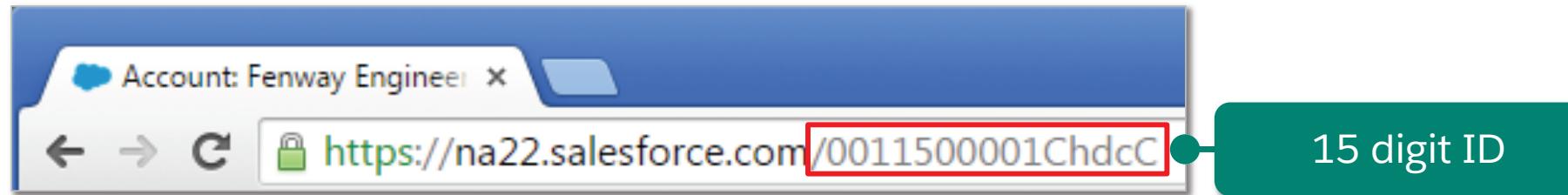
What must be specified if you want to select Certification Attempt records associated with a Certification Element?

```
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 List <Certification_Attempts__c>  
4B }
```

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



IHM, formules, etc

A screenshot of a Salesforce query results page. The title bar says "Query Results - Total Rows: 1". The results table has a single row with an "Id" column containing "0011500001ChdcCAAR". A green callout box points to the "Id" column with the text "18 digit ID".

IL FAUT COMPARER LES ID SUR LE  
MEME FORMAT

Case insensitive



ALERT:  
Never hardcode an Id value in any code.

**Goal:**

Explore the use of record Ids.

**Tasks**

1. Query the Id and name fields for Contact records in the database.
2. Copy the Id from a returned record.
3. Use the Id to display the associated Contact page in the UI.



## KEY TAKEAWAYS

130

salesforce

- 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.



## KNOWLEDGE CHECK

131

salesforce

### 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?

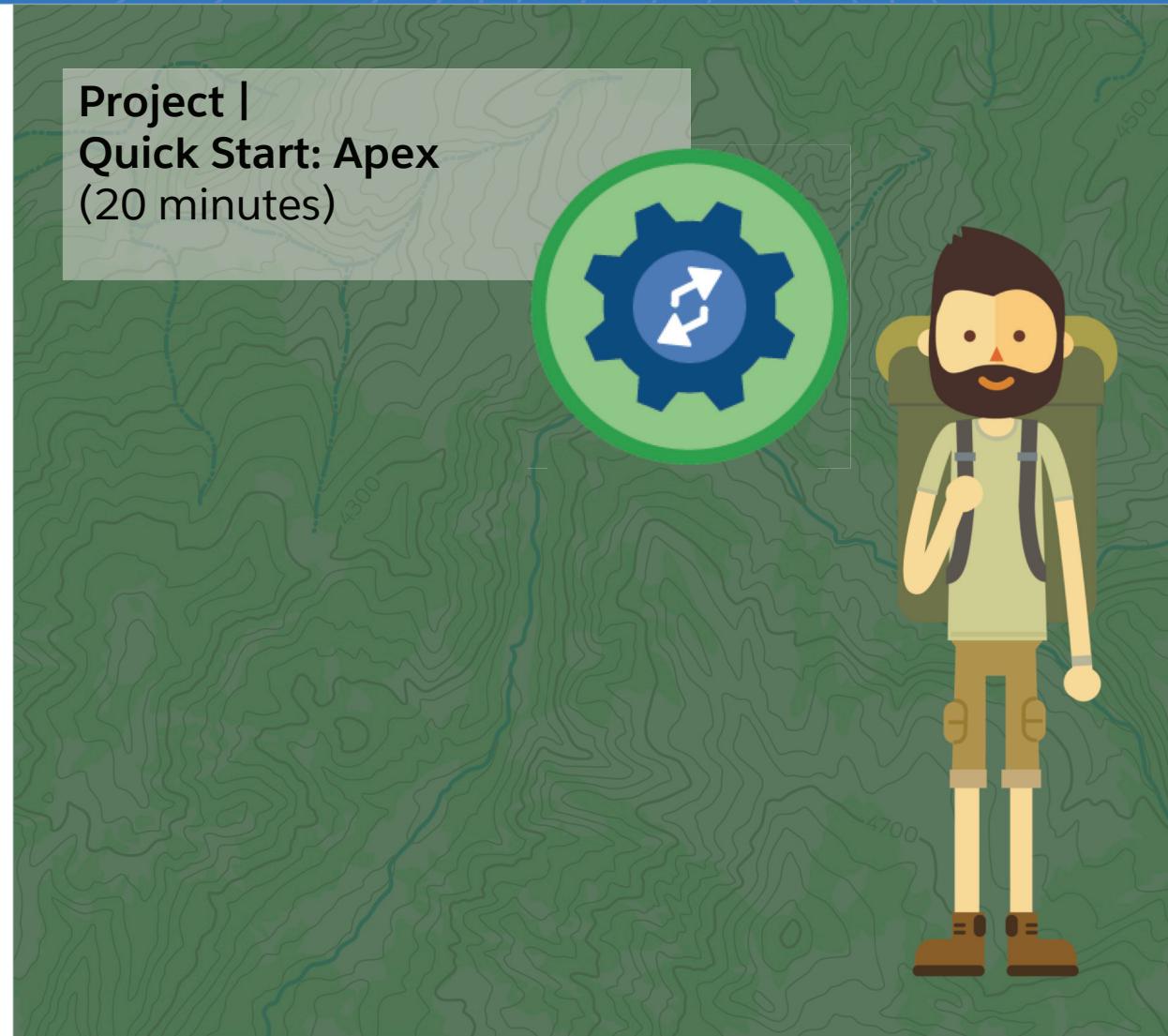
- a) sObj1 == sObj2
- b) sObj1.equals(sObj2)
- c) sObj1==sObj2
- d) sObj1.eq(sObj2)

 KNOWLEDGE CHECK (CONT.)

132

salesforce

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.



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

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE



# WORKING WITH DATA IN APEX: A COMMON PATTERN

135

salesforce

**Retrieve** the Contacts to modify from the database.

**Modify** the Contact sObjects in memory.

**Persist** the data in the modified Contact sObjects to the database.

SOQL

Statements

DML

Apex



Cassie Mitchell

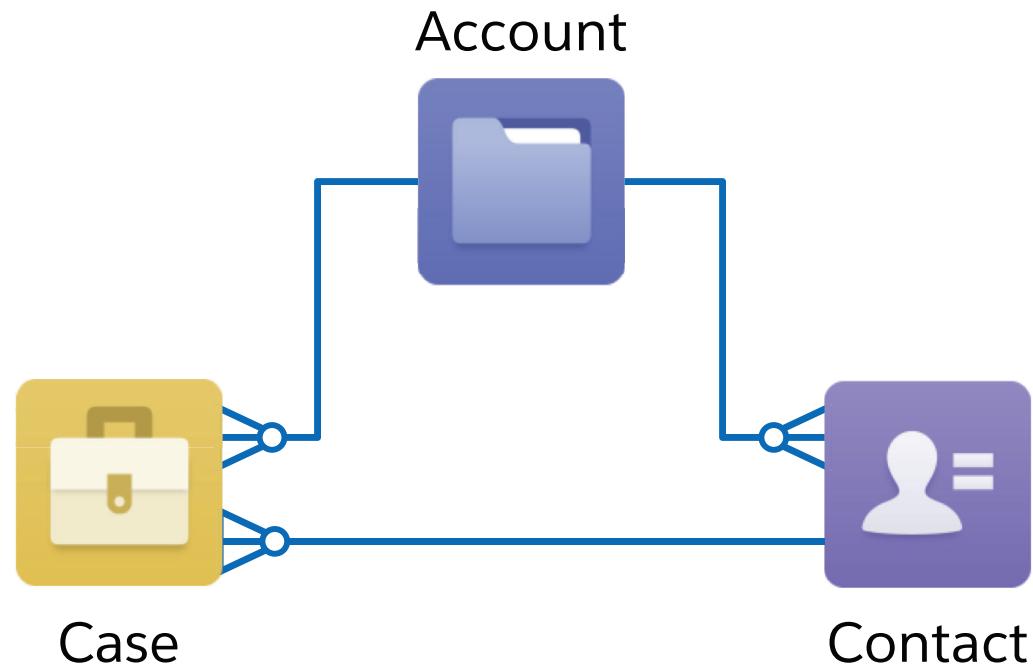
Beginning Developer



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.





## MODULE AGENDA

138

salesforce

### 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?

139

salesforce

Case Detail		<a href="#">Edit</a>	<a href="#">Delete</a>	<a href="#">Close Case</a>	<a href="#">Clone</a>	<a href="#">Sharing</a>
Case Owner	 <a href="#">Tim Howe [Change]</a>			Status	New	
Case Number	00001015 <a href="#">[View Hierarchy]</a>			Priority	High	
Parent Case				Contact Phone	1-713-555-2429	
Contact Name	<a href="#">Jason Cresta</a>			Contact Email	<a href="mailto:jason.cresta@trainingorg-vandelayindustries.com">jason.cresta@trainingorg-vandelayindustries.com</a>	
Account Name	<a href="#">Vandelay Industries</a>			Case Origin	Email	
Backup Agent	<a href="#">?</a>			Type		
Case Record Type	<a href="#">Product Support [Change]</a>					
<b>▼ Additional Information</b>						
Subject	A200 Series Desktop does not shut down.					
Description	A200 Series Desktop does not shut down. This happens intermittently.					
Bug Number						
Product	PC					
<b>▼ System Information</b>						
Date/Time Opened	5/22/2015 8:28 AM	Date/Time Closed				
Created By	<a href="#">Tim Howe, 5/22/2015 8:28 AM</a>	Last Modified By	<a href="#">Tim Howe, 5/22/2015 8:28 AM</a>			

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



WATCH ME

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

140



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.

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.

## Most SQL Variants

## SOQL

Support statements for querying, CRUD, transaction control, schema definition, and more

Support SELECT \*

Support joins, which are written using “left” and “right” keywords

Do not support dot notation syntax to traverse foreign key relationships

Are not governed by limits

Only supports query statements

Does not support SELECT \* 

Supports “relationship queries,” which are written using parent-child syntax 

Supports dot notation syntax to traverse object relationships

Is multi-tenant aware (therefore, governed by limits)



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

# A SOQL EXAMPLE

144

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

Return only 1 record

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

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

145

salesforce

```
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

146

salesforce

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
```

Date values in queries should not be enclosed in quotes.

**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

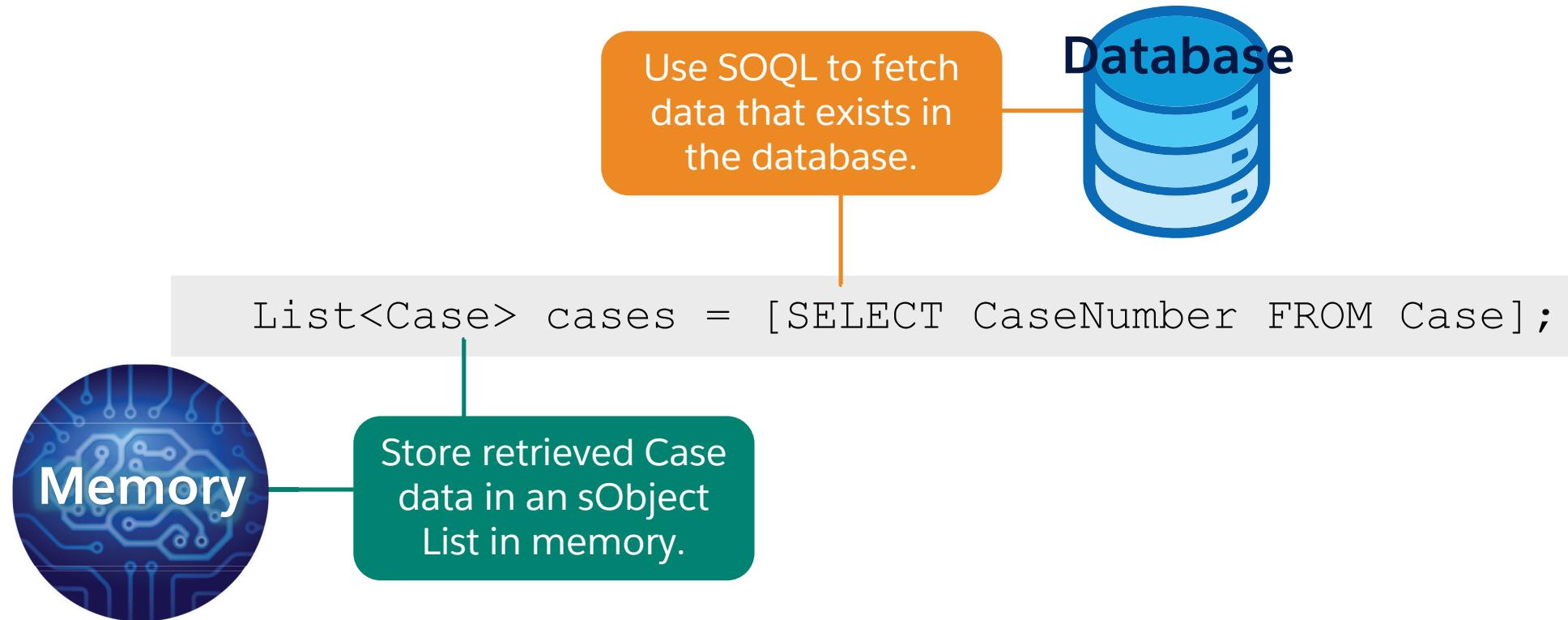
148

salesforce

### 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





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

151

salesforce

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

C'est même une bonne pratique.

```
1 List<Case> cases = [SELECT CaseNumber FROM Case];
2
3
4 for (Case aCase : cases) {
5     System.debug(aCase);
6 }
7
8
9 // alternative loop with integer iterator
10 Integer numCases = cases.size();
11 for (Integer i = 0; i < numCases; i++) {
12     System.debug(cases[i]);
13 }
```

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

NOTE:



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

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

153

salesforce

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];
```

### Execute Anonymous Error

Line: 1, Column: 1

System.QueryException: List has more than 1 row for assignment to SObject

OK

**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.

- 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?

```
1 List<Case> cases = [SELECT CaseNumber FROM Case];
2 // Assume the list "cases" contains at least 1 case.
3
4 // FOR Loop 1
5 for (Case aCase : cases) {
6     System.debug(aCase.Id + ' ' + aCase.CaseNumber);
7 }
8
9 // FOR Loop 2
10 for (Case theCase : cases) {
11     System.debug(theCase.Subject);
12 }
13
14 // FOR Loop 3
15 for (Case currentCase : cases) {
16     currentCase.Subject = 'Printer is jammed';
17 }      Simple affectation aucune raison que ça pose probleme.
```



## EXCEEDING THE HEAP SIZE LIMIT

156

salesforce

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?

```
1 List<Case> cases = [SELECT Description FROM Case];  
2  
3 //... process cases
```

For loop

On ne requête que les éléments dont on a vraiment besoin

# USING List<sObject> ITERATION VARIABLE

157

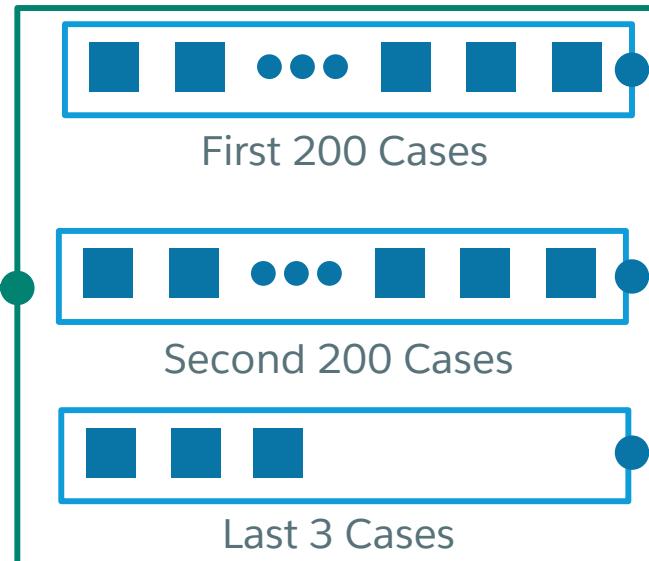
salesforce

```
1 for (List<Case> cases : [SELECT CaseNumber FROM Case]) {  
2     for (Case aCase : cases) {  
3         //... process a single case  
4     }  
5 }
```

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?

158

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?

```
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) !
```

Important à comprendre !



## MODULE AGENDA

159

salesforce

### 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**



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

Example of a bound variable:

```
1B Set<String> caseSet = new Set<String>{'00001003', '00001005'};  
2B List<Case> selectCases = [SELECT CaseNumber  
                           FROM Case  
                           WHERE CaseNumber IN :caseSet];
```

Use the colon operator  
to specify binding.

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

```
1 String criteria = '';  
2 Boolean lookForOpenCases = false;  
3  
4 if (lookForOpenCases) {  
5     criteria = 'Status = \'Working\'';  
6 } else {  
7     criteria = 'ClosedDate < TODAY';  
8 }  
9  
10 List<Case> theCases =  
11     Database.query('SELECT Id, Subject FROM Case WHERE ' + criteria);  
12 System.debug(theCases);
```



ALERT:

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

**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

163

salesforce

- 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.



## KNOWLEDGE CHECK

164

salesforce

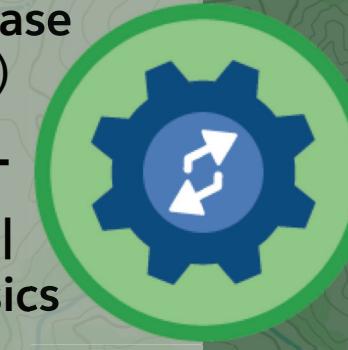
1. If a developer executes this query in the Query Editor of the Developer Console, what would be the result?

```
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



Jason Beck

Developer



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 AGENDA

168

salesforce

### MODULE 6: USE SOQL TO QUERY PARENT-CHILD RELATIONSHIPS

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



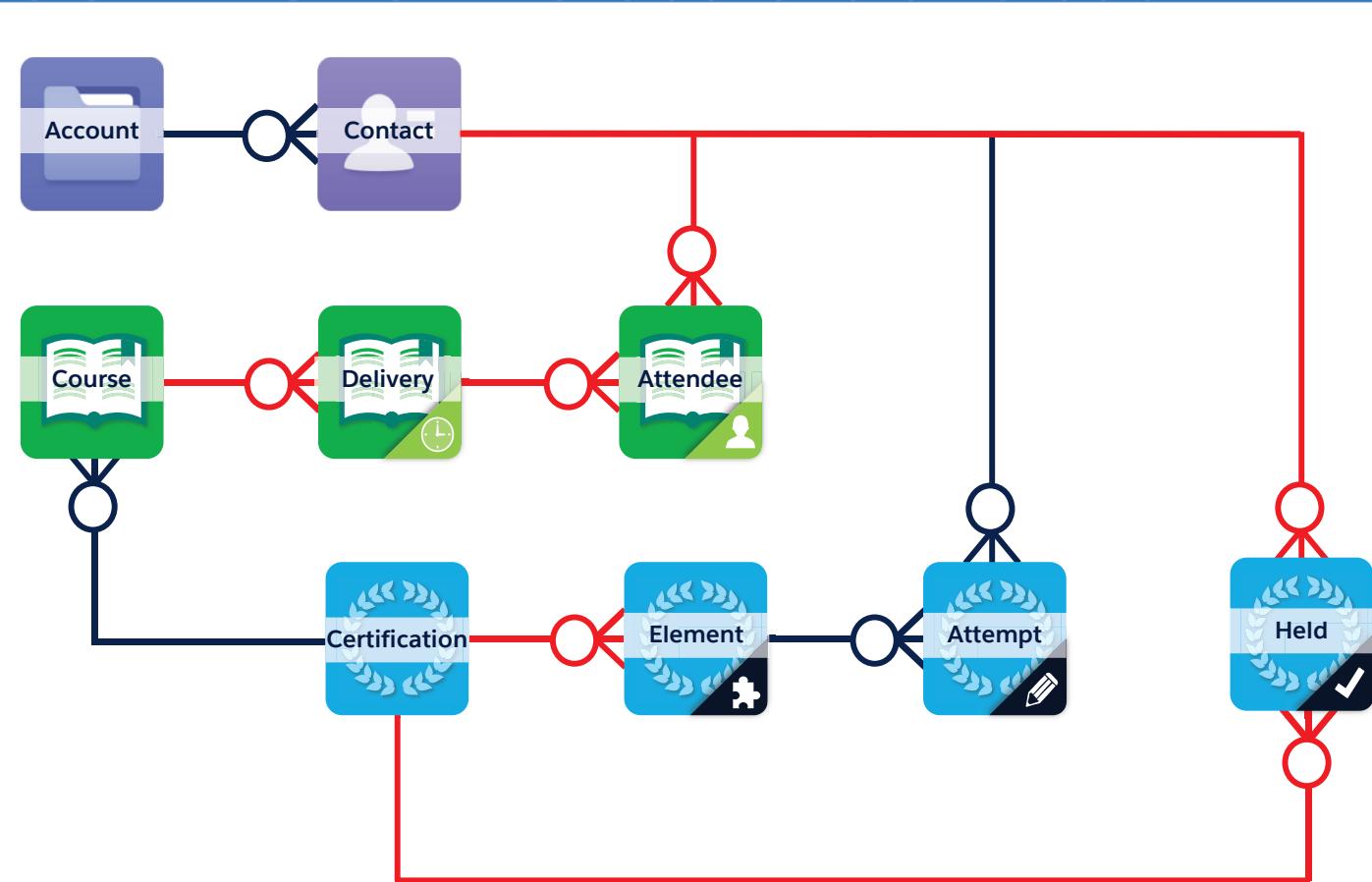
# UNDERSTANDING THE CERTIFICATION APPLICATION

169

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?

170

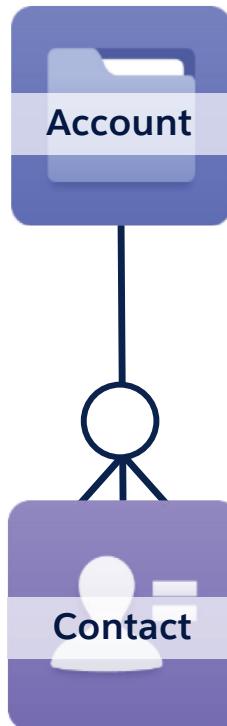
salesforce

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

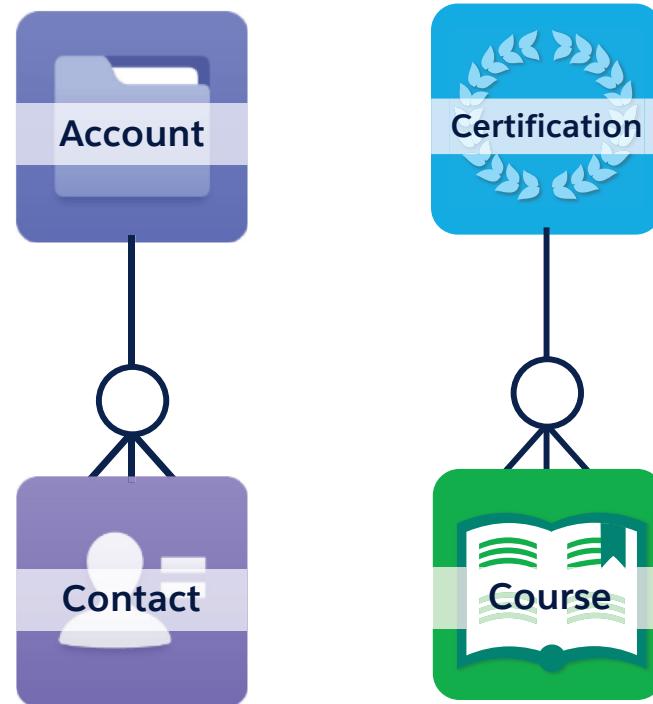
The Account sObject is a parent of the Contact sObject.  
An Account may be related to many Contacts.  
A Contact may be related to a single Account.

# REVIEW OF TYPES OF RELATIONSHIPS

171

salesforce

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



# WHAT IS A RELATIONSHIP QUERY?

172

salesforce

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 AGENDA

173

salesforce

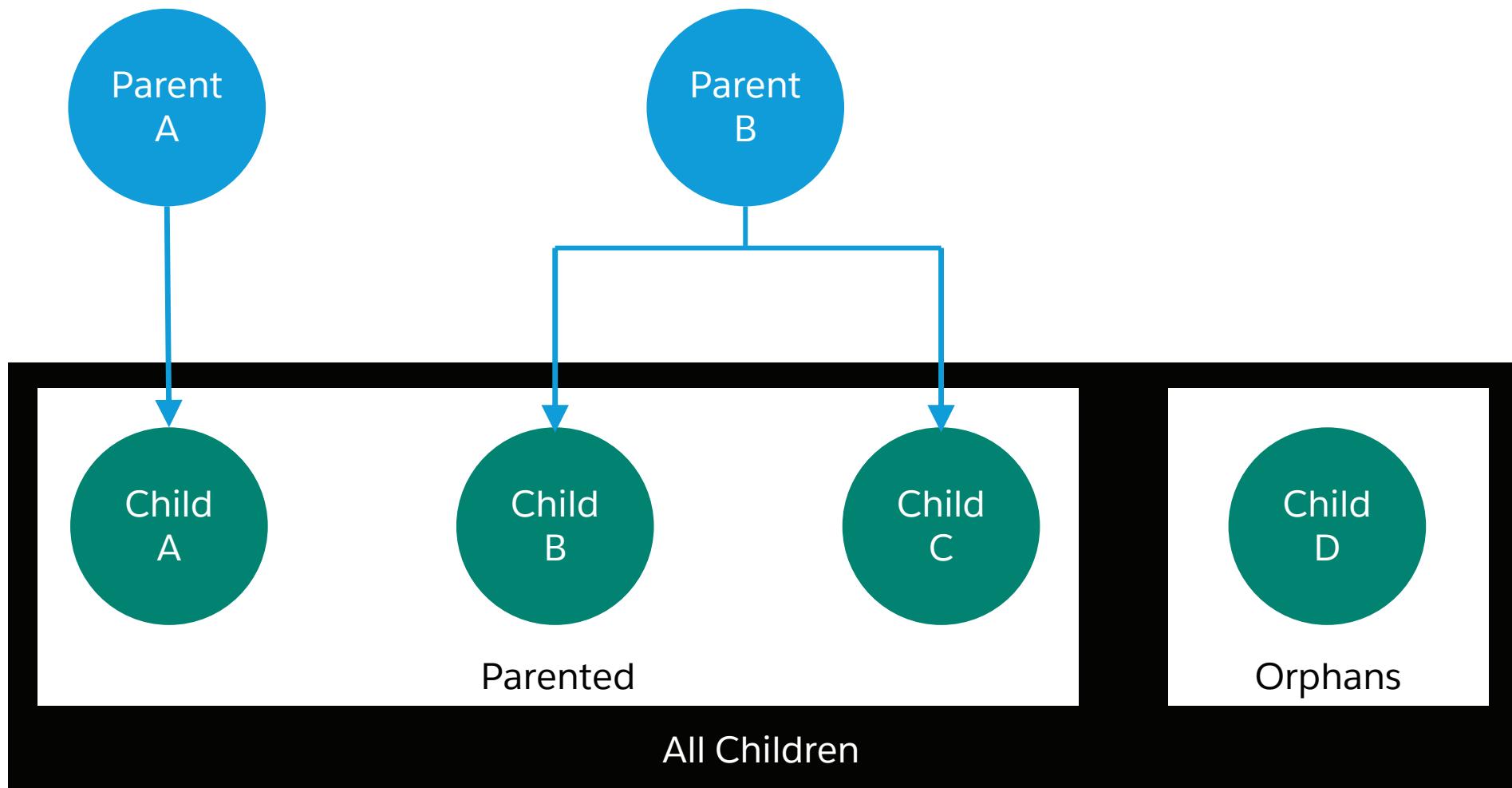
### MODULE 6: USE SOQL TO QUERY PARENT-CHILD RELATIONSHIPS

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



# TERMINOLOGY: TYPES OF CHILD-TO-PARENT RELATIONSHIP QUERIES

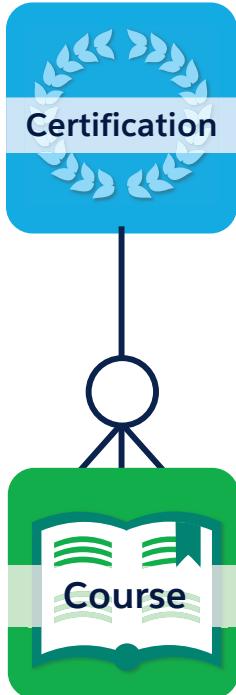
174



# WHICH ROWS WILL BE FETCHED?

175

salesforce

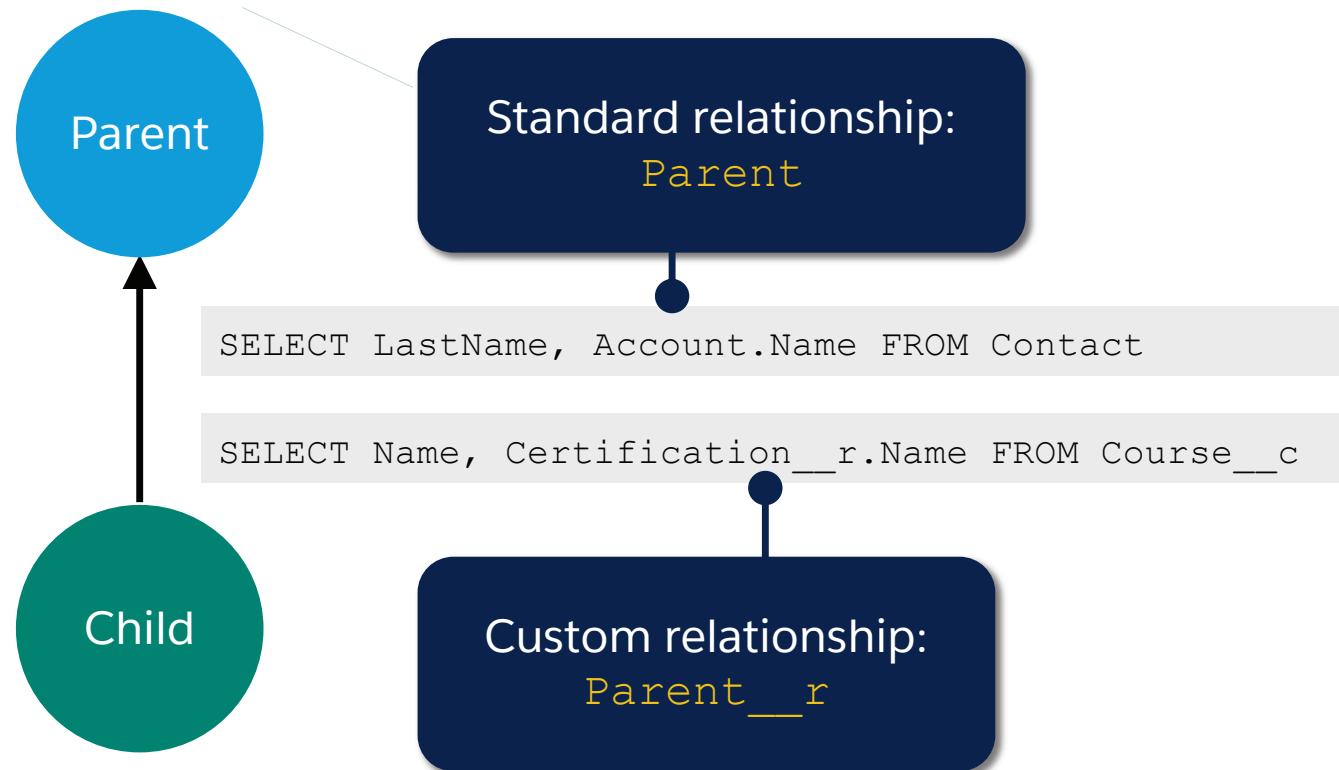


Field Values from Course	
Name	Certification__c
[202] AWCP Network	a03U00000091uRIIAY
[401] Data Recovery	<null>
[402] Managing Network Load	<null>

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

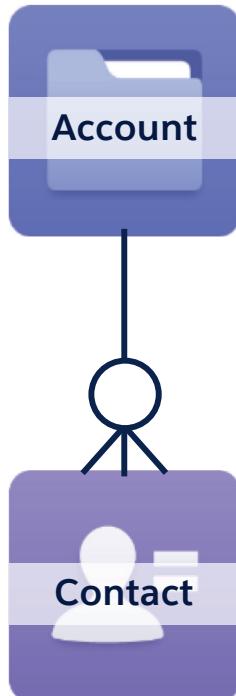
176



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



1

2

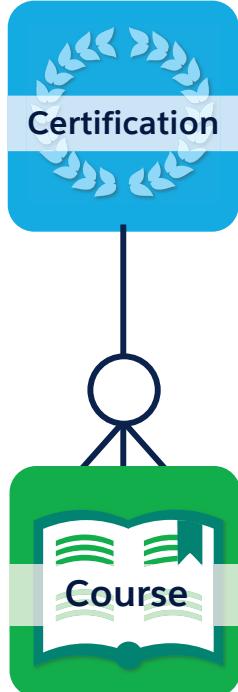
Account.Name

The query above selects all Contacts.

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

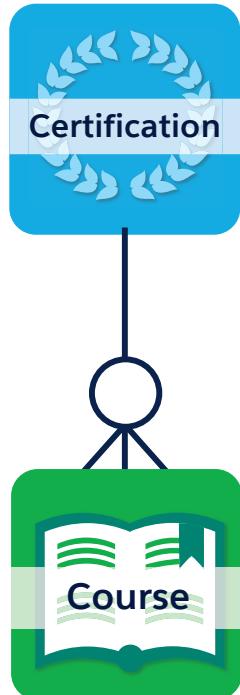


```
SELECT Name, 1, 2 Certification__r.Name
FROM Course__c
```

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>



```
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
<b>Name</b>	<b>Parent Certification's Name</b>
[202] AWCP Network	AWCP Network

**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 AGENDA

181

salesforce

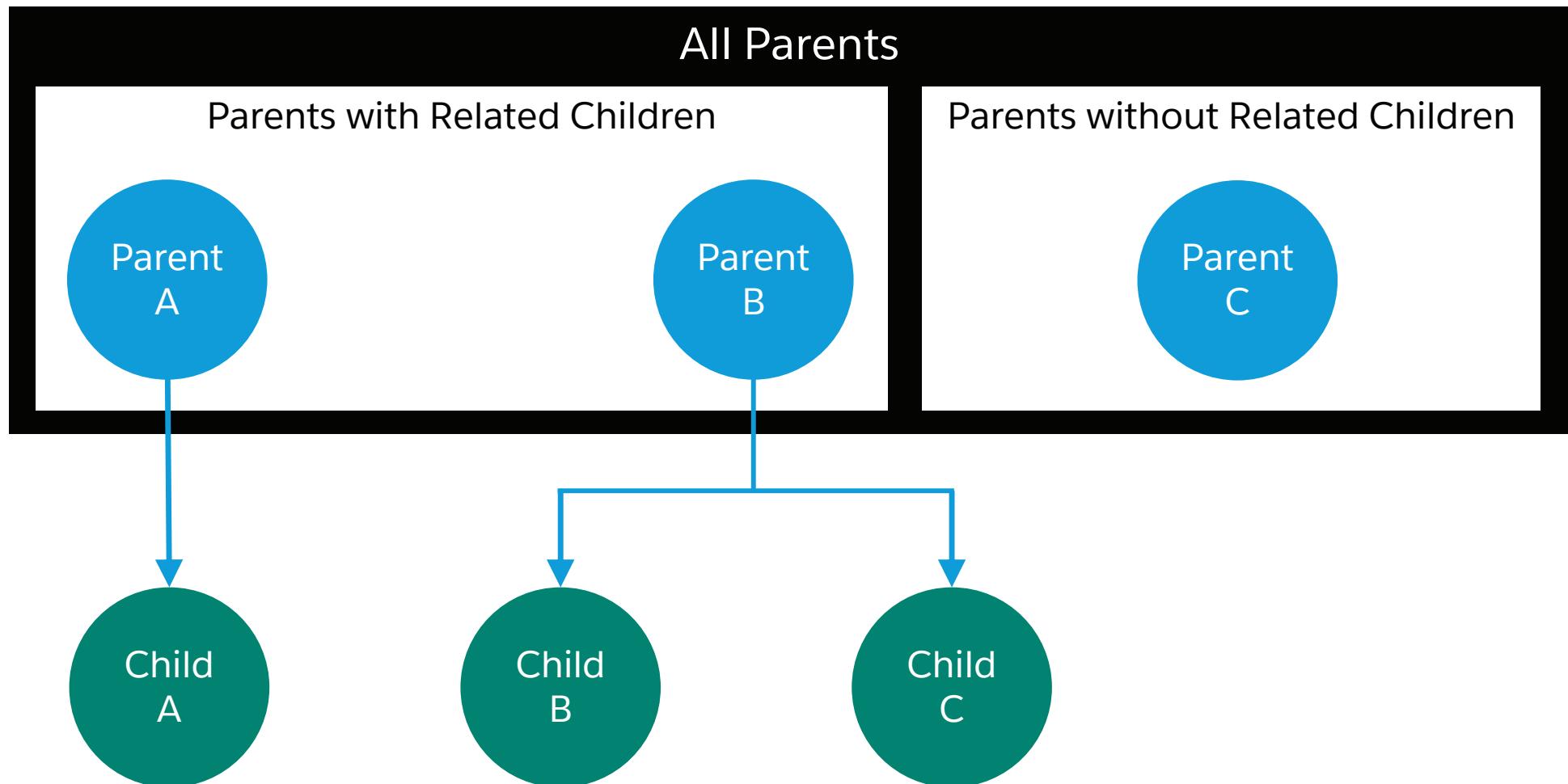
### MODULE 6: USE SOQL TO QUERY PARENT-CHILD RELATIONSHIPS

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



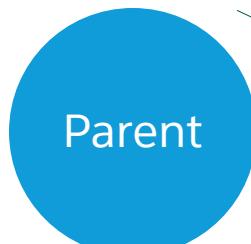
# TERMINOLOGY: TYPES OF PARENT-TO-CHILD RELATIONSHIPS

182



# REFERENCING THE PARENT-TO-CHILD RELATIONSHIP

183

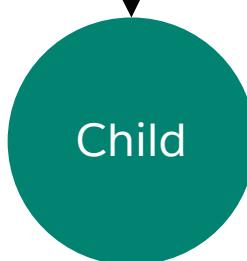


Standard relationship:  
Children

```
SELECT Name, (SELECT Lastname, Firstname FROM Contacts) FROM Account
```



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



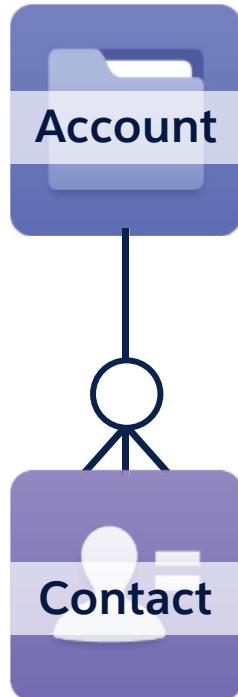
Custom relationship:  
Children\_\_r

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.

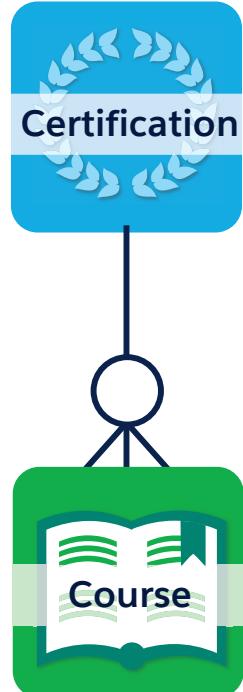


```
SELECT Name,
       (SELECT LastName FROM Contacts)
  FROM Account
```

This query selects all Accounts.

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
<b>Name</b>	<b>List of Contacts</b>
Enmore Installations	[{"LastName":"Morelli"}, {"LastName":"Parkinson"}, {"LastName":"Curran"}, {"LastName":"Yang"}, {"LastName":"Delany"}]



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

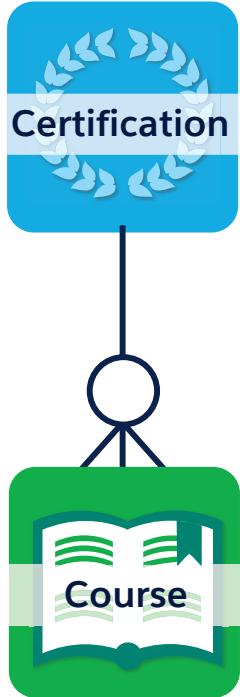
This query selects all Certifications.

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"}]



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

This query selects only Certifications that have related Courses.

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

Field Values from Certification
Certification Name
AWCP Server

**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.





## KEY TAKEAWAYS

188



- 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



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

191

salesforce

### MODULE 7: DML ESSENTIALS

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



# WORKING WITH DATA IN APEX: A COMMON PATTERN

192

salesforce

Retrieve the Contacts  
to modify from  
the database.

Modify the Contact  
sObjects in memory.

Persist the data in the  
modified Contact  
sObjects to the database.

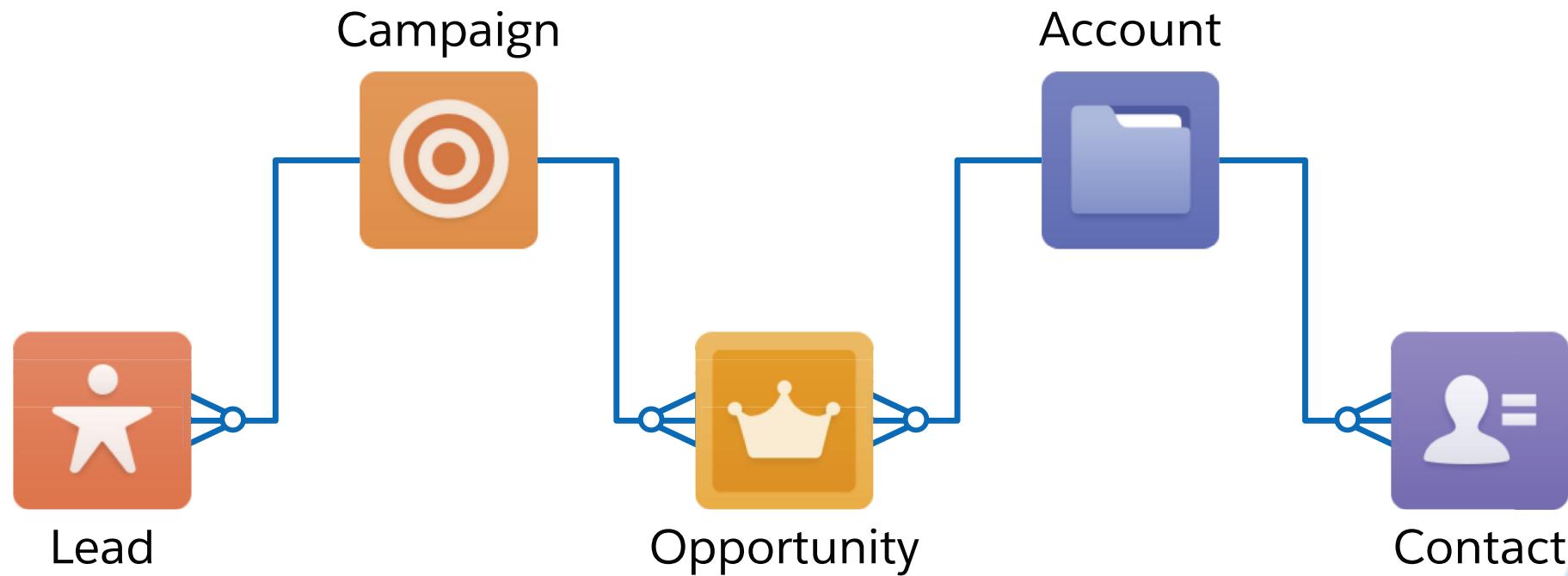
SOQL

Statements

DML

Apex





# DEMO: SAVING CHANGES TO CONTACTS IN THE DEVELOPER CONSOLE

194



Contact@5:59 PM

```
SELECT Id, LastName, LeadSource FROM Contact WHERE LeadSource ='Trade Show'
```

Query Results - Total Rows: 8

Id	LastName	LeadSource
0031400002PBPE0AAP	Bruce	Trade Show
0031400002PBPE4AAP	Farmer	Trade Show
0031400002PBPE0AA5	Maughlin	Trade Show
0031400002PBPETAAS	Decker	Trade Show
0031400002PBPEVAAS	Diesner	Trade Show
0031400002PBPEIAAP	Hardy	Trade Show
0031400002PBPE6AAP	Hensley	Trade Show

Query Grid Save Rows Insert Row Delete Row Refresh Grid Access in Salesforce: Create New Open Detail Page Edit Page

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

195

salesforce

```
1 List<Contact> oldLeadSourceContacts =
  [SELECT LeadSource FROM Contact WHERE LeadSource ='Trade Show'];
2
3 Set<Id> oldLeadSourceContactsIds = new Set<Id>();
4 for (Contact c : oldLeadSourceContacts) {
5   c.LeadSource = 'Other';
6   oldLeadSourceContactsIds.add(c.Id);
7 }
8
9 update oldLeadSourceContacts;
10
11 List<Contact> updatedLeadSourceContacts =
  [SELECT LeadSource FROM Contact WHERE Id IN :oldLeadSourceContactsIds];
12 for (Contact c : updatedLeadSourceContacts) {
13   if (c.LeadSource <> 'Other')
14     System.debug('Update failed');
15 }
```

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

196

salesforce

## Scenario

1. Retrieve into memory a Contact whose LeadSource is 'Trade Show.' Modify its LeadSource to 'Other.' Persist this modification to your org. **UPDATE**
2. Create a new instance of a Contact in memory. Persist this instance to your org. **INSERT**
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**
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.
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**

## DML Command

- A. Insert
- B. Update
- C. Upsert
- D. Delete
- E. Undelete



## MODULE AGENDA

197

salesforce

### MODULE 7: DML ESSENTIALS

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



## 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?

199



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

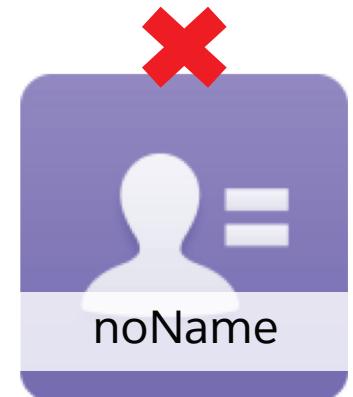
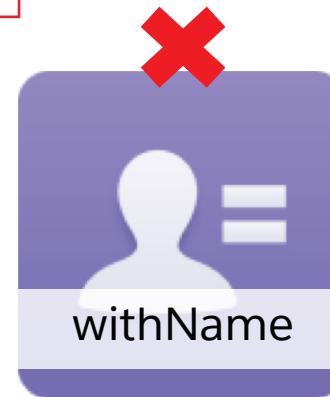
200

salesforce

## Standalone DML

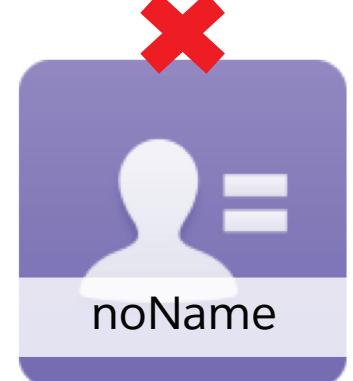
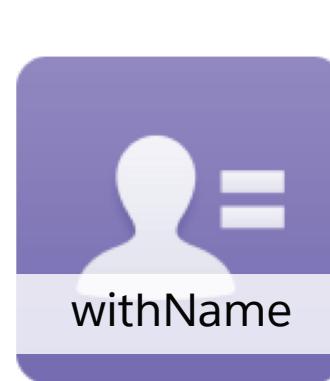
Aucun insert ne rentre si un ou plus ne marche pas

```
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);
```



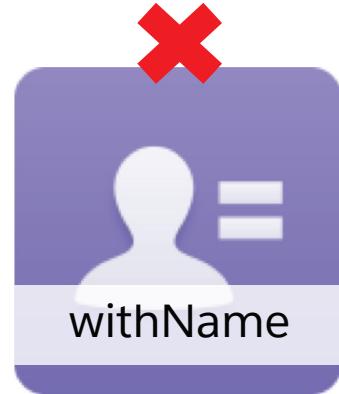
Database.method(param1, param2) :

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.

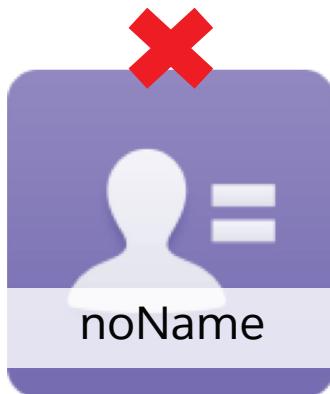
# ARE THERE OTHER OPTIONS FOR PARTIAL PROCESSING?

201

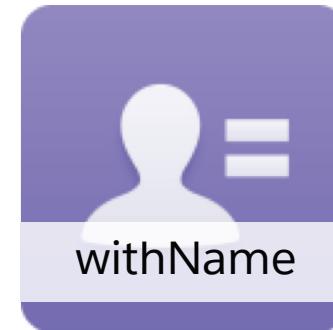
salesforce



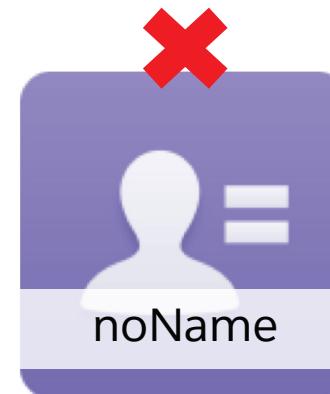
withName



noName



withName



noName

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

202



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

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

**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 AGENDA

204

salesforce

## MODULE 7: DML ESSENTIALS

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



# WHICH CODE BLOCKS RESULT IN AN EXCEPTION?

205

salesforce

```
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 }
```



```
1E List<Contact> contacts = new List<Contact>();  
2E Contact longName = new Contact(LastName =  
    '00085chars00085chars00085chars00085chars00085chars00085chars00085chars00085');  
3E Contacts.add(longName);  
4E insert contacts; //LastName is a Text(80) field
```

85 caractères



# OVERCOMING EXCEPTIONS

206

salesforce

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

No exception.

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

Ensure all the required fields are populated.

```
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;
```

Don't run DML operations on NULL elements.

```
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 }
```

Stay within the DML Limits.

```
1E List<Contact> contacts = new List<Contact>();  
2E Contact longName = new Contact(LastName =  
    '00085chars00085chars00085chars00085chars00085chars00085chars00085chars00085');  
3E Contacts.add(longName);  
4E insert contacts; //LastName is a Text(80) field
```

Ensure that field type restrictions are respected.



```
1 List<Contact> contacts = new List<Contact>();  
2 Contact noName = new Contact();  
3 Contact anotherNoName = new Contact();  
4  
5 Contacts.add(noName);  
6 Contacts.add(anotherNoName);  
7  
8 try {  
9     insert contacts; ● Throws an exception.  
10 } catch (DMLException e){ ● Catches the exception.  
11     System.debug('Caught exception: ' + e);  
12 }
```



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

208

salesforce

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



```
1 Contact withName= new Contact(LastName = 'Lee');
2 Contact noName= new Contact();
3 List<Contact> contacts = new List<Contact>();
4 contacts.add(withName);
5 contacts.add(noName);
6 List<Database.SaveResult> srs = Database.insert(contacts, false);
7
8 for (Database.SaveResult sr : srs) {
9
10    if(sr.isSuccess() == FALSE) {
11
12        List<Database.Error> errors = sr.getErrors();
13
14        Integer i = 1;
15        String debugString = 'Errors: ';
16        for (Database.Error e : errors) {
17            debugString += i + '. ' + e.getMessage() + '; ';
18            i++;
19        }
20
21        System.debug(debugString);
22    }
23 }
```

Loop processes one SaveResult per sObject inserted.

If an error was detected...

... start processing the errors.

getMessage() gets error info.

**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.

```
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

```
1 List<Contact> contacts = new List<Contact>();  
2 for(Contact aContact : [SELECT Id from Contact]) {  
3     //modify aContact  
4     contacts.add(aContact);  
5 }  
6 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?

```
1 for (List<Contact> contacts : [SELECT Id FROM Contact]) {  
2     for (Contact aContact : contacts) {  
3         //modify aContact  
4     }  
5     Database.update(contacts);  
6 }
```



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



## KEY TAKEAWAYS

214

salesforce

- 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



Jason Beck

Developer



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.



## MODULE AGENDA

217

salesforce

### MODULE 8: TRIGGER ESSENTIALS

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



# WORKING WITH DATA IN APEX: A COMMON PATTERN

218

salesforce

**Retrieve** the Contacts to modify from the database.

**Modify** the Contact sObjects in memory.

**Persist** the data in the modified Contact sObjects to the database.

SOQL

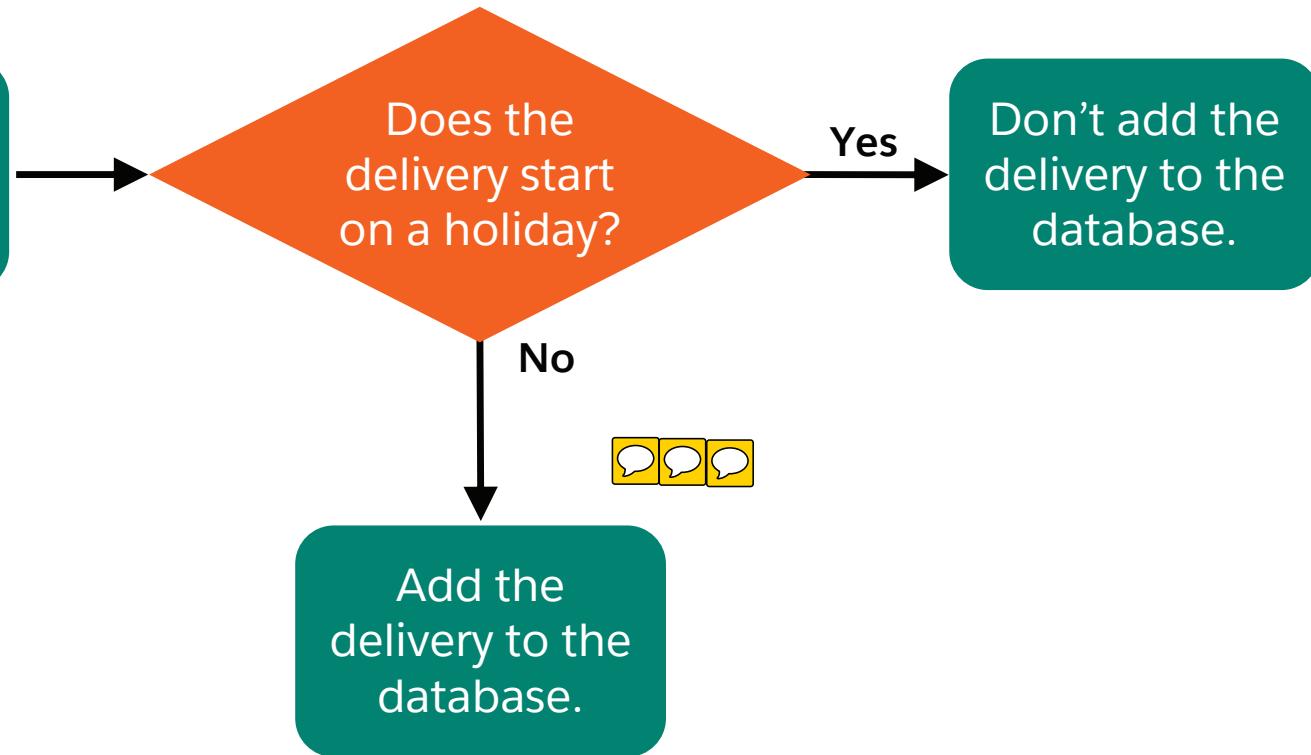
Statements

DML

Apex



An attempt is made to schedule a Course Delivery.



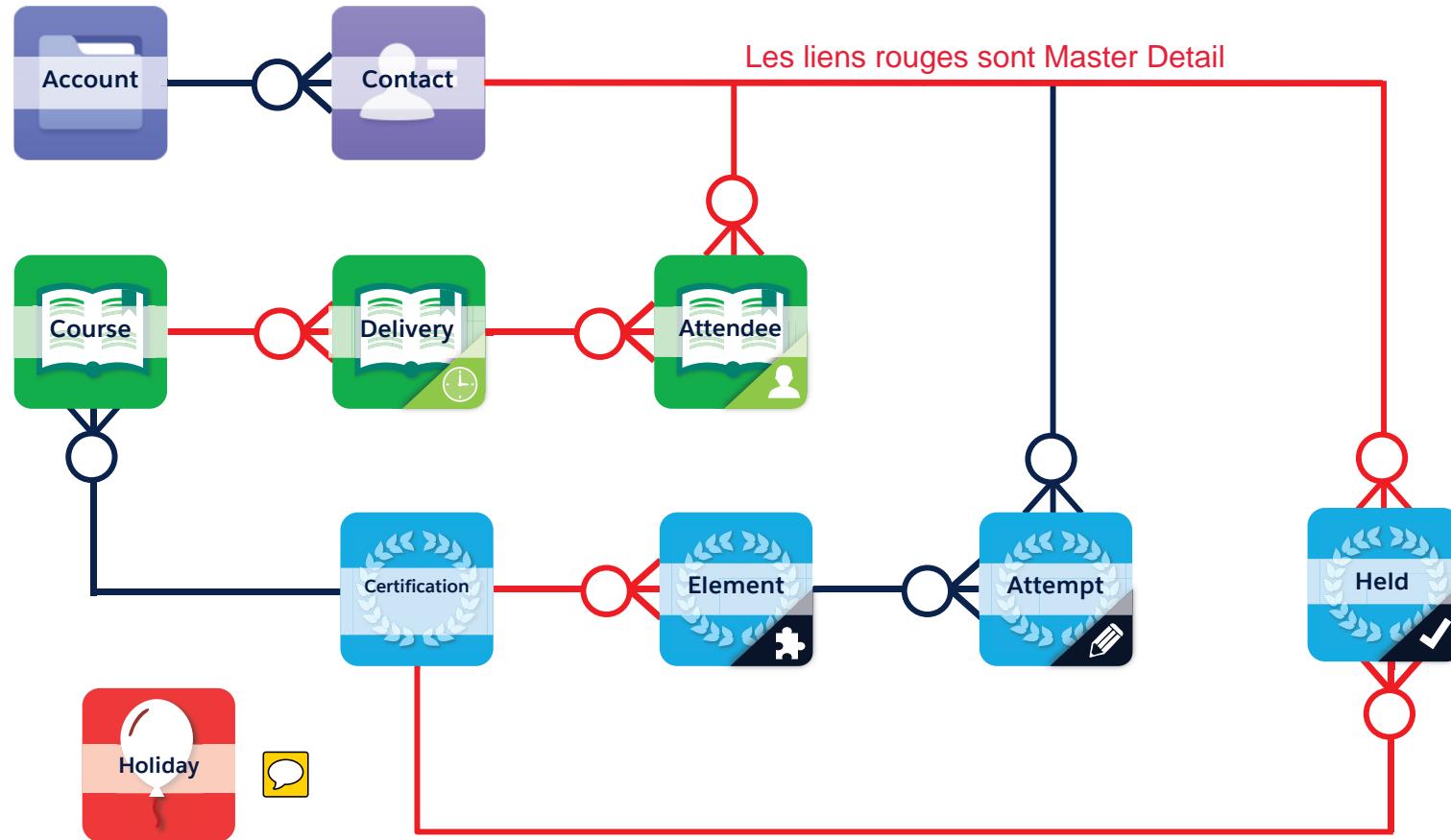
When a training admin tries to save a course delivery record, the system should ensure that the course delivery does not start on a holiday. Holidays are stored in the system-provided Holiday Object.

# WHY CAN'T YOU BUILD THIS LOGIC DECLARATIVELY?

220

salesforce

- **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](https://help.salesforce.com/apex/HTViewHelpDoc?id=process_which_tool.htm)

# A TRIGGER PROVIDES A CODE-BASED SOLUTION

221

salesforce

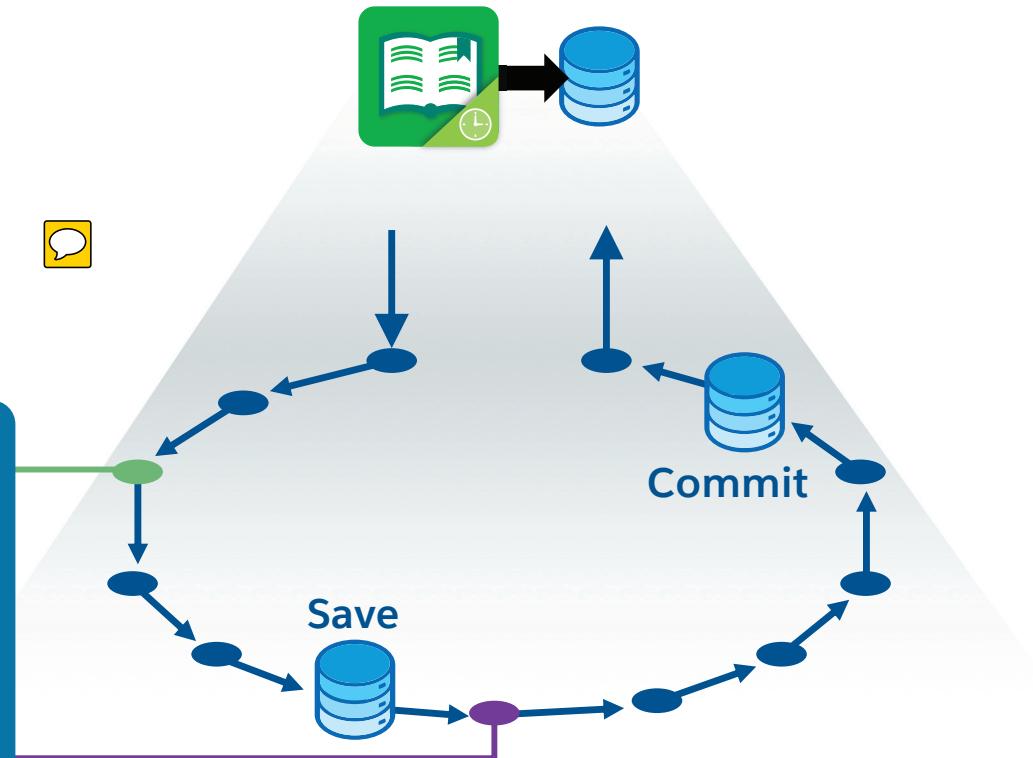
DEFINITION:



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



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?

222

salesforce

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

223

salesforce

## MODULE 8: TRIGGER ESSENTIALS

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



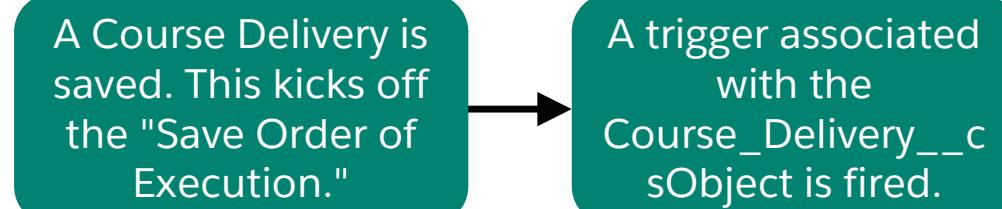
# FOUR KEY QUESTIONS TO ASK BEFORE WRITING A TRIGGER

224

salesforce



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?



The trigger determines: does the Course Delivery start date conflict with a holiday?

Yes

Issue an error and don't save the Course Delivery record.

No

Add the delivery to the database.

# SYNTAX FOR DEFINING A TRIGGER

225

salesforce

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.

```
1 trigger TriggerName on sObject (before insert, before  
update, before delete, after insert, after update, after delete,  
after undelete) {  
2     //Trigger logic ...  
3 }
```

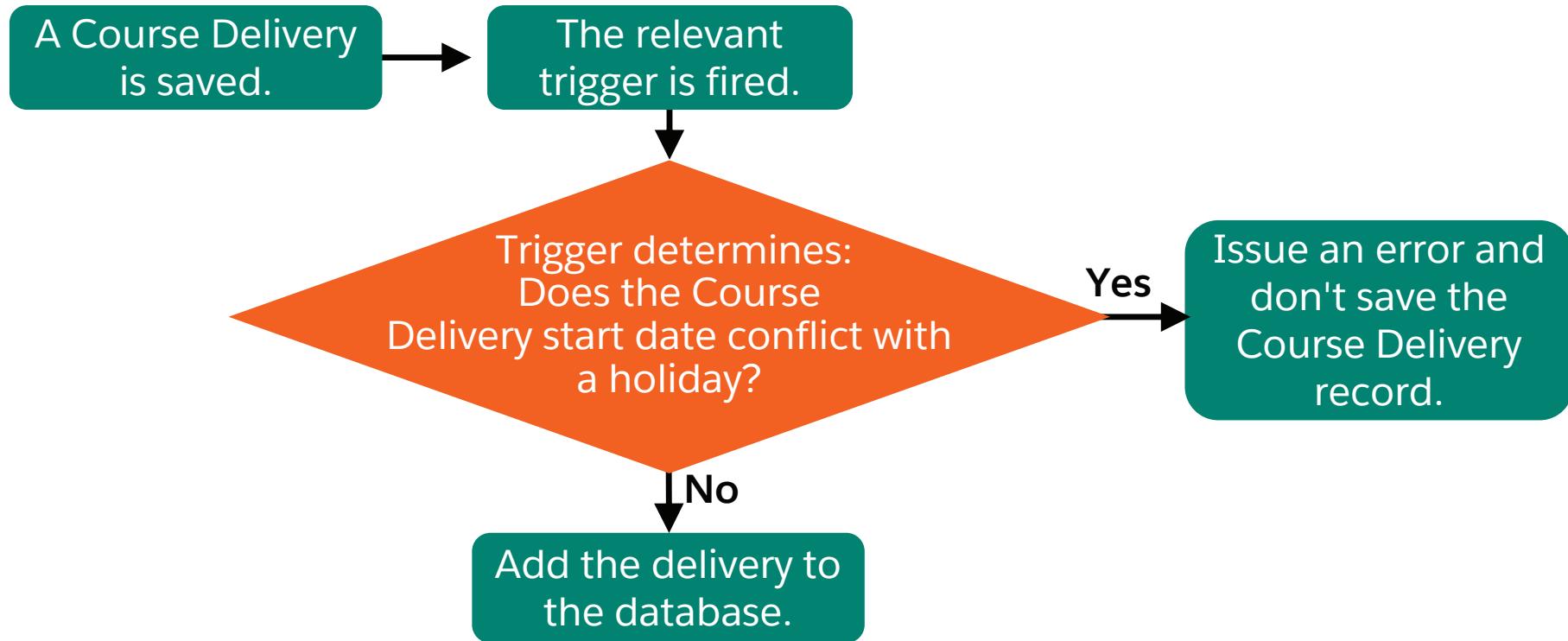
Specify the DML events that fire this trigger.

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

# PUT IT ALL TOGETHER: WRITE THE TRIGGER DEFINITION

226

salesforce



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



## 8-1: DEFINE A TRIGGER

**Goal:**

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

**Task:**

Define a trigger.



# MODULE AGENDA

228

salesforce

## MODULE 8: TRIGGER ESSENTIALS

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



# TRIGGER LOGIC CAN ACCESS THE TRIGGER CONTEXT

229

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

230

salesforce

```
1 trigger MyTrigger on
  MyObject__c (before insert, before update, after update) {
2   if (trigger.isBefore) {
3     if (trigger.isInsert) {
4       // Logic block 1
5     }
6     if (trigger.isUpdate) {
7       // Logic block 2
8     }
9   } else {
10     // Logic block 3
11   }
12 }
```

# WHAT IS IN Trigger.New AND Trigger.Old?

231

salesforce

```
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.

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

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

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

# WORKING WITH Trigger.new AND Trigger.oldMap

232

salesforce

```
1 trigger CourseDeliveryTrigger on Course_Delivery__c (before
    insert, before update, before delete, after insert,
    after update) {
2     if (trigger.isAfter) {
3         if (trigger.isUpdate) {
4             for (Course_Delivery__c cd :trigger.new) {
5                 Date oldDate =
6                     trigger.oldMap.get(cd.id).Start_Date__c;
7                 if (cd.Start_Date__c != oldDate) {
8                     // ... Do some logic
9                 }
10            }
11        }
12    }
```

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

233

salesforce

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.

```
1 trigger CourseDeliveryTrigger on Course_Delivery__c (before
  insert, before update, before delete, after insert,
  after update) {
  2   if (trigger.isAfter) {
  3     if (trigger.isUpdate) {
  4       for (Course_Delivery__c cd : trigger.new) {
  5         Date oldDate =
  6           trigger.oldMap.get(cd.id).Start_Date__c;
  7         if (cd.Start_Date__c != oldDate) {
  8           // ... Do some logic
  9         }
 10       }
 11     }
 12 }
```

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

234



```
1 trigger CourseDeliveryTrigger on Course_Delivery__c (before
  insert, before update, before delete, after insert,
  after update) {
2
3     for (Course_Delivery__c cd : trigger.new) {
4         if (//... Some condition) {
5             //prevent the invoking DML action from completing
6             cdaddError('This sObject cannot be saved.');
7     }
8 }
```

sObject.addError will  
prevent completion of  
the DML action.

Cela arrête l'opération immédiatement.

**Goal:**

Define the business logic of a trigger that only allows a Course Delivery to be saved if it 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

236

salesforce

- 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.
- 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.

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

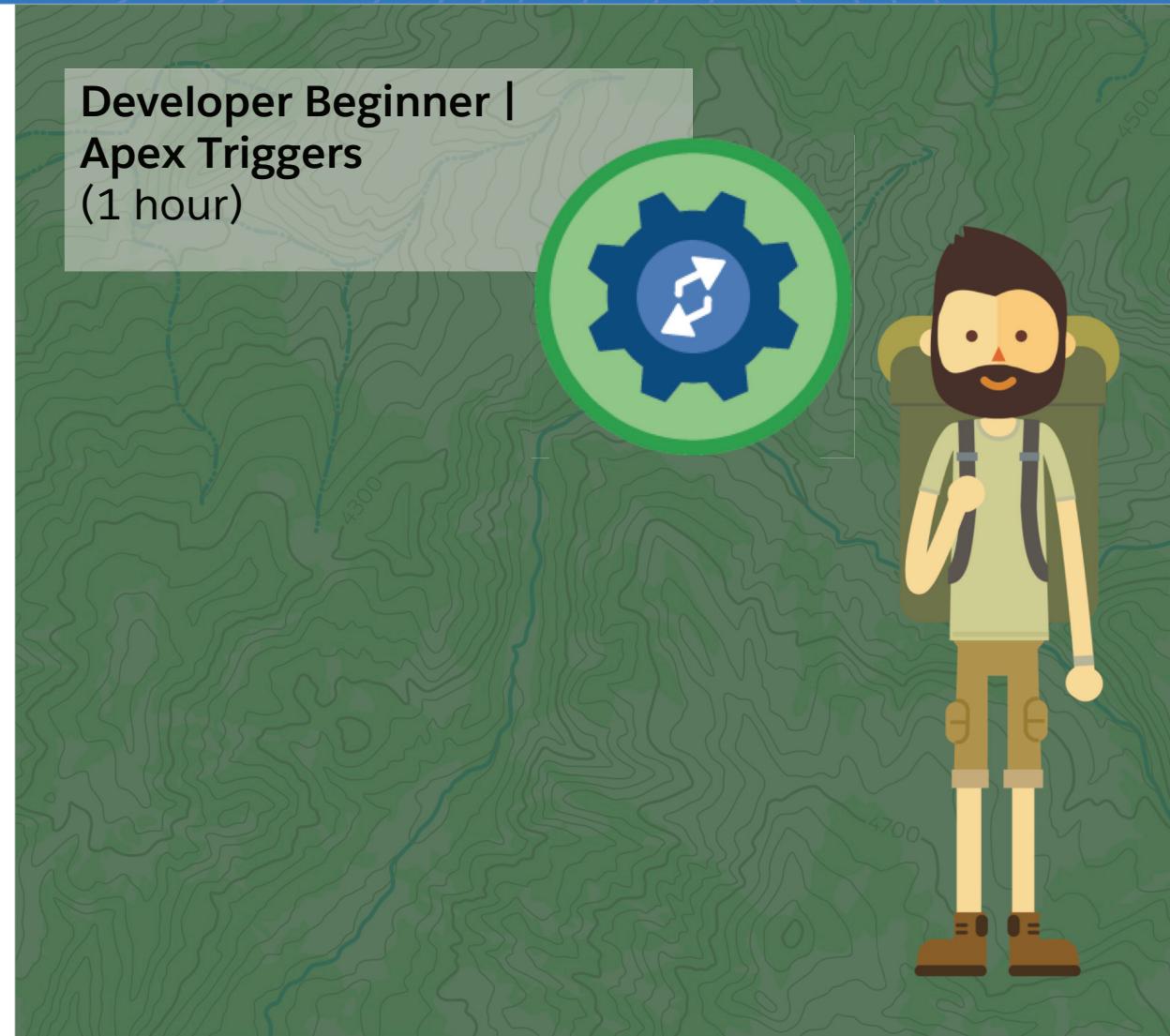


## KNOWLEDGE CHECK

237

salesforce

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.



# MODULE 9:

# APEX CLASS ESSENTIALS

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE



Jason Beck

Developer



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

241

salesforce

## 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?

242

salesforce

A package.

- Managed
- Unmanaged

Your org.

DEFINITION:



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

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 AGENDA

245

salesforce

### MODULE 9: APEX CLASS ESSENTIALS

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



# DEFINING AN APEX CLASS

246

salesforce

Access modifier: who can see this class?

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

```
1 public with sharing class MyClass {  
2     DataType memberVariable;  
3  
4     DataType memberProperty { get; set; }  
5  
6     public MyClass() {  
7         // ... Constructor logic  
8     }  
9  
10    public void memberMethod() {  
11        //... Method logic  
12    }  
13 }
```

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.

Access Modifier Keyword	Applied to a Class	Applied to a Class Member
global 	<ul style="list-style-type: none"> <li>Accessible to all Apex code everywhere</li> <li>Used to define code for asynchronous Apex and services (email, web)</li> </ul>	
Accessible en dehors de SalesForce		
public	Accessible within your application or namespace Accessible à l'intérieur de l'application	
protected 	Not available àoublier, cas d'héritage, très peu utilisé dans SF)	<p>Accessible to any:</p> <ul style="list-style-type: none"> <li>Inner classes in the defining Apex class</li> <li>Classes that extend the defining Apex class</li> </ul>
private 	<ul style="list-style-type: none"> <li>Applied to inner classes to make them accessible locally</li> <li>Can be applied to test classes</li> </ul>	<ul style="list-style-type: none"> <li>The default access modifier</li> <li>A private member is accessible only within the Apex class in which it is defined</li> </ul>

DEFINITION:



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

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

```
1 public class MyClass {  
2     public final Integer FINAL_VAR1;  
3     public final Integer FINAL_VAR2 = 2;  
4     public final static Integer STATIC_FINAL_VAR3;  
5     public final static Integer STATIC_FINAL_VAR4 = 4;  
6  
7     Static {  
8         STATIC_FINAL_VAR3 = 3;  
9     }  
10  
11     Public MyClass() {  
12         FINAL_VAR1 = 1;  
13     }  
14 }
```

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?

```

1A public with sharing class CourseDeliveryTriggerHandler {
2A
3A     public static void logicBlock1() {
4A         // ... Logic block 1
5A     }
6A
7A     private static void logicBlock2() {
8A         // ... Additional logic
9A     }
10A }
```



La classe appellée par le Trigger

En vrai, Private donc ça ne compile pas !

```

1B trigger CourseDeliveryTrigger on Course_Delivery__c (before insert, before update) {
2B     if (trigger.isBefore) {
3B         if (trigger.isInsert) {
4B             CourseDeliveryTriggerHandler.logicBlock1();
```



Le trigger appelant la classe dans son traitement



## 9-1: DEFINE AN APEX CLASS

**Goal:**

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

**Tasks:**

1. Create an Apex class.
2. Invoke the class from the trigger.
3. Test the trigger.



# MODULE AGENDA

251

salesforce

## MODULE 9: APEX CLASS ESSENTIALS

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



# REVIEWING DATA ACCESS

252

salesforce

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

FLS (Field Level Security)

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	6/7/2016	...
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?

... 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

254

salesforce

```
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?

```
1A public with sharing class QueryClass {  
2A     public List<Account> fetchAccounts() {  
3A         return [SELECT Id FROM Account];  
4A     }  
5A }
```



```
1B public without sharing class QueryClass {  
2B     public List<Account> fetchAccounts() {  
3B         return [SELECT Id FROM Account];  
4B     }  
5B }
```



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?

256



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

**When this class is  
invoked by...**

**The Sharing Model is...**

An anonymous  
block

Respected

A trigger

Ignored

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

Another class

- Respected, if the invoking class is "with sharing"
- Ignored, otherwise





## KEY TAKEAWAYS

257

salesforce

- 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.



## KNOWLEDGE CHECK

258

salesforce

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

260

salesforce

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

261

salesforce

### MODULE 10: THE SAVE ORDER OF EXECUTION AND APEX TRANSACTIONS

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





## 10-1: EXPLORE THE IMPLICIT FIRING OF TRIGGERS



262

salesforce

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.



# THE SAVE ORDER OF EXECUTION

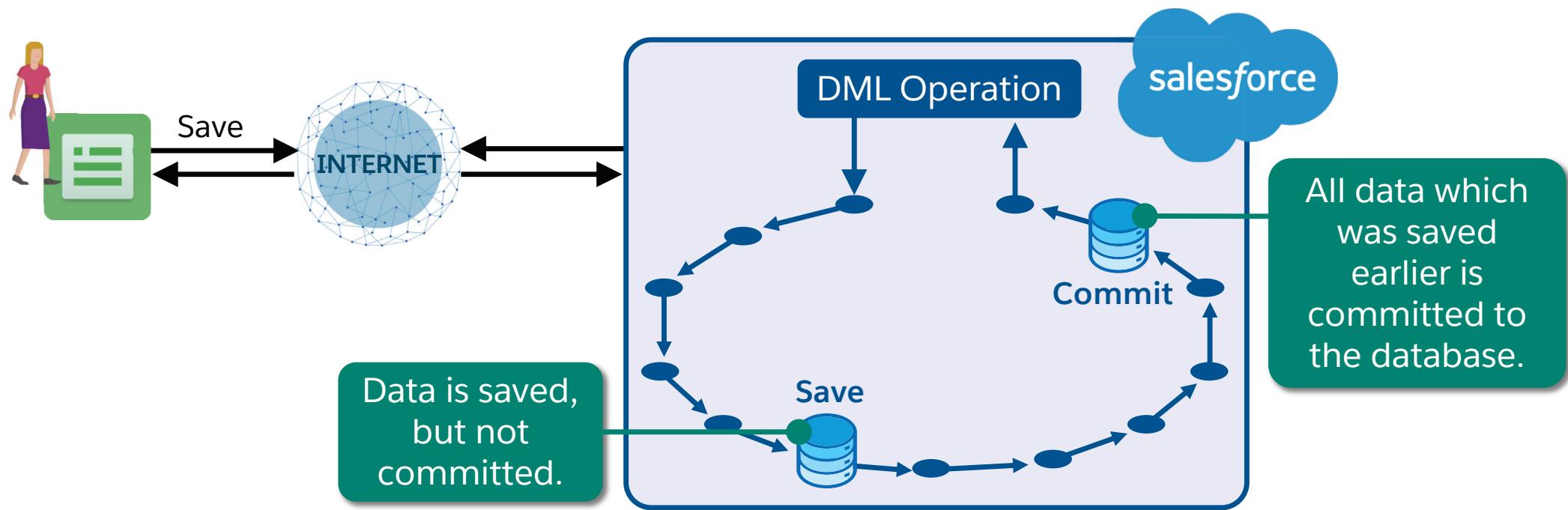
263

salesforce

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?

264



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
- c) Visualforce pages

# WHAT HAPPENS BEFORE THE SAVE TO THE DATABASE?

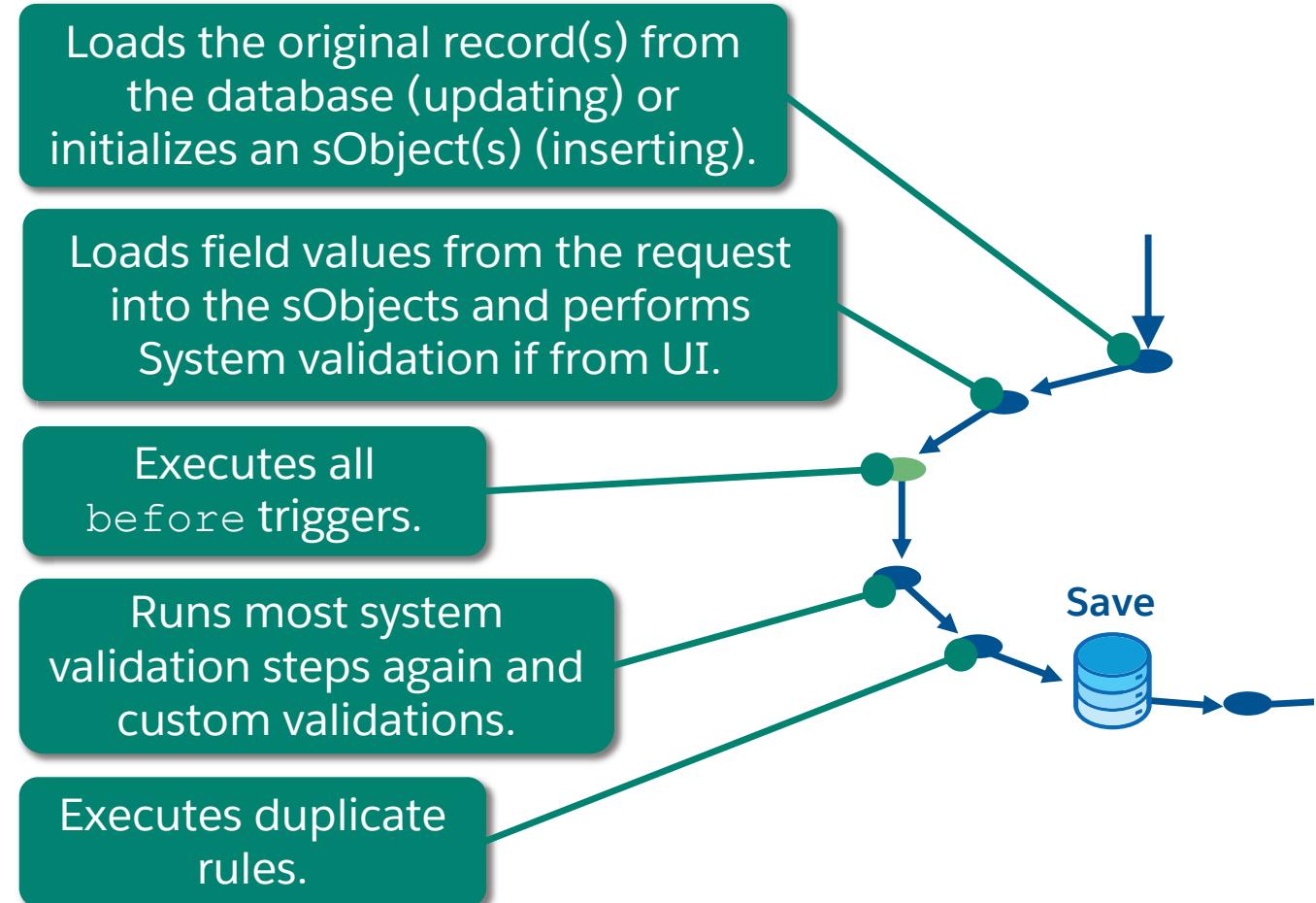
265

salesforce

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.

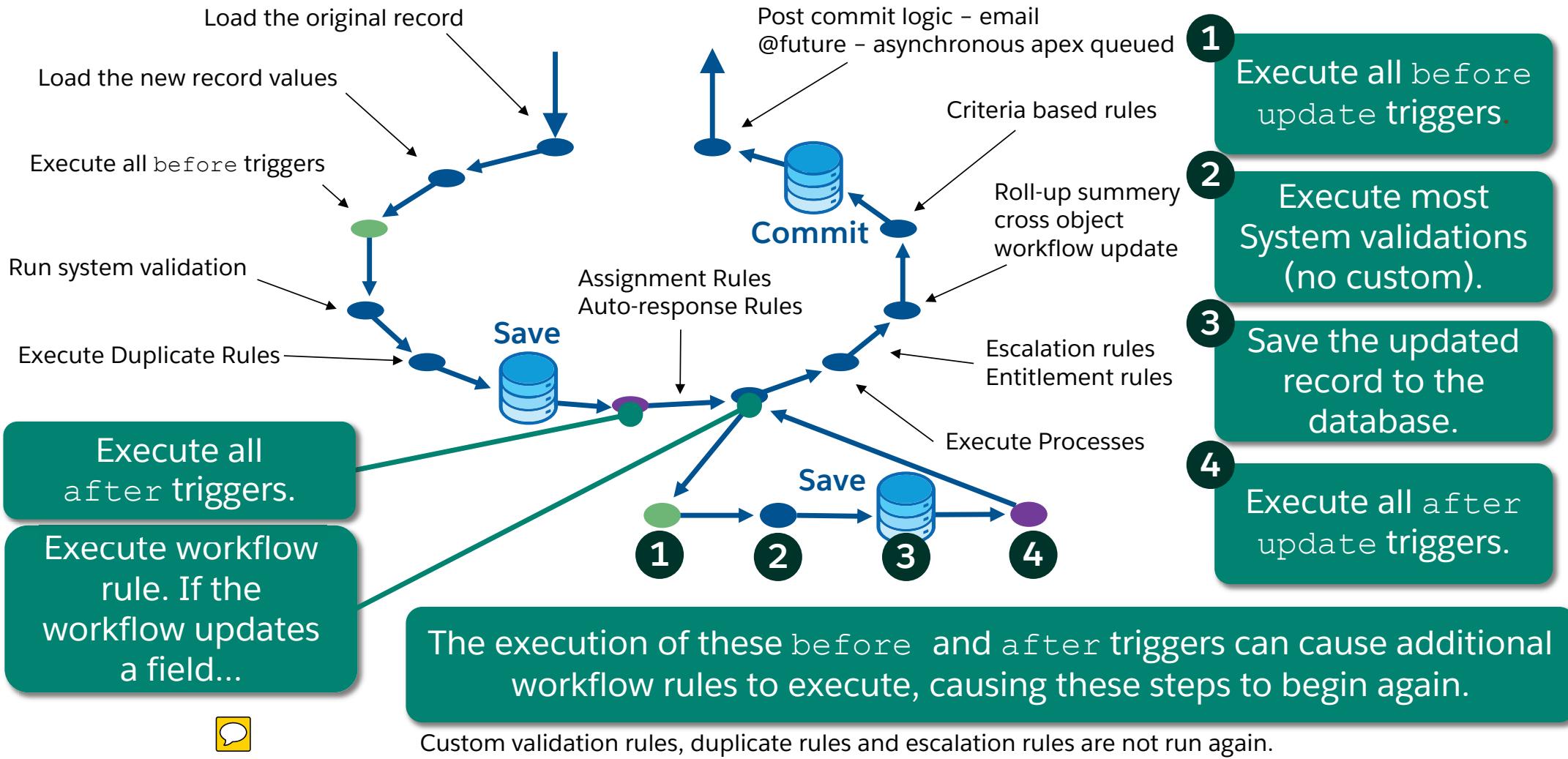
What event is likely to have caused the data to become invalid?



# WHAT HAPPENS AFTER THE SAVE TO THE DATABASE?

## PART 1: TRIGGERS AND WORKFLOW RULES

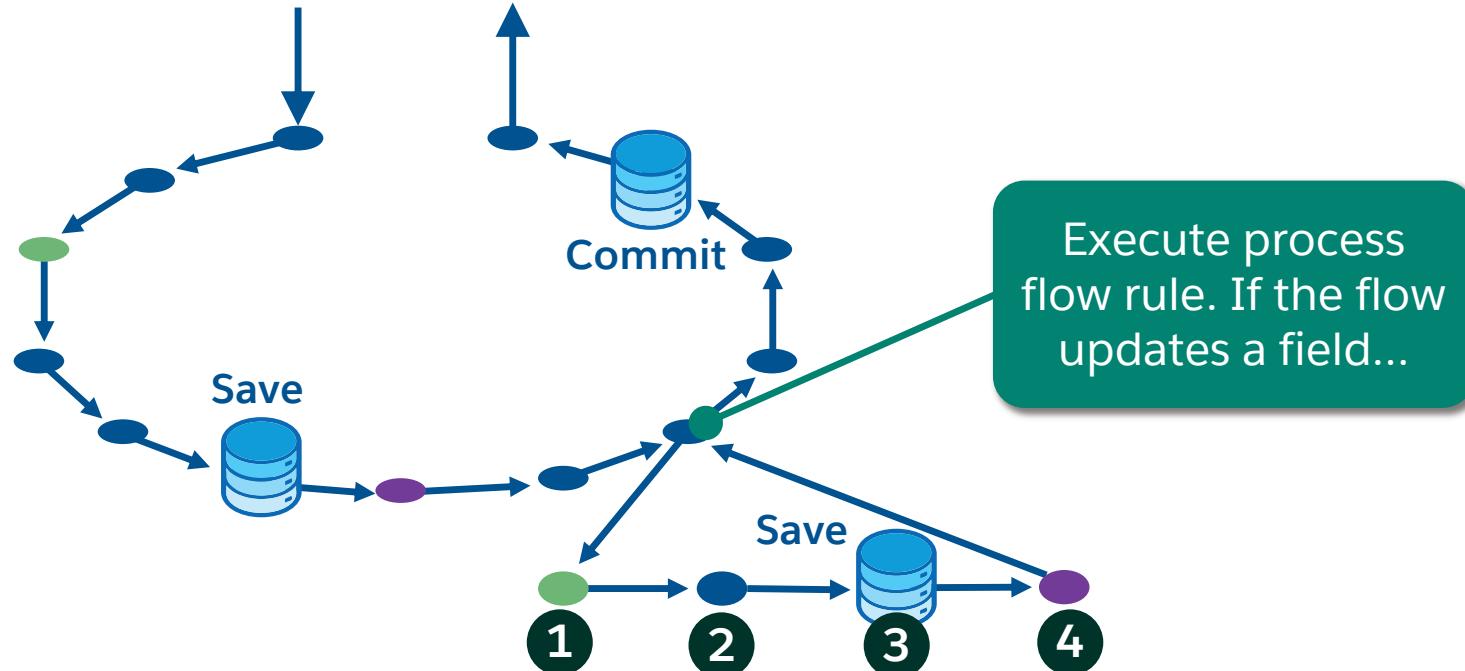
266



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

267

salesforce



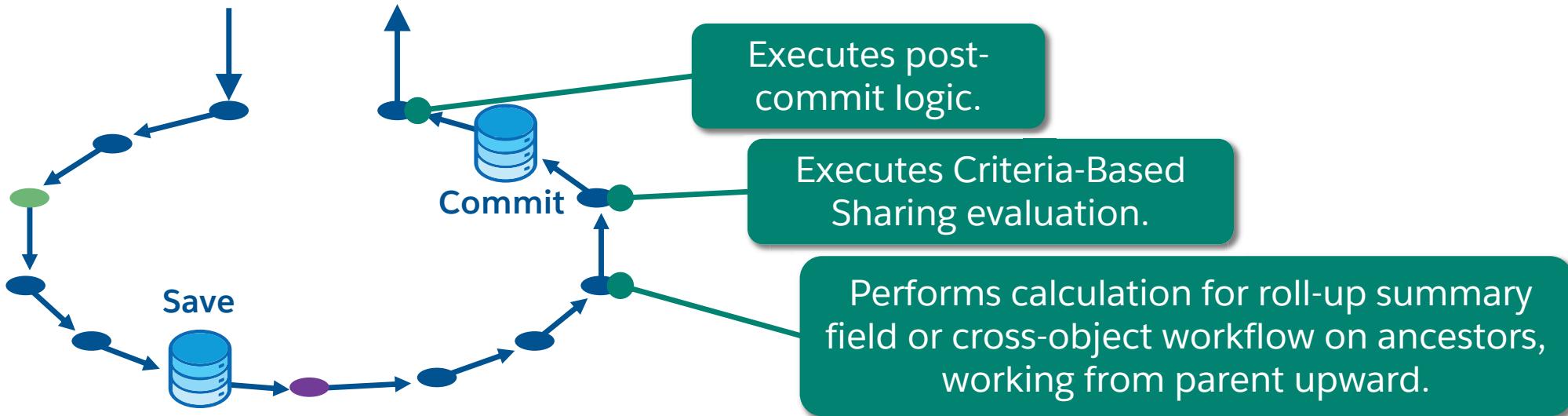
- 1 Execute all before update triggers.
- 2 Execute most System validations (no custom).
- 3 Save the updated record to the database.
- 4 Execute all after update triggers.

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

268

salesforce

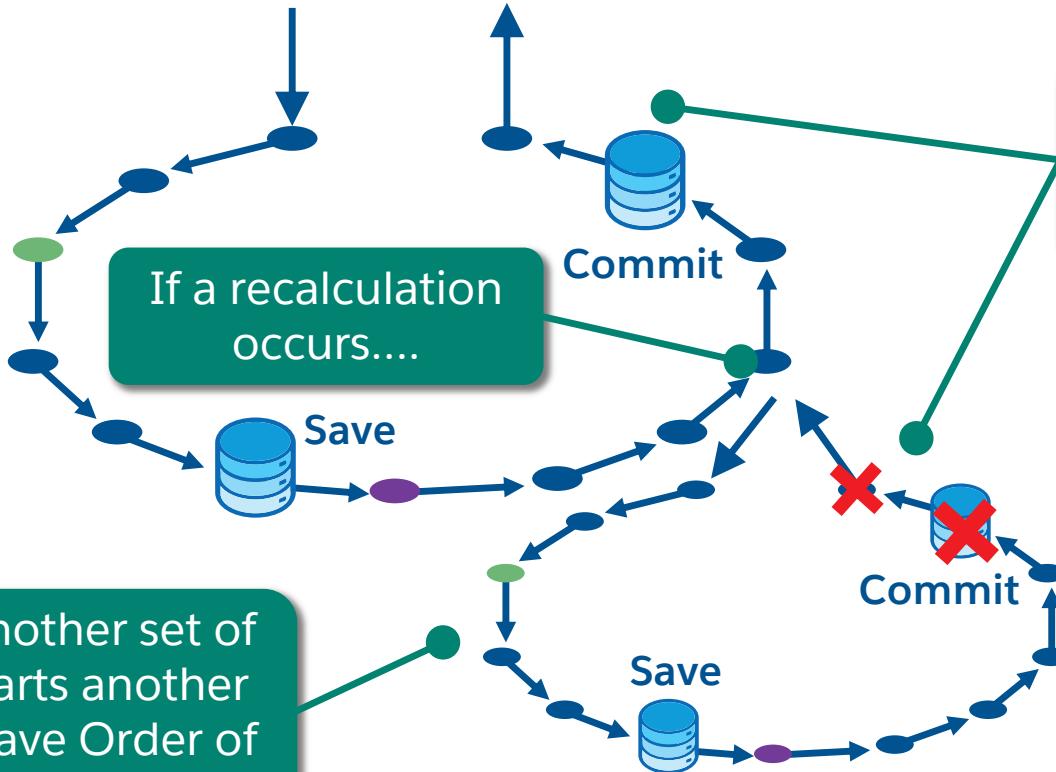


What are the consequences of calculating a roll-up summary field?

# WHAT HAPPENS WHEN ANOTHER DML SAVE OPERATION IS PERFORMED?

269

salesforce



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

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?

270

salesforce

- 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?

271

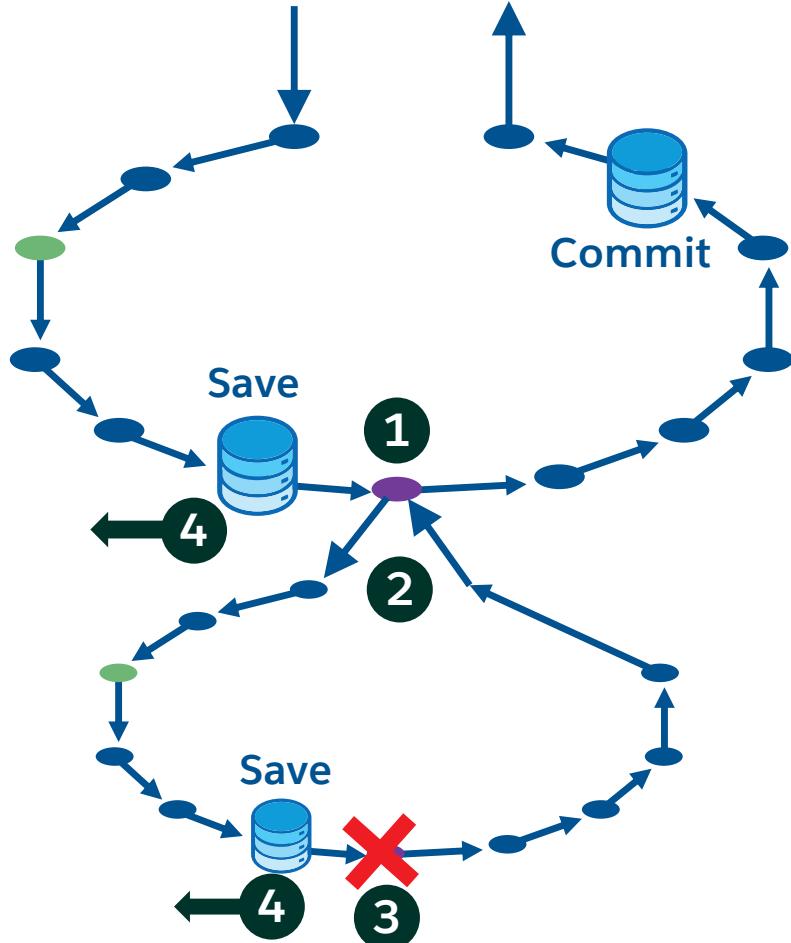


- A system validation?
- A validation rule?
- A trigger?
- A workflow rule?
- A process/flow?

# WHAT HAPPENS IF THERE IS A FAILURE?

272

salesforce



- 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?

273



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.

```
1  ...
2  //within a trigger
3  //don't allow courses with students to be cancelled.
4  for (Course__c c : coursesWithStudents) {
5      caddError('Course has enrolled students');
6  }
7  ...
```

**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 AGENDA

275

salesforce

### 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?

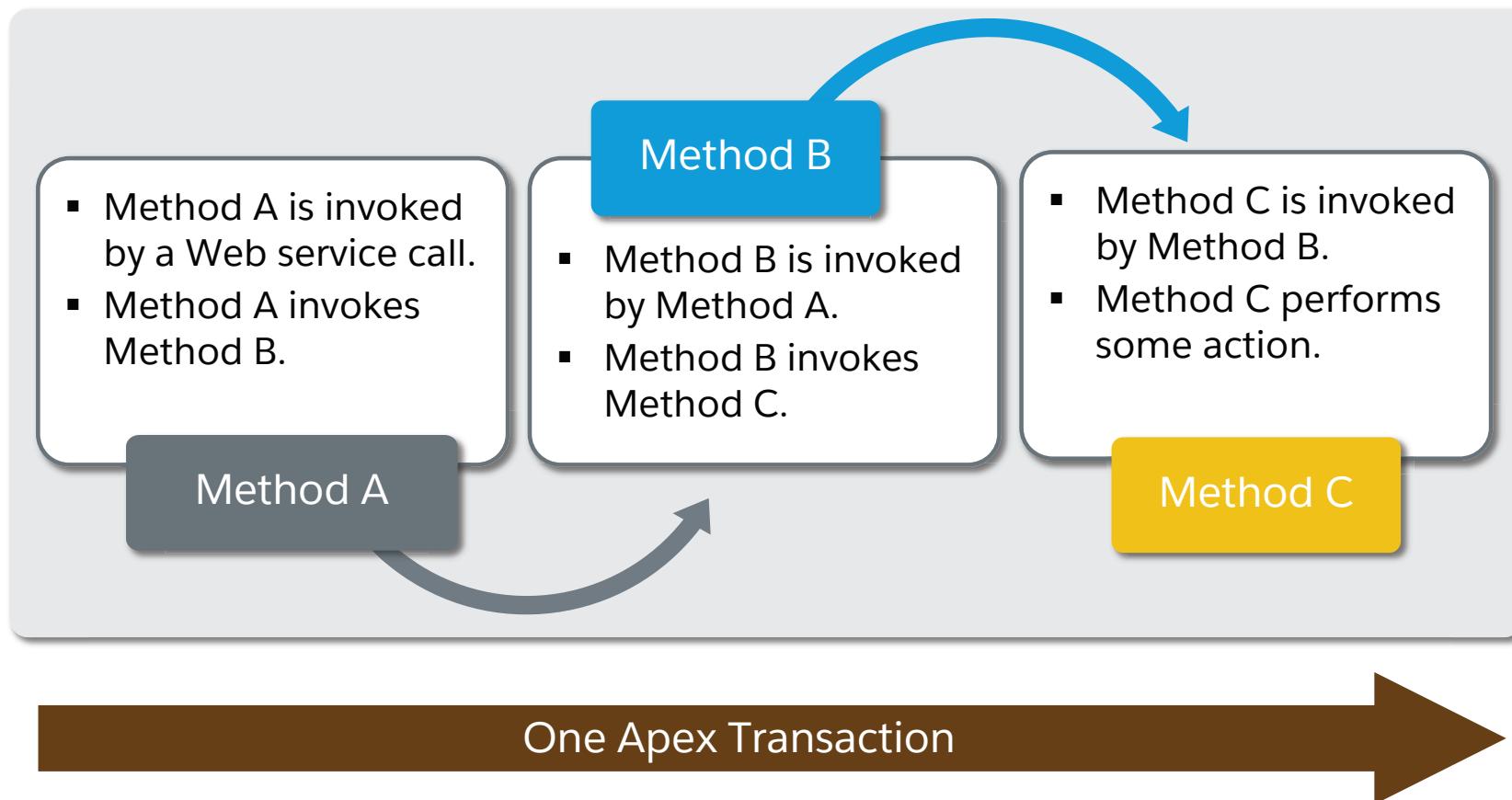
276

salesforce

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!

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



ALERT:

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

# APEX TRANSACTIONS THAT INVOKE TRIGGERS: PART 1

278



## One Apex Transaction

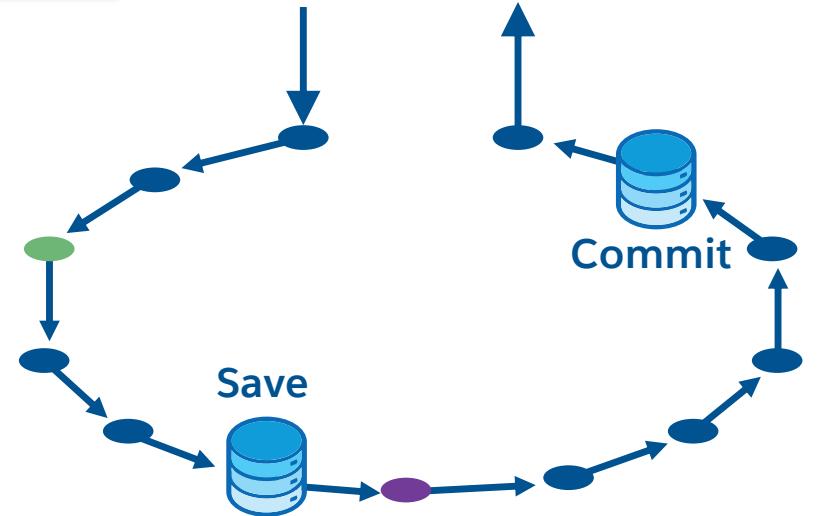
Execute anonymous  
invokes  
ManageContact.load();

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.

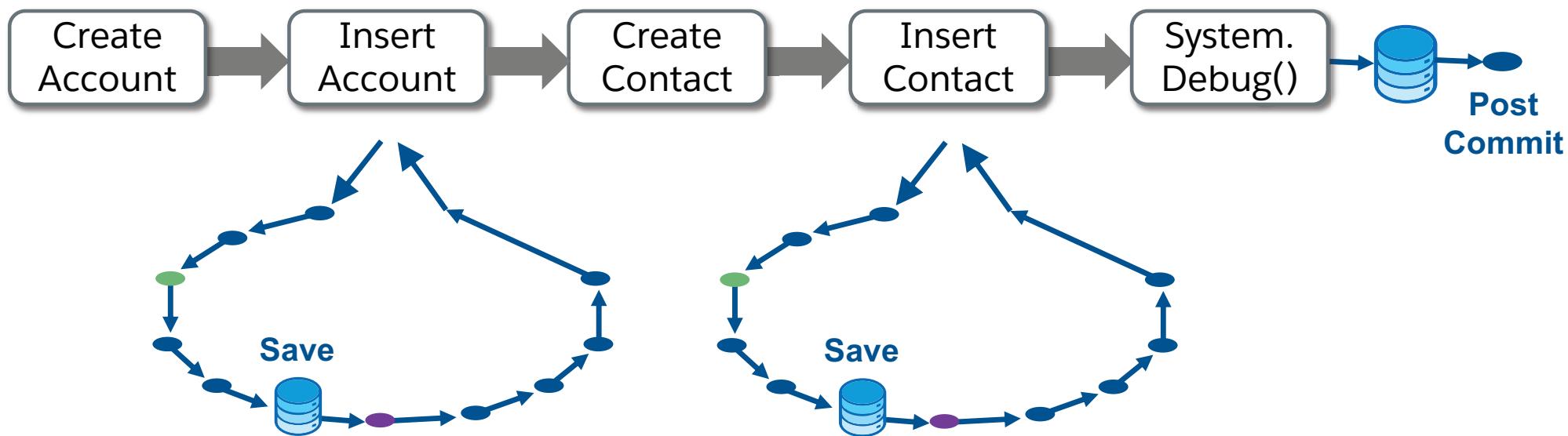


# APEX TRANSACTIONS THAT INVOKE TRIGGERS: PART 2

279



One Apex Transaction

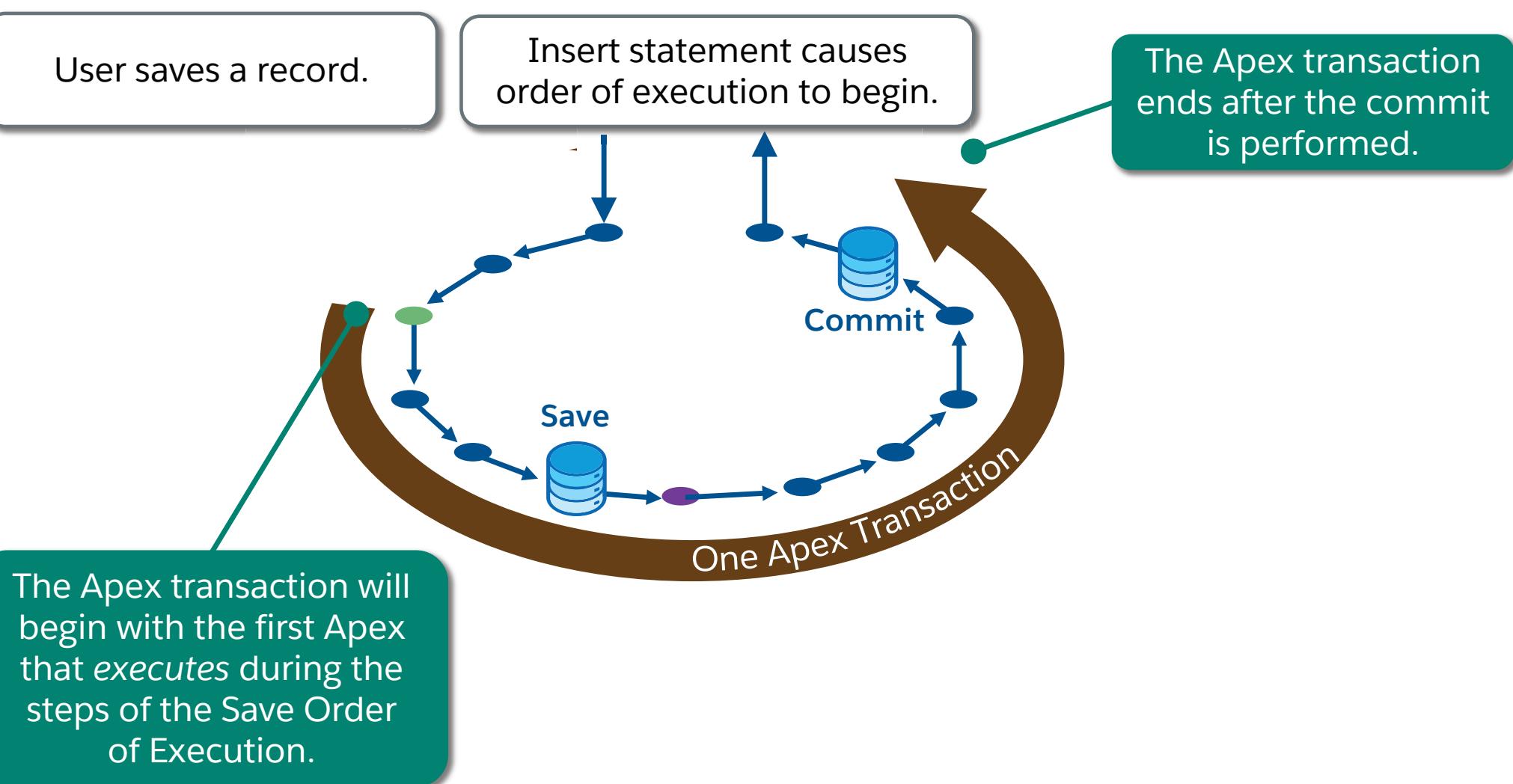


```
1 Account newAccount = new Account(Name = 'Acme');
2 insert newAccount;
3 Contact newContact = new Contact(LastName = 'Manning', AccountId = newAccount.Id);
4 insert newContact;
5 System.debug('About to commit...');
```

# APEX TRANSACTIONS THAT BEGIN WITH TRIGGERS

280

salesforce





# WHAT WILL THIS APEX TRANSACTION INCLUDE?

281

salesforce

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?

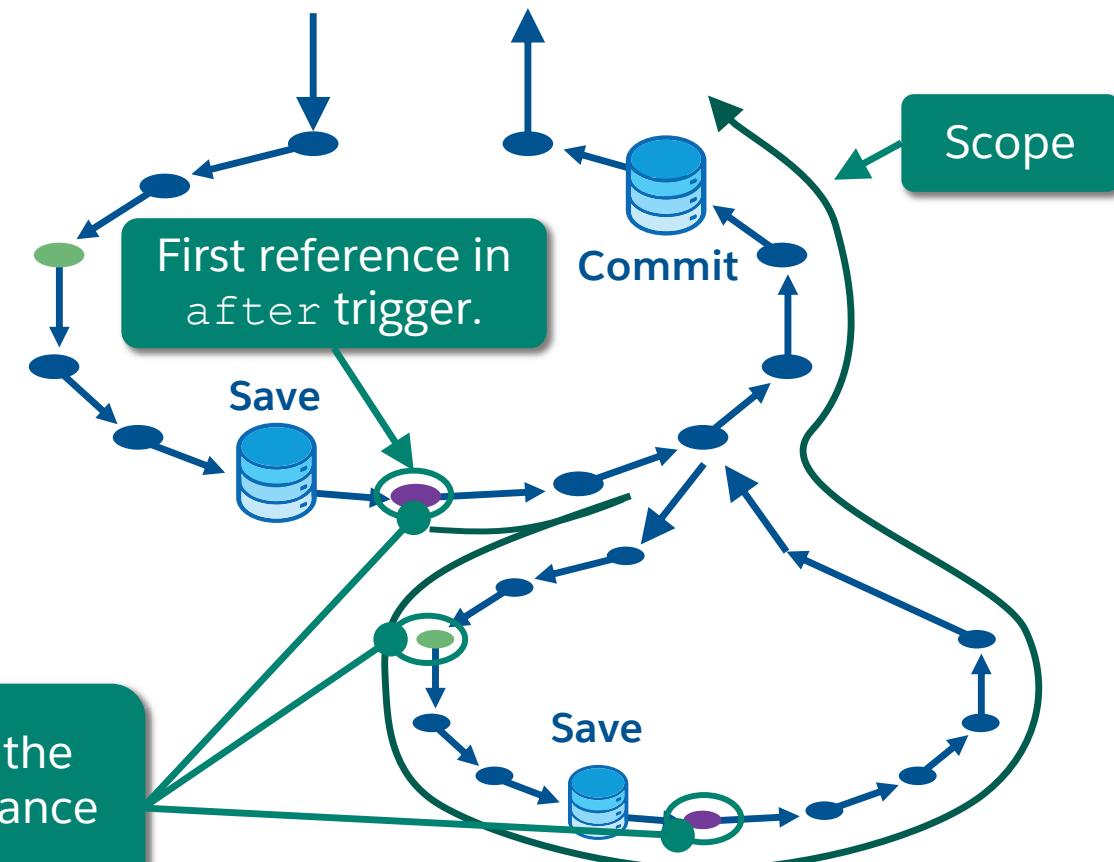
A static attribute within a class:

- 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.



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

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



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.

**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.



- 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.

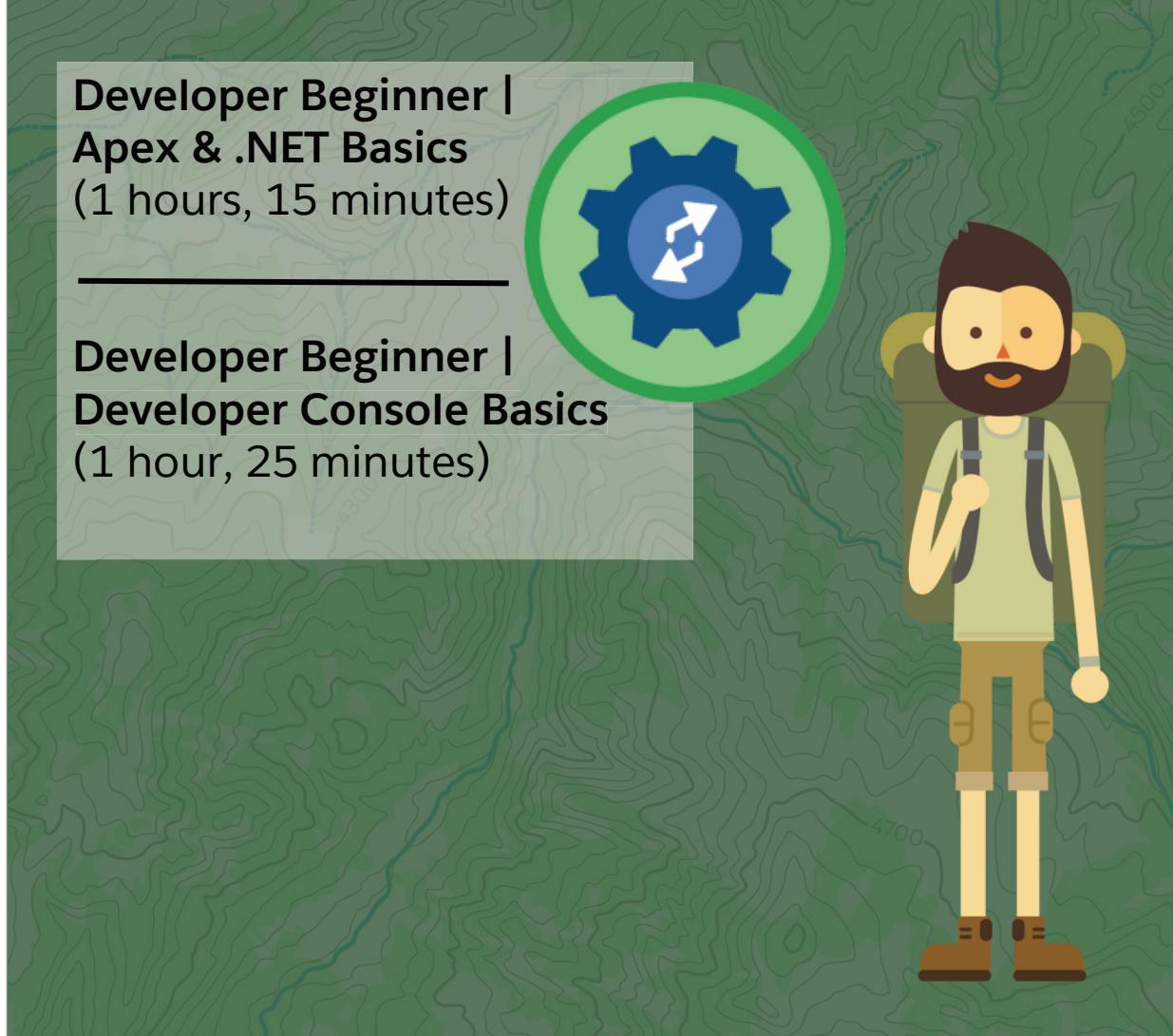
All Apex code should consider the interaction between all events that are part of the order of execution.



**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



Jason Beck

Beginning  
Developer



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 AGENDA

288

salesforce

## MODULE 11: TESTING ESSENTIALS

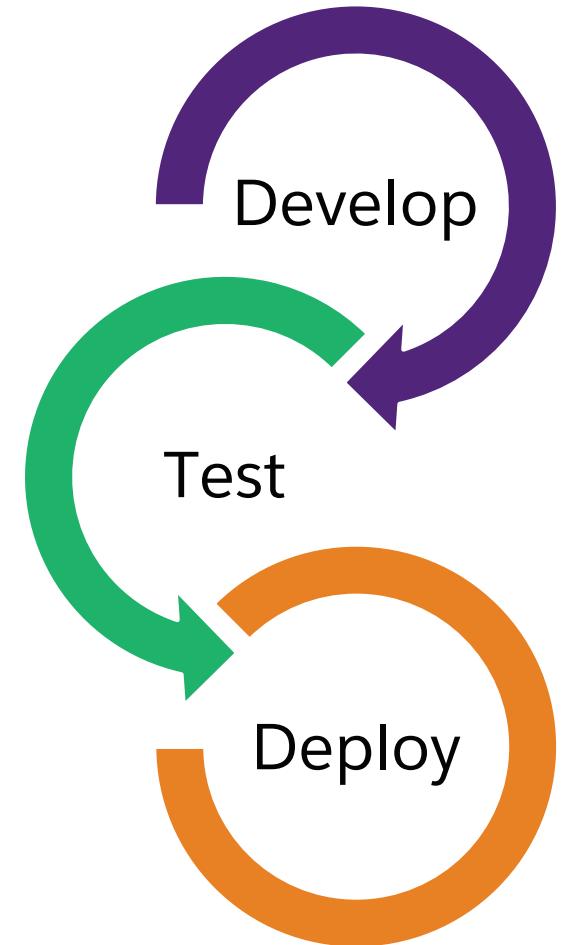
- **Describing Apex's Testing Framework**
- Creating Test Data
- Writing and Running an Apex Test





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:

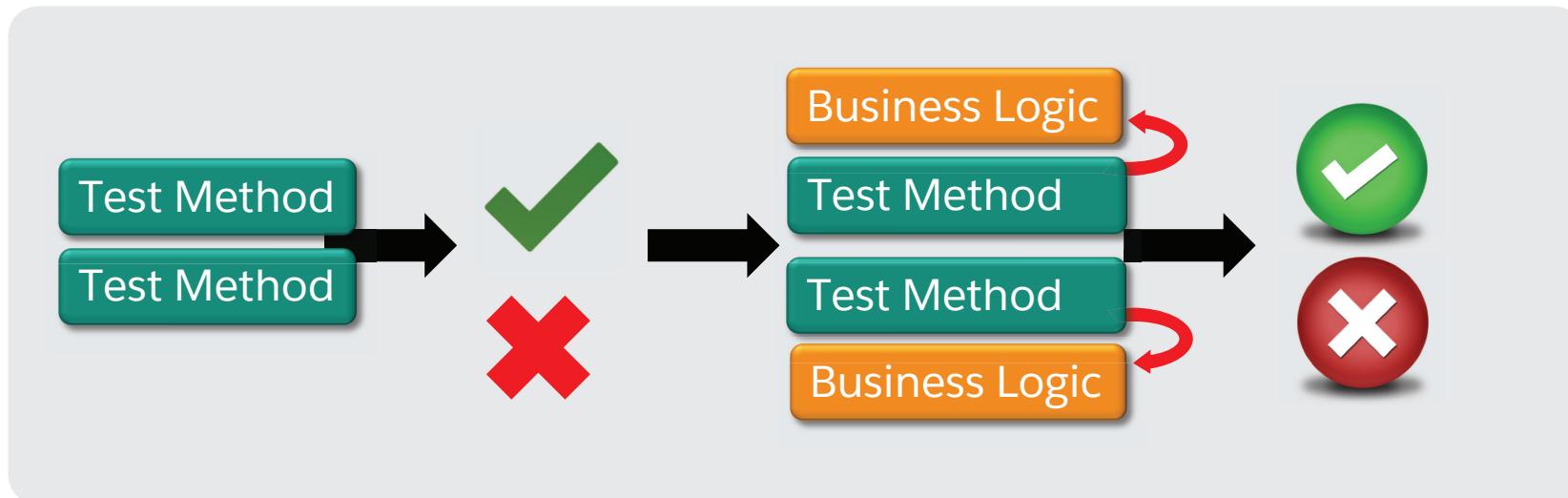
- 75% of Apex code must be executed successfully by test methods. 
- Every Apex trigger must have some coverage in test methods, even though every line of every trigger does not have to be executed by a test. 
- Every Apex test method must execute without throwing any uncaught exceptions or exceeding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



# THE PLATFORM INCLUDES A TESTING FRAMEWORK

290

salesforce



A developer writes test methods inside test classes.

Salesforce receives a specific command to run tests.

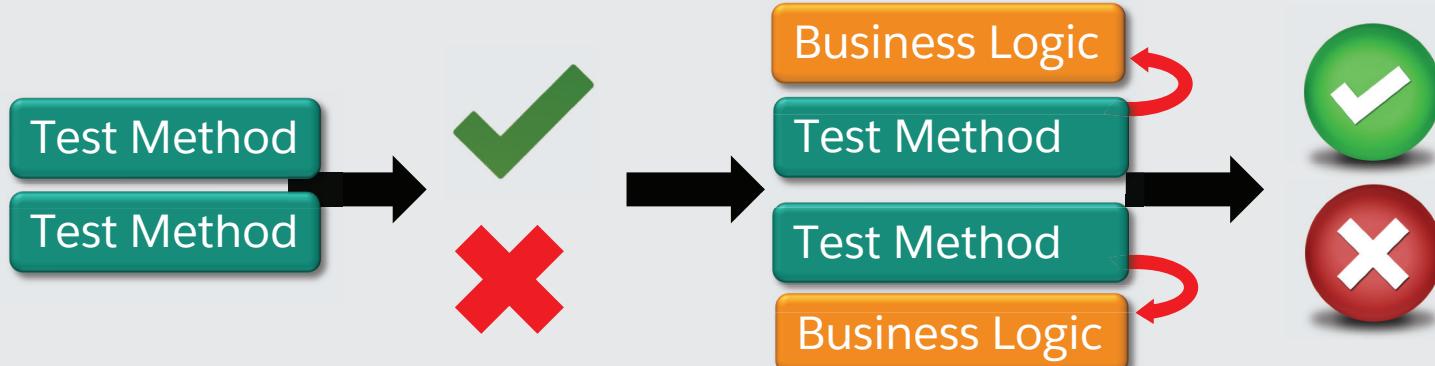
Salesforce runs test methods and any code they invoke.

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

# WHAT IS AN APEX TEST METHOD?

291

salesforce



```
1 @isTest
2 private class MyBusinessLogicClass_Test {
3     private static testMethod void myBusinessLogicTest() {
4         //... test Apex classes and triggers
5     }
6 }
```

DEFINITION:



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



# MODULE AGENDA

292

salesforce

## MODULE 11: TESTING ESSENTIALS

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

Trailhead - Dévl débutant

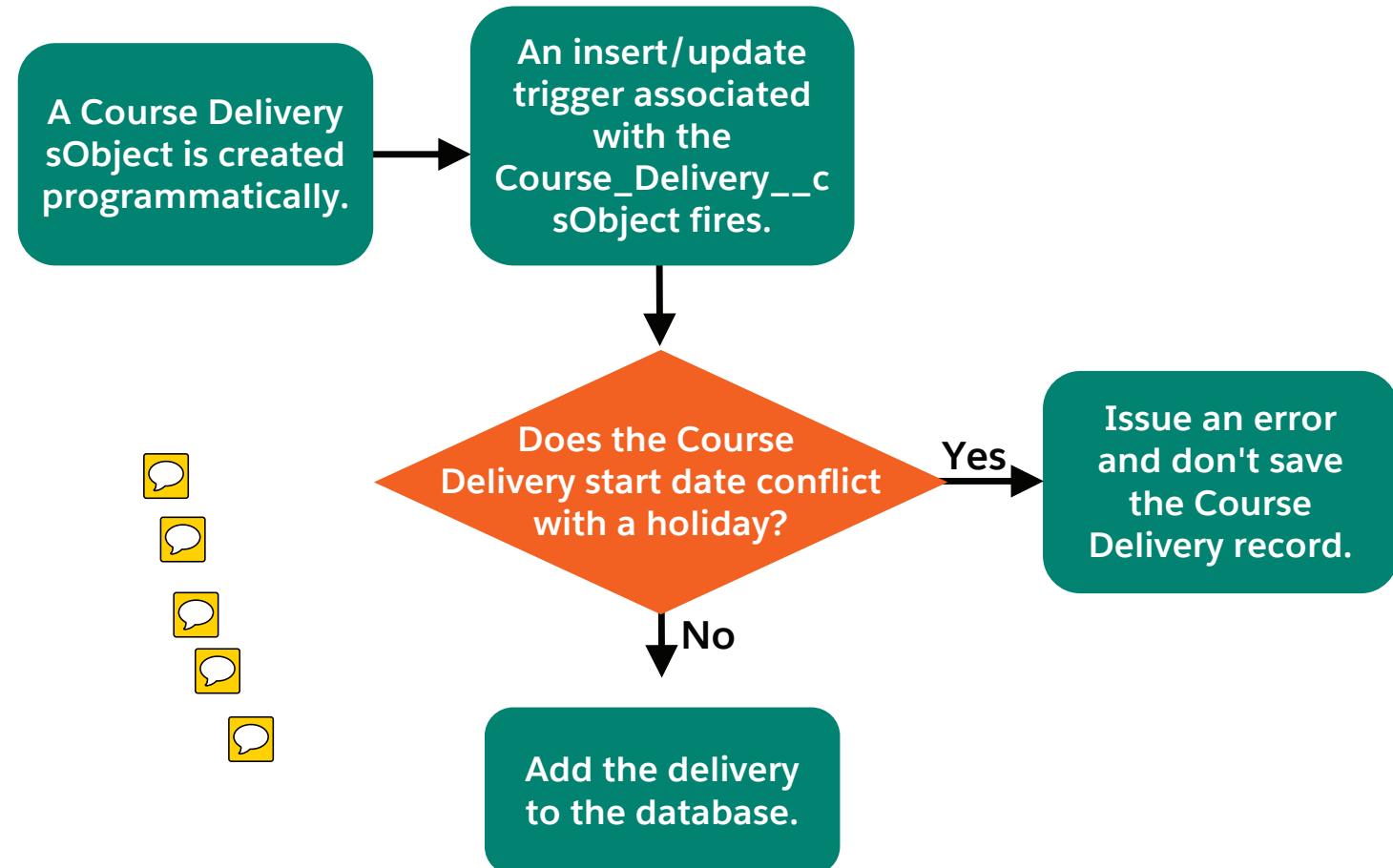
[https://trailhead.salesforce.com/fr/trails/force\\_com\\_dev\\_beginner/modules/apex\\_testing/units/apex\\_testing\\_intro](https://trailhead.salesforce.com/fr/trails/force_com_dev_beginner/modules/apex_testing/units/apex_testing_intro)



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

294

salesforce



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 name =  
'Test\_Holidays'

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

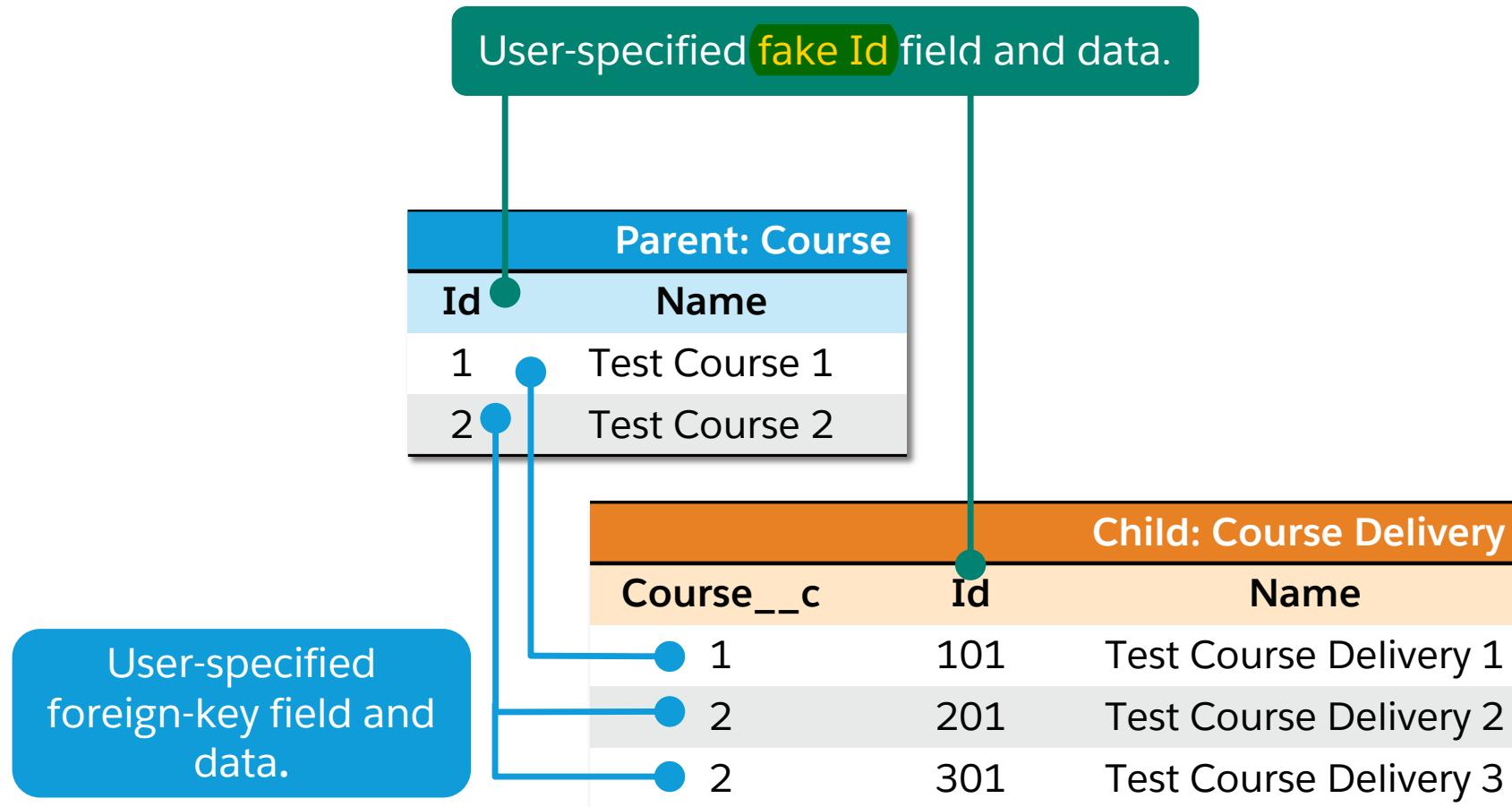
First line must be column names.

CSV file

# LOADING RELATED DATA USING A STATIC RESOURCE

295

salesforce



# CREATING TEST DATA IN A TEST FACTORY CLASS VS. METHOD

296

salesforce

- 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?

```
1A @isTest
2A public class MyTestDataFactory {
3A     public static void insertTestAccounts() {
4A         //... create and insert Accounts
5A     }
6A     public static void insertTestContacts() {
7A         //... create and insert Contacts
8A     }
9A     public static void insertTestHolidays() {
10A        //... create and insert Holidays
11A    }
12A }
```

```
1B @isTest
2B private class MyBusinessLogicClass_Test {
3B     @testSetup
4B     private static void myTestDataSetupMethod() {
5B         //... create and insert Accounts
6B         //... create and insert Contacts
7B         //... create and insert Holidays
8B         //... Etc...
9B     }
10B    private static testMethod void myTestMethod1() {
11B        //... Use the test data created above
12B    }
13B    private static testMethod void myTestMethod2() {
14B        //... Also uses the test data created above
15B    }
16B }
```

# DOES TEST DATA PERMANENTLY CHANGE THE DATABASE?

297

salesforce

```
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? <sup>1</sup>
2. What number will the debug statement on Line 14 print? <sup>2</sup>
3. What number will the debug statement on Line 18 print? <sup>1</sup>
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 :



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?

```
1 @isTest(seeAllData = true) 
2 private class MyBusinessLogicClass_Test {
3     private static testMethod void myBusinessLogicTest() {
4         List<Holiday> holidays = [SELECT ... ];
5     }
6 }
```



## 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

300

salesforce

## MODULE 11: TESTING ESSENTIALS

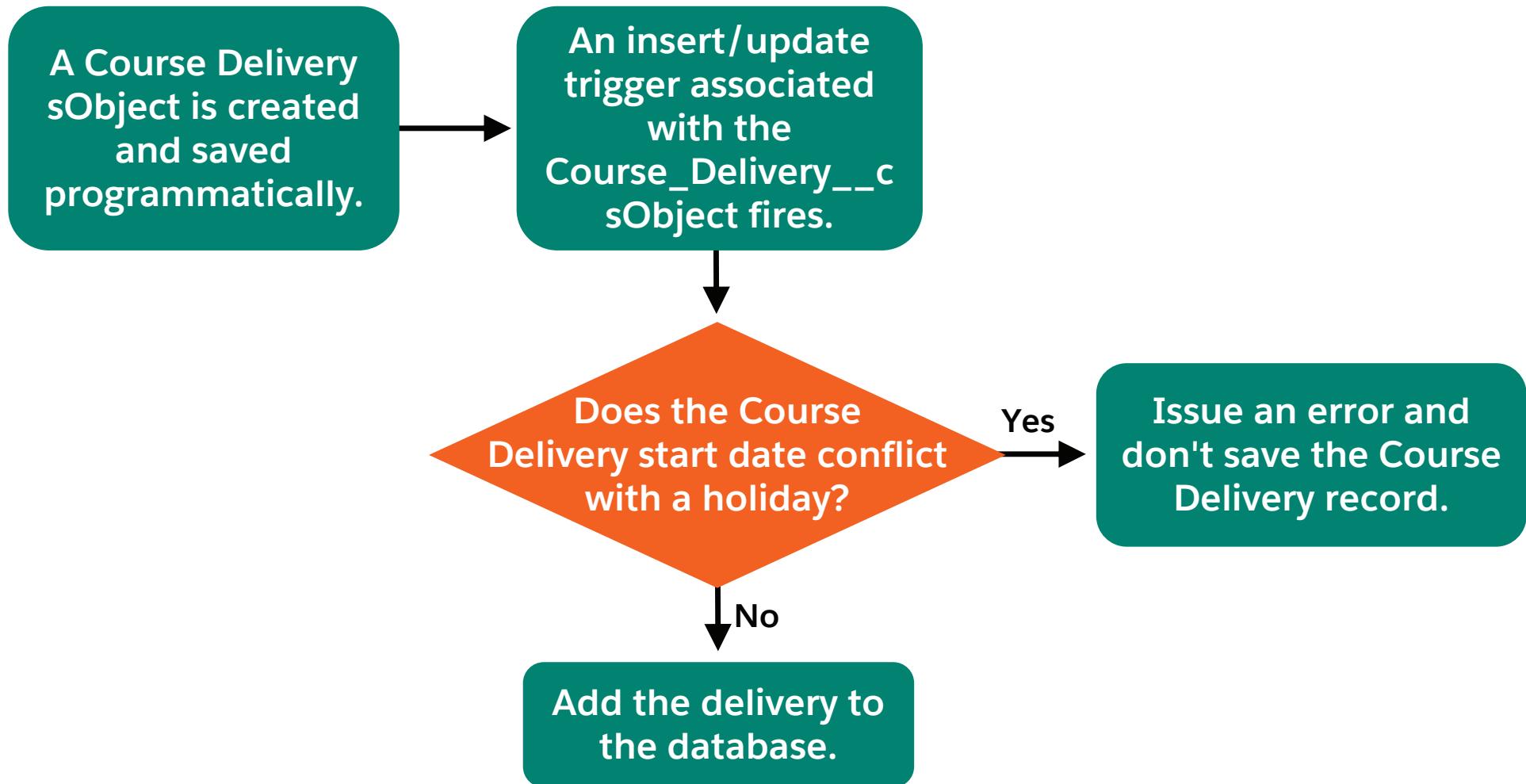
- Describing Apex's Testing Framework
- Creating Test Data
- **Writing and Running an Apex Test**



# WHAT EXPECTED OUTCOMES DO YOU NEED TO TEST?

301

salesforce



# AN APEX TEST METHOD INSIDE AN APEX TEST CLASS

302

salesforce

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.

```
1  @isTest
2  private class MyBusinessLogicClass_Test {
3      @testSetup
4      private static void myTestDataFactory() {
5          //... Create local test data
6      }
7      private static testMethod void myBusinessLogicTest() {
8          //... test saved Apex code
9      }
10 }
```

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

303

salesforce

```
1  @isTest
2  private class MyBusinessLogicClass_Test {
3      private static testMethod void myBusinessLogicTest() {
4          // create a list of 10 valid Courses
5          // insert the list of 10 valid Courses
6          List<Course__c> courses = [SELECT ...]; // retrieve courses that were just inserted
7          Integer numberCoursesInserted = courses.Size();
8
9          //Use assert methods to verify that 10 courses were inserted
10         System.Assert(numberCoursesInserted == 10);
11
12         //An alternative
13         //System.AssertEquals(10, numberCoursesInserted, 'ERROR: Expected 10 Courses');
14     }
15 }
```

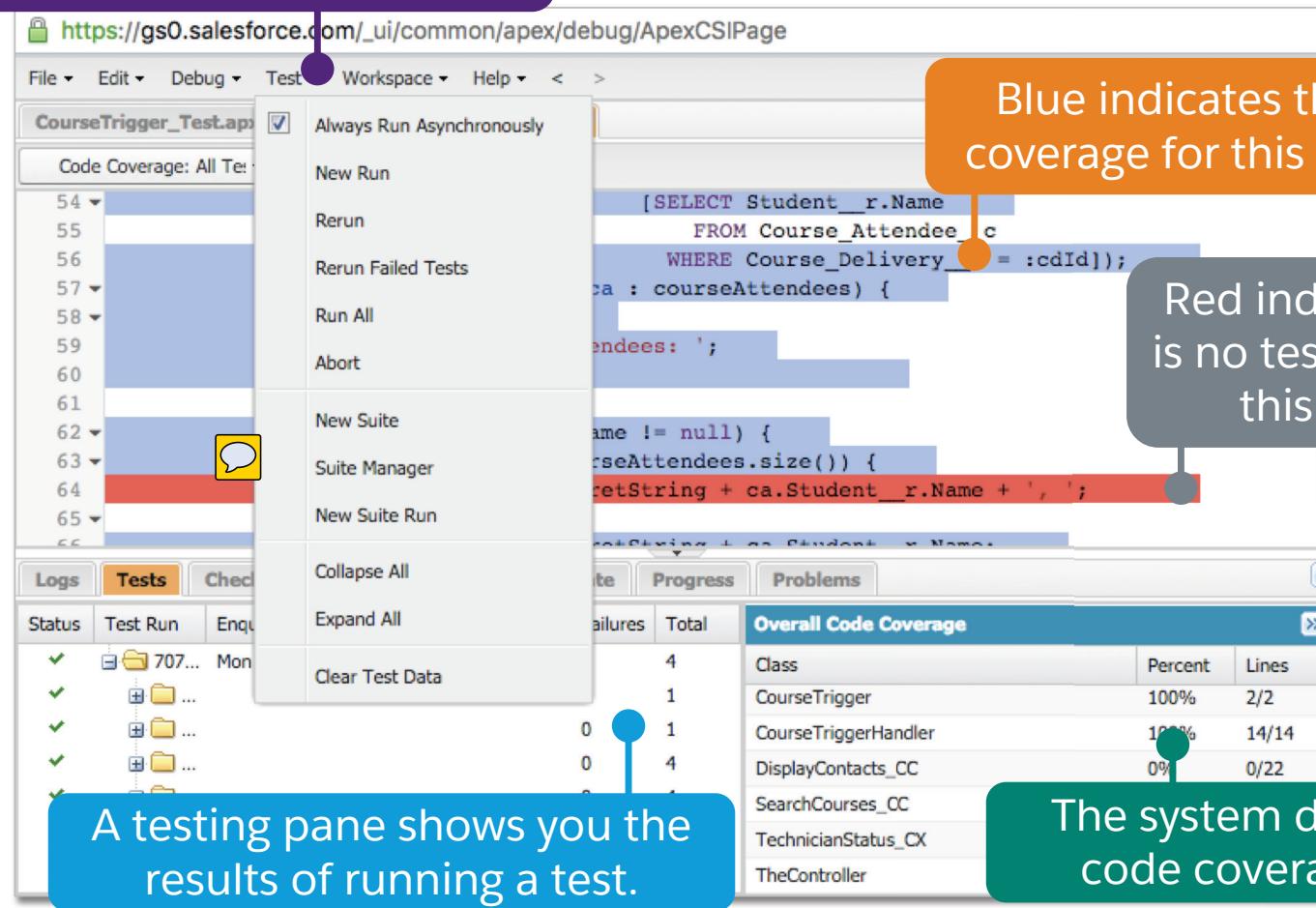
If this condition evaluates to false, a System.AssertException is raised.

# RUNNING A TEST IN THE DEVELOPER CONSOLE

304

salesforce

The test menu lets you choose which tests to run.



The screenshot shows the Salesforce Developer Console interface. The top navigation bar includes 'File', 'Edit', 'Debug', 'Test' (which is highlighted), 'Workspace', and 'Help'. The 'Test' menu is open, showing options: 'Always Run Asynchronously' (checked), 'New Run', 'Rerun', 'Rerun Failed Tests', 'Run All', 'Abort', 'New Suite', 'Suite Manager', 'New Suite Run', 'Collapse All', and 'Expand All'. Below the menu is a code editor with Apex code. The code is annotated with color highlights: blue for lines with coverage and red for lines without. A callout bubble points to a blue line with the text 'Blue indicates that there is coverage for this line of code.' A callout bubble points to a red line with the text 'Red indicates that there is no testing coverage for this line of code.' At the bottom of the code editor is a 'Logs' tab, an active 'Tests' tab, and a 'Check' tab. The 'Tests' tab shows a table of test results with columns 'Status', 'Test Run', 'Enq...', 'Failures', and 'Total'. The table shows 0 failures and 1 total test run. A callout bubble points to this table with the text 'A testing pane shows you the results of running a test.' To the right of the code editor is a 'Overall Code Coverage' table with columns 'Class', 'Percent', and 'Lines'. The table shows coverage for several classes: CourseTrigger (100% coverage, 2/2 lines), CourseTriggerHandler (100% coverage, 14/14 lines), DisplayContacts\_CC (0% coverage, 0/22 lines), SearchCourses\_CC (0% coverage, 0/22 lines), TechnicianStatus\_CX (0% coverage, 0/22 lines), and TheController (0% coverage, 0/22 lines). A callout bubble points to this table with the text 'The system determines the code coverage per class.'

Class	Percent	Lines
CourseTrigger	100%	2/2
CourseTriggerHandler	100%	14/14
DisplayContacts_CC	0%	0/22
SearchCourses_CC	0%	0/22
TechnicianStatus_CX	0%	0/22
TheController	0%	0/22



**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

306

salesforce

- 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.



## KNOWLEDGE CHECK

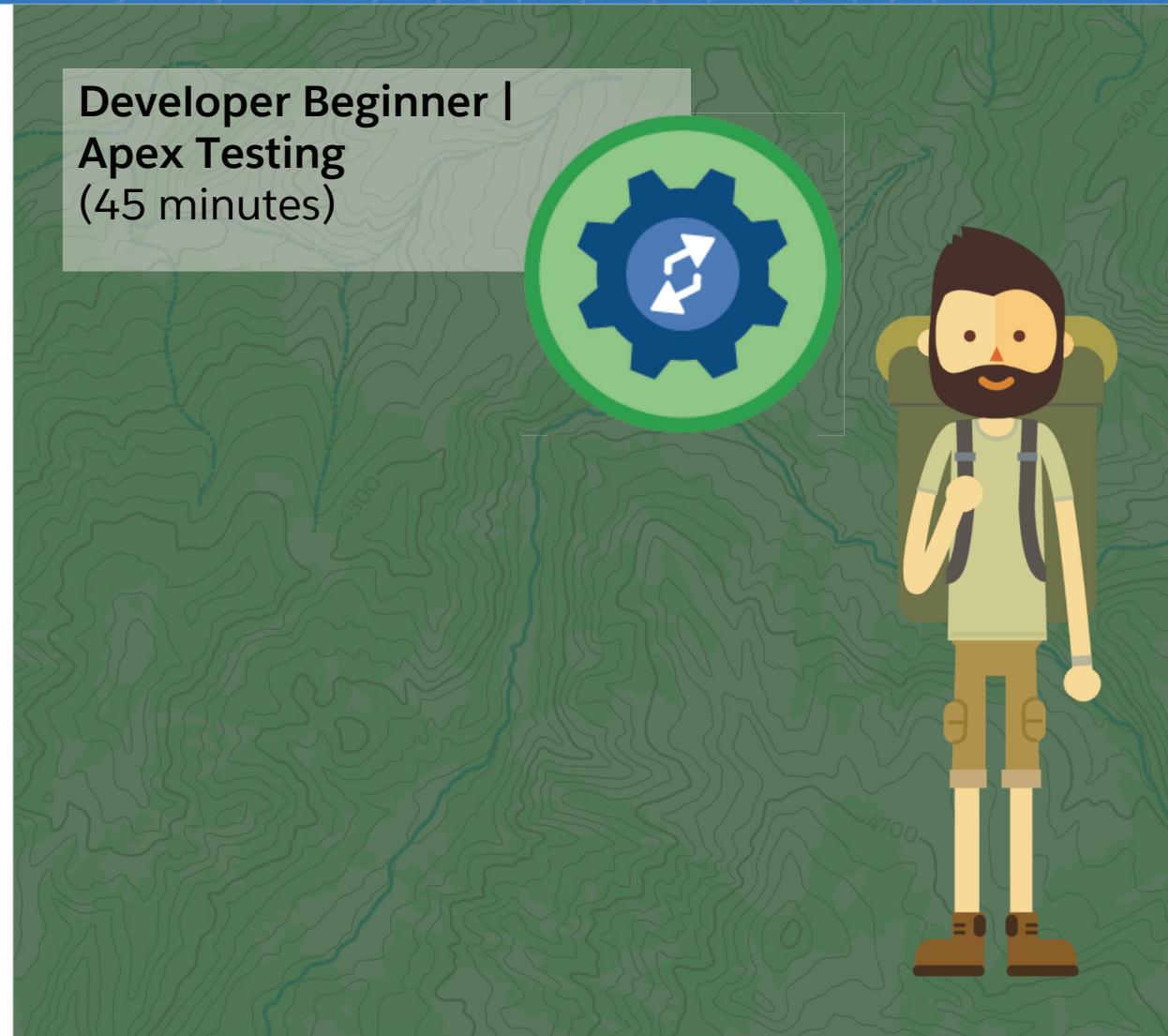
307

salesforce

1. What percentage of Apex code must be executed by test methods?
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? 

75 (mais 90% en vrai)





# MODULE 12:

## TESTING STRATEGIES

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE



**Jason Beck**  
Developer



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

311

salesforce

### MODULE 12: TESTING STRATEGIES

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



# AVOIDING ISSUES CAUSED BY TESTS RUNNING IN PARALLEL

312

salesforce

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.



```
1A @isTest (seeAllData=true)
2A public class AccountClass_Test {
3A     private static testMethod void UpdateTest() {
4A         Account a = // Select a single account
5A         a.Name = 'Foo';
6A         update a;
7A         // test logic
8A     }
9A }
```

```
1B @isTest (seeAllData=true)
2B public class AccountClass_Test2 {
3B     private static testMethod void UpdateTest2() {
4B         Account a = // Select a single account
5B         a.Name = 'Boo';
6B         update a;
7B         // test logic
8B     }
9B }
```

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?

313

salesforce

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?

```
1 // Logic for the AccountTrigger (before insert)
2 public class AccountTriggerHandler {
3     public static void printPhone(List<Account>
4                                     accounts) {
5         String debugString;
6         for (Account a : accounts) {
7             debugString = a.Name + ': ';
8             if (a.Phone == null) {
9                 debugString += 'Null Phone';
10            } else {
11                debugString += a.Phone;
12            }
13        }
14    }
15 }
```

Not Counted

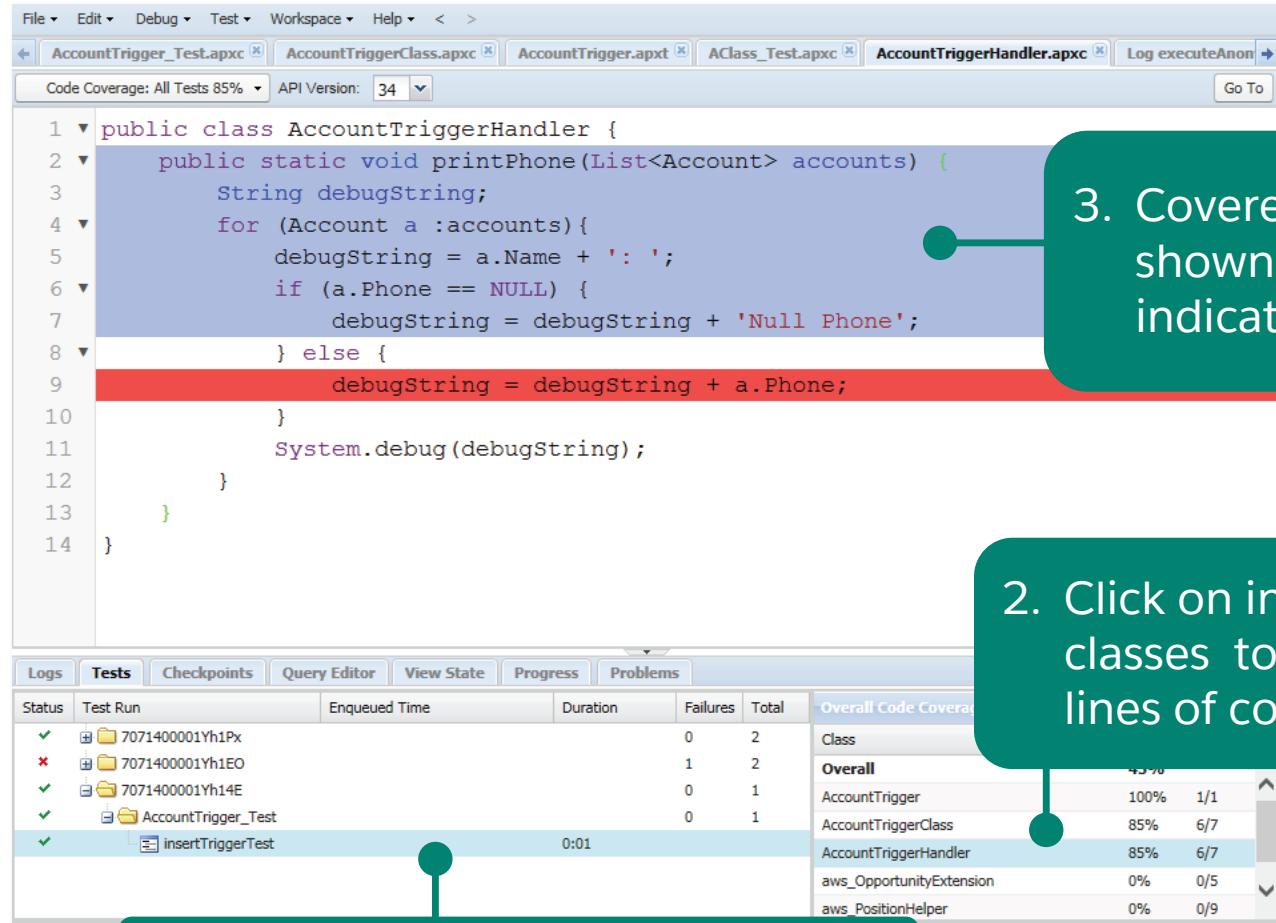
Counted

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

314

salesforce



The screenshot shows the Salesforce IDE interface. At the top, there are tabs for various files: AccountTrigger\_Test.apxc, AccountTriggerClass.apxc, AccountTrigger.apxt, AClass\_Test.apxc, AccountTriggerHandler.apxc, and Log executeAnon. Below the tabs, the code for AccountTriggerHandler is displayed. The code is color-coded: blue for covered lines and red for lines with no coverage. A callout bubble points to the red line at line 10: `if (a.Phone == NULL) {`. The status bar at the bottom of the code editor shows "Code Coverage: All Tests 85%" and "API Version: 34".

Below the code editor is a test results summary table:

Status	Test Run	Enqueued Time	Duration	Failures	Total	Overall Code Coverage
✓	7071400001Yh1Px			0	2	45%
*	7071400001Yh1EO			1	2	
✓	7071400001Yh14E			0	1	
✓	AccountTrigger_Test			0	1	
✓	insertTriggerTest		0:01			

On the right side of the interface, there is a sidebar titled "Overall Code Coverage" which lists the coverage status for various classes:

Class	Overall
AccountTrigger	100% 1/1
AccountTriggerClass	85% 6/7
AccountTriggerHandler	85% 6/7
aws_OpportunityExtension	0% 0/5
aws_PositionHelper	0% 0/9

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.

```
1 // Logic for the AccountTrigger (before insert)
2 public class AccountTriggerHandler {
3     public static void printPhone(List<Account> accounts) {
4         String debugString;
5         for (Account a : accounts) {
6             debugString = a.Name + ': ';
7             if (a.Phone == null) {
8                 debugString += 'Null Phone';
9             } else {
10                 debugString += a.Phone;
11             }
12             System.debug(debugString);
13         }
14     }
15 }
```

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.



## 12-1: EXPLORE CODE COVERAGE

**Goal:**

Explore code coverage using the Developer Console.

**Tasks:**

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



# MODULE AGENDA

317

salesforce

## MODULE 12: TESTING STRATEGIES

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



# TESTING DATA CONDITIONS: VALID, INVALID, AND BULK

318

salesforce

```
1 // Logic for the AccountTrigger (before insert)
2 public class AccountTriggerHandler {
3     public static void printPhone(List<Account> accounts) {
4         String debugString;
5         for (Account a : accounts) {
6             debugString = a.Name + ': ';
7             if (a.Phone == NULL) {
8                 debugString += 'Null Phone';
9                 aaddError('Account has no phone');
10            } else {
11                debugString += a.Phone;
12            }
13            System.debug(debugString);
14        }
15    }
16 }
```



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

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?

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?

## 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.

```
1  @isTest
2  private class MyClass_Test {
3      @testSetup
4      private static void testDataSetup() {
5          Profile p = // Write a query to select a profile
6          User u = new User(Lastname = 'Foo', ProfileId = p.id);
7          insert u;
8      }
9
10     private static testMethod void myClassTestUser() {
11         User testUser = // Find Foo, who has set up above
12         System.runAs (testUser) {
13             // Now, the rest of test runs as testUser.
14             // Otherwise the test would run as System.
15         }
16     }
17 }
```

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

```

1A // The class that needs to be tested
2A public with sharing
3A     class CertficationHeldClass {
4A         public static Integer countDailyCertHeld() {
5A             // Returns the count of
6A             // Certification Held records
7A             // created today.
8A         }

```

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.

```

1B @isTest
2B private class CertficationHeldClass_Test {
3B     @testSetup
4B     static void setupUsers() {
5B         // ... Create and insert a standard user 'su1'
6B         // ... insert 2 Cert Held records for that user
7B         // ... Create and insert standard user 'su2'
8B         // ... Create a Training Admin user 'ta1'
9B     }
10B
11B     private static testMethod void testTrainingAdmin() {
12B         // TODO 1
13B     }
14B
15B     private static testMethod void testStandardUser2() {
16B         // TODO 2
17B     }
18B }

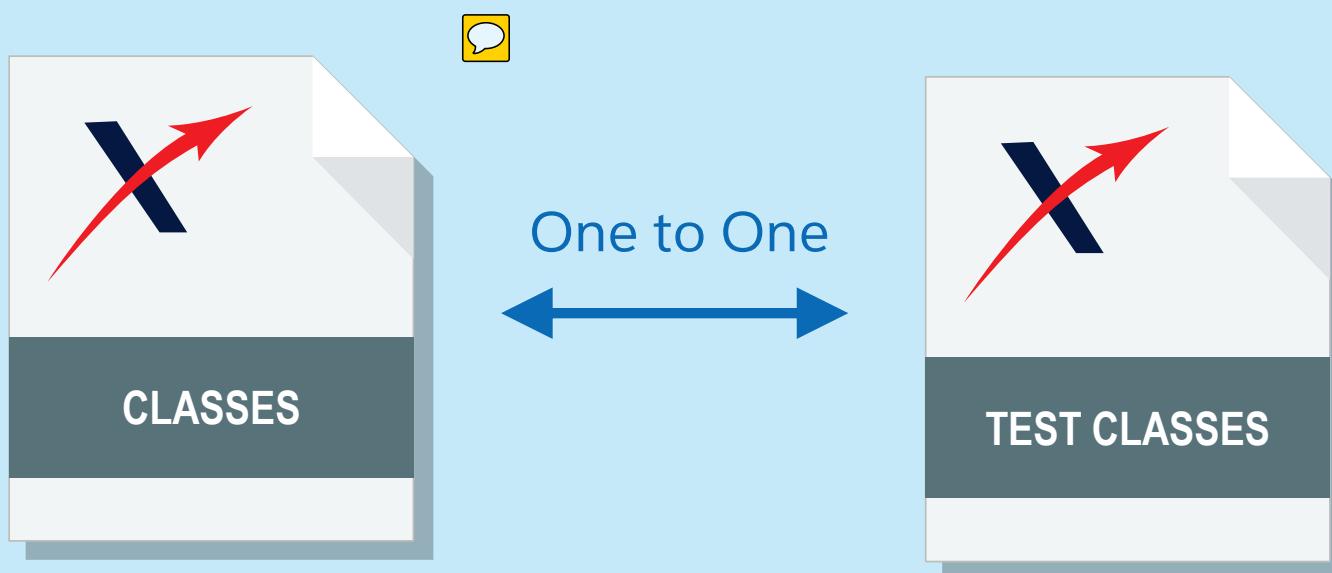
```

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

# BREAKING UP YOUR TEST CLASSES

322

salesforce



# ENSURING TEST DATA IS CREATED PROPERLY WHEN DEPLOYED

323

salesforce

```
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

324

salesforce

- 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 AGENDA

327

salesforce

### MODULE 13: STRATEGIES FOR DESIGNING EFFICIENT APEX SOLUTIONS

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





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

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

```
1 public static void provideAccessLMS (List<Course_Attendee__c> courseAttendees,
2                                     Map<Id, Course_Attendee__c> oldMap) {
3     String emails = '';
4     Integer numberOfNewEnrollees = 0;
5     for (Course_Attendee__c cA: courseAttendees) {
6         if (cA.status__c == 'Enrolled') {
7             List<Course_Attendee__c> attendeeInfo = [SELECT Course_Delivery__r.Name,
8                                                 Student__r.email, Status__c
9                                                 FROM Course_Attendee__c
10                                            WHERE Id = :ca.Id];
11
12             for (Course_Attendee__c cAwithInfo: attendeeInfo) {
13                 emails += '\n' + cAwithInfo.Course_Delivery__r.Name + ': '
14                                         + cAwithInfo.Student__r.email;
15                 numberOfNewEnrollees++;
16             }
17         }
18     }
19
20     ...
21 }
```

Querying  
within a loop.

# PRINCIPLE: USE COLLECTIONS TO MINIMIZE THE NUMBER OF QUERIES

```
1 public class CourseAttendeeTriggerHandler {  
2     public static void provideAccessLMS (List<Course_Attendee__c> courseAttendees,  
3                                         Map<Id,Course_Attendee__c> oldMap) {  
4         Set<Id> courseAttendeesToChangeAccess = new Set<Id>();  
5         for (Course_Attendee__c cA : courseAttendees) {  
6             if (cA.status__c == 'Enrolled') {  
7                 if (oldMap == null || !oldMap.containsKey(cA.Id)  
8                     || oldMap.get(cA.id).status__c != cA.status__c) {  
9                     courseAttendeesToChangeAccess.add(cA.Id);  
10                }  
11            }  
12            if (courseAttendeesToChangeAccess.size() > 0) {  
13                String emails = '';  
14                Integer numberOfNewEnrollees=0;  
15                for (List<Course_Attendee__c> caList : [SELECT Course_Delivery__r.Name, Student__r.email  
16                                                 FROM Course_Attendee__c  
17                                                 WHERE Id in :courseAttendeesToChangeAccess  
18                                                 ORDER BY Course_Delivery__r.Name]) {  
19                    ...  
20                }  
21            }  
22        }  
23    }  
24}
```

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

331



```
[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

332

salesforce

```
1 public class CourseAttendeeTriggerHandler {  
2     public static void provideAccessLMS (List<Course_Attendee__c> courseAttendees,  
3                                         Map<Id,Course_Attendee__c> oldMap) {  
4         Set<Id> courseAttendeesToChangeAccess = new Set<Id>();  
5         for (Course_Attendee__c cA : courseAttendees) {  
6             if (cA.status__c == 'Enrolled') {  
7                 if (oldMap == null || !oldMap.containsKey(cA.Id)  
8                     || oldMap.get(cA.id).status__c != cA.status__c) {  
9                     courseAttendeesToChangeAccess.add(cA.Id);  
10                }  
11            }  
12            if (courseAttendeesToChangeAccess.size() > 0) {  
13                String emails = '';  
14                Integer numberofNewEnrollees=0;  
15                for (List<Course_Attendee__c> caList : [SELECT Course_Delivery__r.Name, Student__r.email  
16                                                 FROM Course_Attendee__c  
17                                                 WHERE Id in :courseAttendeesToChangeAccess  
18                                                 ORDER BY Course_Delivery__r.Name]) {  
19                    ...  
20                }  
21            }  
22        }  
23    }  
24}
```

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

Reduces total number of SOQL queries issued.

## PRINCIPLE: REDUCING THE DATA SET

```
1 public class CourseAttendeeTriggerHandler {  
2     public static void provideAccessLMS (List<Course_Attendee__c> courseAttendees,  
                                         Map<Id, Course_Attendee__c> oldMap) {  
3         Set<Id> courseAttendeesToChangeAccess = new Set<Id>();  
4         for (Course_Attendee__c cA : courseAttendees) {  
5             if (cA.status__c == 'Enrolled') {  
6                 if (oldMap == null || !oldMap.containsKey(cA.Id)  
7                     || oldMap.get(cA.id).status__c != cA.status__c) {  
8                     courseAttendeesToChangeAccess.add(cA.Id);  
9                 }  
10            }  
11        }  
12        if (courseAttendeesToChangeAccess.size() > 0) {  
13            String emails = '';  
14            Integer numberofNewEnrollees=0;  
15            for (List<Course_Attendee__c> caList : [SELECT Course_Delivery__r.Name, Student__r.email  
16                                            FROM Course_Attendee__c  
17                                            WHERE Id in :courseAttendeesToChangeAccess  
18            ]  
19            ) {  
20                for (Course_Attendee__c c : caList) {  
21                    if (c.status__c == 'Enrolled') {  
22                        emails += c.Student__r.email + ',';  
23                    }  
24                }  
25            }  
26            if (emails != '') {  
27                String[] emailsList = emails.split(',');  
28                for (String email : emailsList) {  
29                    // Logic to send email to student  
30                }  
31            }  
32        }  
33    }  
34}
```

Apply filters  
when possible.

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

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

334



```
1 ...
2 if (courseAttendeesToChangeAccess .size() > 0) {
3     String emails = '';
4     Integer numberOfNewEnrollees=0;
5     for (List<Course_Attendee__c> caList : [SELECT Course_Delivery__r.Name, Student__r.email
6                                                 FROM Course_Attendee__c
7                                                 WHERE Id in : courseAttendeesToChangeAccess
8                                                 ORDER BY Course_Delivery__r.Name]) {
9         for (Course_Attendee__c ca: caList) {
10             emails += '\n' + ca.Course_Delivery__r.Name + ':' + ca.Student__r.email;
11             numberOfNewEnrollees++;
12         }
13     }
14 }
```

## FOR LOOP

Records automatically  
batched for efficient  
processing.



Reduces heap size.

# TIP FOR IMPROVING SOQL PERFORMANCE

335

salesforce

Using indexed fields when possible may improve performance.

```
1 SELECT Course_Delivery__r.Name, Student__r.email
2 FROM Course_Attendee__c
3 WHERE Id in :idsToChangeAccess
4 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		<input checked="" type="checkbox"/>	
	Account Name	Name	Name		<input checked="" type="checkbox"/>	
Edit	Account Number	AccountNumber	Text(40)		<input checked="" type="checkbox"/>	
Edit	Account Owner	Owner	Lookup(User)		<input checked="" type="checkbox"/>	
Edit	Account Record Type	RecordType	Record Type		<input checked="" type="checkbox"/>	

**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.

- Total number of DML statements issued. **150**
- Total number of records processed as a result of DML statements. **10,000**
- Total heap size.



```
1 public with sharing class CourseDeliveryTriggerHandler {  
2     public static void cancelEnrollees(List<Course_Delivery__c> triggerNew,  
                                         Map <Id, Course_Delivery__c> oldMap) {  
3         Set<Id> cancelledCourseDeliveries = new Set<Id>();  
4         for (Course_Delivery__c cD : triggerNew) {  
5             if (cD.Status__c == 'Cancelled' && oldMap.get(cD.id).Status__c != 'Cancelled') {  
6                 cancelledCourseDeliveries.add(cD.id);  
7             }  
8         }  
9         for (Course_Attendee__c cA : [SELECT id  
                                         FROM Course_Attendee__c  
                                         WHERE Course_Delivery__c in :cancelledCourseDeliveries  
                                         AND Status__c = 'Enrolled']) {  
10            cA.Status__c = 'Course Cancelled';  
11            try {  
12                update cA;   
13            } catch (System.DmlException ex) {  
14                System.debug(ex);  
15            }  
16        }  
17    }  
18 }
```

One DML statement  
per sObject.

# PRINCIPLE: BATCH DATA FOR DML STATEMENTS

339

salesforce

```
1 public with sharing class CourseDeliveryTriggerHandler {  
2     public static void cancelEnrollees(List<Course_Delivery__c> triggerNew,  
3                                         Map <Id, Course_Delivery__c> oldMap) {  
4         Set<Id> cancelledCourseDeliveries = new Set<Id>();  
5         for (Course_Delivery__c cD : triggerNew) {  
6             if (cD.Status__c == 'Cancelled' && oldMap.get(cD.id).Status__c != 'Cancelled') {  
7                 cancelledCourseDeliveries.add(cD.id);  
8             }  
9         }  
10        List<Course_Attendee__c> courseAttendeesToUpdate = new List<Course_Attendee__c>();  
11        for (Course_Attendee__c cA : [SELECT id  
12                                     FROM Course_Attendee__c  
13                                     WHERE Course_Delivery__c in :cancelledCourseDeliveries  
14                                     AND Status__c = 'Enrolled') {  
15            cA.Status__c = 'Course Cancelled';  
16            courseAttendeesToUpdate.add(cA);  
17        }  
18        if (courseAttendeesToUpdate.size() > 0) {  
19            try {  
20                update courseAttendeesToUpdate;  
21            } catch (System.DmlException ex) {  
22                System.debug(ex);  
23            }  
24        }  
25    } } } //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

340

salesforce



```
1 //More before this...
2 for (List<Course_Attendee__c> cAList :[SELECT id FROM Course_Attendee__c
3                                     WHERE Course_Delivery__c in :cancelledCourseDeliveries
4                                     AND Status__c = 'Enrolled']) {
5     for (Course_Attendee__c cA :cAList) {
6         cA.Status__c = 'Course Cancelled';
7     }
8     try {
9         update cAList;
10    } catch (System.DmlException ex) {
11        System.debug(ex);
12    }
13 }
```

Inputs to loop are batched from SOQL for loop.

DML statement for the batch.

**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 AGENDA

342

salesforce

### MODULE 13: STRATEGIES FOR DESIGNING EFFICIENT APEX SOLUTIONS

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



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



```
1A trigger CourseAttendeeValidateTrigger on Course_Attendee__c (before insert) {  
2A     CourseAttendeeValidateTriggerHandler.validateEnrollment(Trigger.new);  
3A }
```

```
1B trigger CourseAttendeeNotificationTrigger on Course_Attendee__c (after insert, after update) {  
2B     CourseAttendeeNotificationTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);  
3B }
```

```
1C trigger CourseAttendeeLMSTrigger on Course_Attendee__c (after insert, after update) {  
2C     CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);  
3C }
```

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

344

salesforce



```
1 trigger CourseAttendeeTrigger on Course_Attendee_c (before insert, after insert, after update) {  
2     if (Trigger.isBefore && Trigger.isInsert) {  
3         CourseAttendeeTriggerHandler.validateEnrollment(Trigger.new);  
4     } else if (Trigger.isAfter) {  
5         if (Trigger.isInsert || Trigger.IsUpdate) {  
6             CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);  
7             CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);  
8         }  
9     }  
10 }
```

1

2

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

345

salesforce

```
1 trigger CourseAttendeeTrigger on Course_Attendee__c (before insert, after insert, after update) {  
2     if (Trigger.isBefore) {  
3         if (Trigger.isInsert) {  
4             CourseAttendeeTriggerHandler.validateEnrollment(Trigger.new);  
5         }  
6     } else if (Trigger.isAfter) {  
7         if (Trigger.isInsert) {  
8             CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);  
9             CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);  
10        } else if(Trigger.IsUpdate) {  
11            CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);  
12            CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);  
13        }  
14    }  
15 }
```



Easy to add additional logic and order trigger events.

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

```
1 trigger CourseAttendeeTrigger on Course_Attendee__c (before insert, after insert, after update) {  
2     if (Trigger.isAfter) {  
3         if (Trigger.isInsert) {  
4             Set<Id> newToChangeAccess = new Set<Id>();  
5             for (Course_Attendee__c cA : Trigger.new) {  
6                 if (cA.Status__c == 'Enrolled') {  
7                     if (Trigger.oldMap == null || !Trigger.oldMap.containsKey(cA.Id)  
8                         || Trigger.oldMap.get(cA.id).Status__c != cA.Status__c) {  
9                         newToChangeAccess.add(cA.Id);  
10                    }  
11                }  
12            if (newToChangeAccess.size() > 0) {  
13                String emails = '';  
14                Integer numberOfNewEnrollees = 0;  
15                for (List<Course_Attendee__c> caList : [SELECT Course_Delivery__r.Name, Student__r.email  
16                                              FROM Course_Attendee__c  
17                                              WHERE Id in :newToChangeAccess  
18                                              ORDER BY Course_Delivery__r.Name]) {  
19                    for (Course_Attendee__c ca: caList) {  
20                        ...  
21                    }  
22                }  
23            }  
24        }  
25    }  
26}
```



# PRINCIPLE: KEEP BUSINESS LOGIC OUT OF YOUR TRIGGER

347



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

The trigger should at most contain the traffic cop logic.

```
1 trigger CourseAttendeeTrigger on Course_Attendee__c (before insert, after insert, after update) {  
2     if (Trigger.isBefore) {  
3         if (Trigger.isInsert) {  
4             CourseAttendeeTriggerHandler.validateEnrollment(Trigger.new);  
5         }  
6     } else if (Trigger.isAfter) {  
7         if (Trigger.isInsert) {  
8             CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);  
9             CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);  
10        } else if(Trigger.IsUpdate) {  
11            CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);  
12            CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);  
13        }  
14    }  
15 }
```

# SHOULD YOU EXPLICITLY PASS TRIGGER CONTEXT DATA?

348

salesforce

```
1 trigger CourseAttendeeTrigger on Course_Attendee__c (before insert, after insert, after update) {  
2     if (Trigger.isBefore) {  
3         if (Trigger.isInsert) {  
4             CourseAttendeeTriggerHandler.validateEnrollment(Trigger.new);  
5         }  
6     } else if (Trigger.isAfter) {  
7         if (Trigger.isInsert) {  
8             CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);  
9             CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);  
10        } else if(Trigger.IsUpdate) {  
11            CourseAttendeeTriggerHandler.notifyAttendee(Trigger.new, Trigger.oldMap);  
12            CourseAttendeeLMSTriggerHandler.provideAccessLMS(Trigger.new, Trigger.oldMap);  
13        }  
14    }  
15 }
```

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

349

salesforce

## MODULE 13: STRATEGIES FOR DESIGNING EFFICIENT APEX SOLUTIONS

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



# PRINCIPLE: MODULARIZE BUSINESS LOGIC

350

salesforce

Write individual methods for each piece of business logic.

```
1 public class CertificationAttemptTriggerHandler {  
2     public static void grantInstructorSharingAccess(List<Certification_Aattempt__c> triggerNew,  
3                                                 Map<Id, Certification_Aattempt__c> oldMap, Boolean isInsert, Boolean isUpdate) {  
4         //Method does work...  
5         ...  
6     }  
7  
8     public static void validateCertificationAttempt(List<Certification_Aattempt__c> triggerNew) {  
9         //Method does more work  
10        ...  
11    }  
12  
13    //Additional methods here...  
14    ...  
15 }  
16 }
```

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

351

salesforce

- 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?

Insert 200 records

Trigger sent 200 records

Insert 1 record

Trigger sent 1 record



## KEY TAKEAWAYS

352

salesforce

- 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 AGENDA

355

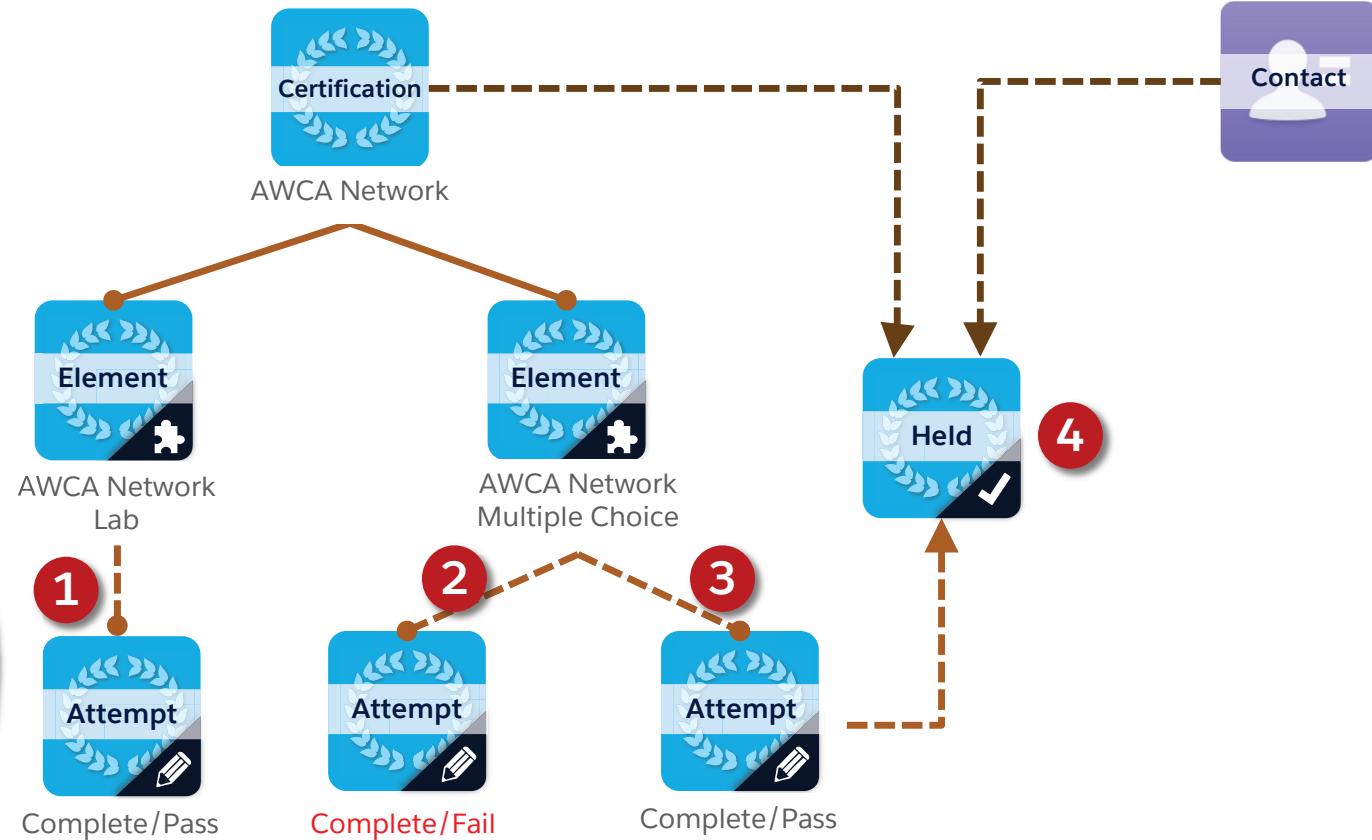
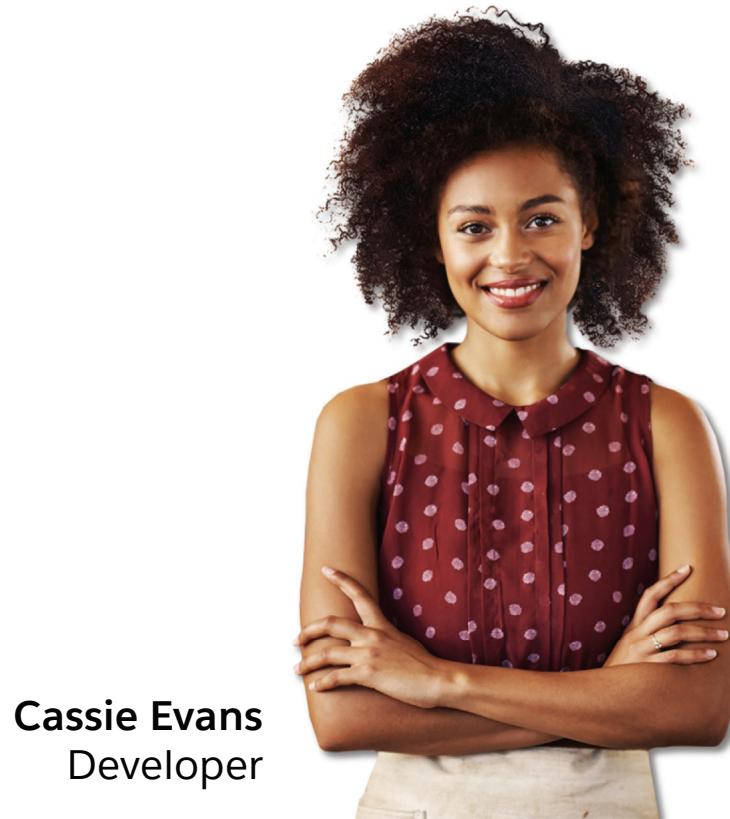
salesforce

## MODULE 14: TRIGGER DESIGN STRATEGY

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



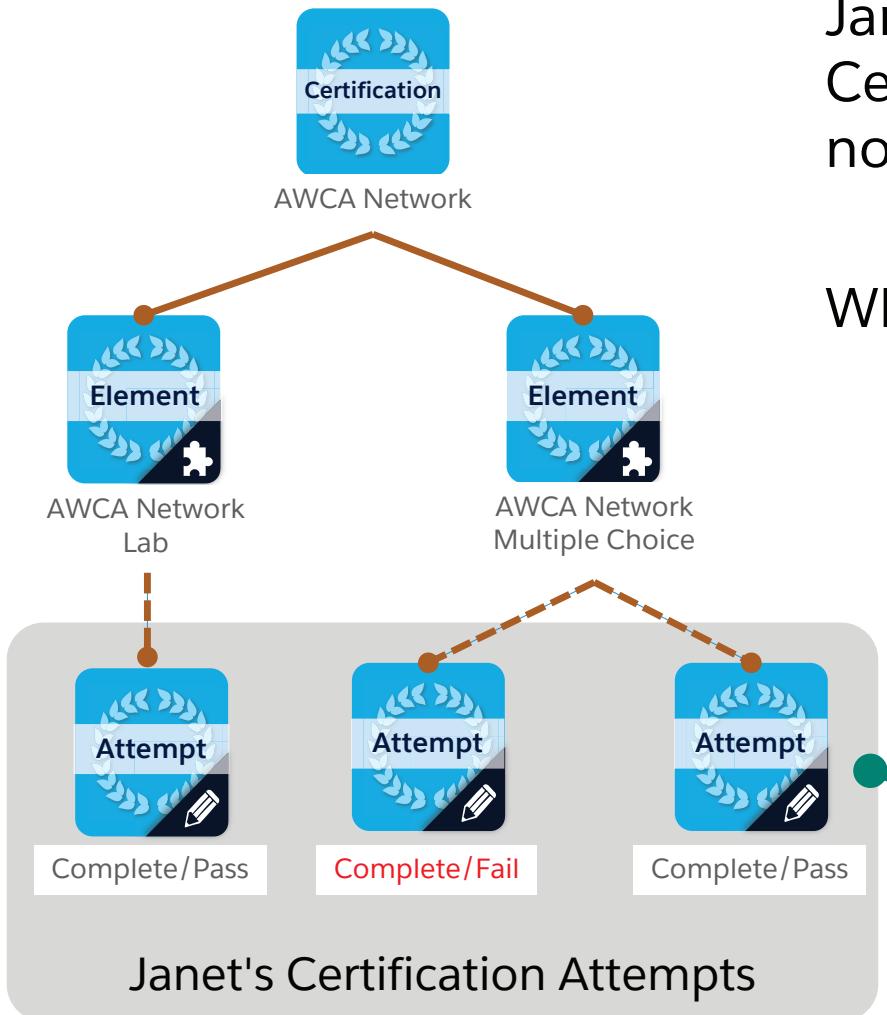
We need a process that automatically creates a Certification Held record. When a candidate passes a Certification Attempt, indicating they have completed a Certification Element, we need to determine if they qualify for the related Certification.



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

357

salesforce



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)

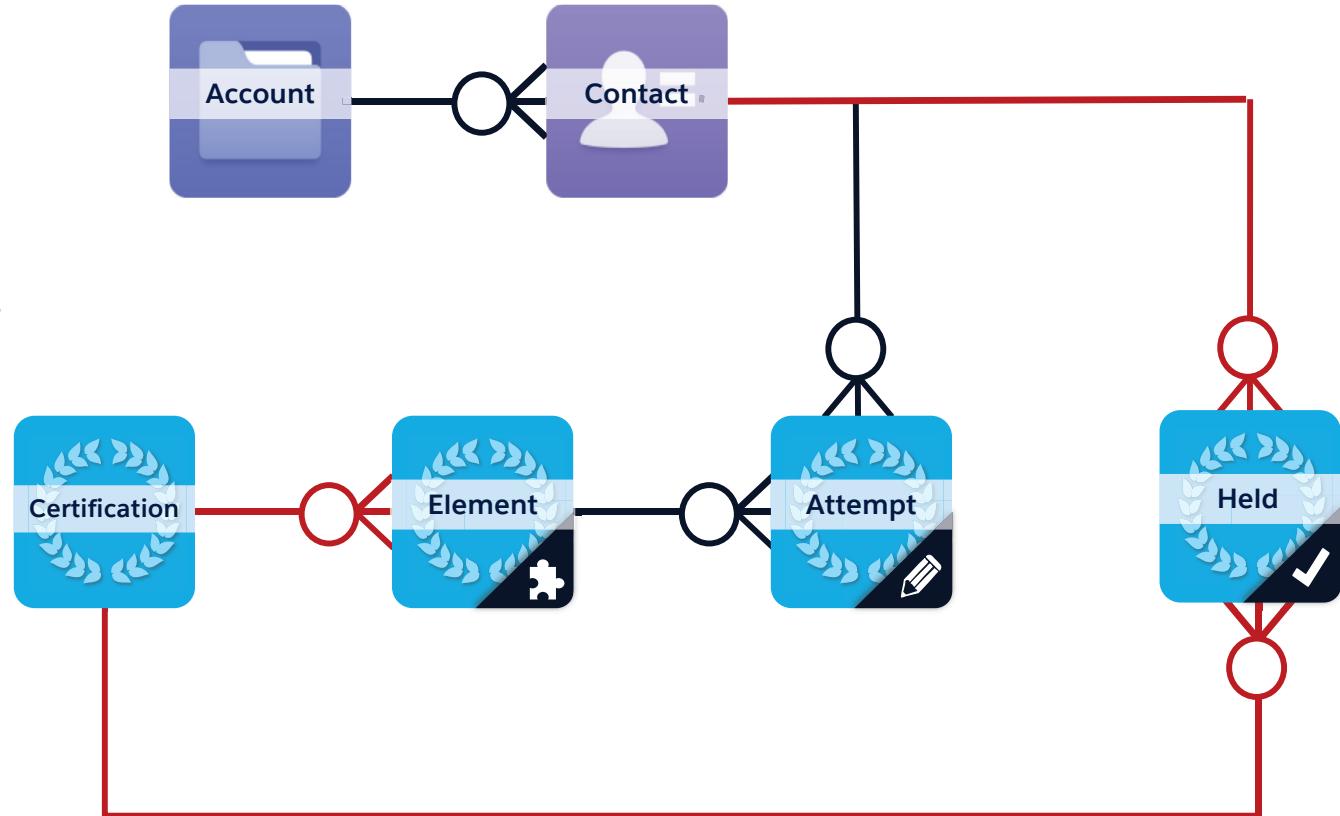
Newly added certification attempt.

# HOW DO YOU GET THE NEEDED DATA?

358

salesforce

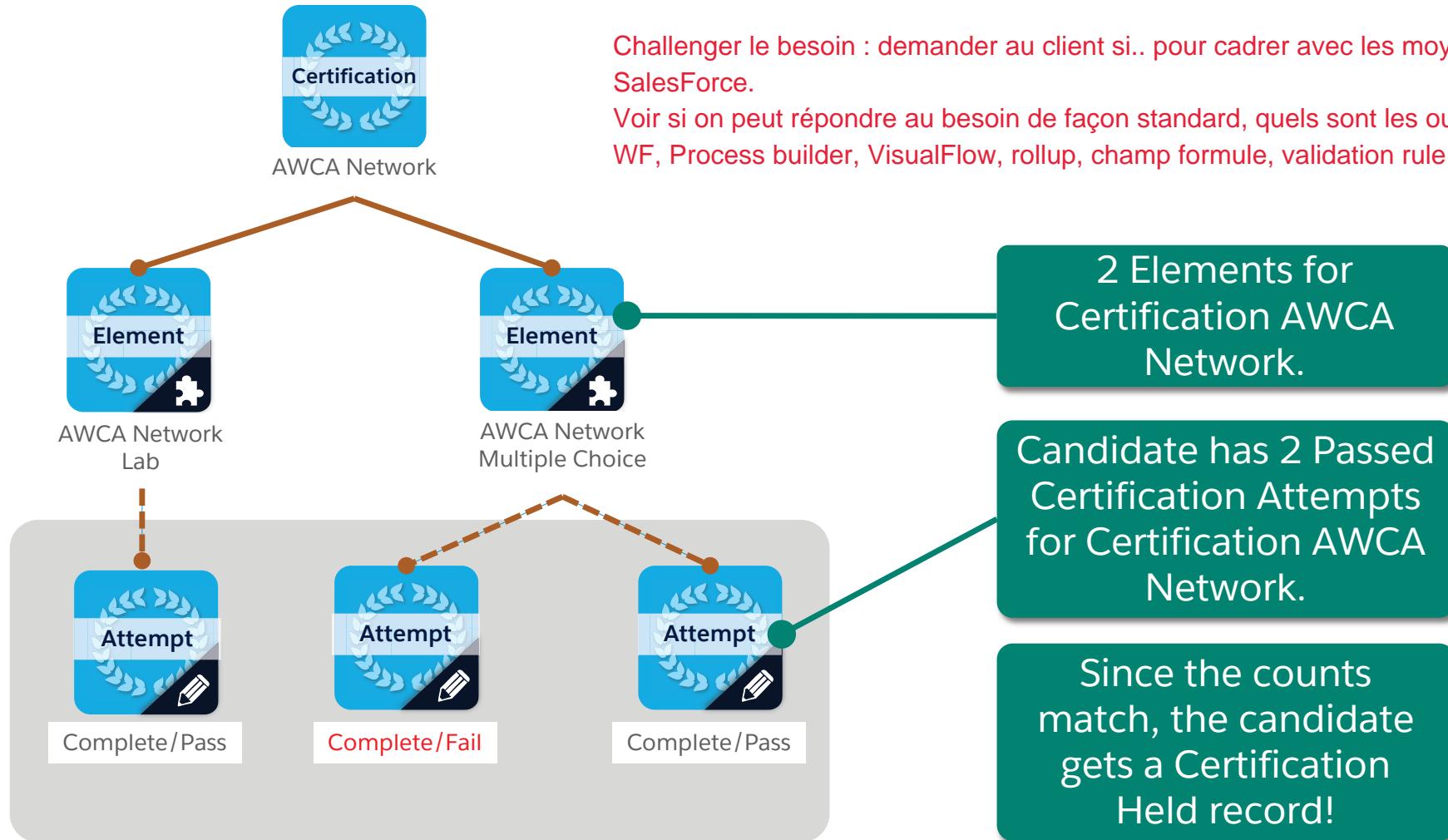
- Changes to which object are potentially firing this automation? **Attempt**
- How do you find the related certification? **Relations**
- How do you find the elements associated with that certification?
- How do you determine whether the candidate has passed those elements?



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?

359



# 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 AGENDA

361

salesforce

## MODULE 14: TRIGGER DESIGN STRATEGY

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



# WHAT TYPE OF TRIGGER IS NEEDED?

362

salesforce

sObject type:

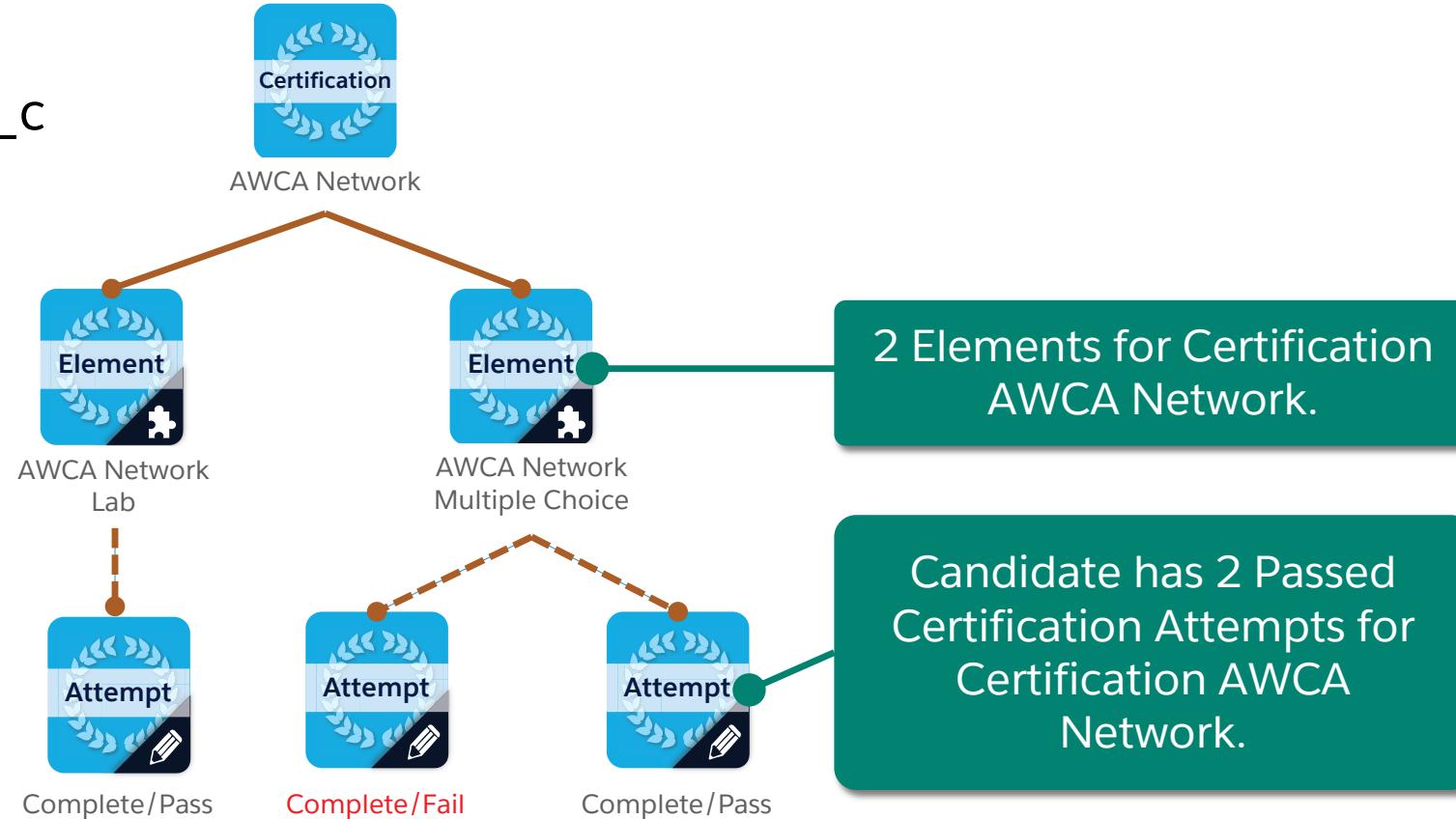
Certification\_Attempt\_\_c

Trigger operation(s)?

insert, update

before or after?

after



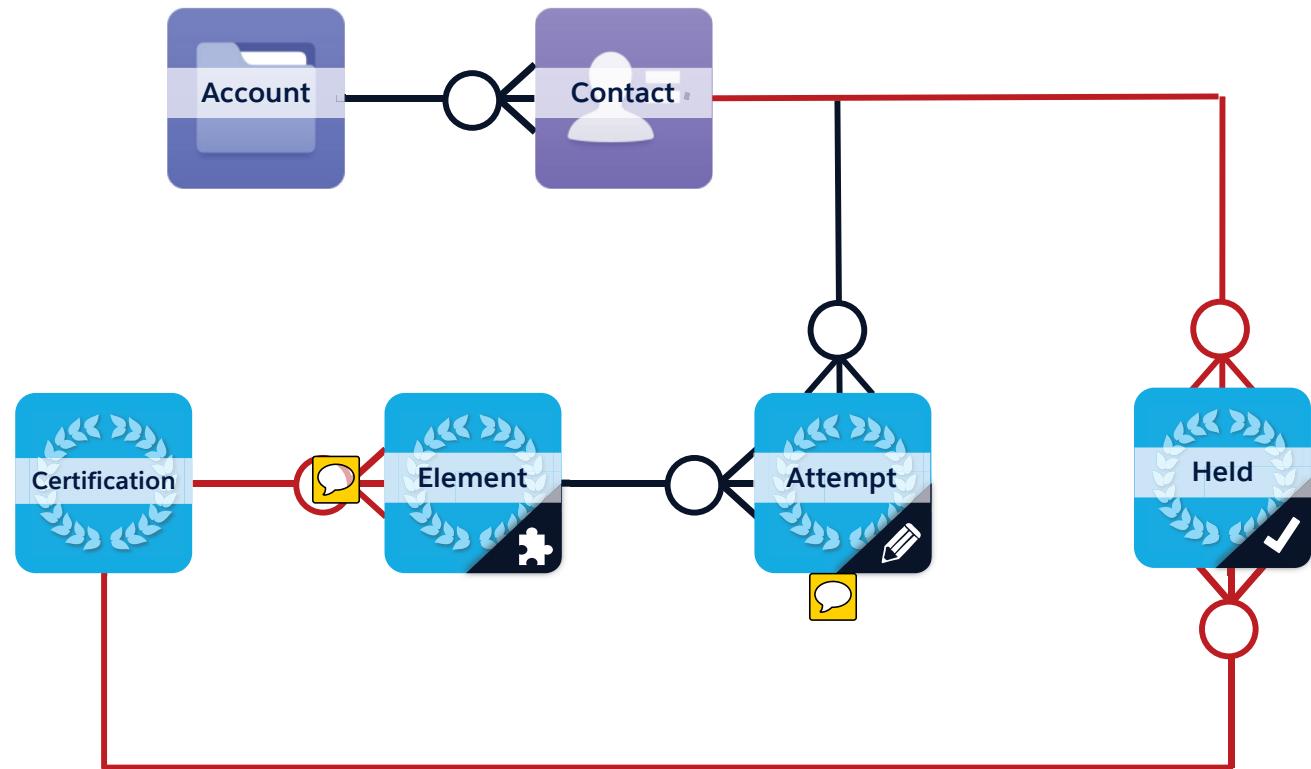
# HOW MANY QUERIES ARE NEEDED FOR THE SOLUTION?

363

salesforce

## 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.



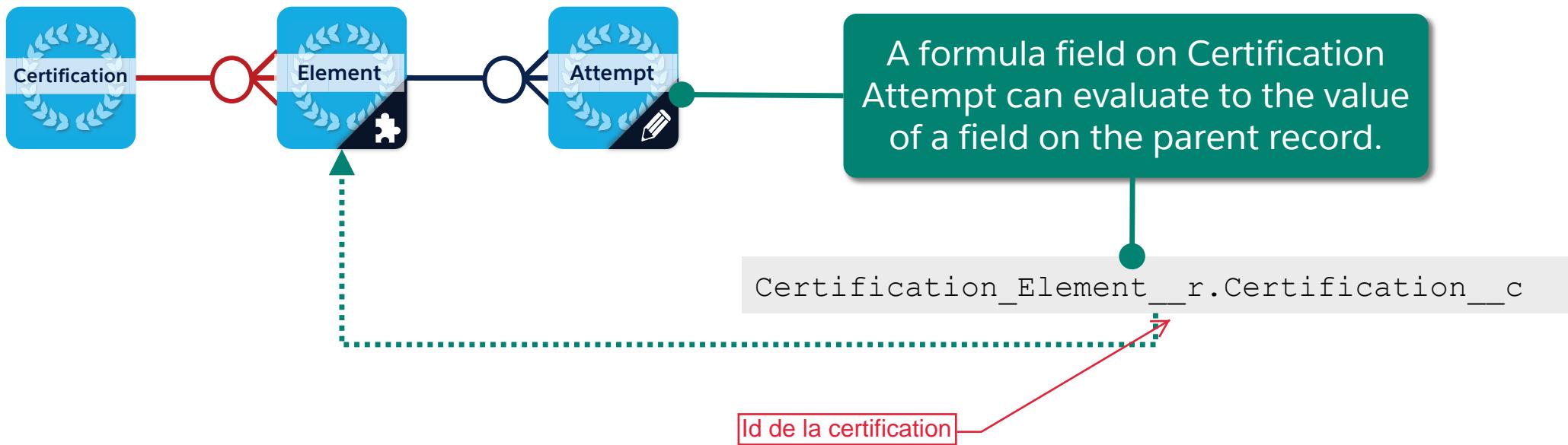
# HOW CAN YOU USE DECLARATIVE FEATURES TO REDUCE THE NUMBER OF QUERIES?

364

salesforce

How do you find the related certification?

Formule de Certification Attempt sur le master -> Certification Element



**Goal:**

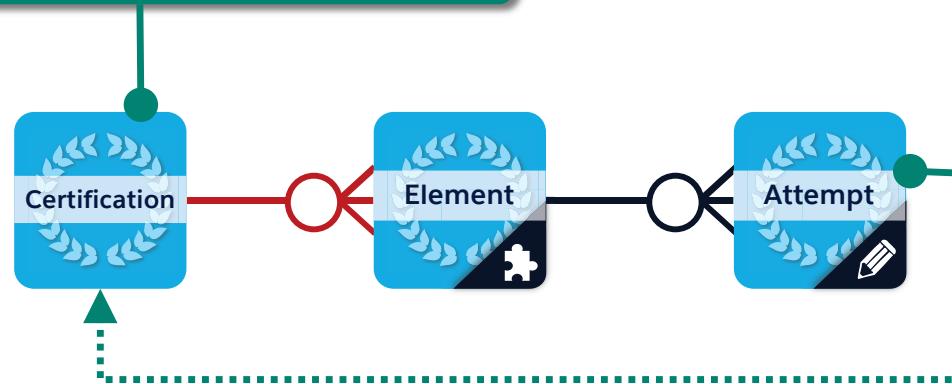
Create a formula field so that the certification Id is available on the Certification Attempt object.

**Task:**

Create a hidden formula field on the Certification Attempt object.

How 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.

**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?

368

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?

369

salesforce

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

## Trigger.new (on insert)

A123
C1
AWCP Security
2
Pass

A104
C2
AWCP Network
2
Pass

A125
C3
S
2
Fail

A103
C3
AWCP Network
2
Pass

A126
C1
S
2
In progress

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

370

salesforce

Trigger.new (on insert)

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

Filter

Certifications:  
AWCP Security,  
AWCP Network



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

The Database

A101	A073	A032	A093	A066	A123
C6	C7	C2	C3	C4	C1
2	2	AWCP Network	1	2	AWCP Security
Fail	Pass	Pass	In progress	Schedule	Pass
A030	A072	A104	A093	A066	A030
C1	C4	C2	C3	C4	C1
AWCP Network					
2	2	2	1	2	2
Pass	Pass	Pass	Pass	Pass	Fail
A017	A082	A031	A103	A083	A002
C2	C1	C7	C3	C4	C1
2	2	2	2	2	2
Fail	Fail	Pass	Pass	Pass	Fail

**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?

372

salesforce

A032	A123	A030	A104	A103
C2	C1	C1	C2	C3
AWCP Network	AWCP Security	AWCP Network	AWCP Network	AWCP Network
2	2	2	2	2
Pass	Pass	Pass	Pass	Pass

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

C2 + AWCP Network → 2

C1 + AWCP Network → 1

C3 + AWCP Network → 1

C1 + AWCP Security → 1

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

**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?

374

salesforce

Trigger.new (on insert)

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

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

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

1 record found.

No record created.

**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?

376

salesforce

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		
Action	Field Label	API Name
Edit   Del	<a href="#">Certification</a>	Certification__c
Edit   Del	<a href="#">Certified Professional</a>	Certified_Professional__c
Edit   Del	<a href="#">Date Achieved</a>	Date_Achieved__c

Solution:

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

Unique  Do not allow duplicate values

**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

378

salesforce

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.



## MODULE AGENDA

381

salesforce

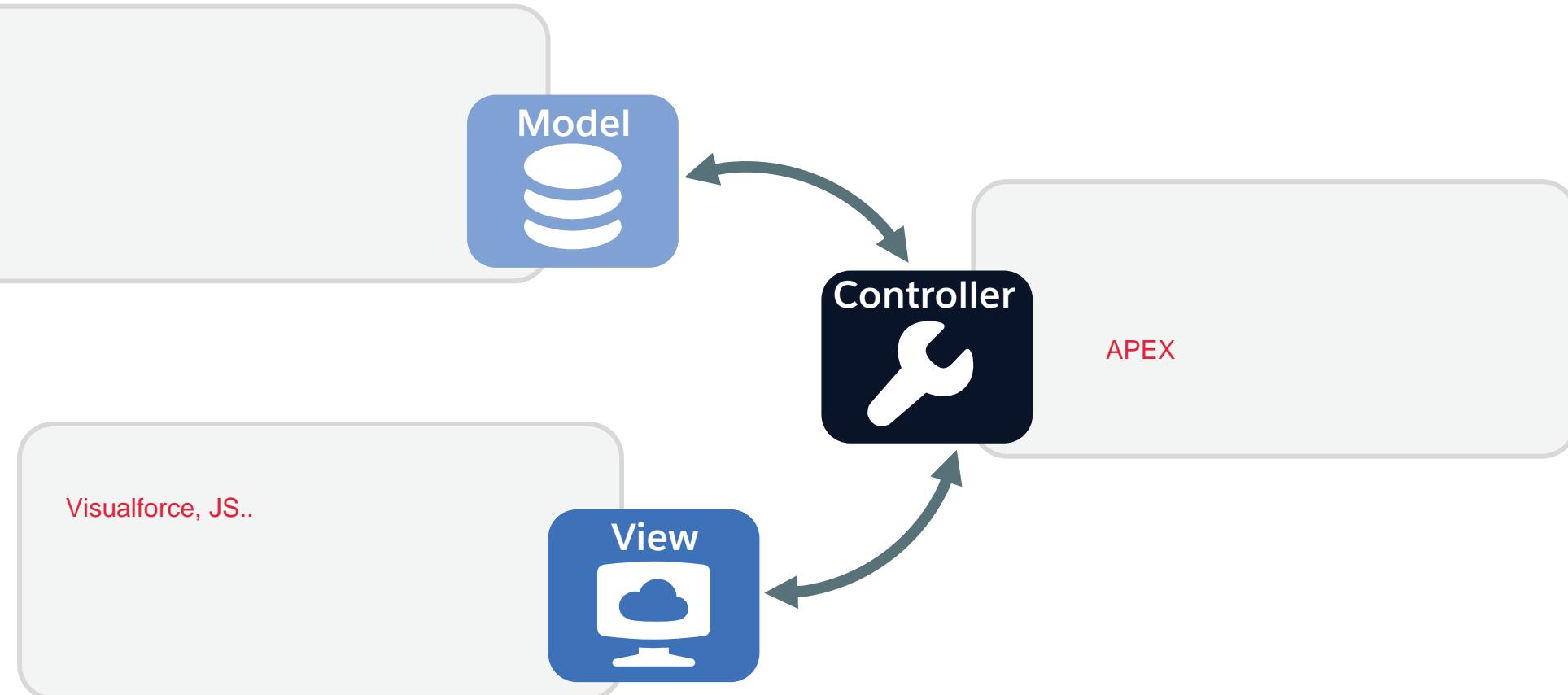
### MODULE 15: CREATING VISUALFORCE PAGES

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



# MAPPING SALESFORCE ELEMENTS TO THE MVC

382



# WHAT IS VISUALFORCE? - MARKUP

383

salesforce

DEFINITION:



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



```
1 <apex:page controller="MyController_CC" tabStyle="Account">
2   <apex:form >
3     <apex:pageBlock id="contactList" title="Verify Accounts">
4       <apex:panelGrid cellspacing="4" columns="6" >
5         ...
6       </apex:pageBlock>
7     </apex:form>
8   </apex:page>
```

# WHAT IS VISUALFORCE? - CONTROLLERS

384

salesforce

DEFINITION:



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

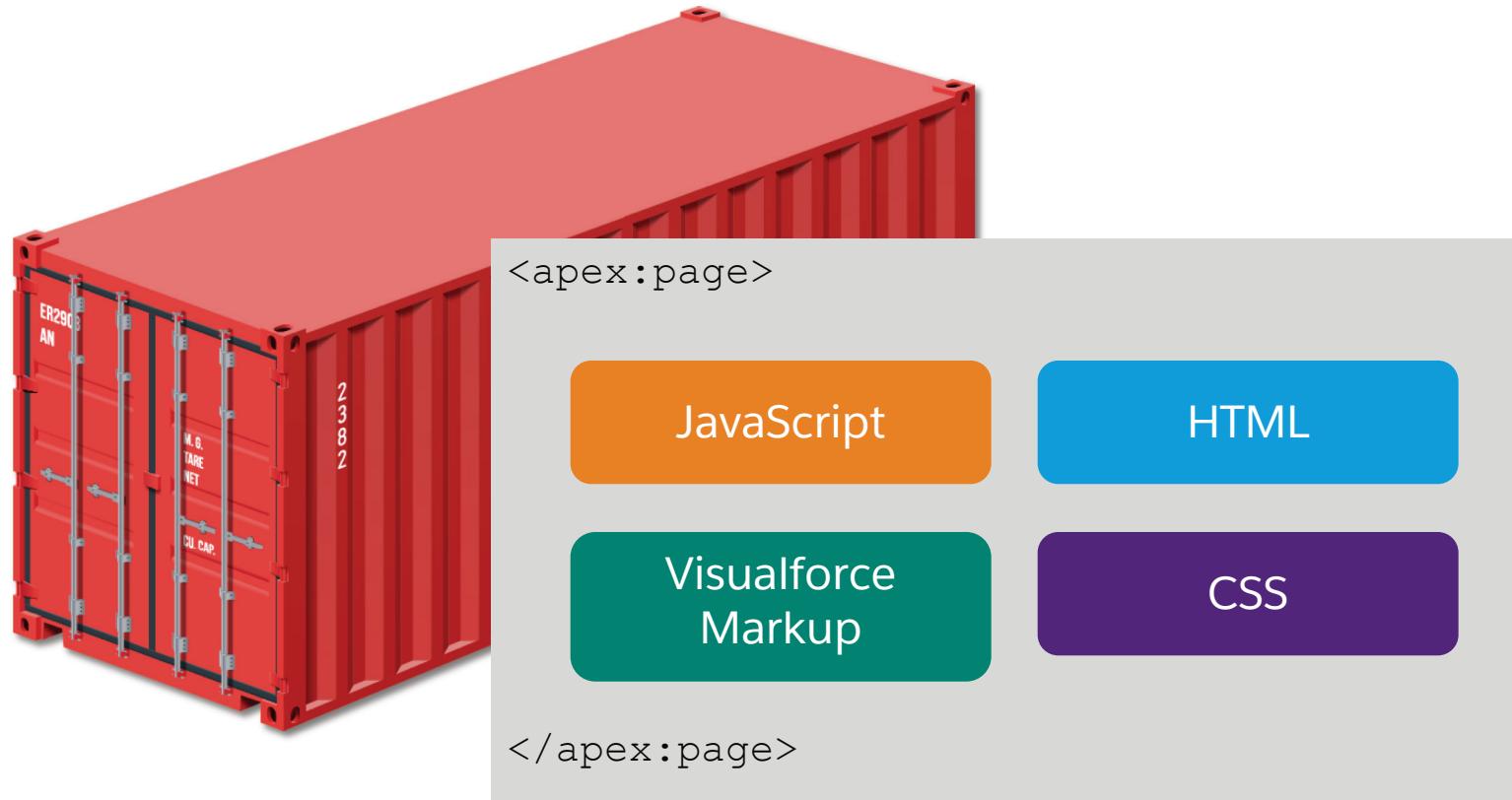
```
1 public class MyController_CC {  
2     private final Account account;  
3     public MyController_CC() {  
4         account = [SELECT Id, Name, Site FROM Account WHERE Id =  
5                     :ApexPages.currentPage().getParameters().get('id')];  
6     }  
7     public PageReference save() {  
8         update account;  
9         return null;  
10    }  
11 }
```

# WHAT IS A VISUALFORCE PAGE?

385

salesforce

A Visualforce page is a container.



## Out of the Box (classic SalesForce)

Standard and  
Custom Objects

Model



Page Layouts

View



Standard Controller

Controller



## Using Visualforce

Standard and  
Custom Objects

Visualforce Pages

Standard and  
Custom Controllers





## MODULE AGENDA

387

salesforce

### MODULE 15: CREATING VISUALFORCE PAGES

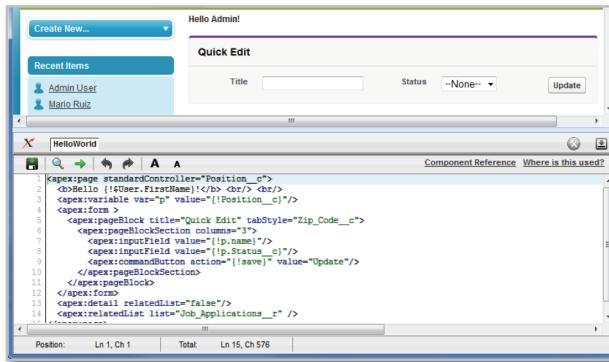
- Understanding Visualforce
- **Creating a Visualforce Page**
- Displaying Record Data and Launching a Visualforce Page



# TOOLS FOR CREATING A VISUALFORCE PAGE

388

salesforce

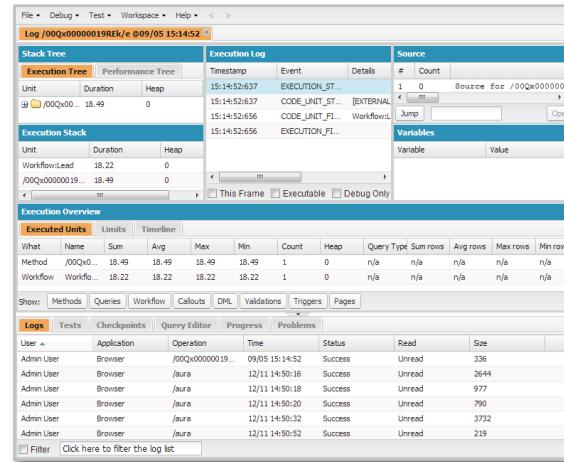


## Inline Editor

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

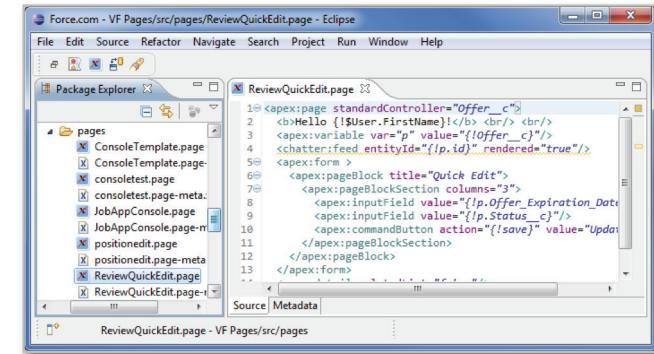


NOTE: There are also a number of other third-party tools for authoring Visualforce.



## Developer Console

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



## Force.com IDE

A plug-in for Eclipse

# ELEMENTS OF A VISUALFORCE PAGE: VISUALFORCE MARKUP

389

salesforce

Identifies Account's standard controller

```
1 <apex:page standardController="Account"  
2   standardStylesheets="false" showHeader="false">  
3   <apex:stylesheet value="{ !$Resource.customCSS }"/>  
4   <apex:includeScript value="{ !$Resource.jQuery }"/>  
5  
6   <apex: dataTable value="{ !account.contacts }" var="c"  
7     headerClass="tablehead" rowClasses="odd,even" >  
8     <apex:column>  
9       <apex:facet name="header">First Name</apex:facet>  
10      <apex:outputText value="{ !c.FirstName }"/>  
11    </apex:column>  
12    <apex:column>  
13      <apex:facet name="header">Last Name</apex:facet>  
14      <apex:outputField value="{ !c.LastName }"/>  
15    </apex:column>  
16  </apex: dataTable>  
17</apex:page>
```

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

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

Displays data about the contacts.

# THE VISUALFORCE COMPONENT REFERENCE

390

salesforce

The screenshot illustrates the Visualforce Component Reference interface. On the left, a code editor shows a Visualforce page with a component reference highlighted. A red box and a callout bubble labeled "Component Reference" point to this area. On the right, a detailed component description is shown for the `<apex:detail>` component. A green box and a callout bubble labeled "Component description" point to the component's definition. Below the description, a table lists the component's attributes with their types and descriptions. A green box and a callout bubble labeled "Usage example" points to the table. A green box and a callout bubble labeled "Component attributes" points to the list of attributes on the right. A green box and a callout bubble labeled "List of components" points to the list of components on the far left.

`<apex:page standardController="Account">`  
`<apex:pageBlock >`  
`<apex:pageBlockSection columns="1" >`  
`<apex:detail subject="{!account}"/>`  
`</apex:pageBlockSection>`  
`<apex:pageBlockSection title="Contact" >`  
`<apex:pageBlockTable value="{!Account}`  
`<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:facet>`
- `<apex:flash>`
- `<apex:form>`
- `<apex:gaugeSeries>`
- `<apex:iframe>`
- `<apex:image>`
- `<apex:include>`
- `<apex:includeScript>`

**Component description**

`<apex:detail>`

The standard detail page for a particular object, as defined by the associated page layout for the object in Setup. This component includes 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	
<b>Attribute Name</b>	<b>Attribute Type</b>	<b>Description</b>
<code>id</code>	String	An identifier that allows the detail component to be referenced by other components in the page.
<code>inlineEdit</code>	Boolean	Controls whether the component supports inline editing.
		See also: <code>&lt; apex:inlineEditSupport &gt;</code>
<code>oncomplete</code>	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.
<code>relatedList</code>	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.
<code>relatedListHover</code>	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.

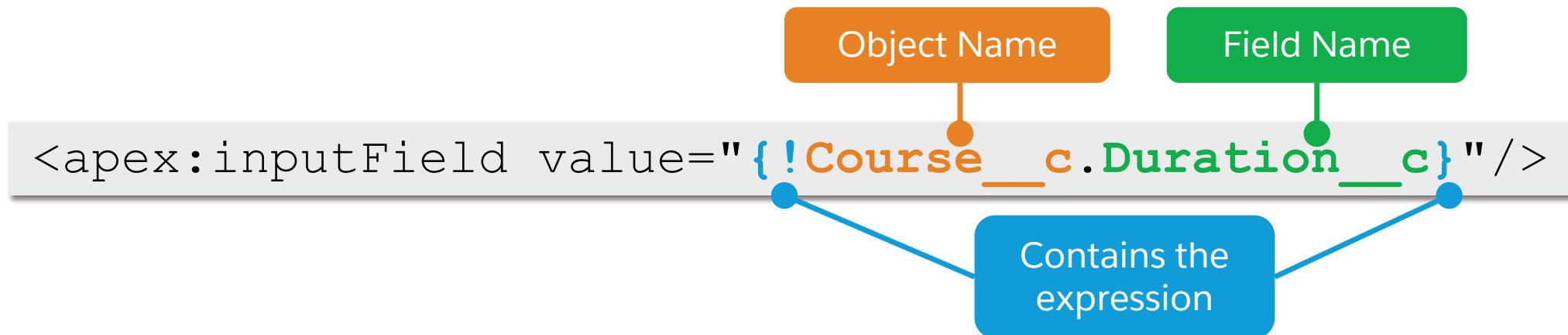
**Component attributes**

**Usage example**

**List of components**



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



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

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

## 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

393

salesforce

### MODULE 15: CREATING VISUALFORCE PAGES

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



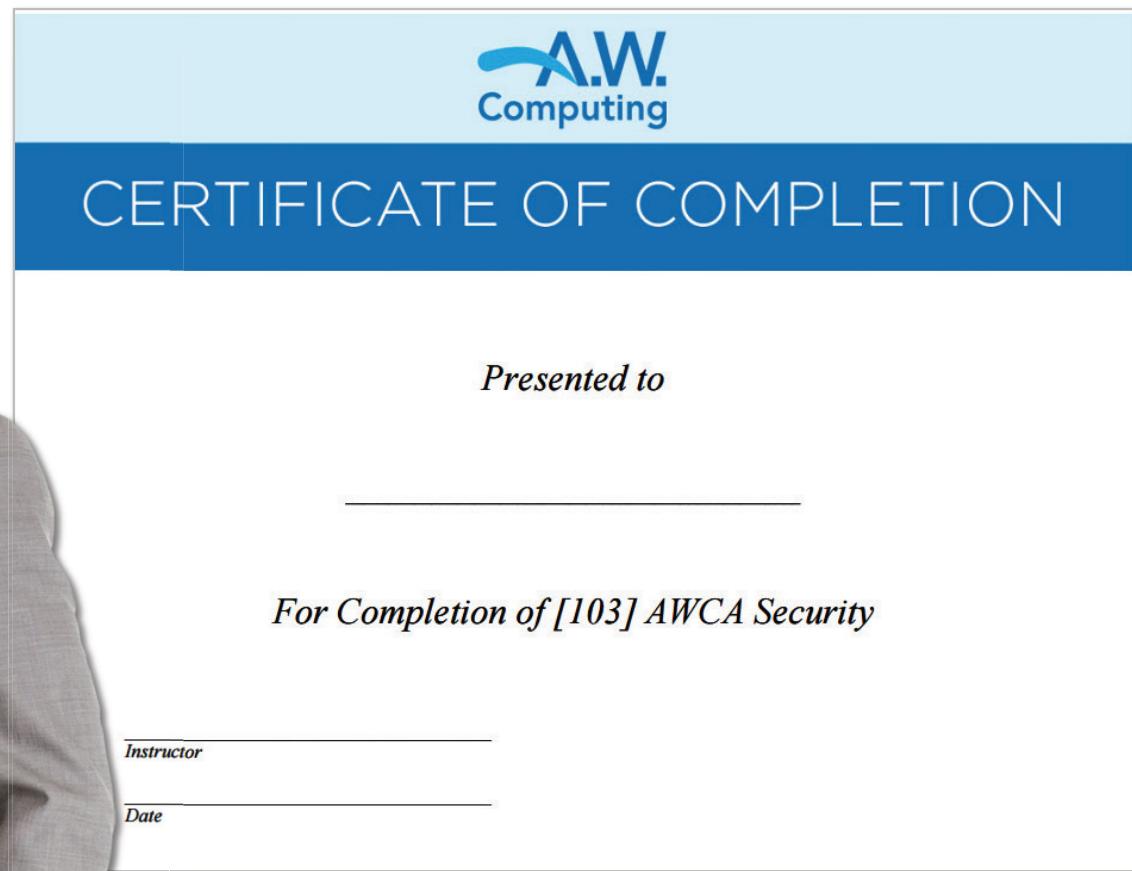


# CREATING A COURSE COMPLETION CERTIFICATE

394

salesforce

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.



**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:



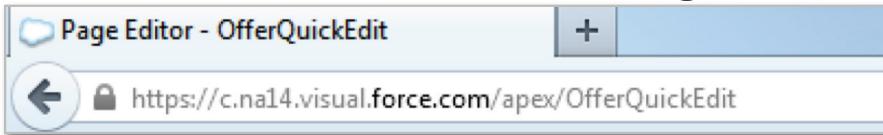
The record Id being passed in must be for a record of the same object as that specified by the standardController attribute.

# LAUNCHING VISUALFORCE PAGES

396

salesforce

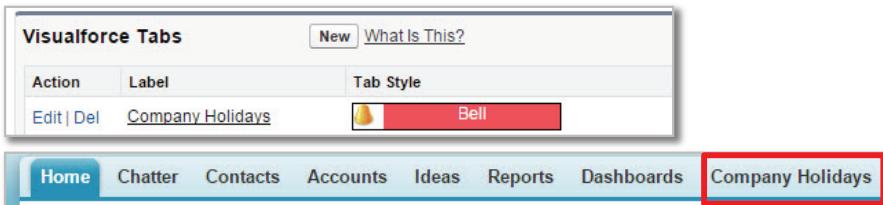
- Provide the URL for the page.



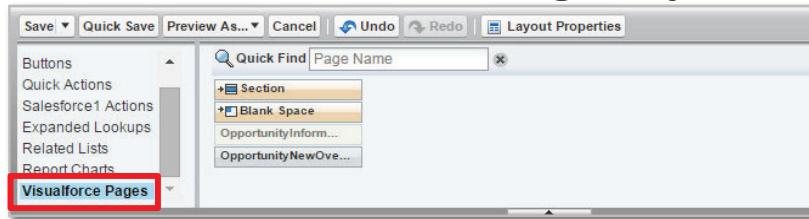
- Create a custom button or link.
- Override one of the standard actions.

Standard Buttons and Links				
Action	Label	Name	Overridden	Display
Edit	Offers Tab	Tab	<input type="checkbox"/>	Standard Salesforce.com Page
Edit	List	List	<input type="checkbox"/>	Standard Salesforce.com Page
Edit	<u>View</u>	View	<input checked="" type="checkbox"/>	<u>OfferQuickEdit (Visualforce Page)</u>

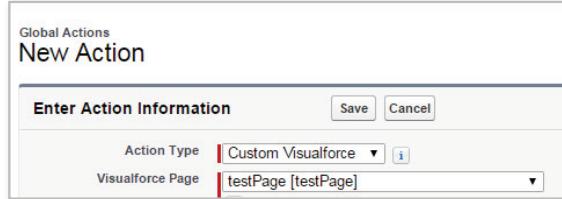
- Create a custom Visualforce tab.



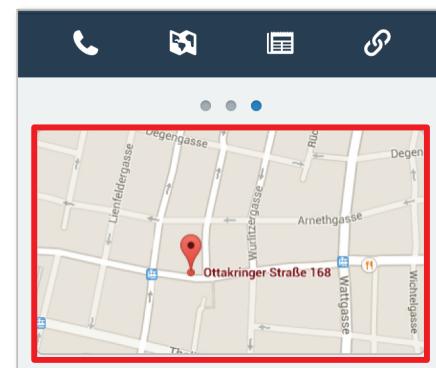
- Include inline in a page layout.



- Create a custom action.



- Create a mobile card.



# LAUNCHING A VISUALFORCE PAGE WITH A CUSTOM BUTTON

397



Contact Custom Button or Link

## New Button or Link

**Custom Button or Link Edit**

Save Quick Save Preview Cancel

**Label**  (highlighted with a yellow background)

**Name**  i

**Description**

**Display Type**

**Detail Page Link** [View example](#)

**Detail Page Button** [View example](#)

**List Button** [View example](#)

**Behavior**  ▼ View Behavior Options

**Content Source**  ▼

**Content**  ▼

Save Quick Save Preview Cancel

NOTE:



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

**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

399

salesforce

- 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



Ryan Jackson

Lead Developer



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 AGENDA

403

salesforce

### 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

404



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:

```
1A  <apex:page standardController="Contact">
2A      <apex:outputField value="{!contact.Account.Owner.FirstName}">
3A  </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:

```
1B  <apex:page standardController="Account">
2B      <apex:pageBlock>
3B          <apex:pageBlockTable value="{!account.Contacts}">
4B          ...
5B      </apex:page>
```

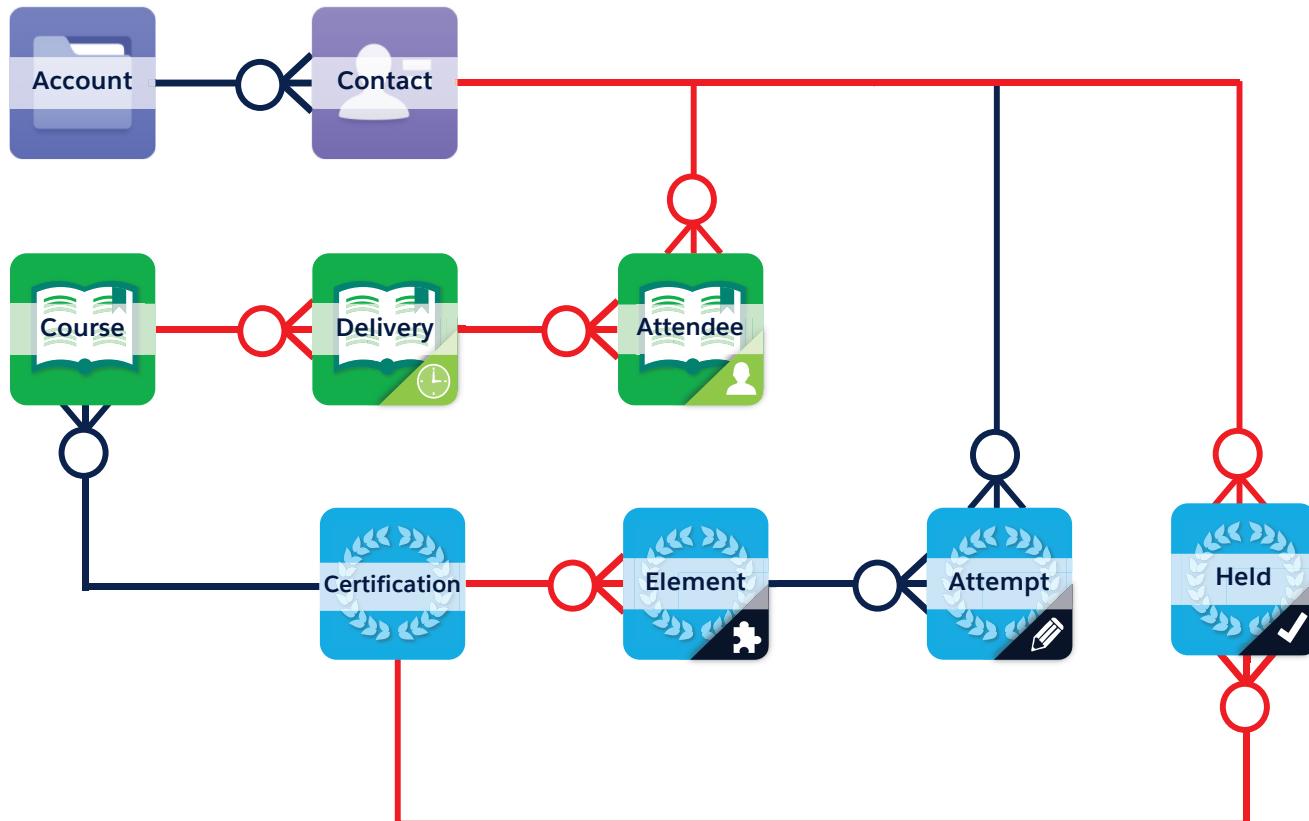


# DETERMINING WHICH CONTROLLER TO USE

405

salesforce

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. 



# CREATING A TECHNICIAN STATUS PAGE

406

salesforce

You need to create a technician status page that looks like this:



Record Detail  
(`<apex:detail/>`)

Contact  
Leo Moreau

Customize Page | Printable View | Help for this Page

Contact Detail

Contact Owner	Joseph Simmons [Change]	Phone	33 0 77 98 41 35
Name	Leo Moreau	Mobile	
Account Name	Marais Tech Solutions	Other Phone	
Title	Technician	Email	lmoreau-training@example.com
Contact Record Type	Technician [Change]	Reports To	Zoe Riccio [View Org Chart]
Certifications Held	1	Status	Active

Address Information

Mailing Address: 16 Rue Chapon, Paris, 75003 France

Other Address: 16 Rue Chapon, Paris, 75003 France

System Information

Created By: Joseph Simmons, 9/8/2015 2:07 PM

Last Modified By: Jay Rao, 9/8/2015 2:07 PM

Courses Attended

Start Date	Name	Certification Name	Certification Description
4/4/2015	Viviane Boudin	AWCA Server	AW Computing Certified Associate Server
4/18/2015	Viviane Boudin	AWCP Network	AW Computing Certified Professional Network

List of Courses Attended

# HOW CAN YOU DISPLAY CHILD RECORDS IN A VISUALFORCE PAGE?

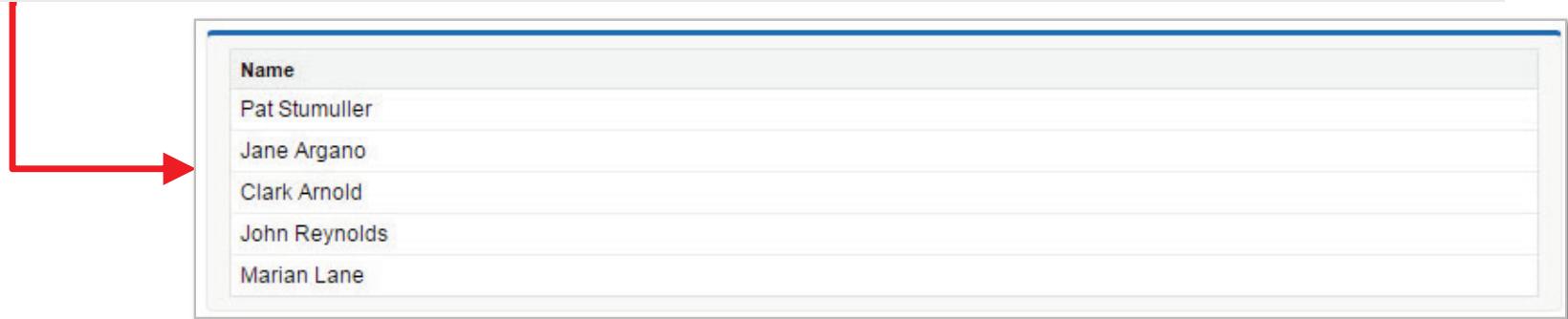
407

salesforce

The name of the variable that represents one element of the collection.

Identifies the collection to be displayed.

```
1 <apex:page standardController="Account">
2     <apex:pageBlock>
3         <apex:pageBlockTable var="item" value="{!account.Contacts}">
4             <apex:column value="{!item.name}" />
5         </apex:pageBlockTable>
6     </apex:pageBlock>
7 </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?

408

salesforce

1. Custom button or link.
2. **Override a standard action.**
3. Custom Visualforce tab.
4. Inline in a page layout.

**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

410

salesforce

### MODULE 16: EXPLORING THE VIEW AND CONTROLLER LAYERS OF VISUALFORCE

- Accessing Data on Related Records
- **Exploring Visualforce Tags and Built-in Styling**





# REFINE THE TECHNICIAN STATUS PAGE

411

salesforce

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	<a href="mailto:lmoreau-training@example.com">lmoreau-training@example.com</a>
Phone	33 0 77 98 41 35	Account	Marais Tech Solutions

**Course Listing**

Date	Name	Certification Name	Certification Description
<a href="#">Sat Apr 04 00:00:00 GMT 2015</a>	Viviane Boudin	AWCA Server	AW Computing Certified Associate Server
<a href="#">Sat Apr 18 00:00:00 GMT 2015</a>	Viviane Boudin	AWCP Network	AW Computing Certified Professional Network
<a href="#">Sat Jun 13 00:00:00 GMT 2015</a>	Viviane Boudin	AWCA Server	AW Computing Certified Associate Server

[Edit Technician Record](#)

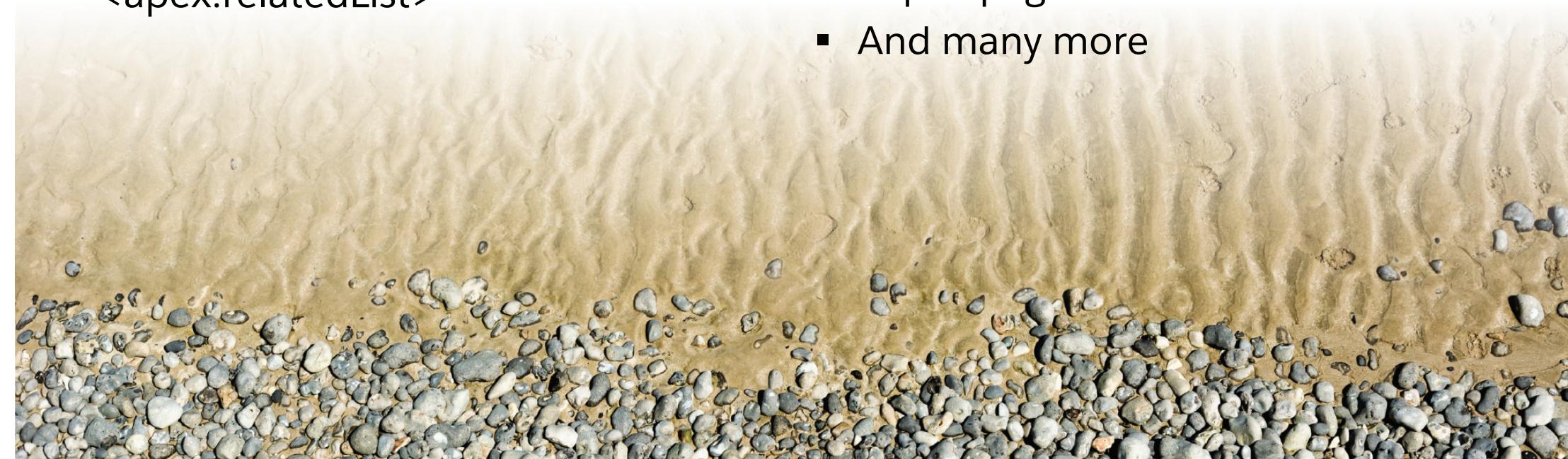
**Ryan Jackson**  
Lead Developer

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?

413

salesforce

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:  which calls the edit method on a standard controller.

Action-Aware Tags	Use Case
<code>&lt;apex:commandButton&gt;</code>	Creates a button that calls an action.
<code>&lt;apex:commandLink&gt;</code>	Creates a link that calls an action.
<code>&lt;apex:actionPoller&gt;</code>	Periodically calls an action. 
<code>&lt;apex:actionSupport&gt;</code>	Makes an event (such as “onclick,” “onmouseover,” etc.) on another (named) component call an action. 
<code>&lt;apex:actionFunction&gt;</code>	Defines a new JavaScript function that calls an action.
<code>&lt;apex:page&gt;</code>	Calls an action when the page is loaded.

The standard controller action methods include: save, quicksave, edit, delete, cancel, and list.



The `value` attribute specifies the destination of the link.

Option 1

The Link will resolve to the page layout for this record.

```
<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.

**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

416

salesforce

- 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



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.



## MODULE AGENDA

419

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





# REFERENCING CUSTOM CONTROLLER CODE

420

salesforce

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.

**Account Detail**

[Edit](#) [Delete](#) [Sharing](#)

Account Owner	<a href="#">Joseph Simmons [Change]</a>	Phone	(408) 555-6000
Account Name	<a href="#">Alveswood Technologies [View Hierarchy]</a>	Website	<a href="http://www.alveswoodtechnologies">http://www.alveswoodtechnologies</a>
Parent Account		Account Record Type	<a href="#">Service Vendor [Change]</a>
		Status	<a href="#">Active</a>

**Certifications Held**

Certification Held Number	Name	Date Achieved
CERTIFICATION-00091	Clara Morales	5/16/2015
CERTIFICATION-00092	Aaryn Patel	5/14/2015
CERTIFICATION-00093	Tammy Rogers	5/4/2015
CERTIFICATION-00094	Diana Tatro	5/4/2015



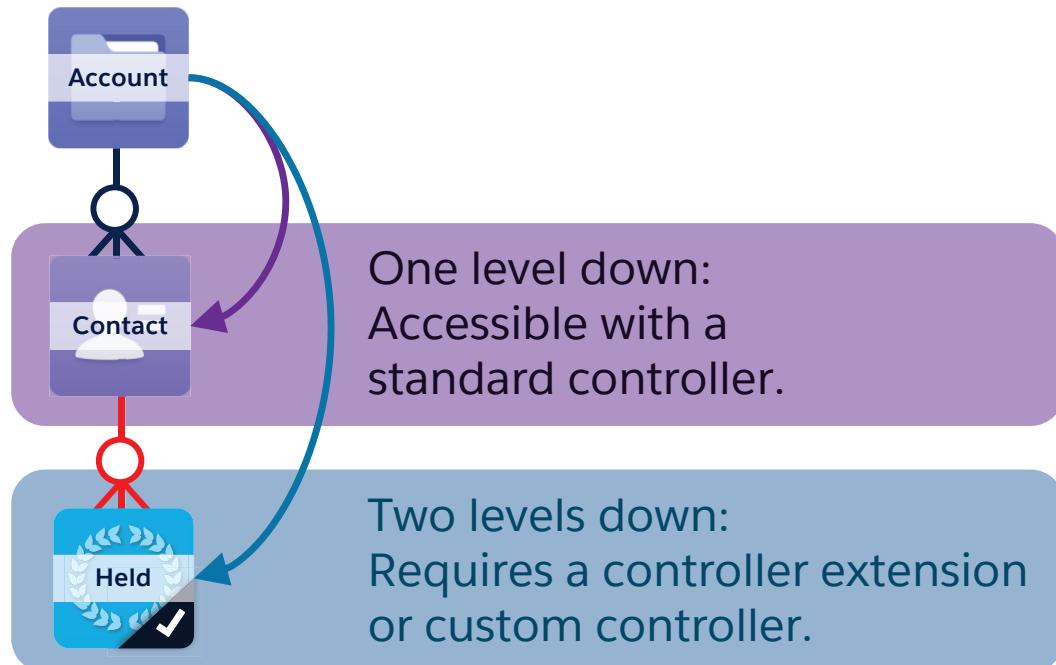
**Cassie Evans**  
Developer

# WHY CAN'T YOU USE A STANDARD CONTROLLER?

421

salesforce

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?

422



**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

```
1  public class TheController {  
2      String searchText;  
3  
4      public String getSearchText() {  
5          return searchText;  
6      }  
7  
8      public void setSearchText(String s) {  
9          searchText = s;  
10     }  
11  
12     public List<Lead> results { get; set; }  
13  
14     public PageReference doSearch() {  
15         results = (List<Lead>) [ FIND :searchText  
16                                     RETURNING Lead(Name, Email) ] [0];  
17  
18     }  
19 }
```

# WHY MUST A PAGE USE A CONTROLLER EXTENSION?

423

salesforce

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

424



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.

- 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

426



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:

```
1  public with sharing class AccountDisplayCertsHeld_CX {  
2  
3      public List<Certification_Held__c> results {  
4          get {  
5              if (results == null) {  
6                  results = [SELECT Id, Name, Date_Achieved__c,  
7                               Certified_Professional__r.Name  
8                               FROM Certification_Held__c  
9                               WHERE Certified_Professional__r.Account.Id = : account.Id];  
10             }  
11             return results;  
12         }  
13         private set;  
14     }  
15  
16     private final Account account;  
17  
18     // Constructor used to get the Account record  
19     public AccountDisplayCertsHeld_CX(ApexPages.StandardController stdController) {  
20         this.account = (Account) stdController.getRecord();  
21     }  
22 }
```

In a Visualforce page, you can reference the list of results using expression syntax: {!results}

**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 AGENDA

428

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 A READ-ONLY PROPERTY

429

salesforce

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



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.

```
1 public class myControllerExtension {  
2     private final Account acct;  
3     public myControllerExtension(ApexPages.StandardController stdController) {  
4         this.acct = (Account)stdController.getRecord();  
5     }  
6     public String getGreeting() {  
7         return 'Hello ' + acct.name + ' (' + acct.id + ')';  
8     }  
9 }
```



NOTE:

Other than the constructor, custom controllers can contain all the same elements seen in controller extensions.

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

```
1  public with sharing class MyController {  
2      public String getName() {  
3          return 'MyController';  
4      }  
5      public Account getAccount() {  
6          return [SELECT Id, Name from Account  
7              WHERE Id = :ApexPages.currentPage().getParameters().get('Id')];  
8      }  
8  }
```



NOTE:

Every value calculated by a controller and displayed in a page must have a corresponding getter method or property.

# REFERENCING A GETTER METHOD

432

salesforce

In a Visualforce page, data from a getter method can be accessed using the `{ !DataName }` expression syntax.

```
1A  public with sharing class MyController {  
2A      public String getName() {  
3A          return 'MyController';  
4A      }  
5A  }
```

```
1B  <apex:page controller="MyController">  
2B      This page is using the { !name } controller!  
3B  </apex:page>
```

Setter methods pass user-specified values from a page to a controller.

```
1  public with sharing class MySearchController {  
2      String privateSearchText;  
3      List<Lead> results;  
4      public String getSearchText() {  
5          return privateSearchText;  
6      }  
7      public void setSearchText(String s) {  
8          privateSearchText = s;  
9      }  
10 }
```

Setter methods always take the form `setDataName()`.



NOTE:

Every time an action method is invoked, all setter methods are automatically executed before the action method.

Visualforce calls setter on primitives only. Controllers have getters returning sObjects but do not need setters for non-primitive values.

```
1 public class myControllerExtension {  
2     private final Account acct;  
3     public myControllerExtension(ApexPages.StandardController  
4                                     stdController) {  
5         this.acct = (Account) stdController.getRecord();  
6     }  
7     public String getGreeting() {  
8         return 'Hello ' + acct.name + ' (' + acct.id + ')';  
9     }  
}
```

This stores the account object in the controller.

- 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:
  - If no additional functionality is required, you can use automatic properties to make coding easier:

```
1A  public class BasicProperty {  
2A      public integer prop {  
3A          get { return prop; }  
4A          set { prop = value; }  
5A      }  
6A  }
```

```
1B  public class AutomaticProperty {  
2B      public integer MyReadOnlyProp { get; }  
3B      public double MyProperty { get; private set; }  
4B  }
```



NOTE:

Properties with a private setter or without a setter are read-only.

**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

437

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



# 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](#)

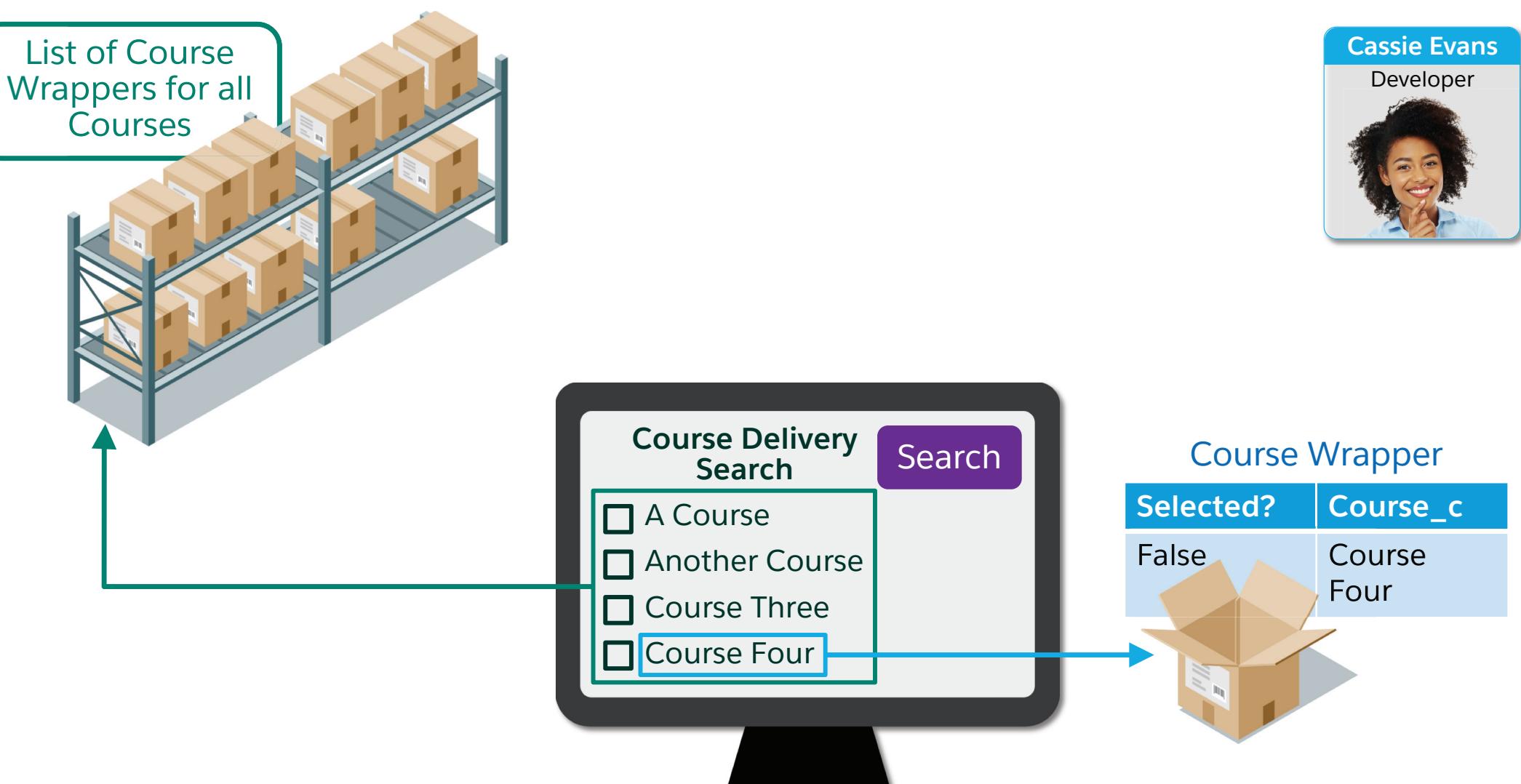
Select	Course Name	Certification Name	Duration	Status
<input checked="" type="checkbox"/>	[201] AWCP Server	AWCP Server	5	Active
<input type="checkbox"/>	[102] AWCA Network	AWCA Network	4	Active
<input type="checkbox"/>	[103] AWCA Security	AWCA Security	4	Active
<input type="checkbox"/>	[302] AWCM Network	AWCM Network	5	Active
<input type="checkbox"/>	[203] AWCP Security	AWCP Security	5	Active
<input type="checkbox"/>	[101] AWCA Server	AWCA Server	4	Active
<input checked="" type="checkbox"/>	[301] AWCM Server	AWCM Server	5	Active
<input checked="" type="checkbox"/>	[303] AWCM Security	AWCM Security	5	Active
<input type="checkbox"/>	[202] AWCP Network	AWCP Network	5	Active
<input type="checkbox"/>	[401] Data Recovery		3	Active
<input type="checkbox"/>	[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?

440

salesforce

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.

```
1 public class CourseWrapper {  
2     public Course__c course {get; set;}  
3     public Boolean checked {get; set;}  
4  
5     public CourseWrapper(Course__c c) {  
6         course = c;  
7         checked = false;  
8     }  
9 }
```

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



**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.

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



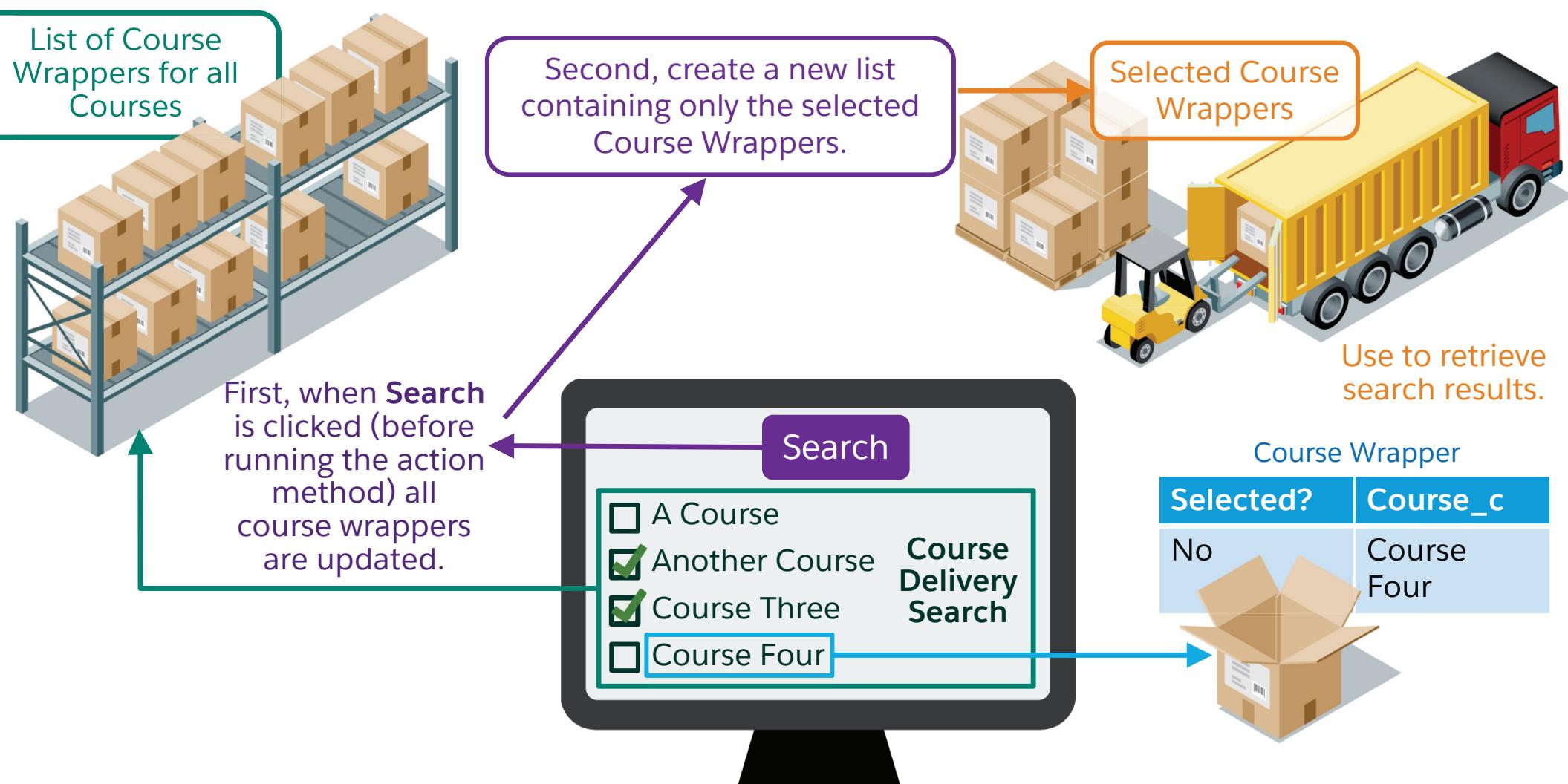
**Cassie Evans**  
Developer



# HOW WILL THE SEARCH PAGE WORK?

443

salesforce



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" >
```

**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

446

salesforce

After the user clicks **Search**, your page should deliver them to a page that displays all of the results of their search like this:



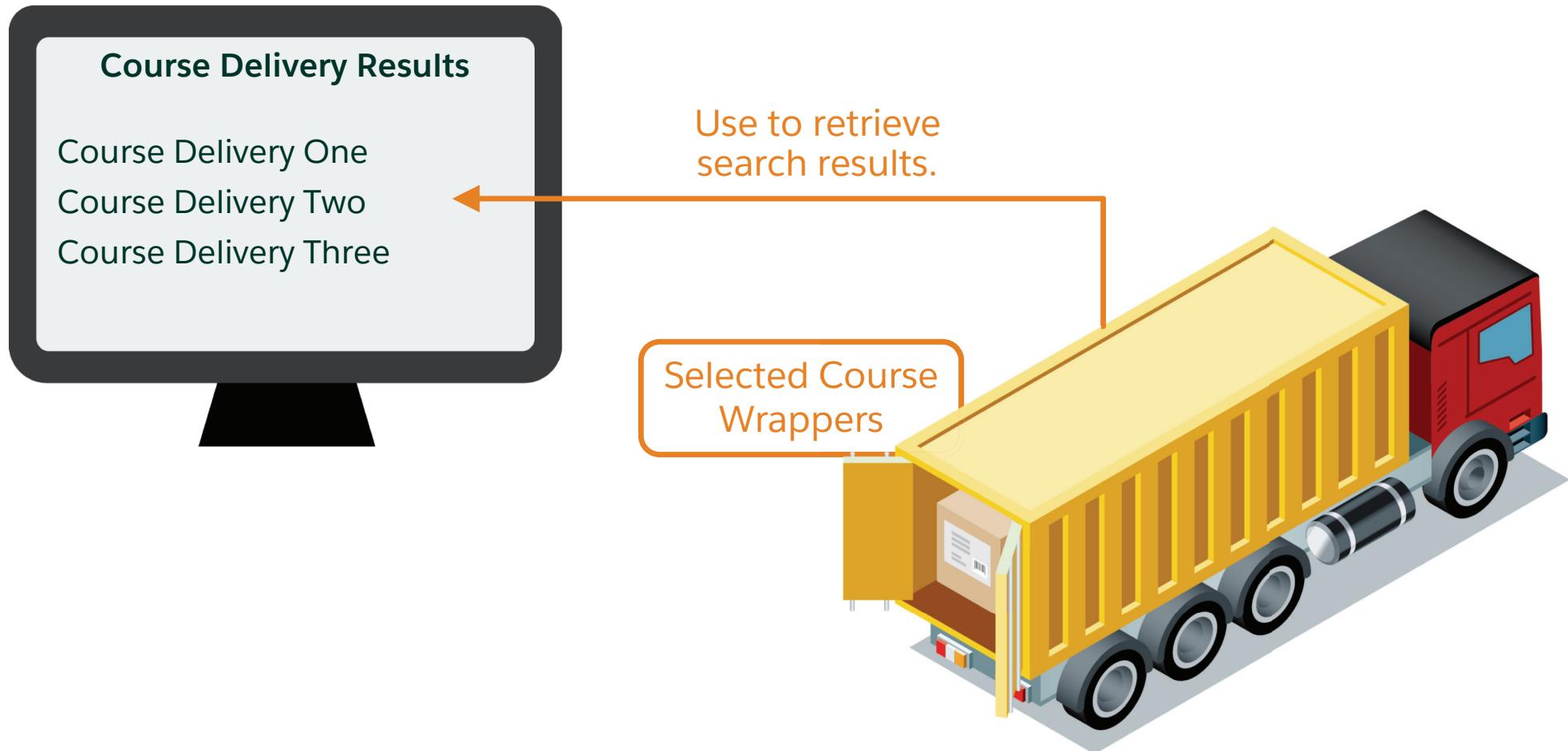
**Cassie Evans**  
Developer

Upcoming Course Deliveries					
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

# HOW WILL THE SEARCH PAGE WORK?

447

salesforce



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)

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)	PageReference pageRef = new PageReference('partialURL');
Salesforce Page	
Non-Salesforce Website	PageReference pageRef = new PageReference('fullURL');



# WORKING WITH URL PARAMETERS

450

salesforce

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

451



To navigate to a new page as part of an action method, return a PageReference:

```
1 public class MySecondController {  
2     Account account;  
3     public Account getAccount() {  
4         if (account == null) account = new Account();  
5         return account;  
6     }  
7     public PageReference save() {  
8         insert account;  
9         PageReference acctPage = new ApexPages.StandardController(account).view();  
10        acctPage.setRedirect(true);  
11        return acctPage;  
12    }  
13 }
```

**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

453

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**



```
1A public PageReference save() {  
2A     try {  
3A         update record;  
4A     }  
5A     catch(System.DMLEException ex) {  
6A         ApexPages.addMessages(ex);  
7A         return null;  
8A     }  
9A  
10A    return new PageReference('/' + record.Id);  
11A }
```

To handle any exception that might be thrown, use try and catch blocks.

```
1B public void check() {  
2B     if ( cert1 == cert2 ) {  
3B         ApexPages.addMessage(new  
4B             ApexPages.message(ApexPages.Severity.INFO,  
5B                 'YEAH! Certifications match'));  
6B     } else {  
7B         ApexPages.addMessage(new  
8B             ApexPages.message(ApexPages.Severity.ERROR,  
9B                 'Certifications do not match'));  
10B    }  
11B }
```

For other error conditions, write custom logic.



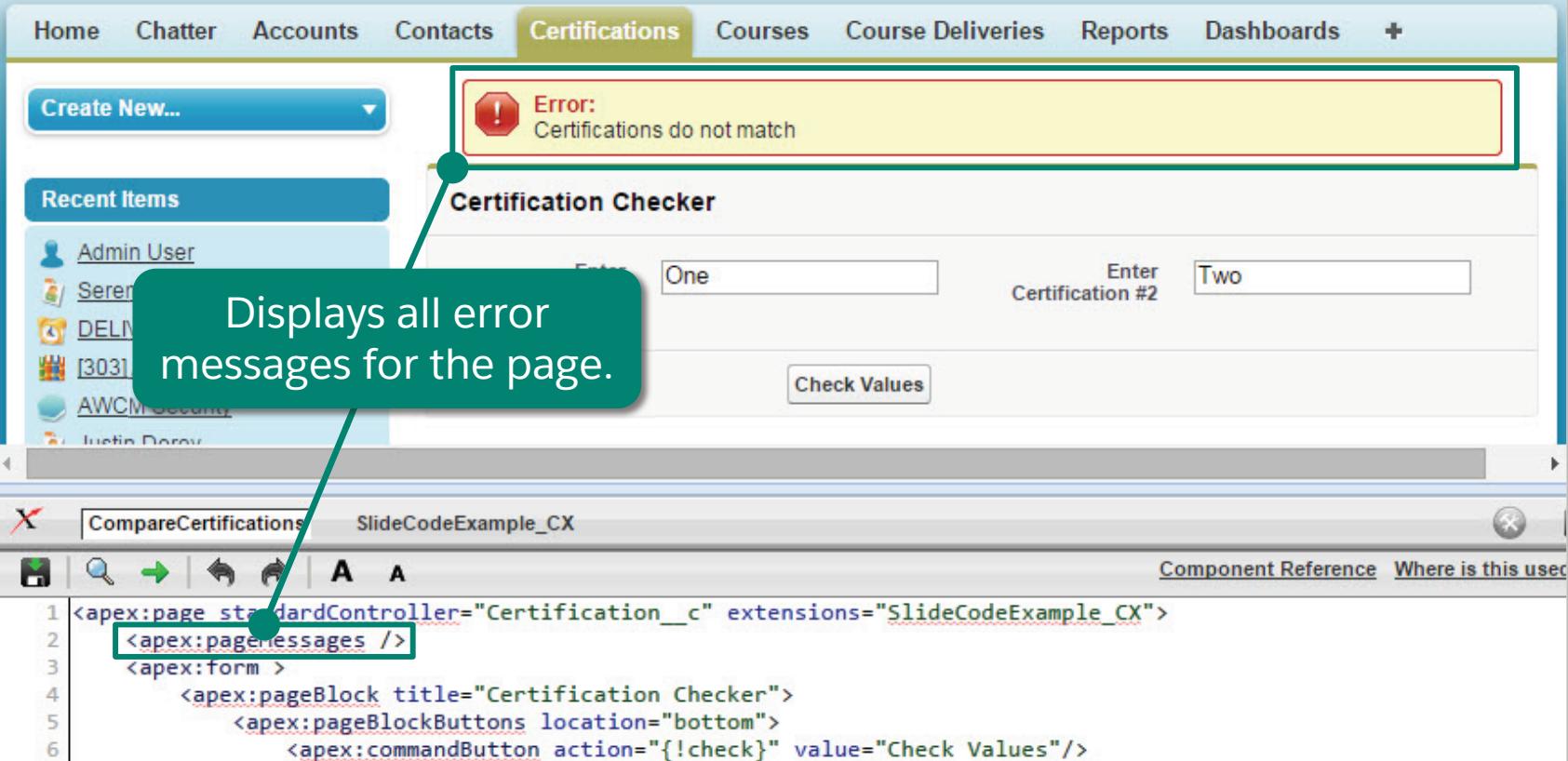
NOTE:

Use the `ApexPages.addMessages` method to add messages to the page.

# DISPLAYING ERROR MESSAGES IN VISUALFORCE PAGES

455

salesforce



Creates a red error message box with a yellow background and a red border. The message box contains an exclamation mark icon and the text 'Error: Certifications do not match'. A green callout bubble with a white background and a green border points to the message box, containing the text 'Displays all error messages for the page.' The Visualforce code editor shows the following markup:

```
1 <apex:page standardController="Certification__c" extensions="SlideCodeExample_CX">
2   <apex:pageMessages />
3   <apex:form>
4     <apex:pageBlock title="Certification Checker">
5       <apex:pageBlockButtons location="bottom">
6         <apex:commandButton action="{!check}" value="Check Values"/>
```



The `<apex:pageMessage>` tag places messages on a Visualforce page with specific styling. When you use the `<apex:form>` tag, use `<apex:pageMessages>`.

**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

457

salesforce

- 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

458

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?

```
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



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.

# EXPLORING GLOBAL SEARCH

461

salesforce

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
<a href="#">Edit</a>	<a href="#">Acme Inc (London)</a>	1-212-555-7500	<a href="#">froberts</a>
<a href="#">Edit</a>	<a href="#">Acme Inc</a>	1-212-555-5555	<a href="#">vmyers</a>

### Contacts (1)

Action	Name	Account Name	Phone	Email	Contact Owner Alias
<a href="#">Edit</a>	<a href="#">Shelby Cooper</a>	<a href="#">Acme Inc (London)</a>	1-212-555-7503	<a href="#">shelby.cooper@trainingorg-acme-ny.com</a>	<a href="#">froberts</a>

### Opportunities (4)

Action	Opportunity Name	Account Name	Stage	Close Date	Opportunity Owner Alias
<a href="#">Edit</a>	<a href="#">Acme Inc - 700 Desktops</a>	<a href="#">Acme Inc</a>	Qualification	7/30/2015	<a href="#">vmyers</a>
<a href="#">Edit</a>	<a href="#">Acme Inc - 20 Desktops</a>	<a href="#">Acme Inc</a>	Proposal/Price Quote	9/8/2015	<a href="#">vmyers</a>
<a href="#">Edit</a>	<a href="#">Acme Inc - 6 Desktops</a>	<a href="#">Acme Inc</a>	Closed Won	5/17/2015	<a href="#">vmyers</a>
<a href="#">Edit</a>	<a href="#">Acme Inc - 600 Desktops</a>	<a href="#">Acme Inc</a>	Closed Lost	5/10/2015	<a href="#">vmyers</a>



## MODULE AGENDA

462

salesforce

### 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

463

salesforce

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" >
```

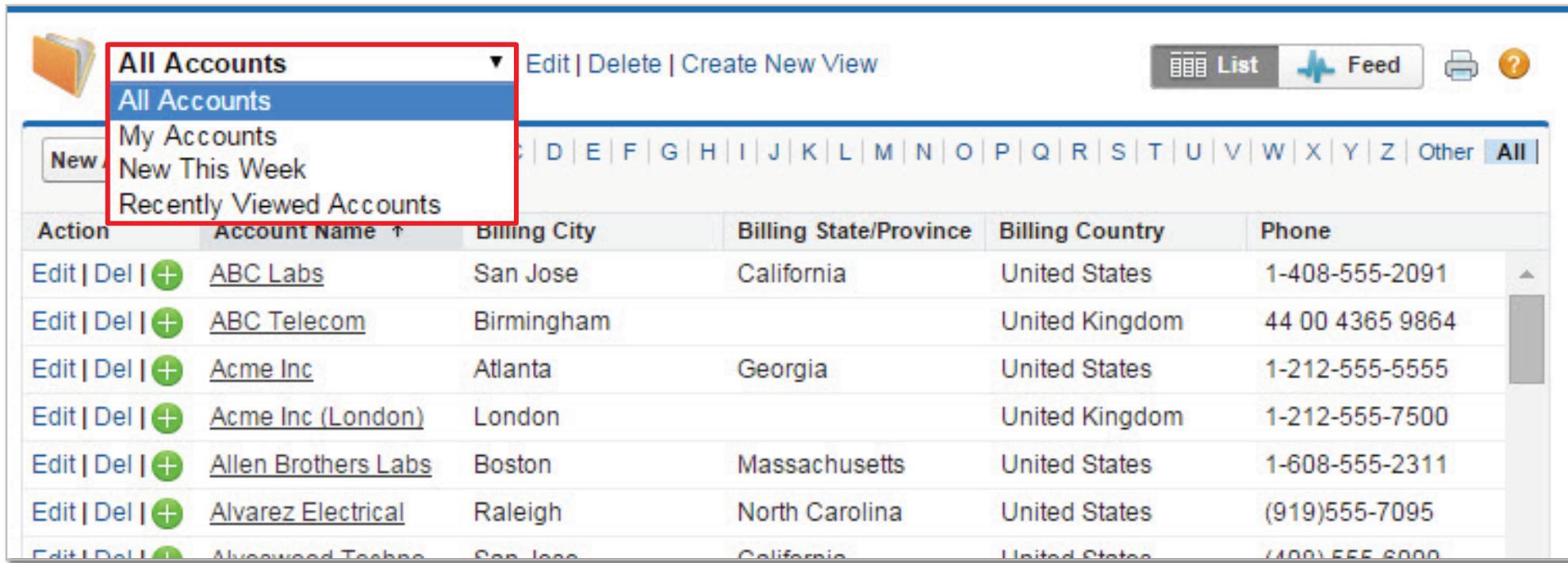
The value of the `recordSetVar` attribute becomes the name of the property that enables page access to the resulting list of records.

# LIST VIEWS AND STANDARD LIST CONTROLLERS

464

salesforce

A list view allows you to quickly view a specific set of records.



Action	Account Name	Billing City	Billing State/Province	Billing Country	Phone
Edit   Del   +	<a href="#">ABC Labs</a>	San Jose	California	United States	1-408-555-2091
Edit   Del   +	<a href="#">ABC Telecom</a>	Birmingham		United Kingdom	44 00 4365 9864
Edit   Del   +	<a href="#">Acme Inc</a>	Atlanta	Georgia	United States	1-212-555-5555
Edit   Del   +	<a href="#">Acme Inc (London)</a>	London		United Kingdom	1-212-555-7500
Edit   Del   +	<a href="#">Allen Brothers Labs</a>	Boston	Massachusetts	United States	1-608-555-2311
Edit   Del   +	<a href="#">Alvarez Electrical</a>	Raleigh	North Carolina	United States	(919)555-7095
Edit   Del   +	<a href="#">Alvewood Techco</a>	San Jose	California	United States	(408) 555-6000

By default, Visualforce pages built using a standard list controller display records using the user's most recently accessed list view.

What will this code display?

```
1 <apex:page standardController="Account" recordSetVar="accounts" >
2     <apex:pageBlock >
3         <apex:pageBlockTable value="{!accounts}" var="acc">
4             <apex:column value="{!acc.name}"/>
5         </apex:pageBlockTable>
6     </apex:pageBlock>
7 </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.

```
1 <apex:pageBlock title="Accounts" id="accountList">
2   <apex:selectList value="{!filterId}" size="1">
3     <apex:selectOptions value="{!listViewOptions}" />
4     <apex:actionSupport event="onchange" reRender="accountList"/>
5   </apex:selectList>
6   ...
7 </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

467

salesforce

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.

```
1  <apex:pageBlock title="Contacts List" id="contacts list">
2
3      <apex:selectList value="{!!filterId}" size="1">
4          <apex:selectOptions value="{!!listViewOptions}" />
5          <apex:actionSupport event="onchange" reRender="contacts list"/>
6      </apex:selectList>
7      ...
8  </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.

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

469

salesforce

```
1  <apex:pageBlock id="accountList" title="Accounts">
2      <apex:pageMes: Calls the standard first() method.
3
4          <apex:pageBlockButtons >
5              <apex:commandButton status="notifyUser" reRender="accountList" value="|<" title="First" action=" {!first}" disabled=" {!NOT(HasPrevious)}" />
6              <apex:commandButton status="notifyUser" reRender="accountList" value="<" title="Previous" action=" {!previous}" disabled=" {!HasPrevious}" />
7              <apex:commandButton status="notifyUser" reRender="accountList" value=">" title="Next" action=" {!next}" disabled=" {!HasNext}" />
8              <apex:commandButton status="notifyUser" reRender="accountList" value=">|>" title="Last" action=" {!last}" disabled=" {!HasNext}" />
9
10             <apex:outputText >
11                 {! (pageNumber * pageSize) + 1 - pageSize}
12                     through
13                     {!IF((pageNumber * pageSize) > resultSize, resultSize, (pageNumber *
14                         pageSize))}
15                     of {!resultSize} records
16             </apex:outputText>
17         </apex:pageBlockButtons>
18     ...
19
20     Disables the button if there
21     are no previous records.
22
23     Indicates which records
```

Disables the button if there are no previous records.

Indicates which records are on the current page.

## 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

471

salesforce

### 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

472

salesforce

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.

```
1 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

473

salesforce

```
[ 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	Acme
"Global	Global Media
Media"	

# THE IN CLAUSE: SPECIFYING A SEARCH GROUP

474

salesforce

```
[ 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 <code>nameField = true</code> .	Includes all email fields.	Includes all phone number fields.

```
[ 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) ];
```

You can execute SOSL searches inside:

- APIs.
- Apex statements, using:
  - Bracket notation.

```
[ Find 'Acme' RETURNING Account ]
```

- `Search.query()`.

```
Search.query('Find {Acme} Returning Account')
```

## 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.

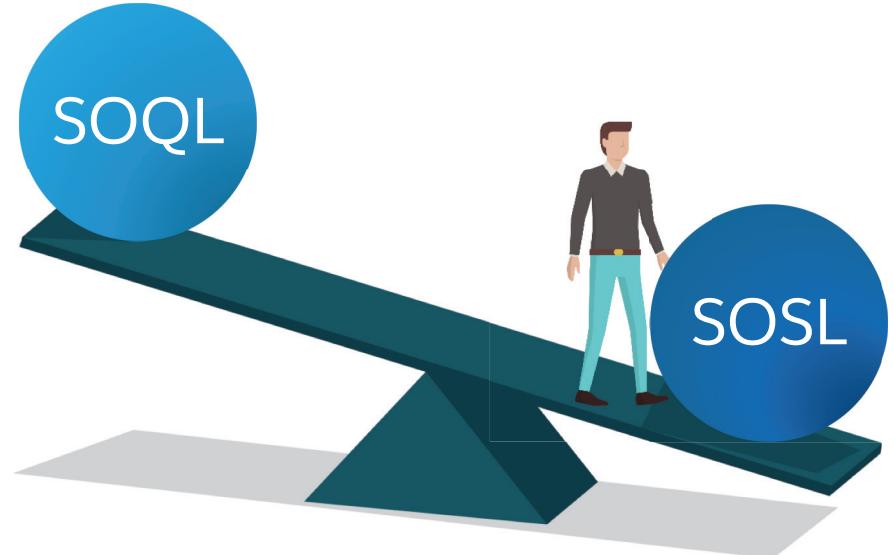
Attention, SOSL pas utilisable dans le CONTEXTE trigger !

## 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?

479

salesforce

SOSL has only one return data type: a List of Lists of sObjects.

```
1 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][0]: 'Acme - 1,200 Widgets' acmes[1][1]: 'Acme - 200 Widgets'
acmes[1] (Opportunities)	acmes[1][2]: 'Acme - 600 Widgets'

Each sObject list contains those sObjects found meeting the search criteria.

# USE MULTIPLE FOR LOOPS TO PROCESS A `<List<List<sObject>>`

480

salesforce

```
1 List<List<sObject>> acmes =  
    [ Find 'Acme'  
      RETURNING Account(Name), Opportunity(Name) ];
```

Find all Account and Opportunity records containing 'Acme.'

```
2  
3 //The first sObject is the List of Accounts  
4 List<Account> acmeAccounts = acmes[0];  
5 for (Account acmeAccount : acmeAccounts) {  
6     System.debug('Account: ' + acmeAccount.Name);  
7 }
```

Parse the Accounts that have been returned.

```
8  
9 //The second sObject is the List of Opportunities  
10 List<Opportunity> acmeOpportunities = acmes[1];  
11 for (Opportunity acmeOpportunity : acmeOpportunities) {  
12     System.debug('Opportunity: ' + acmeOpportunity.Name);  
13 }
```

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

481



You can bind expressions and variables in SOSL searches.

Bound expression

```
1A List<List<sObject>> acmes1 = [ Find :('Ac' + 'me') RETURNING Account ];
```

Bound variable

```
1B String company = 'Acme';
2B List<List<sObject>> acmes2 = [ Find :company RETURNING Account ];
```

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.

**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 AGENDA

484

salesforce

### MODULE 18: WORKING WITH LIST CONTROLLERS AND SOSL QUERIES

- Working with Standard List Controllers
- Writing a Simple SOSL Query
- **Creating a Custom List Controller**





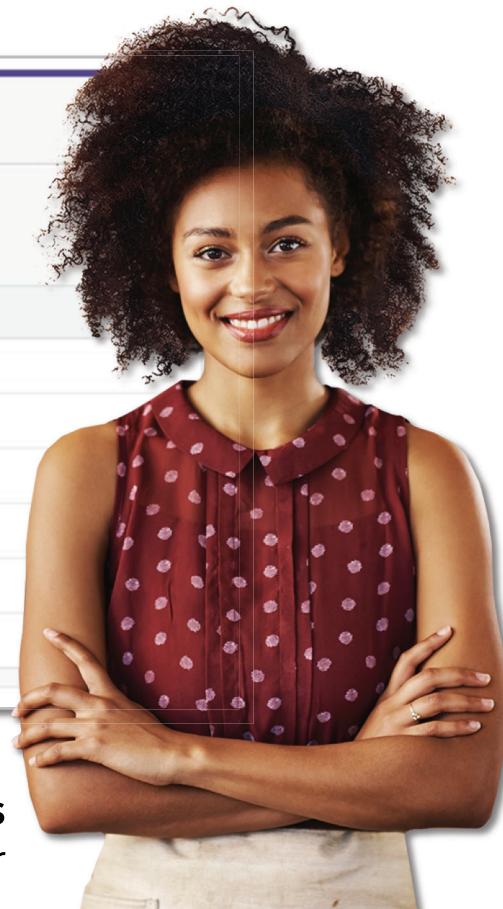
# CREATING A SIMPLE SEARCH PAGE

485

salesforce

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:

<b>Search Contacts</b>			
<b>Search Text:</b> <input type="text" value="Acme*"/> <b>Search</b> <b>&lt;</b> <b>&lt;</b> <b>&gt;</b> <b>&gt;</b>			
<b>Name</b>	<b>Account Name</b>	<b>Phone</b>	<b>Email</b>
<a href="#">Shelby Cooper</a>	Acme Inc (London)	1-212-555-7503	<a href="mailto:shelby.cooper@trainingorg-acme-ny.com">shelby.cooper@trainingorg-acme-ny.com</a>
<a href="#">Valerie Wilson</a>	Acme Inc	1-212-555-5006	<a href="mailto:valerie.wilson@trainingorg-acmeinc.com">valerie.wilson@trainingorg-acmeinc.com</a>
<a href="#">Leanne Tomlin</a>	Acme Inc	1-212-555-5002	<a href="mailto:leanne.tomlin@trainingorg-acmeinc.com">leanne.tomlin@trainingorg-acmeinc.com</a>
<a href="#">Sandra Deely</a>	Acme Inc	1-212-555-5003	<a href="mailto:sandra.deely@trainingorg-acmeinc.com">sandra.deely@trainingorg-acmeinc.com</a>
<a href="#">Carson Connelly</a>	Acme Inc	1-212-555-5004	<a href="mailto:carson.connelly@trainingorg-acmeinc.com">carson.connelly@trainingorg-acmeinc.com</a>
<a href="#">Dave Carroll</a>	Acme Inc	1-212-555-5005	<a href="mailto:dave.carroll@trainingorg-acmeinc.com">dave.carroll@trainingorg-acmeinc.com</a>



**Cassie Evans**  
Developer

# CREATING A CUSTOM LIST CONTROLLER

486

salesforce

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.

```
1  public class DisplayCertifications_CC {  
2      // Standard Set Controller  
3      public ApexPages.StandardSetController setCon {  
4          get {  
5              if(setCon == null) {  
6                  setCon = new ApexPages.StandardSetController(Database.getQueryLocator(  
7                      [SELECT Name, Status__c FROM Certification__c]));  
8              }  
9              return setCon; }  
10             set;  
11     }  
12     // Return a list of records  
13     public List<Certification__c> getCertifications() {  
14         return (List<Certification__c>) setCon.getRecords();  
15     }  
16 }
```

Instantiate the standardSetController to use built-in set controller methods.

Include your own custom methods.

# INSTANTIATING THE StandardSetController CLASS

487



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'));
```

```
1  public with sharing class DisplayContacts_CC {  
2      public String searchText {get; set;}  
3      ...  
4      public void Search() {  
5          if (String.isNotBlank(searchText) && searchText.length() > 1) {  
6              // Search for Contacts  
7              searchText = searchText + '*';  
8              List<sObject> contacts = [FIND :searchText IN ALL FIELDS  
9                                         RETURNING Contact (Id, Name, Email, Phone)][0];  
10             setCon = new ApexPages.StandardSetController(contacts);  
11             ...  
12         }  
13     }  
14 }
```

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.

**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

490

salesforce

- 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

491

salesforce

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

PROGRAMMATIC DEVELOPMENT USING APEX AND VISUALFORCE



Ryan Jackson

Lead Developer



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

494

salesforce

### 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

495

salesforce

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.

**Goal:**

Determine which scenarios are best solved using Visualforce.

**Task:**

Review each scenario, and determine if Visualforce would be the best solution.



## MODULE AGENDA

497

salesforce

### 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?

498



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?

499

salesforce

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.

Declaring variables as transient reduces view state size.

```
1A <apex:page controller="ExampleController">  
2A     T1: {!t1} <br/>  
3A     T2: {!t2} <br/>  
4A     <apex:form>  
5A         <apex:commandLink value="refresh"/>  
6A     </apex:form>  
7A </apex:page>
```

```
1B public class ExampleController {  
2B     DateTime t1;  
3B     transient DateTime t2;  
4B     public String getT1() {  
5B         if (t1 == null) t1 = System.now();  
6B         return '' + t1;  
7B     }  
8B     public String getT2() {  
9B         if (t2 == null) t2 = System.now();  
10B        return '' + t2;  
11B    }  
12B }
```

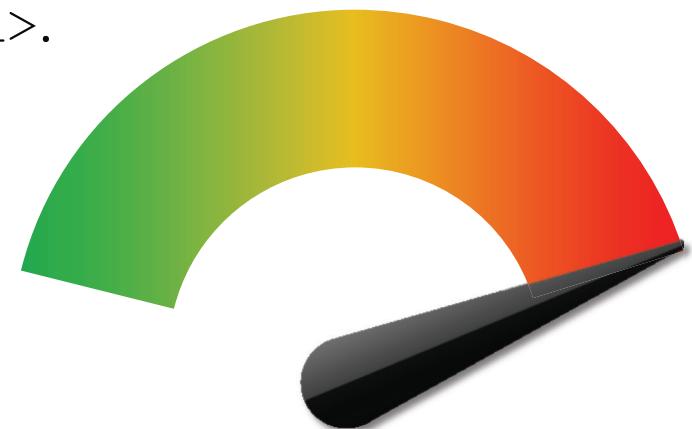
Use the transient keyword to declare instance variables that should not be included in the view state of a Visualforce page.

# MINIMIZING THE RISK OF HITTING GOVERNOR LIMITS

501

salesforce

- 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

502

salesforce

### 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



- 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.

- Use static queries and binding variables if possible.

The previous code can be rewritten like this:

```
1 String queryName = '%' + name + '%';
2 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.

**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.

**Jason Beck**

Beginning Developer





## MODULE AGENDA

507

salesforce

### 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

508

salesforce

- 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

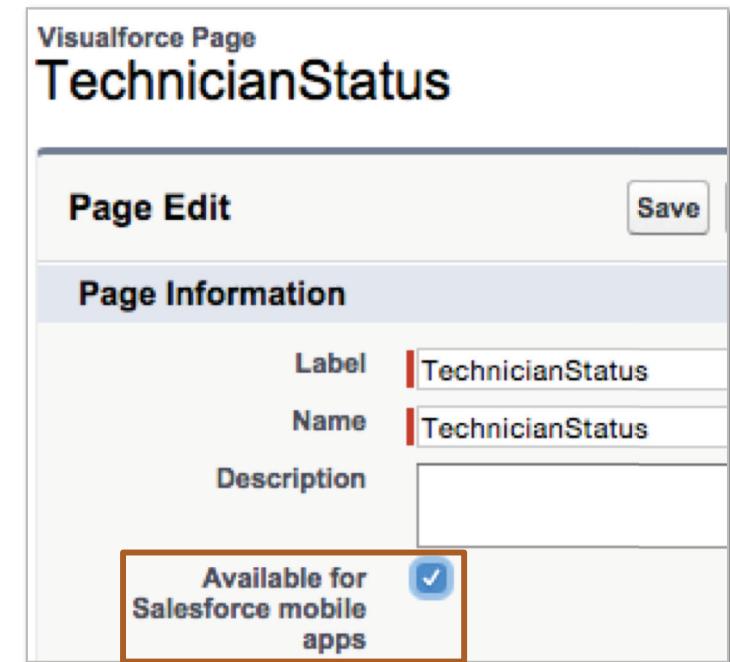
Visualforce Page  
**TechnicianStatus**

Page Edit Save

Page Information

Label	TechnicianStatus
Name	TechnicianStatus
Description	

**Available for Salesforce mobile apps**



# WHAT ARE LIGHTNING COMPONENTS?

509

salesforce

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

510

salesforce

## 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

511

salesforce

- 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.

Static Resource  
**customJS**

**Static Resource Detail**

<b>Name</b>	customJS	<b>Edit</b>	<b>Delete</b>	<b>Where is this used?</b>
<b>Namespace Prefix</b>				
<b>Description</b>	Custom code for various utility methods.			
<b>MIME Type</b>	text/javascript			

Upload the JavaScript files as Static Resources.

```
1 function changeFont(input, textid) {  
2 ... }
```

Use `<apex:includeScript value="{$Resource.customJS}" />` and `$.Resource` to reference the JavaScript file.

```
1 <apex:includeScript value="{$Resource.customJS}" />  
2 <input id="checkbox" type="checkbox" onclick="changeFont(this, '{$Component.thePanel}');" />
```

Call the JavaScript functions.

**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

513

salesforce

- 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



Jason Beck

Beginning  
Developer



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.



## MODULE AGENDA

516

salesforce

### MODULE 20: TESTING VISUALFORCE CONTROLLERS

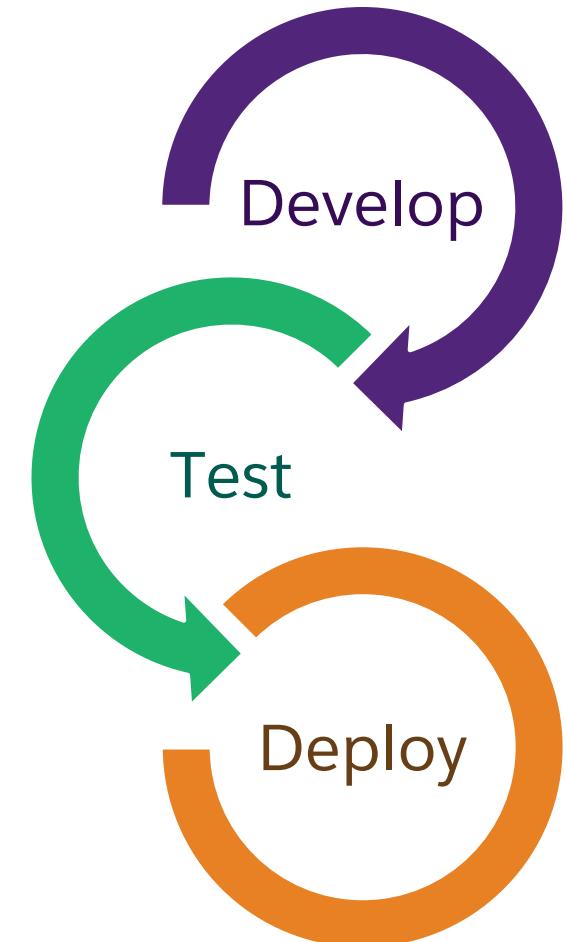
- **Understanding Visualforce Controller Testing**
- Testing a Visualforce Controller Constructor
- Testing Action Methods
- Testing Getters, Setters, and Properties





Just like the Apex triggers and classes we tested earlier, we need to ensure:

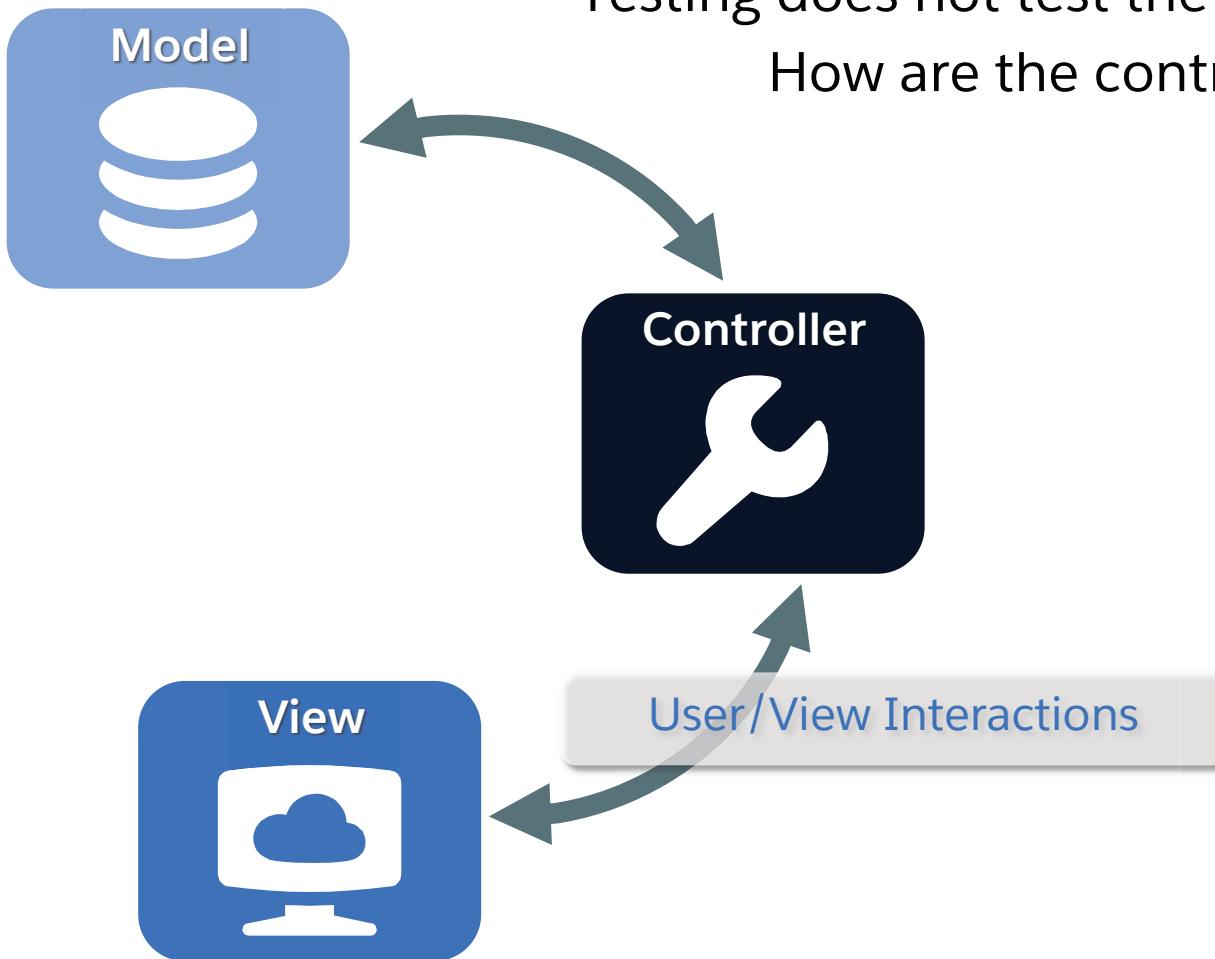
- 75% of Apex code must be executed successfully by test methods.
- Every Apex test method must execute without throwing any uncaught exceptions or exceeding governors.



# WHAT'S DIFFERENT ABOUT TESTING CONTROLLERS?

518

salesforce



Testing does not test the view; it only tests the controller.  
How are the controller methods invoked in a test,  
absent the view?

**Technician Status**

Edit Technician 1 Edit Technician 2 Edit Technician 3

Technician Name	Diana Tatro		
Phone	(408) 555-6991		
Email	dtatro-training@example.com		
Course Listing			
Date Achieved	Name	Certification Name	Certification Description
May 22, 2015	Raymond Montoya	AWCA Server	AW Computing Certified Associate Server
August 21, 2015	Raymond Montoya	AWCA Security	AW Computing Certified Associate Security
Attendees: Clara Morales, Aaryn Patel, Tammy Rogers, Diana Tatro			
Certifications Held			
Date Achieved	Certification Name	Certification Description	
5/30/2015	AWCA Server	AW Computing Certified Associate Server	

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.

Edit Technician 1 Edit Technician 2 Edit Technician 3





## MODULE AGENDA

520

salesforce

### MODULE 20: TESTING VISUALFORCE CONTROLLERS

- Understanding Visualforce Controller Testing
- **Testing a Visualforce Controller Constructor**
- Testing Action Methods
- Testing Getters, Setters, and Properties





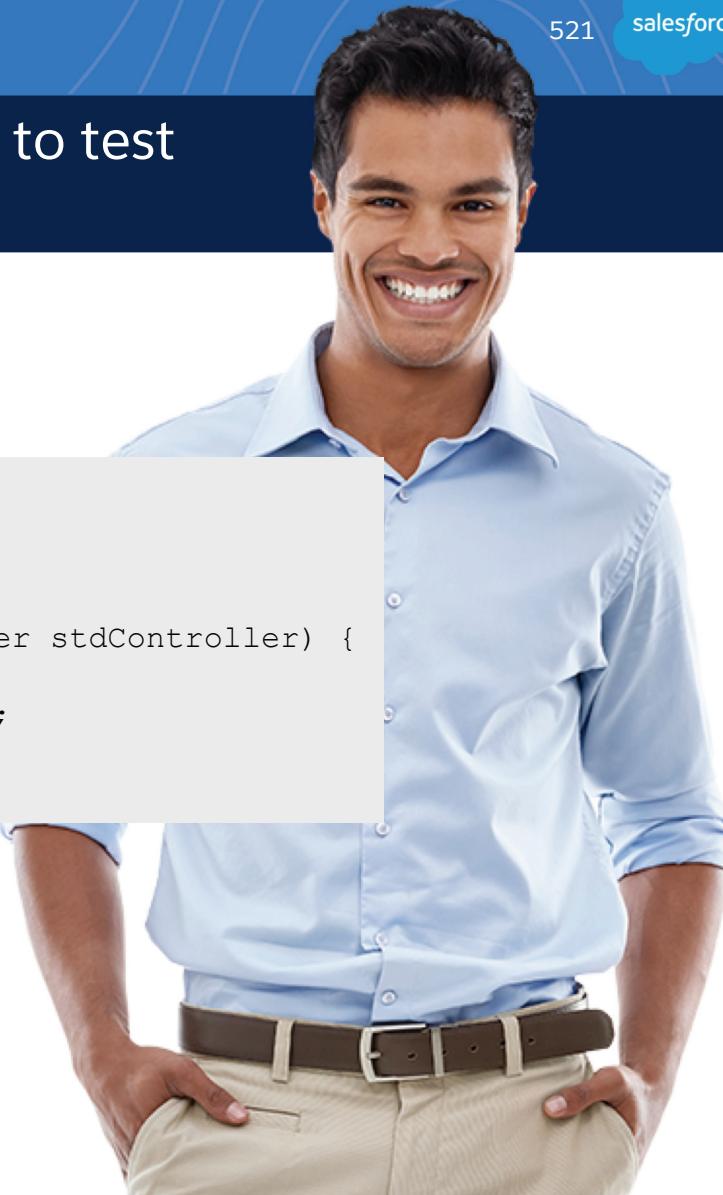
## TESTING THE CONSTRUCTOR

521

salesforce

We'll start by writing the test methods necessary to test the controller extension's constructor.

```
1 public class TechnicianStatus_CX {  
2     private final Contact contact;  
3     private ApexPages.StandardController sController;  
4  
5     public TechnicianStatus_CX(ApexPages.StandardController stdController) {  
6         sController = stdController;  
7         this.contact = (Contact) stdController.getRecord();  
8     }  
9 }
```



# TESTING CONTROLLER CONSTRUCTORS: SETTING currentPage()

522

salesforce

Constructors vary from controller to controller and extension to extension. You should test everything the controller or extension is doing:

```
1A  public class MyController {  
2A      private PageReference sourcePage;  
3A      public MyController() {  
4A          this.sourcePage = ApexPages.currentPage();  
5A      }  
6A  }
```

The constructor sets the current page as the source page.

```
1B  // Test code  
2B  PageReference testPage = Page.MyPage;  
2B  Test.setCurrentPage(testPage);  
3B  MyController testCtrl = new MyController();
```

Use `Test.setCurrentPage()` to assign a mock Visualforce page for `ApexPages.currentPage()`.

# TESTING CONTROLLER CONSTRUCTORS: SETTING PAGE PARAMETERS

523

salesforce

Populate HTTP query string parameters so that they can be found by the controller.

```
1A public class MyController {  
2A     private Id curId;  
3A     public MyController() {  
4A         this.curId = ApexPages.currentPage().getParameters().get('id');  
5A     }  
6A }
```

```
1B // Test code  
2B Account testAccount = new Account();  
3B ... // populate testAccount with good mock data  
4B insert testAccount;  
5B PageReference testPage = Page.MyPage;  
6B testPage.getParameters().put('id', testAccount.id);  
7B Test.setCurrentPage(testPage);  
8B MyController testCtrl = new MyController();
```

The constructor stores the current Id parameter as curId to use it later.

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

524

salesforce

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.

```
1 // Test code
2 Account testAccount = new Account();
3 ... // populate with test data
4 ApexPages.StandardController stdCtrl
   = new ApexPages.StandardController(testAccount);
5 MyControllerExt testExt = new MyControllerExt(stdCtrl);
6 ... // call setters, action, and getter methods
```

Then pass that controller to your extension.

**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

526

salesforce

## MODULE 20: TESTING VISUALFORCE CONTROLLERS

- Understanding Visualforce Controller Testing
- Testing a Visualforce Controller Constructor
- **Testing Action Methods**
- Testing Getters, Setters, and Properties





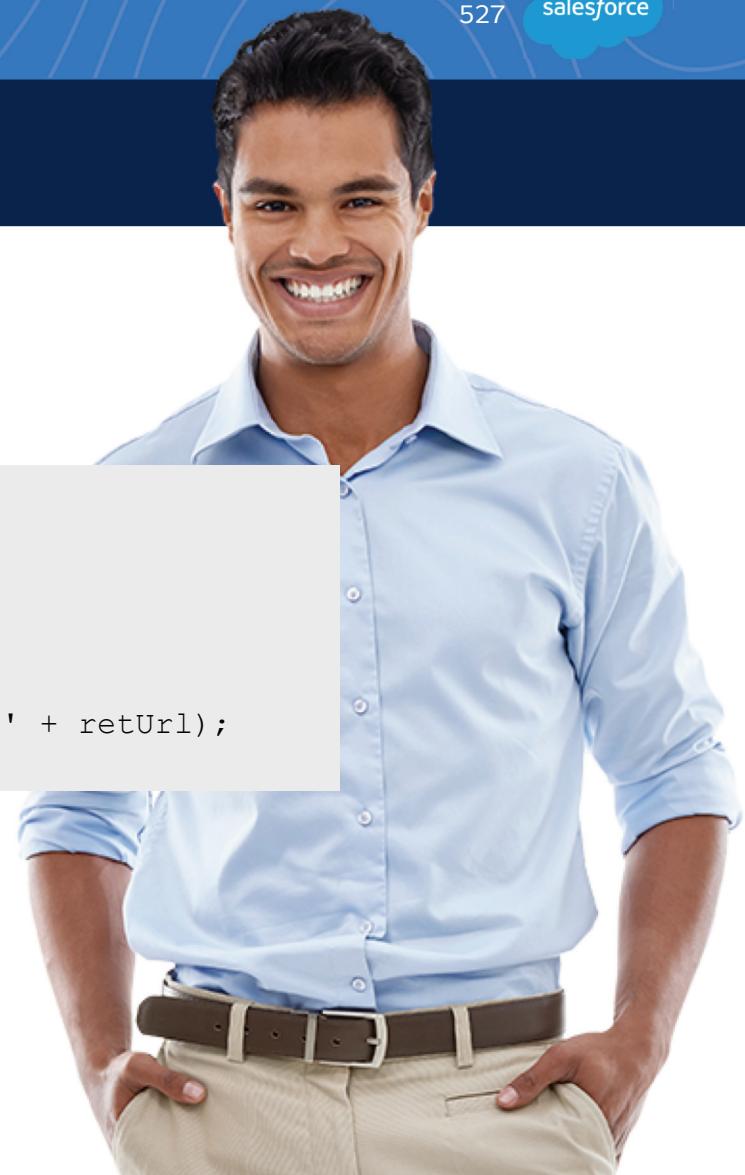
## TESTING ACTION METHODS

527

salesforce

Now you need to test the action methods in this controller extension.

```
1 public PageReference editContact2() {  
2     return sController.Edit();  
3 }  
4  
5 public PageReference editContact3() {  
6     String retUrl = '/apex/TechnicianStatus?id=' + contact.Id;  
7     return new PageReference('/' + contact.Id + '/e' + '?retURL=' + retUrl);  
8 }
```



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

Mimic Visualforce page interactions by explicitly calling methods.

- Button clicks (e.g., { !save }) call action methods.
- Verify what the action "does."

```
1 // Test code
2 MyController testCtrl = new MyController();
3 testCtrl.setName('John');
4 testCtrl.setAccount(new Account());
5 PageReference successPage = testCtrl.save();
```

# VERIFY WHERE THE ACTION "GOES"

530

salesforce

You can verify navigation by comparing PageReference URLs:

```
1A public class MyController {  
2A     ...  
3A     public PageReference save() {  
4A         ... // do something, check results  
5A         if (success) return Page.Success;  
6A         else return Page.Failure;  
7A     }  
8A }
```

```
1B // Test Code  
2B Account testAccount = new Account();  
3B // populate testAccount with good mock data  
4B MyController testCtrl = new MyController();  
5B System.assertEquals(testCtrl.save().getURL(), Page.Success.getURL());  
6B ... // populate testAccount with bad mock data  
7B System.assertEquals(testCtrl.save().getURL(), Page.Failure.getURL());
```

**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

532

salesforce

## 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

533

salesforce

Form fields (e.g., { !obj.field }) call getters and setters.

```
1A public with sharing class MySearchController {  
2A     String searchText;  
3A     public String getSearchText() {  
4A         return searchText;  
5A     }  
6A     public void setSearchText(String s) {  
7A         searchText = s;  
8A     }  
9A }
```

```
1B String testString = 'Test String';  
2B Test.startTest();  
3B controller.setSearchText(testString);  
4B Test.stopTest();  
5B 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.

Depending on your code, you may wish to check whether your method returns:

- Specific results.
- Results containing a specific value.
- A valid result (!null).

```
1A // List of Course Attendees associated with the Course Delivery Date that the user hovers over
2A public List<Course_Attendee__c> getAttendeeList() {
3A     String cdId = apexPages.currentPage().getParameters().get('courseDeliveryId');
4A     return [SELECT Student__r.Name FROM Course_Attendee__c WHERE Course_Delivery__c = :cdId];
5A }
```

```
1B Test.startTest();
2B // Mimics the user getting the attendeeList for a specific Course Delivery
3B List<Course_Attendee__c> attendeeList = controllerExt.getAttendeeList();
4B Test.stopTest();
5B System.assert(attendeeList.size() > 0);
```

- Calling a setter method in a test class

```
controller.setMethodName(value);
```

- Setting a property in a test class

```
controller.propertyName = value;
```

**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

537

salesforce

- 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](http://www.webassessor.com/salesforce)  
to register.

**Take the next step!** Register for  
the next course! Go to  
[www.salesforce.com/training](http://www.salesforce.com/training).

# TRAINING AND CERTIFICATION RESOURCES



**Training:**

[www.salesforce.com/training](http://www.salesforce.com/training)



**Salesforce Certification:**  
[certification.salesforce.com](http://certification.salesforce.com)



**Follow us on Twitter:**  
@Trailhead



**Trailhead:**  
[trailhead.salesforce.com](http://trailhead.salesforce.com)