Spring Web Flow uses the Spring MVC framework. The DispatcherServlet handles the request. A FlowHandlerMapping is used to map the request to a particular Web Flow. The FlowHandlerMapping is provided an instance of FlowRegistry. The FlowRegistry has the definitions for the flow. The FlowHandlerMapping maps the request to a FlowHandlerAdapter which is responsible for handling the flow requests. The FlowHandlerAdapter is provided with an instance of FlowExecutor which is responsible for the actual flow lifecycle. Lets look at the above classes in brief

Important classes

* *FlowHandlerMapping* – This is a HandlerMapping that creates a URL from the ids registered in the flow registry. It returns a FlowHandler if a matching id is found in the flow registry.
* *FlowHandler* – This is a controller helper that has the reference to the actual flow. It handles the execution of the flow, its outcomes and exceptions.
* *FlowHandlerAdapter* – This is the HandlerAdapter for web flows. It delegates work to the mapped FlowHandler.
* *FlowExecutor* – This is the central class of the Flow and is the facade for the actual flow. It manages creating of new flows or resuming existing flows. It is an entry into the Spring web flow system.
* *FlowDefinition* – This class stores the definition of the flow. It contains the set of states that form part of the flow. Each flow has one start state.
* *FlowRegistry* – This class contains all the flow definitions.

What are the different parts of a flow

The web flow has three main parts – States, Transition and Data.   
States – States are points on the flow where an action happens or decision is made or a view is displayed. The end of a flow is also a state. Sometimes, to increase brewity, a subset of a flow can be combined together to form a sub-flow and then the sub-flow is a state in the main flow.   
Transitions – Transition connects two states. A movement of request from one state to another is called a transition.   
Data – Data is the information that is carried from one state to another. The life span of the data depends on the scope with which it is declared.

**States**– There are five kinds of states

1. View States – A view state displays information to a user or obtains user input using a form. a model can be passed to bind the elements of the form. The view may be implemented as any view defined in spring mvc. (e.g. JSP)
2. Action States – Action states are where actions are perfomed by the spring beans. The action state transitions to another state. The action states generally have an evaluate property that describes what needs to be done. The evaluate is generally a method of one of the registered spring beans. The evaluate expression is a SpEL. More than one action can be configured in which case the actions will be executed one after the other.
3. Decision States: Decision state is where a decision is made. A decision state has two transitions depending on whether the decision evaluates to true or false.
4. Subflow States – It may be advantageous to call another flow from one flow to complete some steps. The sub flow returns to the original flow when it is complete. Data may be passed from the calling flow into the subflow and output data from the subflow may be retrieved into the calling flow.
5. End States – The end state signifies the end of flow. If the end state is part of a root flow then the execution is ended. However, if its part of a sub flow then the root flow is resumed.

**Flow Data Scopes**  
Flow Data can be stored as variables that can be created using the ‘var’ element or using the evaluate expression.  
  
Variable scope – The lifespan of the variable depends on the scope with which it is declared. The scopes available are:

* Conversation – The conversation scope starts when a flow starts and ends when the flow ends. It is available in sub flows
* Flow – Available within a flow. Not available in sub flows
* Request – Available during the life of a request in a flow
* Flash – Available during the lifetime of a flow. However, once a state is rendered, the variable is cleared.
* View – Available only during the lifetime of a view. Created when a view is created and destroyed once a view is destroyed