01

Utilizando o package no projeto

Transcrição

Nesta aula criaremos em nosso projeto uma **dependência com outras bibliotecas**.

No arquivo `pom.xml`, podemos inserir quaisquer dependências de nossa escolha. Começaremos pelo *Stella Core*, uma biblioteca da Caelum disponível no Maven Repository (https://mvnrepository.com/artifact/br.com.caelum.stella/caelum-stella-core). Nesta biblioteca encontraremos APIs para se trabalhar com padronizações brasileiras, como CPFs e boletos específicos.

Usaremos a versão `2.1.2`, e inseriremos seu código de implementação no arquivo `pom.xml` do projeto.

```
<dependency>
    <groupId>br.com.caelum.stella</groupId>
    <artifactId>caelum-stella-core</artifactId>
    <version>2.1.2</version>
</dependency>
```

Serão baixados todos os `.jar`s necessários e, no diretório "Libraries > Maven Dependencies", está o arquivo `caelum-stella-core-2.1.2.jar`. Vamos testar se os conteúdos da biblioteca Stella podem ser utilizados no projeto. Na classe `ContatoServlet`, invocaremos `CPF` e `isValid()`, por exemplo. A biblioteca pode ser usada normalmente.

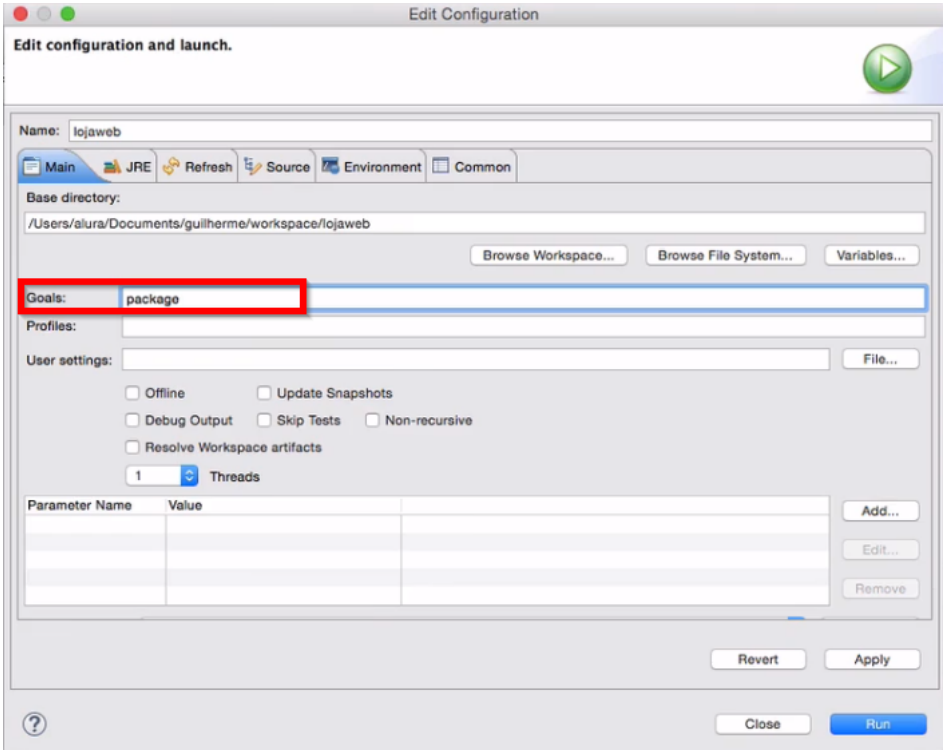
```
<***!***>

@WebServlet(urlPatterns={"/contato"})
public class ContatoServlet extends HttpServlet{

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletExcept
        new CPF("2222222222").isValido();
        PrintWriter writer = resp.getWriter();
        writer.println("<html><h2>Bata um papo conosco</h2></html>");
        writer.close();
    }
}
```

A próxima etapa é aprendermos como levar o projeto para o servidor de produção. Não precisamos utilizar todo o código fonte: em um projeto web de Java, nós geramos um arquivo `.war`. No terminal, poderíamos começar com o comando `mvn package`, mas optaremos por fazer todo o procedimento via Eclipse.

Na área "Project Explorer", selecionaremos o diretório `lojaweb` e clicaremos sobre ele com o botão direito. Assim feito, escolheremos as opções "Run As > Maven buid...". Teremos acesso a uma nova caixa de diálogo, e configuraremos o *goal* a ser executado como `package`.



Dessa forma iremos gerar o pacote do projeto, e teremos como produto o arquivo `lojaweb.war`. Vamos explorar o conteúdo deste arquivo no terminal:

```
cd target/
ls
unzip -l
```

Teremos o `MANIFEST.MF`, `index.jsp`, a classe que criamos, `ContatoServlet.class`, a dependência do Stella e o nosso arquivo `pom.xml`, dentre outros arquivos.

Está tudo pronto para enviarmos o arquivo para o servidor de aplicação. Temos então que o *goal* `mvn package` gera o pacote do nosso projeto web. O mesmo procedimento aconteceria se utilizássemos este comando para gerar o pacote do nosso antigo projeto `produtos`.

O `mvn package` irá gerar os pacotes, tanto `.jar` quanto `.war`. No primeiro caso, serão geradas apenas as classes, no segundo, teremos as classes, bibliotecas, WEB-INF, `.jsp`s, e assim por diante. Assim, o arquivo `.war` comporta todos os conteúdos do nosso projeto, inclusive as dependências.