Platform Developer 2:

https://developer.salesforce.com/platform/overview

Force.com :

Fastest way to create enterprise cloud apps.

Force.com delivers an on-demand cloud database with easy tools for building data driven applications and APIs for integrating with other apps.

API, Mobile SDK, Programmatic, Drag & Drop business logic, Social, Security

Heroku :

Platform as a Service supporting several programming languages (Java,Ruby,Scala,Node.js,Python,Php etc.). Develop amazing customer apps that sync with salesforce.

Heroku Connect - Bi directional data sync between Postgres and Salesforce so you can build apps that leverage your core customer data.

On-demand Scale

Heroku Connect, Languages you love, Add-on market place, Instant, on-demand scale.

Fuel :

Create Digital Marketing apps.

Fuel is an integrated collection of platform technologies that is open to third party development, enabling you to build upon, extend and integrate with ExactTarget's digital marketing products.

APIs, SDKs, Fuel UX, Fuel Cloud Editor

Sharing Table:

https://developer.salesforce.com/page/Using_Apex_Managed_Sharing_to_Create_Custom_Record_Sharing_Logic

Standard Salesforce objects support "Programatic Sharing" while custom objects support Apex managed sharing. More specifically, object shares can be written to both standard and custom objects, however custom sharing reasons can only be defined for shares written to custom objects.

All objects that have a default sharing setting of the either "Private" or "Public Read Only" also have a related "Share" object that is similar to an access control list (ACL) found in other platforms. All share objects for custom objects are named as MyCustomObject__Share, where MyCustomObject__c is the name of the related custom object. A share object includes records supporting all three types of sharing:

1. **Force.com managed sharing** (Record Ownership, Role Hierarchy, Territories, Teams and Sharing rules)

2. **User managed sharing** (Manual Sharing) : User managed sharing is removed when the record owner changes or when the access granted in the sharing does not grant additional access beyond the object's organization-wide sharing default access level.

3. **Apex managed sharing**.

A custom object's share object allows four pieces of information to be defined:

1. ParentId - The record being shared.

2. UserOrGroupId - The User or Group with whom the object is being shared.

3. AccessLevel - The permission level being granted to the User or Group.

4. RowCause - The reason why the User or Group has been granted sharing access.

A record can be shared multiple times with a user or group using different Apex sharing reasons.

Flow & Process Builder can be used to create share record for an object.

https://rakeshistom.wordpress.com/tag/sharing-table-in-salesforce/

Field History:

https://help.salesforce.com/HTViewHelpDoc?id=tracking_field_history.htm

The field history data is retained for up to 18 months.

If a trigger causes a change on an object the current user doesn't have permission to edit, that change is not tracked because field history honors the permissions of the current user.

Field Audit Trail lets you define a policy to retain archived field history data up to ten years, independent of field history tracking.

This feature helps you comply with industry regulations related to audit capability and data retention.

Changes to fields with more than 255 characters are tracked as edited, and their old and new values are not recorded.

Multi Currency:

On Enabling Multi-Currency, all existing records are stamped with a default currency code that you provide in your enablement request.

If you have only one currency in your multi-currency organization, you can set a preference to display currency symbols instead of ISO codes

After enablement, all currency fields display the ISO code of the currency before the amount.

https://help.salesforce.com/apex/HTViewHelpDoc?id=admin_enable_multicurrency_implications.htm&language=en_US

For organizations that have the multicurrency option enabled, the CurrencyIsoCode field is defined for any object that can have currency fields.

Salesforce locale settings determine the display formats for date and time, users' names, addresses, and commas and periods in numbers.

https://help.salesforce.com/HTViewHelpDoc?id=admin_supported_locales.htm&language=en_US

WorkBench:

https://help.salesforce.com/apex/HTViewHelpDoc?id=customize_wbench.htm

Compound Fields:

Compound fields group together multiple elements of primitive data types, such as numbers or strings, to represent complex data types, such as a location or an address. Compound fields are an abstraction that can simplify application code that handles the values, leading to more concise, understandable code.

Code that references individual component fields is unaffected by the new compound fields.

Compound fields are read-only. Changes are performed by writing to the individual component fields.

Compound fields are available only through the SOAP and REST APIs.

- Although geolocation fields appear as a single field in the user interface, custom geolocation fields count as *three*custom fields towards your organization's limits: one for latitude, one for longitude, and one for internal use.

A compound geolocation field value is returned as the structured data type `Location`.

In API versions earlier than 30.0, SOAP calls return compound geolocation field values as strings, instead of as a structured data type, for backward compatibility.

https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/compound_fields.htm

Primitive Data Types

If you set ID to a 15-character value, Apex converts the value to its 18-character representation.

All invalid ID values are rejected with a runtime exception.

Strings have no limit on the number of characters they can include. Instead, the heap size limit is used to ensure that your Apex programs don't grow too large.

AnyType. AnyType is used within the Force.com platform database exclusively for sObject fields in field history tracking tables.

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/langCon_apex_primitives.htm

External Id:

External Id is a field that usually references an ID from another (external) system.

https://help.salesforce.com/apex/HTViewSolution?id=000005395&language=en_US

You can identify up to seven (7) custom fields on an object as being an external ID field.

Index field

Default Number of custom indexes an entity/object can have is 13

https://help.salesforce.com/apex/HTViewSolution?id=000005081

Custom Settings

Custom settings are similar to custom objects and enable application developers to create custom sets of data, as well as create and associate custom data for an organization, profile, or specific user. All custom settings data is exposed in the application cache, which enables efficient access without the cost of repeated queries to the database. This data can then be used by formula fields, validation rules, flows, Apex, and the SOAP API.

List Custom Settings: provides a reusable set of static data that can be accessed across your organization.

Hierarchy Custom Settings: uses a built-in hierarchical logic that lets you "personalize" settings for specific profiles or users.

https://help.salesforce.com/apex/HTViewHelpDoc?id=cs_about.htm

The total amount of cached data allowed for your organization is the lesser of these two values:

10 MB OR 1 MB multiplied by the number of full-featured user licenses in your organization

Each custom setting counts against the total number of custom objects available for your organization.

- No owner is assigned when a custom setting is created, so the owner can't be changed.

Record size is based on the maximum field size of each field type, not the actual storage that's used in each field.

https://help.salesforce.com/apex/HTViewHelpDoc?id=cs_limits.htm&language=en_US

Formula fields only work for hierarchy custom settings; they can't be used for list custom settings.

{!$Setup.CustomSettingName__c.CustomFieldName__c}

Custom settings that have Privacy defined as Public are exposed to the API in the same way custom objects are exposed.

https://help.salesforce.com/HTViewHelpDoc?id=cs_accessing.htm&language=en_US

Triggers and Order of Execution

Workflow field updates that run based on an approval process or time-dependent action do not trigger any rules

https://help.salesforce.com/HTViewSolution?id=000005694&language=en_US

Formula fields calculate & display their results real-time whenever the field is accessed in any way. So for example: if a Workflow Rule uses a Formula Field in its criteria or formula, the formula field is evaluated when the Workflow Rule criteria is checked.

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_triggers_order_of_execution.htm

Flow:

Apex can be invoked from Flow using

1. Apex Plugin : Implement Process.Plugin Interface (Blob, Collection, sObject, Time & Bulk operations not supported).

2. Call Apex : Method with @InvocableMethod annotation (Generic Object, Generic sObject, Sets, Maps, Enum not supported).

Bulk operation is supported.

Classes with @InvocableMethod annotation are available in Flow, Processes (Process Builder) and REST API

https://help.salesforce.com/HTViewHelpDoc?id=vpm_designer_elements_apex.htm&language=en_US

Users can run only flows that have an active version. If the flow you embed doesn't have an active version, users see an error message.

If the flow you embed includes a subflow element, the flow that is referenced and called by the subflow element must have an active version.

<flow:interview name="MyUniqueFlowName"/>

To run the flow, external users (such as on a community) need access to the Visualforce page. To run the flow, internal users need access to the Visualforce page and either:

1. The "Run Flows" permission

2. The Force.com Flow User field enabled on their user detail page

In flow:interview use finishLocation attribute to redirect.

Set the allowShowPause attribute to false to prevent users from pausing in the flow.

If the flow is from a managed package, the `name` attribute must be in this format:`namespace.flowuniquename`.

https://help.salesforce.com/HTViewHelpDoc?id=vpm_admin_add_flow_to_vfpage.htm&language=en_US

Process:

Process Builder is a workflow tool that helps you easily automate your business processes by providing a powerful and user-friendly graphical representation of your process as you build it.

Automated processes in the Process Builder are based on records and consist of:

1. Criteria that determine when to execute action groups.

2. Immediate and scheduled actions to execute when those criteria are met.

Process builder doesn't support outbound messages.

With the Process Builder, you can:

1. Create a record

2. Update any related record—not just the record or its parent

3. Use a quick action to create a record, update a record, or log a call

4. Launch a flow—you can't schedule this action with workflow

5. Send an email

6. Post to Chatter

7. Submit for approval

Lightning Process Builder: https://help.salesforce.com/HTViewHelpDoc?id=process_overview.htm

Extending Lightning Process Builder: http://andyinthecloud.com/2015/03/01/extending-lightning-process-builder-and-visual-workflow-with-apex/

Invocable Method Considerations

Invocable methods are called with the REST API and used to invoke a single Apex method

- Only one method in a class can have the `InvocableMethod` annotation.
- Triggers can't use invocable methods.
- The invocable method must be static and public/global, and its class must be an outer class.
- Other annotations can't be used with the `InvocableMethod` annotation.

The use of **global** is only important if you plan to expose the action from an AppExchange package, if your developing a solution in a Sandbox for deployment to Production, you can use **public**.

Use the `future` annotation to identify methods that are executed asynchronously. When you specify `future`, the method executes when Salesforce has available resources.

Process can iterate up to five times after the original execution

https://developer.salesforce.com/blogs/developer-relations/2015/03/process-builder-taking-point-click-development-new-level.html

- In the Update a Record action, you *can* set a text based on a Formula field on the same record or a lookup record. This makes the previous bullet point less of an issue.

An exception denotes an error that disrupts the normal flow of code execution.

https://help.salesforce.com/HTViewHelpDoc?id=vpm_designer_elements_connector_fault.htm&language=en_US

```
{!$Flow.FaultMessage}
```

Transaction Control

All requests are delimited by the trigger, class method, Web Service, Visualforce page or anonymous block that executes the Apex code.

If the entire request completes successfully, all changes are committed to the database.

References to savepoints cannot cross trigger invocations because each trigger invocation is a new trigger context.

Static variables are not reverted during a rollback.

The ID on an sObject inserted after setting a savepoint is not cleared after a rollback.

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/langCon_apex_transaction_control.htm

Transaction:

An Apex transaction represents a set of operations that are executed as a single unit. All DML operations in a transaction either complete successfully, or if an error occurs in one operation, the entire transaction is rolled back and no data is committed to the database. The boundary of a transaction can be a trigger, a class method, an anonymous block of code, a Visualforce page, or a custom Web service method.

All operations that occur inside the transaction boundary represent a single unit of operations.

Transactions are useful when several operations are related, and either all or none of the operations should be committed.

This keeps the database in a consistent state.

https://developer.salesforce.com/docs/atlas.en-us.apex_workbook.meta/apex_workbook/apex_transactions.htm

SOSL queries can be used for text searches.

Eg: [FIND 'map*' IN ALL FIELDS RETURNING Account (Id, Name), Contact, Opportunity, Lead]

Unlike SOQL, which can only query one object at a time, SOSL enables you to search text, email, and phone fields for multiple objects simultaneously.

Dynamic Apex:

1. Access sObject and field describe information

2. Access Salesforce app information

3. Write dynamic SOQL queries, dynamic SOSL queries and dynamic DML

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_dynamic.htm

Single Sign-on:

Authentication providers let your users log in to your Salesforce organization using their login credentials from an external service provider. Salesforce supports the OpenId Connect protocol that allows users to log in from any OpenID provider such as Google, Paypal, LinkedIn and other services supporting OpenID Connect. When authentication providers are enabled, Salesforce does not validate a user's password. Instead, Salesforce uses the user's login credentials from the external service provider to establish authentication credentials.

Federated authentication using Security Assertion Markup Language (SAML)

Delegated authentication single sign-on enables you to integrate Salesforce with an authentication method that you choose. You manage delegated authentication at the permission level.

https://help.salesforce.com/HTViewHelpDoc?id=security_overview_auth.htm

StandardSetController

The maximum record limit for StandardSetController is 10,000 records. Instantiating StandardSetController using a query locator returning more than 10,000 records causes a LimitException to be thrown. However, instantiating StandardSetController with a list of more than 10,000 records doesn't throw an exception, and instead truncates the records to the limit.

https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/apex_pages_standardsetcontroller.htm

Visualforce

Visualforce Component that is defined in the same namespace as an associated page can also use the c namespace prefix.

https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_comp_cust_def.htm

https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/pages_comp_cust_creating.htm

Once your component has been created, you can view it at
http://mySalesforceInstance/apexcomponent/nameOfNewComponent

Lighting:

Lightning components give you a client-server framework that accelerates development, as well as app performance, and is ideal for use with the Salesforce1 mobile app

The Lightning App Builder empowers you to build apps visually, without code, quicker than ever before using off-the-shelf and custom-built Lightning components.

Lightning Component framework is built on the open-source Aura framework.

Force namespace contains components specific to Salesforce.

Lightning components are client-side centric, making them more dynamic and mobile friendly.

Contrastingly, Visualforce components are page-centric and rely heavily on server calls.

https://developer.salesforce.com/docs/atlas.en-us.lightning.meta/lightning/

Methods in the Apex controller that can be called must be static and annotated with @AuraEnabled.

Additionally, any properties or accessor methods on the returned object must be @AuraEnabled.

https://developer.salesforce.com/blogs/developer-relations/2015/03/lightning-components.html

Streaming:

Use Streaming API to receive notifications for changes to Salesforce data that match a SOQL query you define, in a secure and scalable way.

These events can be received by:

1. Pages in the Salesforce application.

2. Application servers outside of Salesforce.

3. Clients outside the Salesforce application.

The sequence of events when using Streaming API is as follows:

1. Create a PushTopic based on a SOQL query. This defines the channel.

2. Clients subscribe to the channel.

3. A record is created, updated, deleted, or undeleted (an event occurs). The changes to that record are evaluated.

4. If the record changes match the criteria of the PushTopic query, a notification is generated by the server and received by the subscribed clients.

https://developer.salesforce.com/docs/atlas.en-us.api_streaming.meta/api_streaming/

publish/subscribe model

REST API:

The REST-based Salesforce1 Reporting API gives you programmatic access to your report and dashboard data as defined in the report builder and dashboard builder. The API lets you integrate the data into any web or mobile application, inside or outside the Salesforce platform.

At a high level, the API resources let you query and filter report data. You can:

1. Run tabular, summary, or matrix reports synchronously or asynchronously.

2. Filter for specific data on the fly.

3. Query report metadata.

You can also work with dashboard resources to:

1. Get a list of recently used dashboards.

2. Get dashboard metadata and data.

3. Query dashboard status.

4. Refresh dashboards.

Limits are enforced against the aggregate of all API calls made by the organization in a 24 hour period; limits are not on a per-user basis. When an organization exceeds a limit, all users in the organization may be temporarily blocked from making additional calls.

Any action that sends a call to the API counts toward usage limits, except the following:

- Outbound messages
- Apex callouts

Compression algorithm used, either gzip or deflate

https://developer.salesforce.com/docs/atlas.en-us.api_analytics.meta/api_analytics/

https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/implementation_considerations.htm

Unit Test

Classes defined with the isTest annotation don't count against your organization limit of 3 MB for all Apex code.

Classes defined as isTest must be top-level classes and can't be interfaces or enums.

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_testing_unit_tests.htm

https://help.salesforce.com/HTViewHelpDoc?id=code_test_execution.htm&language=en_US

The isTest(SeeAllData=true) annotation is used to open up data access when applied at the class or method level.

If the test class or a test method has access to organization data by using the @isTest(SeeAllData=true) annotation, test setup methods aren't supported in this class

```
@testSetup static void methodName()
```

However, using isTest(SeeAllData=false) on a method doesn't restrict organization data access for that method if the containing class has already been defined with the isTest(SeeAllData=true) annotation. In this case, the method will still have access to all the data in the organization.

Use the `transient` keyword to declare instance variables that can't be saved, and shouldn't be transmitted as part of the view state for a Visualforce page. Declaring variables as `transient` reduces view state size.

Static variables also don't get transmitted through the view state.

Use the `Continuation` class to make callouts asynchronously to a SOAP or REST Web service.

Asynchronous callouts that are made from a Visualforce page don't count toward theApex limit of 10 synchronous requests that last longer than five seconds.

These hundred simultaneous actions exceed the limit of concurrent long-running requests of 10, but by using asynchronous callouts, the requests aren't subjected to this limit and can be executed.

You can make up to three asynchronous callouts in a single continuation. Add these callout requests to the same continuation by using the `addHttpRequest` method of the `Continuation` class.

You can chain callout requests.

- Only one method in a class can have the `InvocableMethod` annotation.
- Triggers can't use invocable methods.
- The invocable method must be static and public or global, and its class must be an outer class.
- Other annotations can't be used with the `InvocableMethod` annotation.

- Other annotations can't be used with the `InvocableVariable` annotation.

The invocable variable can't be one of the following:

- A type such as an interface, class, or enum.
- A non-member variable such as a static or `local` variable.
- A property.
- A final variable.
- `Protected` or private.

Methods with the `future` annotation cannot be used in Visualforce controllers in either `get`*MethodName* or `set`*MethodName* methods, nor in the constructor.

When you use @future method, you completly gets disconnected with current transaction and cannot show output of that method on Visualforce page, if you want to...

In case of continuatioon object, User can work on VF page and once output is available, it will be rendered in Visualforce page, means response from Async is still available in Contuniation object.

I will say both are very useful depending in situation. Chances are very high that you will end up with using @future methods more than continuation object.

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_web_services_methods.htm

You cannot use the `webService` keyword when defining a class.

- System-defined enums cannot be used in Web service methods.
- You must define any method that uses the `webService` keyword as static.

They also cannot be marked as return values

- Maps
- Sets
- Pattern objects
- Matcher objects
- Exception objects

You must use the `webService` keyword with any member variables that you want to expose as part of a Web service. You should not mark these member variables as static.

Use `convertCurrency()` in the `SELECT` statement of a SOQL query to convert currency fields to the user's currency. This action requires that the org has multiple currencies enabled.

Use an ISO code that your org has enabled and made active. If you don't put in an ISO code, the numeric value is used instead of comparative amounts.

SELECT Id, Name FROM Opportunity WHERE Amount > USD5000

Currency data returned by using an aggregate function, such as `SUM()` or `MAX()`, is in the org's default currency.

You can use aggregate functions without using a `GROUP BY` clause.

Note that any query that includes an aggregate function returns its results in an array of AggregateResult objects. AggregateResult is a read-only sObject and is only used for query results.

apex:actionSupport: A component that adds AJAX support to another component, allowing the component to be refreshed asynchronously by the server when a particular event occurs, such as a button click or mouseover.

apex:actionFunction:

A component that provides support for invoking controller action methods directly from JavaScript code using an AJAX request. An <apex:actionFunction> component must be a child of an <apex:form> component.

Unlike <apex:actionSupport>, which only provides support for invoking controller action methods from other Visualforce components, <apex:actionFunction> defines a new JavaScript function which can then be called from within a block of JavaScript code.

apex:actionPoller

A timer that sends an AJAX request to the server according to a time interval that you specify. Each request can result in a full or partial page update.

Action methods used by <apex:actionPoller> should be lightweight. It's a best practice to avoid performing DML, external service calls, and other resource-intensive operations.

- Avoid using this component with enhanced lists.

apex:remoteObjects

Use this component, along with
child `apex:remoteObjectModel` and `apex:remoteObjectField` components, to specify the sObjects and fields to access using Visualforce Remote Objects. These components generate models in JavaScript that you can use for basic create, select, update, and delete operations in your client-side JavaScript code.

You can use Apex classes that implement the `Messaging.InboundEmailHandler`interface to handle an inbound email message.

```
global class CreateTaskEmailExample implements Messaging.InboundEmailHandl
er {
```

Apex provides the built-in `WebServiceMock` interface and the `Test.setMock` method.

The `Database.Batchable` interface contains three methods that must be implemented.

- If the Apex flex queue has the maximum number of 100 jobs, `Database.executeBatch` throws a `LimitException` and doesn't add the job to the queue.
- If you specify `Database.Stateful` in the class definition, you can maintain state across these transactions
- Use the Test methods `startTest` and`stopTest` around the `executeBatch` method to ensure that it finishes before continuing your test
- Asynchronous calls, such as @future or `executeBatch`, called in a `startTest`, `stopTest` block, do not count against your limits for the number of queued jobs.
- The maximum number of batch Apex method executions per 24-hour period is 250,000, or the number of user licenses in your org multiplied by 200—whichever is greater. This limit is for your entire organization and is shared with all asynchronous Apex: Batch Apex, QueueableApex, scheduled Apex, and future methods.
- Methods declared as `future` aren't allowed in classes that implement the `Database.Batchable` interface.
  - Methods declared as `future` can't be called from a batch Apex class.
- Each batch Apex invocation creates an `AsyncApexJob` record. To construct a SOQL query to retrieve the job's status, number of errors, progress, and submitter, use the `AsyncApexJob` record's ID
- For each 10,000 `AsyncApexJob` records, Apex creates an `AsyncApexJob` record of type `BatchApexWorker` for internal use.
- Minimize the number of batches, if possible.
- If more than 2,000 unprocessed requests from a single organization are in the queue, any additional requests from the same organization will be delayed while the queue handles requests from other organizations.

- To ensure fast execution of batch jobs, minimize Web service callout times and tune queries used in your batch Apex code.
- You can start another batch job from an existing batch job to chain jobs together. You can chain a batch job by calling `Database.executeBatch` or `System.scheduleBatch` from the `finish` method of the current batch class.
- *`implements Schedulable {}`*
- `global void execute(SchedulableContext sc){}`
    - You can also programmatically query the CronTrigger and CronJobDetail objects to get the count of Apex scheduled jobs. CronTrigger: Contains schedule information for a scheduled job.
- If there are one or more active scheduled jobs for an Apex class, you cannot update the class or any classes referenced by this class through the Salesforce user interface. However, you can enable deployments to update the class with active scheduled jobs by using the Metadata API .
- Though it's possible to do additional processing in the `execute` method, we recommend that all processing take place in a separate class.
- The scheduler runs as system—all classes are executed, whether or not the user has permission to execute the class.
- Use extreme care if you're planning to schedule a class from a trigger. You must be able to guarantee that the trigger won't add more scheduled classes than the limit.
- The `System.Schedule` method uses the user's timezone for the basis of all schedules.
- Synchronous Web service callouts are not supported from scheduled Apex. To be able to make callouts, make an asynchronous callout by placing the callout in a method annotated with @future(callout=true) and call this method from scheduled Apex.
- `@RestResource,@HttpDelete, @HttpGet`
- site must be registered in the Remote Site Settings page, or the call will fail

WebService keyword: It can't be used on a class itself, or an interface or interface methods or variables.

An Apex trigger can execute a callout when the callout is invoked within a method defined as asynchronous: that is, defined with the @future keyword.

- Because there are no SOAP analogs for certain Apex elements, methods defined with the `webservice` keyword cannot take the following elements as parameters. While these elements can be used within the method, they cannot be used as return values.
    Maps
    Sets
    Pattern objects
    Matcher objects
    Exception objects

You must use the `webservice` keyword with any member variables that you want to expose as part of a web service. You should not mark these member variables as static.

There are two main ways to develop callouts with Apex: (a) Import a WSDL into Apex (b) HTTP (RESTful) Services classes.

The following data types are only supported when used as call ins, that is, when an external Web service calls an Apex Web service method. These data types are not supported as callouts, that is, when an Apex Web service method calls an external Web service.

- blob
- decimal
- enum

On top of class `@RestResource(urlMapping='/FieldCase/*')`

Schema Builder is enabled by default and lets you add the following to your schema:

- Custom objects
- Lookup relationships
- Master-detail relationships
- All custom fields except: Geolocation

Messaging.InboundEmail, Messaging.InboundEmailResult, Messaging.InboundEnvelope

How are Apex triggers stored?: As metadata in the application, under the object with which they are associated.

apex:PageMessages is a containing component where any messages that have been added to the page will appear.

apex:pageMessage is a component that adds a single message to the page.

apex:message allows you to associate a message with a component

I've not used apex:messages - I can't see how it particularly differs from apex:pagemessages

ApexPages.Message is the class that is used to model a message. A message isn't associated with a page until it is added via the ApexPages class.

ApexPages is a class that allows you to access the current page (through ApexPages.CurrentPage()) and manage messages for the current page.

apex:message is used to display an error on only a very specific field. It is used to allow the developer to place field specific errors in a specific location.

apex:messages is similar to apex:message, but it displays all of the errors. These errors are displayed as a list with no styling.

apex:pageMessages is used to display all of the messages on a page. It will display Salesforce generated messages as well as custom messages added to the ApexPages class.

- Use OFFSET for pages in which users typically do not move beyond the first page or results, or for pages in which the SOAP API is used to retrieve a relatively small data set for pagination.
- Use the StandardSetController for most Web applications built on the Force.com platform, except mobile applications and other applications in which users are not likely to move past the first page of results.

Implement the `Schedulable` interface in an Apex class that instantiates the class you want to run. Alternatively, you can call the `System.scheduleBatch` method to schedule the batch job to run once at a future time

```
global class scheduledBatchable implements Schedulable {
    global void execute(SchedulableContext sc) {
       batchable b = new batchable();
       database.executebatch(b);
    }
 }
```

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/connectAPI_overview.htm

Chatter in Apex methods execute in the context of the context user, who is also referred to as the *context user*. The code has access to whatever the context user has access to. It doesn't run in system mode like other Apex code.

The reason why sObjects can't be passed as arguments to future methods is because the sObject might change between the time you call the method and the time it executes.

https://releasenotes.docs.salesforce.com/en-us/summer14/release-notes/rn_apex_price_books_in_tests.htm

Compound fields can *only*be used in SOQL queries made through the SOAP and REST APIs.
https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql_geolocate.htm

Location-based SOQL queries let you compare and query location values stored in Salesforce. You can calculate the distance between two location values, such as between a warehouse and a store.

SOQL queries made using the SOAP and REST APIs also support using geolocation fields, including address fields that have a geolocation component, directly in SOQL statements. This often results in simpler SOQL statements. Compound fields can *only* be used in SOQL queries made through the SOAP and REST APIs.

- `DISTANCE` and `GEOLOCATION` are supported in `WHERE` and `ORDER BY` clauses in SOQL, but not in `GROUP BY`. `DISTANCE` is supported in `SELECT` clauses.
  - When using the `GEOLOCATION` function in SOQL queries, the geolocation field must precede the latitude and longitude coordinates. For example, `DISTANCE(warehouse_location__c, GEOLOCATION(37.775,-122.418), 'km')` works but `DISTANCE(GEOLOCATION(37.775,-122.418), warehouse_location__c, 'km')` doesn't work.
  - Apex bind variables aren't supported for the units parameter in `DISTANCE` or `GEOLOCATION` functions. This query doesn't work.

```
1 String units = 'mi';
2 List<Account> accountList =
3     [SELECT ID, Name, BillingLatitude, BillingLongitude
4      FROM Account
5      WHERE DISTANCE(My_Location_Field__c, GEOLOCATION(10,10), :units)
  < 10];
```

https://developer.salesforce.com/page/Paginating_Data_for_Force.com_Applications

When building pagination on the Force.com platform with Visualforce, do not execute a query to retrieve data for pagination--the other tools can provide the same functionality and minimize page state, which increases page response time.

Although developers might think that the OFFSET clause has made the query and queryMore calls obsolete in pagination design, this assumption is only true in some cases. The query and queryMore calls still provide a performance edge when you're paginating over large data sets because they hold a server-side cursor (batch), which in turn stores the data in the API Cursor Server (ACS).

https://help.salesforce.com/apex/HTViewHelpDoc?id=code_dev_console_view_system_log.htm&language=en#apex_log_execution

The **Apex Code** log level must be set to `Finest` for variable assignments to be logged.

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_rest_methods.htm

`@RemoteAction` methods are static, and therefore can't see the current page state directly, while `apex:actionFunction` methods are instance methods, and so can see the entire page state.

Methods annotated with `@HttpGet` or `@HttpDelete` should have no parameters.

same class can't have two methods annotated with `@HttpGet`.

Error

```
{
2     "x" : "value1",
3     "x" : "value2"
4 }
```

In JSON, you can specify `null` as the value

Queueable Apex

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, which is an enhanced way of running your asynchronous Apex code compared to using future methods.

```
public class AsyncExecutionExample implements Queueable {
     public void execute(QueueableContext context) {
          Account a = new Account(Name='Acme',Phone='(415) 555-1212');
          insert a;
     }
 }
```