07

Mãos na massa: Primeiros passos

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

Criando e configurando o projeto

1) Baixe o **JBoss Forge** acessando a sua página de Download (http://forge.jboss.org/download) e, em ***Get Forge for your OS***, baixe o zip específico para o seu sistema operacional. Atualmente a versão completa seria a **"JBoss Developer Studio"**.

2) Extraia o zip baixado no passo anterior **dentro do seu workspace**. Em seguida, abra o seu terminal (ou Prompt de Comando), e acesse o seu **workspace**.

3) Dentro do seu **workspace**, inicie o terminal do **JBoss Forge**, acessando a sua subpasta **bin**, e executando o aplicativo **forge**, por exemplo:

```
alura-Mac-mini:workspace alura$ forge/bin/forge
```

4) Agora, crie o projeto **casadocodigo**:

```
project-new --named casadocodigo
```

5) Abra o Eclipse e no ***Project Explorer***, clique com o botão direito do mouse e acesse ***Import -> Import...***. Selecione ***Existing Maven Projects*** e clique em ***Next***. Em ***Root Directory***, clique em ***Browse...***, selecione o projeto criado acima e clique em ***Finish***.

6) Para rodar a aplicação, é necessário um servidor. Então, faça o download do Tomcat aqui (https://tomcat.apache.org/download-70.cgi) e adicione-o no Eclipse. Para isso, extraia o zip, volte ao Eclipse e, através do atalho **CTRL + 3**, acesse a view ***Servers***, clique para criar um novo servidor, escolha o Tomcat na versão 7 e aponte para a pasta do Tomcat que você extraiu anteriormente.

7) No Eclipse, na view ***Servers***, associe o projeto **casadocodigo** com o Tomcat. Basta clicar com o botão da direita do mouse sobre o servidor, selecionar ***Add and Remove...***, clicar no projeto **casadocodigo** e clicar em ***Add*** e em seguida em ***Finish***.

8) Para usar o **Spring MVC**, adicione a sua dependência ao projeto, no arquivo **pom.xml**. Abra esse arquivo e, após o fechamento da *tag* `<build>` e antes da abertura da *tag* `<properties>`, adicione as seguintes dependências:

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>4.1.0.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.apache.tomcat</groupId>
        <artifactId>tomcat-servlet-api</artifactId>
```

```
            <version>7.0.30</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.1</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp.jstl</groupId>
            <artifactId>jstl-api</artifactId>
            <version>1.2</version>
            <exclusions>
                <exclusion>
                    <groupId>javax.servlet</groupId>
                    <artifactId>servlet-api</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.glassfish.web</groupId>
            <artifactId>jstl-impl</artifactId>
            <version>1.2</version>
            <exclusions>
                <exclusion>
                    <groupId>javax.servlet</groupId>
                    <artifactId>servlet-api</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>1.6.1</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>jcl-over-slf4j</artifactId>
            <version>1.6.1</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.6.1</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.16</version>
            <scope>runtime</scope>
        </dependency>
    </dependencies>
```

Criando o primeiro Controller

9) Crie a classe `HomeController` dentro do pacote `br.com.casadocodigo.loja.controllers`. Em seguida, anote a classe com `@Controller`, para que o Spring MVC a reconheça como um *controller*.

```
@Controller
public class HomeController {

}
```

10) Agora, dentro da classe `HomeController`, crie o método `home`, que será o responsável por atender a *home* da aplicação, ou seja, o endereço `/`. Por fim, anote esse método com `@RequestMapping("/")`, para atender o endereço mencionado anteriormente, imprima a mensagem **Exibindo a home da CDC** e retorne o nome da página JSP que será exibida, **home**:

```
@Controller
public class HomeController {

    @RequestMapping("/")
    public String index() {
        System.out.println("Exibindo a home da CDC");
        return "home";
    }
}
```

11) Configure a *Servlet* do Spring MVC para que ela atenda as requisições recebidas pelo servidor. Para isso, crie a classe `ServletSpringMVC` dentro do pacote `br.com.casadocodigo.loja.conf`. Em seguida, estenda a classe `AbstractAnnotationConfigDispatcherServletInitializer` e implemente os métodos conforme abaixo:

```
public class ServletSpringMvc extends AbstractAnnotationConfigDispatcherServletInitializer{

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return null;
    }

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[] { AppWebConfiguration.class};
    }

    @Override
    protected String[] getServletMappings() {
        return new String[] {"/"};
    }

}
```

12) Dentro do método `getServletConfigClasses()`, está sendo utilizada uma referência da classe `AppWebConfiguration`. A partir dessa classe, faça a configuração do MVC para a *Servlet* do Spring. Portanto, crie também essa classe, no mesmo pacote `br.com.casadocodigo.loja.conf`, com o seguinte código:

```
@EnableWebMvc
@ComponentScan(basePackageClasses={HomeController.class})
public class AppWebConfiguration {

}
```

13) Dentro dessa classe, crie o método `internalResourceViewResolver`, responsável por indicar qual diretório estão as views:

```
@Bean
public InternalResourceViewResolver internalResourceViewResolver() {
    InternalResourceViewResolver resolver = new InternalResourceViewResolver();
    resolver.setPrefix("/WEB-INF/views/");
    resolver.setSuffix(".jsp");
    return resolver;
}
```

A partir dos métodos `setPrefix()` e `setSuffix()`, você define o local (**WEB-INF/views/**) e extensão (**jsp**) das views, respectivamente. Por fim, anotando esse método com `@Bean`, você transforma-o em um ***Bean***, para que o Spring o gerencie.

14) Resta criar o diretório do projeto no qual o Spring vai procurar as views, isto é, as suas páginas. Para isso, dentro do diretório **src/main/webapp** crie o diretório **WEB-INF/views**, onde as views serão armazenadas.

15) Agora, dentro do diretório **src/main/webapp/WEB-INF/views**, crie a página **home.jsp**, com o seguinte conteúdo:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Livros de Java, Android, iPhone, PHP, Ruby e muito mais - Casa do Código</title>
</head>
<body>
    <h1>Casa do Código</h1>
    <table>
        <tr>
            <td>Java 8 Prático</td>
            <td>Certificação OCJP</td>
        </tr>
        <tr>
            <td>TDD na Prática - Java</td>
            <td>Google Android</td>
        </tr>
    </table>
</body>
</html>
```

16) Por fim, inicie o servidor e acesse http://localhost:8080/casadocodigo/ (http://localhost:8080/casadocodigo/).