03

Modificando nosso web.xml

Transcrição

Aprendemos que criar um projeto web no Eclipse é muito simples, bastando clicar com o botão direito, selecionar "Create > New Project", fazer as configurações necessárias e acionar o `mvn jetty:run` no terminal. Porém, há alguns pontos nebulosos, como a mensagem de erro em nosso arquivo `index.jsp`.

No momento da instalação do plugin, nós ignoramos todo o conteúdo a partir da tag `<configuration>` ao passarmos o código para o `pom.xml` do projeto. Evidentemente estamos lidando com informações opcionais, mas adiante veremos quais são as configurações possíveis.

Primeiramente, verificaremos a mensagem de erro que surge em nosso arquivo `index.jsp`. O Eclipse aponta que `The superclass "javax.servlet.http.HttpServlet whas not found on the Java Build Path`. O servidor, neste caso o Jetty, transforma o `.jsp` em Servlet, compilando-a e executando-a automaticamente. O Eclipse detecta que não temos a Servlet no projeto.

No arquivo `pom.xml` temos apenas o JUnit, e o Eclipse nos adianta que a ausência de uma Servlet pode ocasionar problemas no projeto. Na prática, o servidor de aplicação adiciona esse `.jar`, por isso o projeto consegue ser executado. Contudo, essa não é uma ação detectável pelo Eclipse, e para resolvermos isso adicionaremos a *Page Servlet* no arquivo `pom.xml`.

Devemos seguir duas etapas, começando pela versão de Servlet que iremos utilizar. No *Maven Repository* procuraremos por "servlet" e selecionaremos a opção "Java Servlet API", versão "3.1.0". Assim, teremos acesso ao código a ser utilizado e o inseriremos no arquivo `pom.xml`.

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
</dependency>
```

Ao acessarmos novamente o localhost, notaremos que a mensagem `Minha loja web!` é exibida normalmente. Contudo, ainda existe uma segunda etapa. Antigamente, era necessário inserir na API de Servlet um arquivo no diretório `WEB-INF` chamado `web.xml`. Este arquivo foi criado automaticamente:

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
    <display-name>Archetype Created Web Application</display-name>
</web-app>
```

O arquivo está configurando a Servlet `2.3`. Para resolvemos o problema, usaremos a versão `3.1`. Procuraremos na internet por "web.xml servlet 3.1 exemple", e encontraremos um exemplo simples no site Mkyong (https://www.mkyong.com/web-development/the-web-xml-deployment-descriptor-examples/):

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">
</web-app>
```

Substituiremos o código antigo do arquivo `web.xml` por este modelo. Feito isso, executaremos o comando `mvn jetty:run`. O servidor está funcionando perfeitamente, e a mensagem é exibida no localhost.

Na prática, queremos utilizar uma versão mais recente da API de Servlet. Quando criamos um projeto Maven web novo, o primeiro passo é configurar a Servlet e trocar o código de `web.xml`. A mensagem de erro que era exibida para nós no começo da aula já não tem mais razão de existir.

A seguir, entenderemos melhor a configuração do Jetty.