

**FIAP GRADUAÇÃO**

# DIGITAL BUSINESS ENABLEMENT

Prof. Me. Thiago T. I. Yamamoto

#02 – SPRING MVC – BOOTSTRAP E TEMPLATES



thiagoyama



thiagoyama@gmail.com

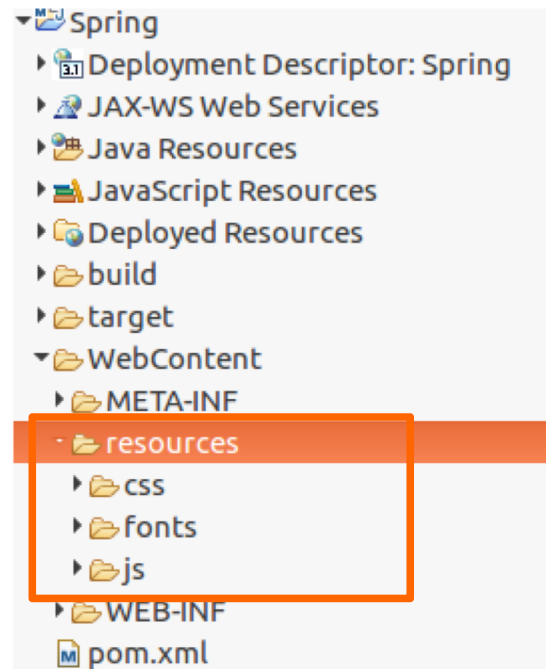
## #02 – BOOTSTRAP E TEMPLATES

---

- Spring MVC e Bootstrap
- Templates
- Atributos no Template
- JSP Fragments

# BOOTSTRAP

- Para utilizar o **bootstrap** no projeto **Spring MVC**, primeiro faça o download dos arquivos em: <http://getbootstrap.com/> e <http://jquery.com/>
- Crie um diretório dentro de **WebContent** com o nome **resources** e copie os arquivos js, css e font.



- É preciso configurar o Spring MVC para que o Front Controller (servlet do spring) não processe os arquivos de css, js, fontes e imagens.
- Para isso adicione no arquivo **servlet-context.xml** a configuração:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.1.xsd">

    <context:component-scan base-package="br.com.exemplo" />

    <mvc:annotation-driven />

    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>

    <mvc:default-servlet-handler/>

</beans>
```

- Por fim, vamos adicionar as referências do css e js.
- Porém, precisamos da taglib `<c:url>` para referenciar o path correto dos arquivos **css** e **js**.

```
<link rel="stylesheet" type="text/css"
      href="<c:url value="/resources/css/bootstrap.min.css"/>">

<script src="<c:url value="/resources/js/jquery-3.1.1.min.js"/>"></script>
<script src="<c:url value="/resources/js/bootstrap.min.js"/>"></script>
```

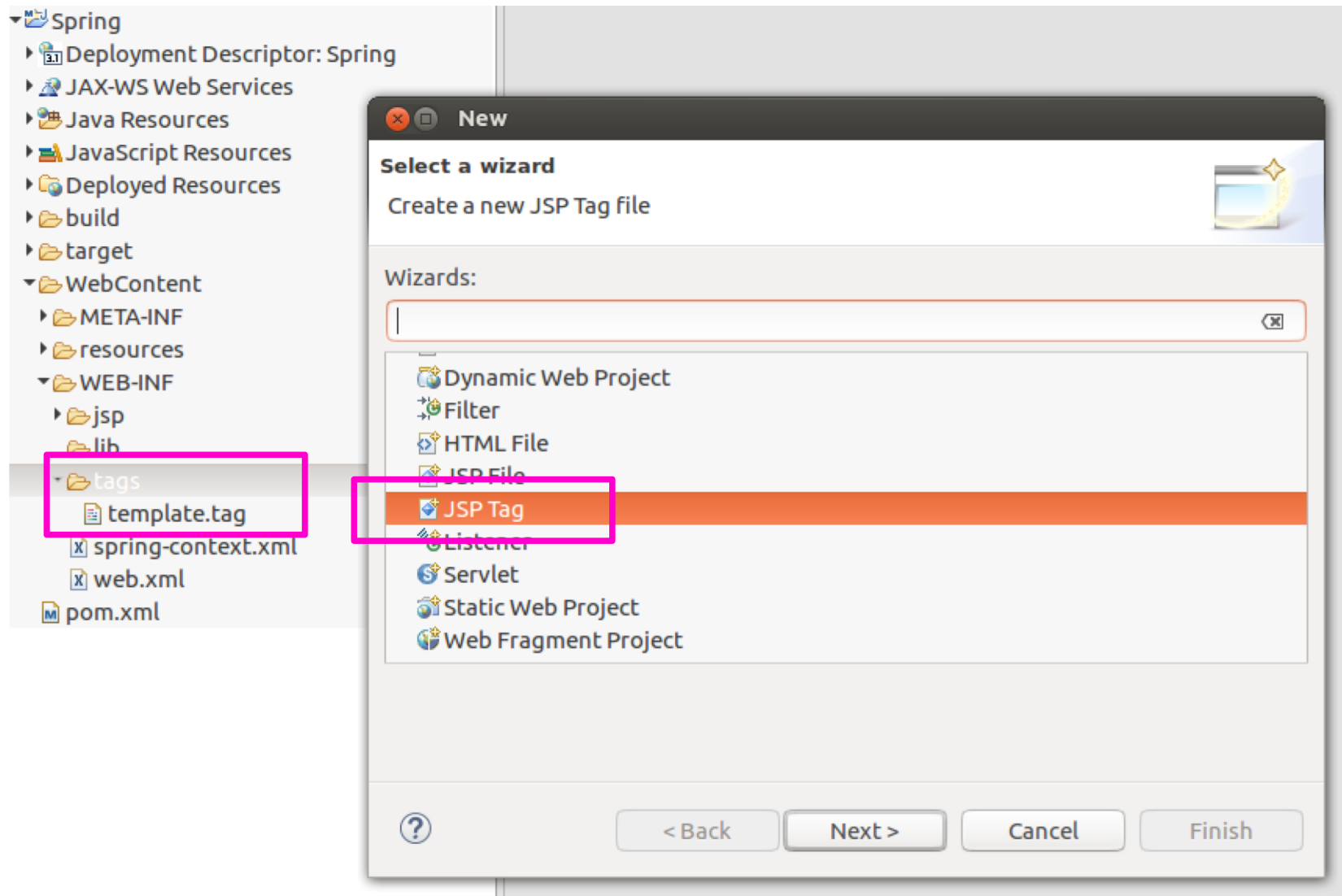
# TEMPLATE



- Existem várias formas de trabalhar com templates no Spring MVC: *plain JSP, custom tag JSP, Apache Tiles, Thymeleaf.*
- Podemos utilizar a diretiva **include** para **adicionar JSPs** nas páginas;
- Nós vamos utilizar **custom tag JSP**, ou seja, vamos criar uma tag JSP:

## Passos:

- Dentro da pasta **WEB-INF** crie um diretório chamado **tags**;
- Nesta pasta crie uma nova tag JSP, clique com o botão direito do mouse e escolha **new → other → JSP tag**.
- Dê um nome qualquer, como por exemplo: *template.tag*



- Na nova tag, vamos adicionar a estrutura básica do HTML, taglibs, css, js (*bootstrap*), menu, rodapé e etc..
- A tag **<jsp:doBody/>** irá receber o conteúdo das outras páginas.

```
<%@ tag language="java" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Spring MVC | Home</title>
<link rel="stylesheet" type="text/css" href="<c:url value="/resources/css/bootstrap.min.css"/>">
</head>
<body>

<h1>Spring MVC - Exemplo</h1>

<div class="container">
  <jsp:doBody />
</div>

<p>Todos os Direitos Reservados</p>

<script src="<c:url value="/resources/js/jquery-3.1.1.min.js"/>"></script>
<script src="<c:url value="/resources/js/bootstrap.min.js"/>"></script>

</body>
</html>
```

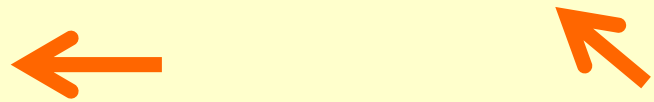


**Para utilizar o template**, precisamos declarar a taglib para acessar a nova tag (template) e depois utiliza-la;

2

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="tags" tagdir="/WEB-INF/tags" %>

<tags:template>
    <h1>Cadastro de Produto</h1>
    <form action="cadastrar" method="post">
        <input type="text" name="titulo" placeholder="Digite o título">
        <input type="text" name="preco" placeholder="Digite o preço">
        <input type="submit" value="Salvar">
    </form>
</tags:template>
```



# SPRING MVC - TEMPLATE

- É possível passar atributos (valores) para o *template*;
- Para isso é preciso declarar o nome do atributo e se é obrigatório ou não. Depois é só utilizá-lo com a **Expression Language**;

```
<%@ tag language="java" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ attribute name="title" required="true" %>

<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content Type" content="text/html; charset=UTF-8">
<title>Spring MVC ${title }</title>
<link rel="stylesheet" type="text/css" href="<c:url value="/resources/css/bootstrap.min.css"/>">
</head>
<body>

<h1>Spring MVC -Exemplo</h1>

<div class="container">
    <jsp:doBody />
</div>

<p>Todos os Direitos Reservados</p>

<script src="<c:url value="/resources/js/jquery-3.1.1.min.js"/>"></script>
<script src="<c:url value="/resources/js/bootstrap.min.js"/>"></script>

</body>
</html>
```

Declaração do atributo obrigatório

Utilização do atributo

Para enviar o atributo para o *template* basta adicionar atributo dentro da tag, passando o seu valor.

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="tags" tagdir="/WEB-INF/tags" %>

<tags:template title="Cadastro de Produto">

    <h1>Cadastro de Produto</h1>
    <form action="cadastrar" method="post">
        <input type="text" name="titulo" placeholder="Digite o título">
        <input type="text" name="preco" placeholder="Digite o preço">
        <input type="submit" value="Salvar">
    </form>

</tags:template>
```

- Dentro do *template* podemos definir outras áreas para as páginas adicionarem outros trechos de código.
- Isso é útil para os **javascripts**, pois cada página pode ter o seu código **js** específico.

## Passos:

- Declare um atributo informando o nome e definindo que é um fragmento;
- Utilize a tag **<jsp:invoke>** para definir onde será inserido o novo trecho de código.

```
<%@ tag language="java" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

1 <%@ attribute name="title" required="true" %>
  <%@ attribute name="scripts" fragment="true" %>

<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Spring MVC | ${title }</title>
<link rel="stylesheet" type="text/css" href="<c:url value="/resources/css/bootstrap.min.css"/>">
</head>
<body>

<h1>Spring MVC -Exemplo</h1>

<div class="container">
<jsp:doBody />
</div>

<p>Todos os Direitos Reservados</p>

<script src="<c:url value="/resources/js/jquery-3.1.1.min.js"/>"></script>
<script src="<c:url value="/resources/js/bootstrap.min.js"/>"></script>

2 <jsp:invoke fragment="scripts"></jsp:invoke>

</body>
</html>
```



- Na página JSP é preciso realizar alguns ajustes. Com o fragmento, é obrigatório utilizar a tag **<jsp:body>** para definir o conteúdo da página.
- Para adicionar o fragmento de código é preciso utilizar a tag **<jsp:attribute>**, informando o nome do fragment.

```
<tags:template title="Cadastro de Produto">

    <jsp:attribute name="scripts">
        <!-- Código JS... -->
    </jsp:attribute>

    <jsp:body>
        <!-- Código da página... -->
    </jsp:body>

</tags:template>
```

**Atenção:** Os fragmentos devem ser declarados antes do <jsp:body>!

**Copyright © 2017 - 2018 - Prof. Me. Thiago T. I. Yamamoto**

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

*Algumas pessoas sonham com o sucesso, outras levantam  
cedo e batalham para alcançá-lo.  
Eike Batista*