What is flow?

Summary

Flow

A flow encapsulates a reusable sequence of steps that can execute in different contexts. Below is a Garrett Information Architecture diagram illustrating a reference to a flow that encapsulates the steps of a hotel booking process.

```
<?xml version="1.0" encoding="UTF-8"?>
<flow xmlns="http://www.springframework.org/schema/webflow"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.springframework.org/schema/webflow
    http://www.springframework.org/schema/webflow/spring-webflow-2.0.xsd">

</flow>
```

What is the makeup of a typical flow?

In Spring Web Flow, a flow consists of a series of steps called "states". Entering a state typically results in a view being displayed to the user.
In this view, the event occurs that controls the Sate. These events cause Transition(transition) that moves to other views as a result.
All state is defined in <flow/>. State that is defined first becomes the starting point of Flow.

How are flows authored?

Flows are authored by web application developers using a simple XML-based flow definition language. The next steps of this guide will walk you through the elements of this language.

Description

Essential language elements

```
<?xml version="1.0" encoding="UTF-8"?>
<flow xmlns="http://www.springframework.org/schema/webflow"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.springframework.org/schema/webflow
                      http://www.springframework.org/schema/webflow/spring-webflow-2.0.xsd">

          <view-state id="enterBookingDetails" />

          <view-state id="enterBookingDetails">
                    <transition on="submit" to="reviewBooking" />
          </view-state>

          <end-state id="bookingCancelled" />

</flow>
```

- view-state: component defining the Sate showing the screen during Flow
  - In the directory where flow definition file is located, the screen template matching view-state id in the directory with flow definition file is matched.
- transition: component that controls the events to be generated in Sate. Cause screen movement.
- end-state: define the results of Flow

Actions

Most of flows need to express more than just view navigation logic. Typically they also need to invoke business services of the application or other actions.

Within a flow, there are several points where you can execute actions. These points are:

- On flow start
- On state entry
- On view render
- On transition execution
- On state exit
- On flow end

Actions are defined using a concise expression language. Spring Web Flow uses the Unified EL by default. The next few sections will cover the essential language elements for defining actions.

evaluate

The action element you will use most often is the evaluate element. Use the evaluate element to evaluate an expression at a point within your flow. With this single tag you can invoke methods on Spring beans or any other flow variable. For example:

```
<!-- [1] Call by inserting the booking object in persist method of entityManager Bean and .   -->
<evaluate expression="entityManager.persist(booking)" />

<!-- [2] Call findHotels method and save the execution result Hotels object in flowScope data model. --
>
<evaluate expression="bookingService.findHotels(searchCriteria)" result="flowScope.hotels" />

<!-- [3] Call findHotels method and convert to dataModel type when saving execution result Hotels
object in flowScope data model.   -->
<evaluate expression="bookingService.findHotels(searchCriteria)" result="flowScope.hotels" result-
type="dataModel"/>
```

In the following example, create and save the Booking object in the flow scope when starting Flow. hotelId receives as input property of flow.

```
<flow xmlns="http://www.springframework.org/schema/webflow"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.springframework.org/schema/webflow
    http://www.springframework.org/schema/webflow/spring-webflow-2.0.xsd">

          <input name="hotelId" />

          <on-start>
                    <evaluate expression="bookingService.createBooking(hotelId, currentUser.name)"          re
          </on-start>

          <view-state id="enterBookingDetails">
                    <transition on="submit" to="reviewBooking" />
          </view-state>

          <view-state id="reviewBooking">
                    <transition on="confirm" to="bookingConfirmed" />
                    <transition on="revise" to="enterBookingDetails" />
                    <transition on="cancel" to="bookingCancelled" />
          </view-state>

          <end-state id="bookingConfirmed" />
```

```
            <end-state id="bookingCancelled" />
</flow>
```

Input/output mapping

Each flow has a well-defined input/output contract. Flows can be passed input attributes when they start, and can return output attributes when they end. In this respect, calling a flow is conceptually similar to calling a method with the following signature:

```
FlowOutcome flowId(Map<String, Object> inputAttributes);
```

FlowOutcome to be returned has the method declaration part as shown below.

```
public interface FlowOutcome {
            public String getName();
            public Map<String, Object> getOutputAttributes();
}
```

input

```
<!-- [1] Value of relevant variable is saved in flow scope as hotelId.   -->
<input name="hotelId" />

<!-- [2] Property can be designated as type property. If the type does not match, try type conversion -
->
<input name="hotelId" type="long" />

<!-- [3] Allocate the input value with value property -->
<input name="hotelId" value="flowScope.myParameterObject.hotelId" />

<!-- [4] Force not to be empty or null as required property -->
<input name="hotelId" type="long" value="flowScope.hotelId" required="true" />
```

output

Use the output element to declare a flow output attribute. Output attributes are declared within end-states that represent specific flow outcomes.

```
<end-state id="bookingConfirmed">
   <output name="bookingId" />
</end-state>

<!—Designate the value of target directly -->
<end-state id="bookingConfirmed">
   <output name="confirmationNumber" value="booking.confirmationNumber" />
</end-state>
```

Input/output mapping: Sample

```
<flow xmlns="http://www.springframework.org/schema/webflow"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.springframework.org/schema/webflow
                        http://www.springframework.org/schema/webflow/spring-webflow-2.0.xsd">
            <input name="hotelId" />

            <on-start>
```

```
          <evaluate expression="bookingService.createBooking(hotelId, currentUser.name)"
          result="flowScope.booking" />
           </on-start>

           <view-state id="enterBookingDetails">
                     <transition on="submit" to="reviewBooking" />
           </view-state>

           <view-state id="reviewBooking">
                     <transition on="confirm" to="bookingConfirmed" />
                     <transition on="revise" to="enterBookingDetails" />
                     <transition on="cancel" to="bookingCancelled" />
           </view-state>

           <end-state id="bookingConfirmed">
                     <output name="bookingId" value="booking.id" />
           </end-state>
           <end-state id="bookingCancelled" />
</flow>
```

Above Flow receives hotelId as input value now and when new reservation is over, it returns bookingId output property as result.

Variables

A flow may declare one or more instance variables. These variables are allocated when the flow starts. Any @Autowired transient references the variable holds are also rewired when the flow resumes.

Use the var element to declare a flow variable:

```
<var name="searchCriteria" class="com.mycompany.myapp.hotels.search.SearchCriteria"/>
```

Make sure your variable's class implements java.io.Serializable, as the instance state is saved between flow requests.

Calling Subflows

A flow may call another flow as a subflow. The flow will wait until the subflow returns, then respond to the subflow outcome.

subflow-state

Use the subflow-state element to call another flow as a subflow:

```
<subflow-state id="addGuest" subflow="createGuest">
           <transition on="guestCreated" to="reviewBooking">
                     <evaluate expression="booking.guests.add(currentEvent.attributes.guest)" />
           </transition>
           <transition on="creationCancelled" to="reviewBooking" />
</subfow-state>
```

The above example calls the createGuest flow, then waits for it to return. When the flow returns with a guestCreated outcome, the new guest is added to the booking's guest list.

Passing subflow input

Use the input element to pass input to the subflow:

```
<subflow-state id="addGuest" subflow="createGuest">
            <input name="booking" />
            <transition to="reviewBooking" />
</subfow-state>
```

Mapping subflow output

Simply refer to a subflow output attribute by its name within a outcome transition

```
 <subflow-state   ..>
  <transition on="guestCreated" to="reviewBooking">
    <evaluate expression="booking.guests.add(currentEvent.attributes.guest)" />
  </transition>
  ..
```

In the above example, guest is the name of an output attribute returned by the guestCreated outcome.

Sample: Calling Sub Flow

Following is sample code.

```
<flow xmlns="http://www.springframework.org/schema/webflow"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.springframework.org/schema/webflow
                        http://www.springframework.org/schema/webflow/spring-webflow-2.0.xsd">

            <input name="hotelId" />
            <on-start>
                        <evaluate expression="bookingService.createBooking(hotelId, currentUser.name)"
                                    result="flowScope.booking" />
            </on-start>

            <view-state id="enterBookingDetails">
                        <transition on="submit" to="reviewBooking" />
            </view-state>

            <view-state id="reviewBooking">

                        <transition on="addGuest" to="addGuest" />
                        <transition on="confirm" to="bookingConfirmed" />
                        <transition on="revise" to="enterBookingDetails" />
                        <transition on="cancel" to="bookingCancelled" />
            </view-state>

            <subflow-state id="addGuest" subflow="createGuest">
                        <transition on="guestCreated" to="reviewBooking">
                                    <evaluate expression="booking.guests.add(currentEvent.attributes.guest)"
/>
                        </transition>
                        <transition on="creationCancelled" to="reviewBooking" />
            </subfow-state>

            <end-state id="bookingConfirmed">
                        <output name="bookingId" value="booking.id" />
            </end-state>
            <end-state id="bookingCancelled" />
```

</flow>

Reference

- Spring Web Flow reference 2.0.x
- Spring Web-Flow Framework Reference beta with Korean (by Park Chan Wook)