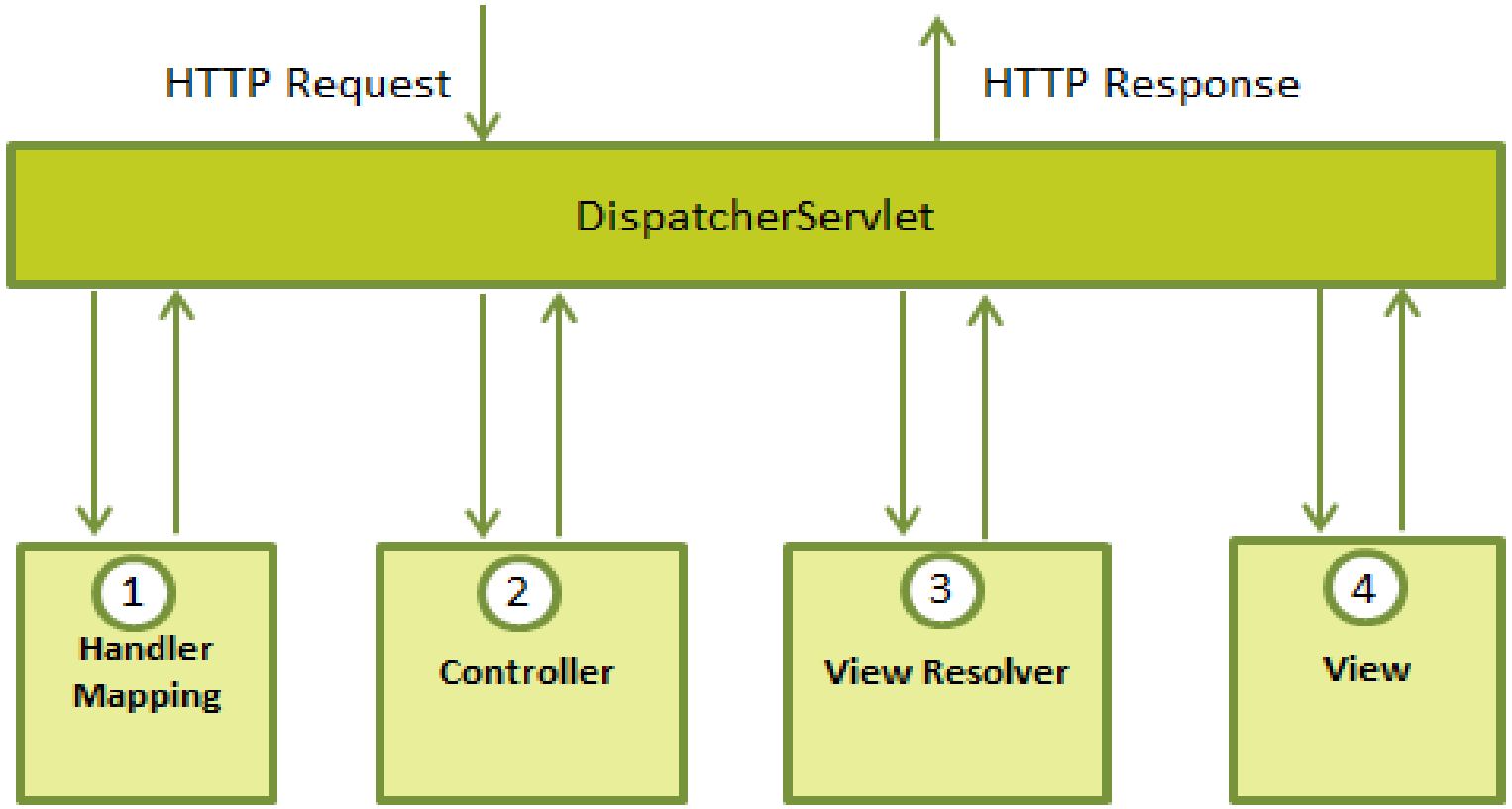
FreeMarker in Spring Web

Marin Kalapać

Agenda

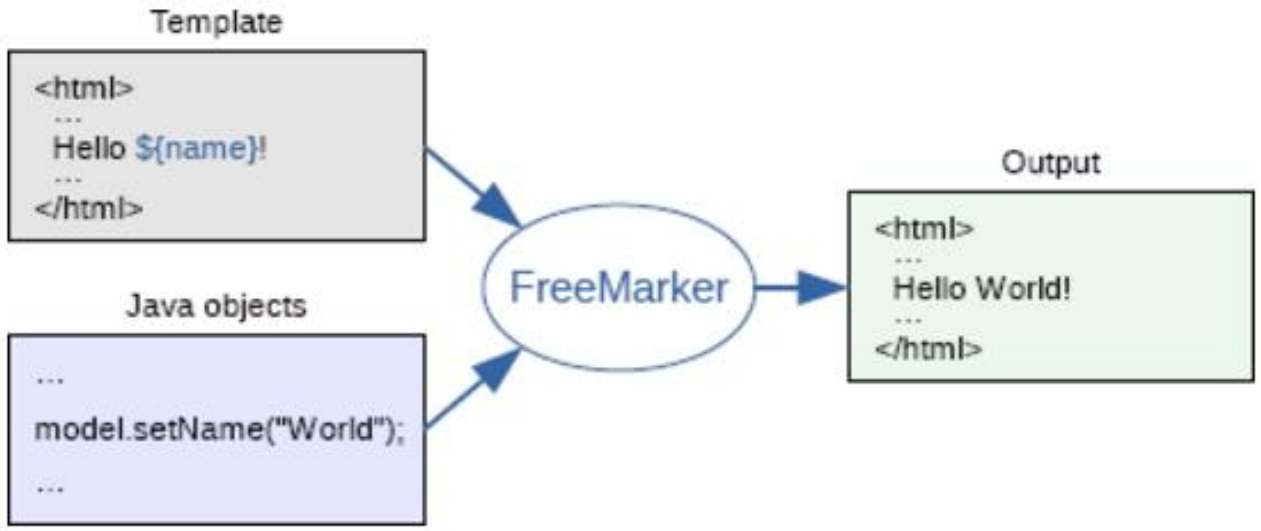
- Spring MVC view resolving in general
- FreeMarker – what is it and basics
- Configure Spring MVC to use Freemarker as view engine instead of jsp
- Commonly used components

Spring MVC view resolving – How it works



Apache FreeMarker

- FreeMarker is a template engine
  - Template engine generates text output (like HTML, e-mails etc.) based on templates and changing data provided to it



Apache FreeMarker

- FreeMarker was originally created with generating HTML pages in MVC web application frameworks in mind, but it can be used for any other purpose

- Unlike jsp, it isn't bound to servlets so it can be used in non-web application environments as well.

- Since it is not bound to servlets, you can even do Unit test on your templates

Apache FreeMarker

- It is free software, part of Apache Software Foundation (currently in Apache Incubator)
- Official page (with detailed manual and other useful info)
  - http://freemarker.org/

Apache FreeMarker basics

- Freemarker files are stored in .ftl files (for example home.ftl)
  - ftl = **F**reeMarker **T**emplate **L**anguage
- Overall structure of file

```
<html>
<head>
</head>
<body>
    <#-- Greet conference -->
    Hello ${conference}
    My name is ${user.name} ${user.surname}
    <#if attendes.size <= 1>
        Tumbleweed rolling...
    </#if>
</body>
</html>
```

FreeMarker in Spring

- Officially supported for use in Spring projects
- Even recommended as template engine by official page
  - *„As of Spring Framework 4.3, Velocity support has been deprecated due to six years without active maintenance of the Apache Velocity project.* ***We recommend Spring's FreeMarker support instead****, or Thymeleaf which comes with Spring support itself."*
- Officially supported for use as view technology within Spring MVC
  - Rumoured to be only supported in future releases of Spring
    - https://goo.gl/joZxOl
    - https://goo.gl/iftKVV

Replacing jsp with freemarker (Classic Spring MVC)

- Add dependency
  - Maven repository or include Jar directly
- Make changes to servlet-config

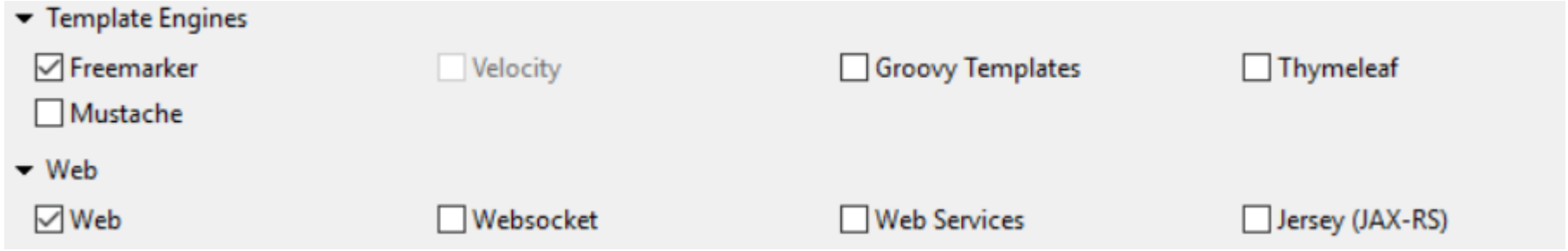
```
<beans:bean id="freemarkerConfig"
    class="org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer">
    <beans:property name="templateLoaderPath" value="/WEB-INF/views/" />
</beans:bean>

<beans:bean id="viewResolver"
    class="org.springframework.web.servlet.view.freemarker.FreeMarkerViewResolver">
    <beans:property name="cache" value="true" />
    <beans:property name="prefix" value="" />
    <beans:property name="suffix" value=".ftl" />
</beans:bean>
```

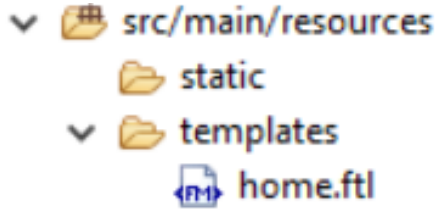
- Include home.ftl file in WEB-INF/views/ folder

Replacing jsp with freemarker (Spring Boot)

- Just tell Spring boot that you are going to use freemarker ☺



- Spring boot autoconfigures view resolver to check for .ftl files in templates folder, found in src/main/resources



Syntax of commonly used components

- Setup for examples:
  - Spring MVC and Freemarker added via Spring boot
  - IDE: Eclipse STS with Jboss Freemarker plugin installed

Syntax - variables

1. Prints value of variable conference

```
${conference}
```

2. Prints value of variable conference or ignores it if is null

```
${conference!}
```

3. Prints value of variable conference or prints a default value if is null

```
${conference!'some default value'}
```

4. Prints subvariable name of variable user

```
${user.name}
```

5. Prints subvariable name of variable user or prints a default value if is null

```
${(user.username)!'Not available'}
```

Syntax – variables

- Jsp comparison – write variable, or default value if variable is empty

JSP

```
<c:if test="${empty conference}">
    <c:out value="${conference}" />
</c:if>
<c:if test="${not empty conference}">
    some default value
</c:if>
```

FreeMarker

```
${conference!'some default value'}
```

Syntax - collections

- Freemarker can work with most of collection types in java
- Collections iteration

```
<#list users as user>
    Doing something for every user in users...
</#list>
```

- FTL offers some useful built-ins for lists, for example:
  - Size of list `${users?size}`
  - Null and empty check

```
<#if users?has_content>
    <#list users as user>
    Doing something for every user in users...
    </#list>
<#else>
    list is null or empty, doing something else
</#if>
```

Syntax – collections

- Jsp comparison – write out list or default value if list is empty

JSP

```
<c:choose>
    <c:when test="${not empty users}">
        <c:forEach items="${users}" var="user">
            <tr>
                <td><c:out value="${user.name}" /></td>
                <td><c:out value="${user.surname}" /></td>
            </tr>
        </c:forEach>
    </c:when>
    <c:otherwise>
        <p>No content!</p>
    </c:otherwise>
</c:choose>
```

FreeMarker

```
<#if users?has_content>
    <#list users as user>
        <tr>
            <td>${user.name}</td>
            <td>${user.surname}</td>
        </tr>
    </#list>
<#else>
    <p>No content!</p>
</#if>
```

Include templates in templates

- We often have some repetitive code in our view files, like menu headers, copyright in footer etc.

- FTL provides means to put repetitive code in separated template files and then inclusion of those files in template.

- Example bellow includes template file menu.ftl in place where we define #include directive

```
<#include "common/menu.ftl">
```

Macros

- A macro is a template fragment associated with a variable.
  - Basically a way to define your own directives
- Macro definition

```
<#macro greet conference>
  <font size="+2">Hello people at ${conference}!</font>
</#macro>
```

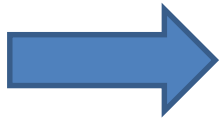
- Macro usage

```
<@greet conference = 'Javacro'/>
```

Macros

- Better example of macro usage

```
<#macro userstable collection>
    <#if collection?has_content>
            <table class="table table-bordered table-striped">
                    <thead>
                        <tr>
                            <th>Name</th>
                            <th>Surname</th>
                        </tr>
                    </thead>
                    <tbody>
                    <#list collection as element>
                        <tr>
                            <td>${element.name}</td>
                            <td>${element.surname}</td>
                        </tr>
                    </#list>
                    </tbody>
            </table>
        <#else>
                <p>No content!</p>
        </#if>
</#macro>
```



```
<@userstable users/>
```

Spring Macro

- First we need to import spring library into ftl file

```
<#import "/spring.ftl" as spring />
```

- Afterwards we use it as a classic macro
  - Example: macro to produce relative url

```
<link rel="stylesheet" href="<@spring.url '/css/bootstrap.min.css'/>" />
```

- We can also configure freemarkerConfig bean to auto import libraries to all ftl files

List of available spring macros

macro	FTL definition
message (output a string from a resource bundle based on the code parameter)	`<@spring.message code/>`
messageText (output a string from a resource bundle based on the code parameter, falling back to the value of the default parameter)	`<@spring.messageText code, text/>`
url (prefix a relative URL with the application's context root)	`<@spring.url relativeUrl/>`
formInput (standard input field for gathering user input)	`<@spring.formInput path, attributes, fieldType/>`
formHiddenInput * (hidden input field for submitting non-user input)	`<@spring.formHiddenInput path, attributes/>`
formPasswordInput * (standard input field for gathering passwords. Note that no value will ever be populated in fields of this type)	`<@spring.formPasswordInput path, attributes/>`
formTextarea (large text field for gathering long, freeform text input)	`<@spring.formTextarea path, attributes/>`
formSingleSelect (drop down box of options allowing a single required value to be selected)	`<@spring.formSingleSelect path, options, attributes/>`
formMultiSelect (a list box of options allowing the user to select 0 or more values)	`<@spring.formMultiSelect path, options, attributes/>`
formRadioButtons (a set of radio buttons allowing a single selection to be made from the available choices)	`<@spring.formRadioButtons path, options separator, attributes/>`
formCheckboxes (a set of checkboxes allowing 0 or more values to be selected)	`<@spring.formCheckboxes path, options, separator, attributes/>`
formCheckbox (a single checkbox)	`<@spring.formCheckbox path, attributes/>`
showErrors (simplify display of validation errors for the bound field)	`<@spring.showErrors separator, classOrStyle/>`

Summary

- Why use it instead of jsp (opinionated view ☺)
  - Better looking syntax
  - Useful built-in functions
  - Not dependent on servlets
  - Using templates inside templates
  - Macros
  - Performance

Questions?

- Thank you!

- Contact: marin.kalapac@trilix.eu