Apex Triggers

Learning Objectives

- Write a trigger for a Salesforce object.
- Use trigger context variables.
- Call a class method from a trigger.
- Use the sObject addError() method in a trigger to restrict save operations.

Apex can be invoked by using *triggers*. Apex triggers enable you to perform custom actions before or after changes to Salesforce records, such as insertions, updates, or deletions.
A trigger is Apex code that executes before or after the following types of operations:

- insert
- update
- delete
- merge
- upsert
- undelete

For example, you can have a trigger run before an object's records are inserted into the database, after records have been deleted, or even after a record is restored from the Recycle Bin.

There are two types of triggers:

- *Before triggers* are used to update or validate record values before they're saved to the database.
- *After triggers* are used to access field values that are set by the system (such as a record's Id or LastModifiedDate field), and to affect changes in other records, such as logging into an audit table or firing asynchronous events with a queue. The records that fire the *after trigger* are read-only.

Asynchronous Apex

Apex offers multiple ways for running your Apex code asynchronously. Choose the asynchronous Apex feature that best suits your needs.

Asynchronous Apex Feature

Take control of your asynchronous Apex processes by using the Queueable interface. This interface enables you to add jobs to the queue and monitor them. Using the interface is an enhanced way of running your asynchronous Apex code compared to using future methods.

Package

A group of Lightning Platform components and applications that are made available to other organizations through the AppExchange. You use packages to bundle an app along with any related components so that you can upload them to AppExchange together.

Package Dependency

This is created when one component references another component, permission, or preference that is required for the component to be valid. Components can include but are not limited to:

- Standard or custom fields
- Standard or custom objects
- Visualforce pages
- Apex code

Permissions and preferences can include but are not limited to:

- Divisions
- Multicurrency
- Record types

Apex Testing

- Describe the key benefits of Apex unit tests.
- Define a class with test methods.
- Execute all test methods in a class and inspect failures.
- Create and execute a suite of test classes.

Important

- ➤ Unit tests must cover at least 75% of your Apex code, and all of those tests must complete successfully.
- ➤ Note the following.
- ➤ Every trigger must have some test coverage.
- ➤ All classes and triggers must compile successfully.
- ➤ When deploying Apex to a production organization, each unit test in your organization namespace is executed by default.
- ➤ Calls to System.debug are not counted as part of Apex code coverage.
- ➤ Test methods and test classes are not counted as part of Apex code coverage.

Create Test Data for Apex Tests

- Create a test utility class.
- Use a test utility method to set up test data for various test cases.
- Execute all test methods in a class.

Apex Integration

Make Callouts to External Services from Apex
An Apex callout enables you to tightly integrate your Apex code with an external service. The callout makes a call to an external web service or sends an HTTP request from Apex code, and then receives the response.

Apex callouts come in two flavors.

Web service callouts to SOAP web services use XML, and typically require a WSDL document for code generation.
HTTP callouts to services typically use REST with JSON.
These two types of callouts are similar in terms of sending a request to a service and receiving a response. But while WSDL-based callouts apply to SOAP Web services, HTTP callouts can be used with any HTTP service, either SOAP or REST.

SOAP is a mature protocol with a complete spec and is designed to expose individual operations – or pieces of operations – as web services. One of the most important characteristics of SOAP is that it uses XML rather than HTTP to define the content of the message.

The Argument For SOAP

SOAP is still offered by some very prominent tech companies for their APIs (Salesforce, Paypal, Docusign). One of the main reasons: legacy system support. If you built a connector between your application and Salesforce back in the day, there's a decent probability that connection was built in SOAP.

There are a few additional situations:

SOAP is good for applications that require formal contracts between the API and consumer since it can enforce the use of formal contracts by using WSDL (Web Services Description Language).
Additionally, SOAP has built in WS-Reliable messaging to increase security in asynchronous execution and processing.
Finally, SOAP has built-in stateful operations. REST is naturally stateless, but SOAP is designed to support conversational state management.
Some would argue that because of these features, as well as support for WS_AtomicTransaction and WS_Security, SOAP can benefit developers when there is a high need for transactional reliability.

Apex Web Services

Apex class and methods so that external applications can access your code and your application through the REST architecture. This is done by defining your Apex class with

the @RestResource annotation to expose it as a REST resource. Similarly, add annotations to your methods to expose them through REST. For example, you can add the @HttpGet annotation to your method to expose it as a REST resource that can be called by an HTTP GET request. For more information

Enterprise WSDL

A strongly-typed WSDL for customers who want to build an integration with their Salesforce organization only, or for partners who are using tools like Tibco or webMethods to build integrations that require strong typecasting. The downside of the Enterprise WSDL is that it only works with the schema of a single Salesforce organization because it is bound to all of the unique objects and fields that exist in that organization's data model.