Spring Web Flow 1.0 Upgrade Guide

By Keith Donald and Erwin Vervaet

Spring Web Flow 1.0 delivers powerful features such as render actions (1), evaluate actions (2), set actions (3), flow execution attributes (4), and flash scope (5). It provides enhanced documentation, strong flow definition validation, smart defaults, and a complete custom Spring 2.0 configuration schema for configuring the flow execution engine.

On the road to the generally available 1.0 release, Spring Web Flow has undergone several user affecting changes. Most of these changes occurred between PR5 and 1.0 EA and from 1.0 RC3 to 1.0 RC4. This upgrade guide exists to communicate the notable changes and support existing users in an upgrade to 1.0 final from 1.0 RC3 or earlier.

We sincerely appreciate you journeying with us to Spring Web Flow 1.0 over the last 18 months. Because of your feedback we can say confidently Spring Web Flow is the most powerful and complete UI flow engine available today. 1.0 final is an important milestone, delivering stability, innovation, and a proven foundation to build upon. We hope it continues to add great value in your projects.

The main user affecting changes begin on the next page, ordered by their relevance.

1. Flow definitions now use XML schema instead of XML DTD.

Previous:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE flow PUBLIC "-//SPRING//DTD WEBFLOW 1.0//EN"
        "http://www.springframework.org/dtd/spring-webflow-1.0.dtd">

<flow start-state="startingPoint">

</flow>
```

Now:

```
<?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-1.0.xsd">

    <start-state idref=”startingPoint”/>

</flow>
```

Note: the XSD support requires Xerces 2 on JDK 1.4 or JDK 5.

2. Bean (POJO) Action XML has changed.

Previous:

```
<action-state id="processSale">
    <action bean="saleProcessor" method="process(${flowScope.sale})"/>
    <transition on="success" to="finish"/>
</action-state>
```

Now:

```
<action-state id="processSale">
    <bean-action bean="saleProcessor" method="process">
        <method-arguments>
            <argument expression="flowScope.sale"/>
        </method-arguments>
    </bean-action>
    <transition on="success" to="finish"/>
</action-state>
```

3. A stable public API has been extracted from the more-volatile flow execution engine. As a result, 1.0 RC3 public APIs such as the RequestContext have been moved around.

Recommendation: when upgrading to 1.0 from 1.0 RC3 or earlier run a fresh Organize Imports on your Spring Web Flow types such as Actions to import relocated types from their new locations.

4. Some signatures on the RequestContext have changed.

```
- public Flow getActiveFlow() throws IllegalStateException;
+ public FlowDefinition getActiveFlow() throws IllegalStateException;
```

Flow definition is an extracted interface part of the stable definition package. “Flow” is now the default FlowDefinition implementation part of the more volatile engine and encapsulated.

```
- public State getCurrentState() throws IllegalStateException;
+ public StateDefinition getCurrentState() throws IllegalStateException;
```

State definition is an extracted interface part of the stable definition package. “State” is now the base StateDefinition implementation part of the more volatile engine and encapsulated.

```
- public AttributeMap getRequestScope();
+ public MutableAttributeMap getRequestScope();
```

In 1.0 RC3, Attribute was a class and part of the root webflow package. Now, MutableAttributeMap is an extracted interface part of the stable core.collection package. “AttributeMap” is its superinterface also part of the core.collection package. Core map operations such as get and put are the same. The map implementation is encapsulated.

```
- public AttributeMap getFlowScope();
+ public MutableAttributeMap getFlowScope();
```

The same notes above apply.

```
- public AttributeMap getConversationScope();
+ public MutableAttributeMap getConversationScope();
```

The same notes above apply.

5. Some system defaults have changed.

```
FormAction.formErrorsScope
- ScopeType.REQUEST
+ ScopeType.FLOW

alwaysRedirectOnPause
- false
+ true

repositoryType
- simple (default)
+ continuation
```

6. The way you configure the Spring Web Flow system has changed.

Previous on Spring 1.2.8:

```
<!-- Launches new flow executions and resumes existing executions. -->
<bean id="flowExecutor" class="org.springframework.webflow.executor.FlowExecutorImpl">
    <constructor-arg ref="repositoryFactory"/>
    <property name="redirectOnPause" value="true"/>
</bean>

<bean id="repositoryFactory" class="org.springframework.webflow.execution.repository.
continuation.ContinuationFlowExecutionRepositoryFactory">
    <constructor-arg ref="flowRegistry"/>
</bean>

<!-- Creates the registry of flow definitions for this application -->
<bean id="flowRegistry" class="org.springframework.webflow.registry.XmlFlowRegistryFactoryBean">
    <property name="flowLocations" value=”/WEB-INF/flows/**/*-flow.xml”>
</bean>
```

Now on Spring 1.2.8:

```
<!-- Launches new flow executions and resumes existing executions: Spring 1.2 config version -->
<bean id="flowExecutor" class="org.springframework.webflow.config.FlowExecutorFactoryBean">
        <property name="definitionLocator" ref="flowRegistry"/>
</bean>

<!-- Creates the registry of flow definitions for this application: Spring 1.2 config version -->
<bean id="flowRegistry" class="org.springframework.webflow.engine.builder.xml.XmlFlowRegistryFactoryBean">
    <property name="flowLocations" value="/WEB-INF/flows/**/*-flow.xml">
</bean>
```

If you are using Spring 2.0 you can take advantage of the new `webflow-config` XSD:

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

        <!-- Launches new flow executions and resumes existing executions. -->
        <flow:executor id="flowExecutor" registry-ref="flowRegistry"/>

        <!-- Creates the registry of flow definitions for this application -->
        <flow:registry id="flowRegistry">
                <flow:location path="/WEB-INF/flows/**/*-flow.xml"/>
        </flow:registry>

</beans>
```

This is generally recommended as it gives you auto-complete and better validation. This form is Spring 2.0 only.

7. Transition event id matching expressions are now case sensitive.

This may affect your flow definitions if you were mixing case between event ids in your flows with those returned from actions or views. In the Spring Web Flow samples, numberguess was affected:

```
- <transition on="correct" to="showAnswer"/>
+ <transition on="CORRECT" to="showAnswer"/>
```

… as the criteria for transition was the GuessResult JDK 5 enum, whose enum names were all caps.

8. Input and output attribute elements have been added to provide attribute mapping convenience.

For example, instead of saying:

```
<?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-1.0.xsd">

    <input-mapper>
```

```
        <mapping source=”id” target=”flowScope.id”/>
    </input-mapper>

</flow>
```

… to map an input attribute into flow scope you may simply say:

```
<?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-1.0.xsd">

    <input-mapper>
        <input-attribute name=”id”/>
    </input-mapper>

</flow>
```

This form is more concise but less flexible. The original form above still is supported. See the XSD documentation for the input-mapper and output-mapper elements for more information.

Did we miss documenting something that affected you?
- **Visit http://forum.springframework.org to tell us about it.**
- **We also recommend you review the sample applications, reference manual, and change log.**

References

(1) render actions are always executed before a view (response) is rendered.

(2) evaluate-actions evaluate expressions against the state of the flow. Expressions can retrieve property values, invoke methods, etc. Evaluation results can drive state transitions and can be exposed to the flow automatically.

(3) set-actions set flow variable values during the course of flow execution. You may set values in any supported scope: request, flash, flow, or conversation.

(4) User-assigned attributes may influence flow execution behavior. The “alwaysRedirectOnPause” attribute forces a redirect each time an execution pauses, to achieve POST+REDIRECT+GET automatically.

(5) Data placed in flash scope lives through the current request and the next request into the active flow session. Flash scope is shorter than flow scope but longer than request scope.