



DIGITAL FACTORY 7.0

## MVC using Spring Web Flow

Rooted in Open Source CMS, Jahia's Digital Industrialization paradigm is about streamlining Enterprise digital projects across channels to truly control time-to-market and TCO, project after project.

**Jahia Solutions Group SA**

9 route des Jeunes,  
CH-1227 Les acacias  
Geneva, Switzerland

<http://www.jahia.com>

# Summary

1	Introduction.....	4
2	Why Spring Web Flow .....	5
2.1	Drivers .....	5
2.2	Criteria .....	6
2.3	Spring Web Flow as a choice .....	6
3	What Is Spring Web Flow .....	7
3.1	Definition .....	7
3.2	History .....	7
3.3	Basic concepts .....	7
4	How Is It Integrated in Digital Factory .....	8
4.1	Rendering concepts .....	8
4.1.1	Templates, areas and views.....	8
4.1.2	Request rendering flow .....	9
4.1.3	Views.....	9
4.1.4	View (script) types .....	10
4.2	What Digital Factory does for you .....	11
4.3	What you need to use it.....	11
4.3.1	Bean definitions file .....	12
4.3.2	View of type “flow” .....	12
4.3.3	Flow definition file.....	12

4.3.4	Validation.....	14
5	Examples.....	15
5.1	Spring Web Flow showcase .....	15
5.2	Digital Factory server and site panels .....	15
6	Summary .....	16

# 1 Introduction

This document covers basic aspects of using Spring Web Flow for implementing advanced MVC-based content rendering, form processing, data binding and input validation in Digital Factory 7.0.

The Spring Web Flow integration was first introduced during a [JahiaOne](http://www.jahiaone.com/jahiaOne-2014/slides-videos-pictures) technical session and the corresponding slides and video are available online at <http://www.jahiaone.com/jahiaOne-2014/slides-videos-pictures>.

## 2 Why Spring Web Flow

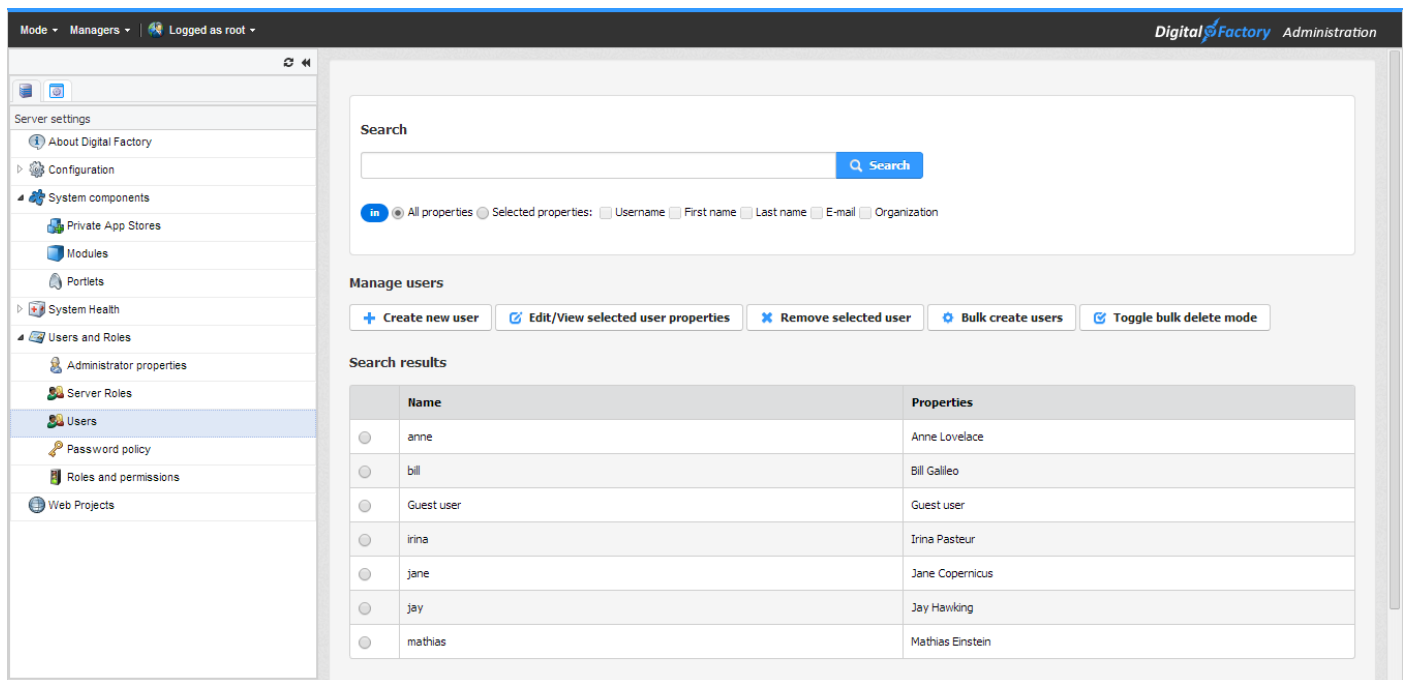
### 2.1 Drivers

The main drivers for introducing Spring Web Flow into Digital Factory were mainly related to our customers' and integrator's feedback claiming the need for powerful:

- Form processing and automatic data binding
- User input validation
- Business logic
- Integration points with external services and systems



And the second driver – effectively the triggering one – is our effort for refactoring Server and Site Management parts in Digital Factory.



The screenshot shows the 'Digital Factory Administration' interface. On the left is a sidebar with a tree view containing categories like 'Server settings', 'Configuration', 'System components', 'System Health', 'Users and Roles', and 'Web Projects'. The 'Users' item under 'Users and Roles' is selected. The main content area is titled 'Manage users' and includes a search bar, a filter section for 'All properties' (Username, First name, Last name, E-mail, Organization), and a table of search results.

	Name	Properties
<input type="radio"/>	anne	Anne Lovelace
<input type="radio"/>	bill	Bill Galileo
<input type="radio"/>	Guest user	Guest user
<input type="radio"/>	irina	Irina Pasteur
<input type="radio"/>	jane	Jane Copernicus
<input type="radio"/>	jay	Jay Hawking
<input type="radio"/>	mathias	Mathias Einstein

## 2.2 Criteria

After identifying the needs, we've defined the main criteria for what we wanted to achieve:

- Seamless integration into Jahia rendering system
- Automatic data binding and form processing
- Support for complex / custom validation
- Advanced control over “page navigation”
- Easy to learn and use

## 2.3 Spring Web Flow as a choice

We finally have chosen Spring Web Flow mainly because:

- It matched our goals
- It comes from the Spring landscape (bean definitions, MVC, type converters, binding, etc.)
- It is a control layer on top of MVC (Spring MVC, JSF, Struts, etc.)

Finally we came up with the new type of content views in Digital Factory, which are based on Web Flow.

## 3 What Is Spring Web Flow

### 3.1 Definition

We would like to cite an original definition of the Web Flow from the official site:

*Spring Web Flow builds on Spring MVC and allows implementing the "flows" of a web application. A flow encapsulates a sequence of steps that guide a user through the execution of some business task.*

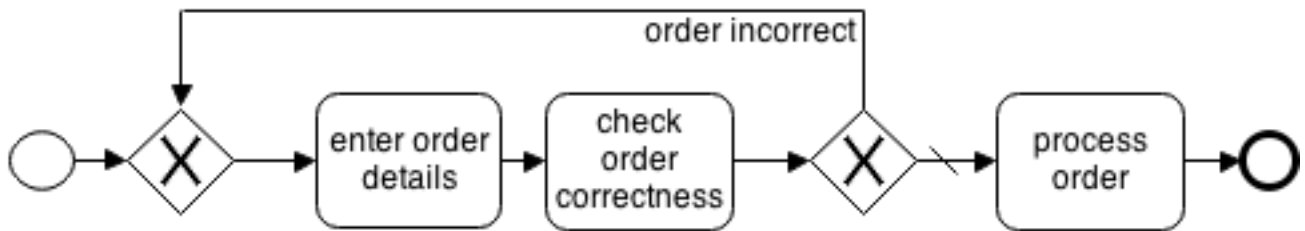
(source: <http://projects.spring.io/spring-webflow/>)

### 3.2 History

- Started as an extension of Spring MVC by Erwin Vervaeke in 2004
- Released in October 2006
- Version 2.3.2 was released in January 2013

### 3.3 Basic concepts

- Introduced flow concept
- Extended application scopes
- Essential language elements: flow, view-state, transition, end-state etc.
- Actions (<evaluate>): can be executed on various flow points



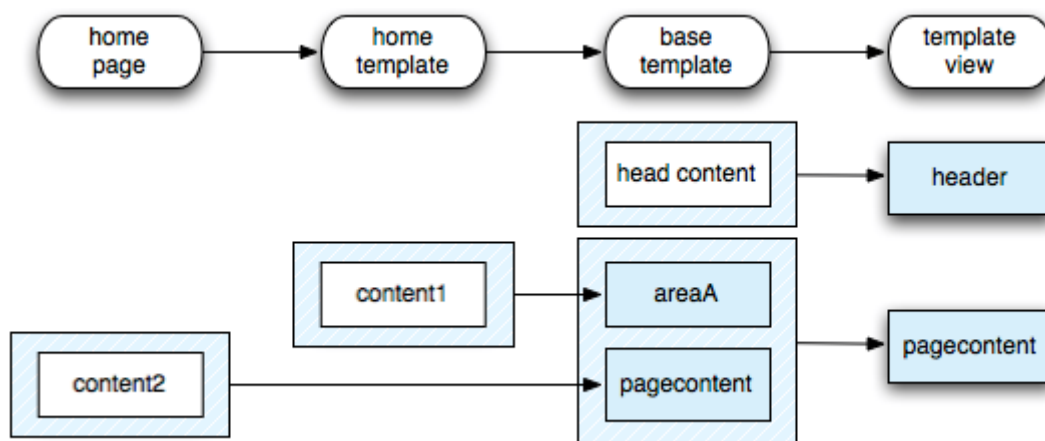
## 4 How Is It Integrated in Digital Factory

### 4.1 Rendering concepts

In this section we give as a reminder a quick overview of rendering concept in Digital Factory.

#### 4.1.1 Templates, areas and views

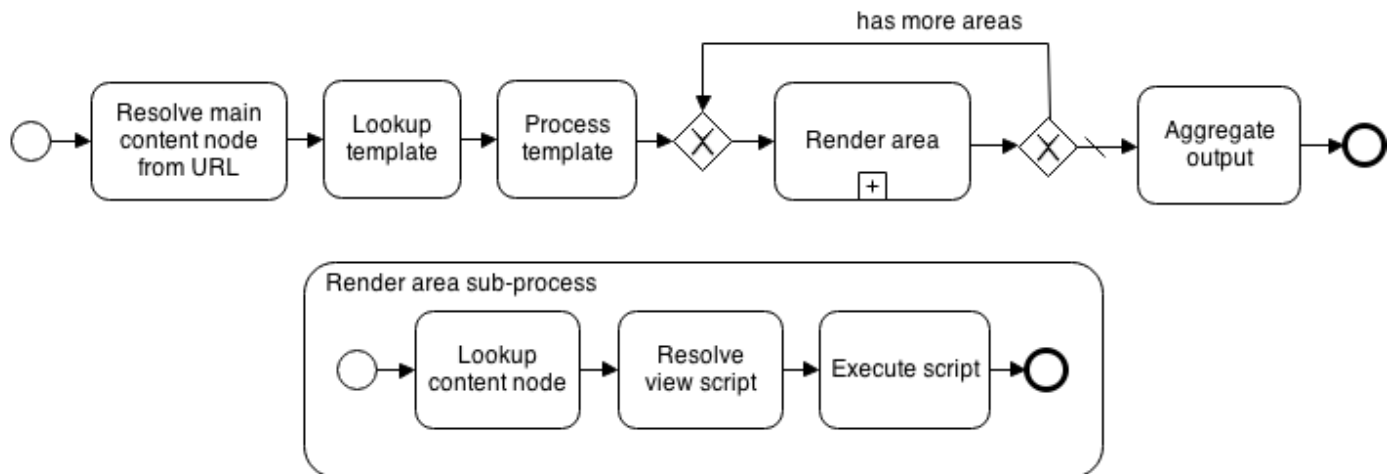
- Each content node needs a *template* (full page view)
- Template will define which elements to display and where to display them using *areas*
- An area renders a content node using a *view*





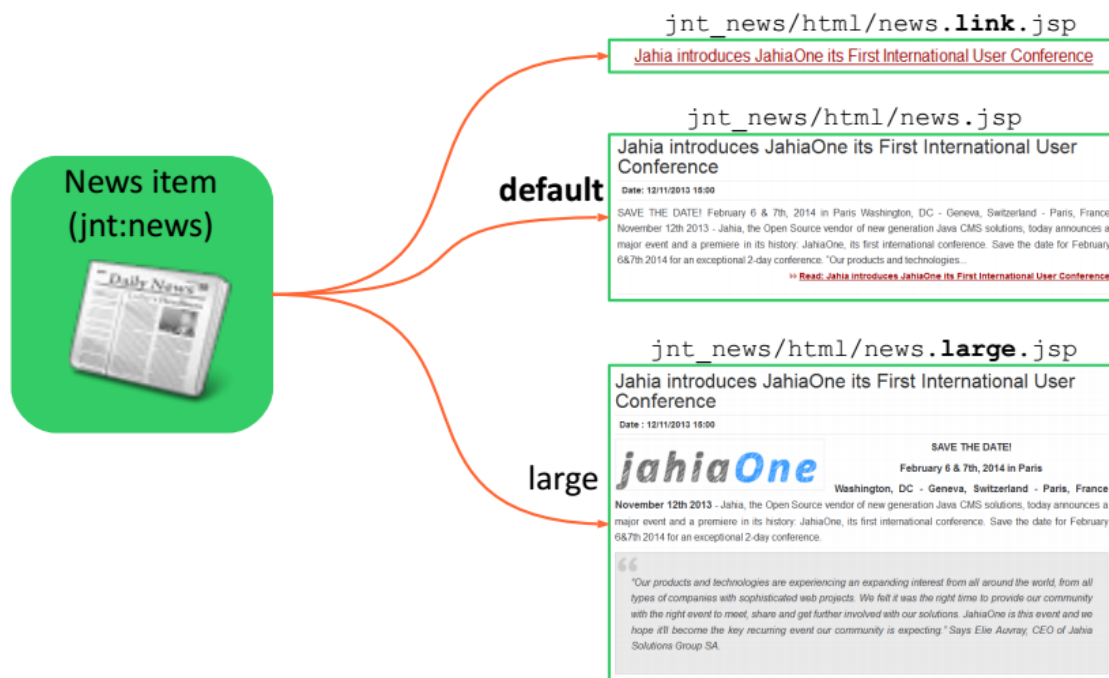
## 4.1.2 Request rendering flow

The request rendering flow in Digital Factory can be represented as follows:



## 4.1.3 Views

To render a *content node* Jahia needs a script file(s) called a *view*:



The example above shows how a same content item, a news entry in this case, can have multiple rendering views, depending on the context or needs.

## 4.1.4 View (script) types

The following view (script) types are supported in Digital Factory:

- JSP (`javax.servlet.RequestDispatcher`)
- JSR-223: Groovy, Velocity, FreeMarker, etc. (`javax.script.ScriptEngineManager`)
- Web Flow (`javax.servlet.RequestDispatcher`) dispatching to a dedicated controller (`/flow/*`)

In Digital Factory 7.0 we added the Web Flow type, which is acting as an intermediate layer, dispatching the request to a dedicated flow controller. The flow controller is responsible for handling flow initialization and executing “navigation” control and logic.

## 4.2 What Digital Factory does for you

The Digital Factory automatically:

- detects and registers/unregisters flow definitions in modules;
- does proper dispatching to controller, URL mappings, handler mapping, view resolution, message source resolution (i18n), etc.

It means the flow-based views are automatically detected in your module and registered during start of the module and automatically unregistered when the module is stopped.

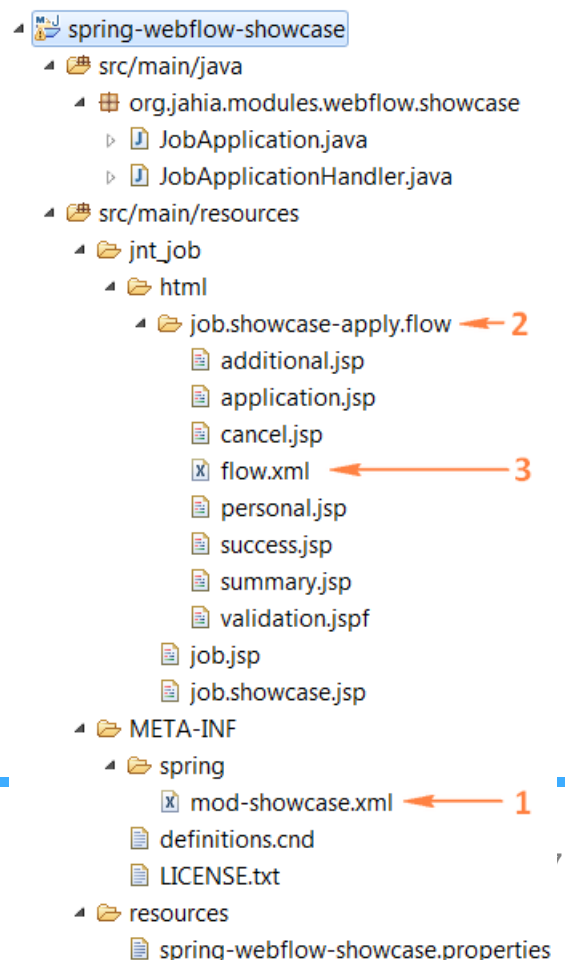
Thus new views will be available for content rendering.

All the view resolution part (i.e. resolving a view script for a particular flow view state) and dispatching to it is also done transparently by the Digital Factory without the need to introduce special URL and handler mappings.

## 4.3 What you need to use it

Here are the things that you have to include into your module:

- a Spring bean definitions file (1)
- a view (folder) of type “flow” (2)
- a flow definition file – flow.xml (3)
- flow view template files
- action handlers
- validation
- i18n resources



### 4.3.1 Bean definitions file

The bean definitions file is mandatory to “tell” Digital Factory that the module contains flow definitions to lookup and register. The bean definitions file doesn’t have to contain any beans, it just has to be present.

Nevertheless, if you are using resource bundles for internationalization you should include the following bean definition into that file:

```
<bean id="messageSource"  
class="org.jahia.utils.i18n.ModuleMessageSource"/>
```

And in case the JSR-303 “Bean Validation” annotations are used the following entry has to be present also:

```
<mvc:annotation-driven conversion-  
service="springTypeConversionService"/>
```

### 4.3.2 View of type “flow”

The view of type “flow” is actually represented by a folder, not a single file as in case of normal views.

The folder name by convention has the same format as usual view files, but also ends with “.flow” to indicate a particular view type. In the example above the folder is named `job.showcase-apply.flow`, which means it is a rendering a view with the name `showcase-apply` for the node type `jnt:job`. And the type of the view is “flow”, indicating that a dispatching to the Spring Web Flow controller is done for rendering the output.

### 4.3.3 Flow definition file

A single flow definition file, which by convention has the name `flow.xml`, is located in the view folder.

This file contains all the elements, required to describe and process the flows:

- view states
- transitions between views
- action handlers
- model binding and validation switches
- etc.

```
<?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
      ">

    <var name="handler"
class="org.jahia.modules.webflow.showcase.JobApplicationHandler"/>
    <var name="jobApplication"
class="org.jahia.modules.webflow.showcase.JobApplication"/>

    <view-state id="personal" model="jobApplication">
        <transition on="next" to="additional"/>
    </view-state>

    <view-state id="additional" model="jobApplication">
        <transition on="previous" to="personal" />
        <transition on="next" to="application" />
    </view-state>

    <view-state id="application" model="jobApplication">
        <transition on="previous" to="additional" />
        <transition on="next" to="summary" />
    </view-state>

    ...
```

Please, refer to “Chapter 3. Defining Flows” of the “[Spring Web Flow Reference Guide](http://docs.spring.io/spring-webflow/docs/2.3.2.RELEASE/reference/html/ch03.html)” for the syntax and meaning of various flow definition elements: <http://docs.spring.io/spring-webflow/docs/2.3.2.RELEASE/reference/html/ch03.html>

## 4.3.4 Validation

Spring Web Flow supports advanced input validation on state transitions, also including [JSR-303](#) (Bean Validation) support.

Here is an example of annotation-based validation:

```
public class JobApplication implements Serializable {

    @NotEmpty
    @Email
    private String email;

    private UploadedFile resume;

    private int salary;

    @DateTimeFormat(iso = ISO.DATE)
    private Date startDate;

    @Pattern(regexp = "application/*.pdf")
    public String getCoverLetterContentType() {
        return coverLetter != null ? coverLetter.getContentType() :
"application/pdf";
    }

    @Max(value = 1 * 1024 * 1024L)
    public long getCoverLetterLength() {
        return coverLetter != null ? coverLetter.getFile().length() : 0;
    }

    ...
}
```

## 5 Examples

In order to ease the start with the Spring Web Flow views in Digital Factory various examples could be used, which are described in brief in the next sections.

### 5.1 Spring Web Flow showcase

This is a custom module for the Digital Factory platform that shows the use of Spring Web Flow MVC technology for creating powerful content views, supporting automatic form data binding, advanced validation and integration of complex business processing logic.

Module source code: <https://github.com/shyrkov/jahia-spring-webflow-showcase>

Presentation slides (JahiaOne): <http://www.slideshare.net/Jahia/112-jahia-onespringwebflow>

Video recording of the presentation (JahiaOne): <http://youtu.be/TUESY3l5Xlw>

### 5.2 Digital Factory server and site panels

In Digital Factory the Server and Site Administration parts were refactored using Spring Web Flow based views and move to the selector (left side panel) in edit mode.

All those management parts could serve also as a nice examples and quick-starts for your Web Flow development.

## 6 Summary

As a summary we will remind, what are the main application areas of the new Web Flow based views and what can be done with it:

- Flow-based data processing
- Validation
- Complex business logic
- External data / service integration

With this new powerful mechanism of data processing we have met our goals and our needs are covered.

Please, do not hesitate to send us your feedback or suggestions!





**Jahia Solutions Group SA**

9 route des Jeunes,  
CH-1227 Les acacias  
Geneva, Switzerland

<http://www.jahia.com>

