3.5. Special EL variables

There are several implicit variables you may reference from within a flow. These variables are discussed in this section.

flowScope

Use `flowScope` to assign a flow variable. Flow scope gets allocated when a flow starts and destroyed when the flow ends. With the default implementation, any objects stored in flow scope need to be Serializable.

```
<evaluate expression="searchService.findHotel(hotelId)" result="flowScope.hotel" />
```

viewScope

Use `viewScope` to assign a view variable. View scope gets allocated when a `view-state` enters and destroyed when the state exits. View scope is *only* referenceable from within a `view-state`. With the default implementation, any objects stored in view scope need to be Serializable.

```
<on-render>
    <evaluate expression="searchService.findHotels(searchCriteria)" result="viewScope.hotels"
              result-type="dataModel" />
</on-render>
```

requestScope

Use `requestScope` to assign a request variable. Request scope gets allocated when a flow is called and destroyed when the flow returns.

```
<set name="requestScope.hotelId" value="requestParameters.id" type="long" />
```

flashScope

Use `flashScope` to assign a flash variable. Flash scope gets allocated when a flow starts, cleared after every view render, and destroyed when the flow ends. With the default implementation, any objects stored in flash scope need to be Serializable.

```
<set name="flashScope.statusMessage" value="'Booking confirmed'" />
```

conversationScope

Use `conversationScope` to assign a conversation variable. Conversation scope gets allocated when a top-level flow starts and destroyed when the top-level flow ends. Conversation scope is shared by a top-level flow and all of its subflows. With the default implementation, conversation scoped objects are stored in the HTTP session and should generally be Serializable to account for typical session replication.

```
<evaluate expression="searchService.findHotel(hotelId)" result="conversationScope.hotel" />
```

requestParameters

Use `requestParameters` to access a client request parameter:

```
<set name="requestScope.hotelId" value="requestParameters.id" type="long" />
```

currentEvent

Use `currentEvent` to access attributes of the current `Event`:

```
<evaluate expression="booking.guests.add(currentEvent.attributes.guest)" />
```

currentUser

Use `currentUser` to access the authenticated `Principal`:

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

messageContext

Use `messageContext` to access a context for retrieving and creating flow execution messages, including error and success messages. See the `MessageContext` Javadocs for more information.

```
<evaluate expression="bookingValidator.validate(booking, messageContext)" />
```

resourceBundle

Use `resourceBundle` to access a message resource.

```
<set name="flashScope.successMessage" value="resourceBundle.successMessage" />
```

flowRequestContext

Use `flowRequestContext` to access the `RequestContext` API, which is a representation of the current flow request. See the API Javadocs for more information.

flowExecutionContext

Use `flowExecutionContext` to access the `FlowExecutionContext` API, which is a representation of the current flow state. See the API Javadocs for more information.

flowExecutionUrl

Use `flowExecutionUrl` to access the context-relative URI for the current flow execution view-state.

externalContext

Use `externalContext` to access the client environment, including user session attributes. See the `ExternalContext` API JavaDocs for more information.

```
<evaluate expression="searchService.suggestHotels(externalContext.sessionMap.userProfile)"
          result="viewScope.hotels" />
```

3.6. Scope searching algorithm

When assigning a variable in one of the flow scopes, referencing that scope is required. For example:

```
<set name="requestScope.hotelId" value="requestParameters.id" type="long" />
```

When simply accessing a variable in one of the scopes, referencing the scope is optional. For example:

```
<evaluate expression="entityManager.persist(booking)" />
```

If no scope is specified, like in the use of `booking` above, a scope searching algorithm will be employed. The algorithm will look in request, flash, view, flow, and conversation scope for the variable. If no such variable is found, an `EvaluationException` will be thrown.