09

Deploy da aplicação

Transcrição

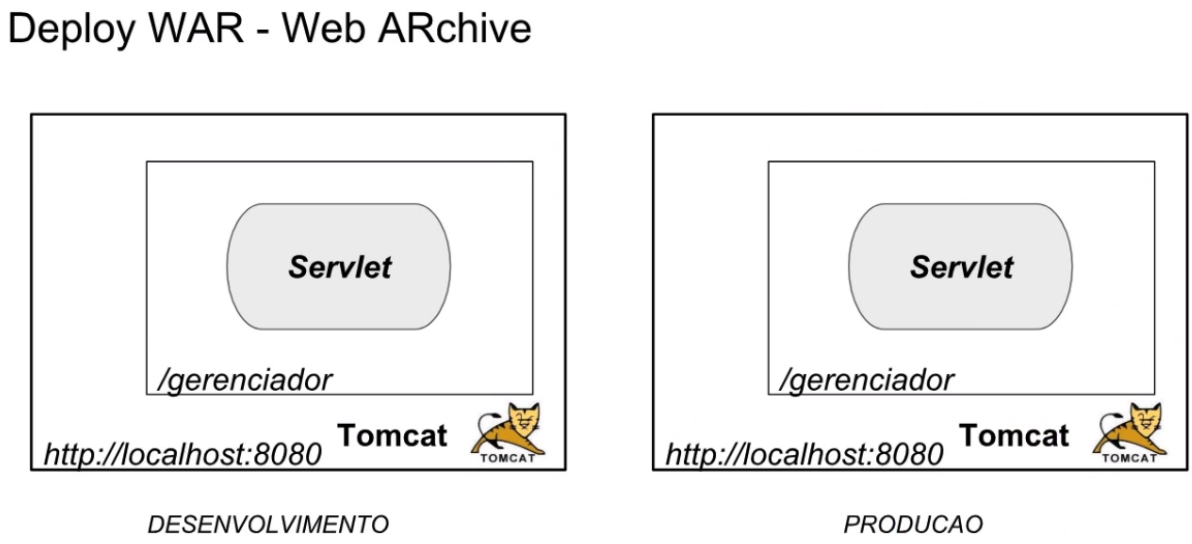
Bem-vindos! Esta aula teremos mais abordagens práticas, já que terminamos de abordar o tópico conceitual de inversão de controle. Para finalizarmos a primeira parte do nosso curso, realizaremos o **deploy**.

Estamos criando uma aplicação em nossa máquina e em algum momento precisaremos e gostaríamos de colocar nosso código em produção, o que significa executá-lo em outra máquina, seja para que o cliente veja o projeto ou para que aplicação seja de fato utilizada, afinal o usuário final não acessará nossa máquina de desenvolvimento.

Para simular o processo de deploy, criaremos mais uma instância no Tomcat. Temos ainda o arquivo zip do download do Tomcat, no início de nosso curso, versão 9.0.11. Copiaremos esse arquivo e colaremos na raiz da pasta `D:` , feito isso extrairemos o conteúdo do arquivo zip. Renomearemos esse arquivo extraído de `apache-tomcat-9.0.11` para `apache-tomcat-9.0.11-prod` . É muito comum utilizar o sufixo "prod", refere-se ao termo "produção" ou "*production*". Agora temos um Tomcat cru, não é o mesmo que utilizamos para o desenvolvimento da aplicação.

Passaremos os arquivos do projeto para a nova instância do Tomcat, e temos vários! São classes Java, arquivos de configuração, `web.xml` , bibliotecas e outros arquivos formato jsp. A ideia é compactarmos os arquivos e depois transferi-los, semelhante ao processo que fazemos com biblioteca, que nada mais são várias classes `.jar` compactadas em um zip.

Existe uma organização parecida no mundo web, mas o formato dos arquivos não são `.jar`, e sim `.war`, isto é, *Web Archive*. Passaremos os arquivos de **desenvolvimento** para a instância de **produção**, ou seja, realizaremos o processo de *deploy*.



Este processo pode conter diversas instâncias intermediárias, por exemplo uma instância responsável pela execução de testes, homologação ou produção, constituindo uma *pipeline* que garantirá a qualidade do projeto.

Neste caso, simularemos nosso deploy com apenas duas instâncias do Tomcat. Criaremos o arquivo `.war`, e para isso usaremos o Eclipse. Clicaremos com o botão direito sobre `gerenciador` selecionaremos as opções "Export > WAR File". Caso esta opção não está disponível para você, acessaremos " Export > Export...". Na nova caixa de diálogo, escreveremos "WAR" na opção "Select an export wizard", e em seguida configuraremos o destino desse arquivo para a pasta raiz D:, com o nome de `gerenciador.war`.

O arquivo `.war` é semelhante ao `.jar`, trata-se um zip renomeado. Vamos fazer o teste acessando nossa pasta raiz e renomeando o arquivo `gerenciador.war` para `gerenciador.zip`. Em seguida, extrairemos o arquivo zipado. Neste novo arquivo extraído, notaremos que há todo o conteúdo da pasta "WebContent", como "META-INF", "WEB-INF" `formAlteraEmpresa.jsp`, `novaEmpresaCriada.jsp` e assim por diante.

Temos, contudo, uma pequena mudança: ao acessarmos a pasta "WEB-INF" veremos o arquivo `web.xml`, "lib" e "classes", Esta última pasta não é exibida no Eclipse. O padrão da especificação Servlet é que existe na "WEB-INF" uma pasta "classes" que abrigará todas as classes compiladas do projeto. Dentro desse arquivo war teremos todo o código - não o .java, mas o .class - responsável pela execução do programa.

Renomearemos novamente `gerenciador.zip` para `gerenciador.war`, dessa forma o Tomcat poderá ser utilizado. Arrastaremos este arquivo para a pasta `apache-tomcat-9.0.11-prod`. Não é apenas o Tomcat que poderíamos usar, existem diversos servidores Java como Jetty, que também sabem trabalhar com o formato `.war` e especificação Servlet. Precisaremos inserir `gerenciador.war` na pasta de deploy, que normalmente possui o mesmo nome. Neste caso, especificamente, o Tomcat atribuiu à pasta de deploy o nome "webapps".

Com os arquivos alocados, resta subirmos a nova instância no Tomcat. Não queremos realizar a configuração via Eclipse, afinal não o instalaremos em uma máquina na nuvem da Amazon, e talvez essa máquina nem tenha uma interface gráfica. Iremos inicializar a máquina na linha de comando.

Abriremos uma Prompt de Comando, isto é, o terminal. Acessaremos o Tomcat de produção:

```
C:\Users\Alura>d:

D:\>cd apache-tomcat-9.0.11-prod
```

COPIAR CÓDIGO

Devemos achar a pasta para subir o Tomcat, que normalmente é chamada de `bin` e contém os arquivos de inicialização:

```
D:\>cd apache-tomcat-9.0.11-prod\bin>startup
```

COPIAR CÓDIGO

No sistema operacional Linux, o arquivo de inicialização é o `startuo.sh`, já no Windows `startup.bat`. Usaremos este último.

```
D:\>cd apache-tomcat-9.0.11-prod\bin>startup.bat
```

COPIAR CÓDIGO

Será aberta uma nova janela do terminal em que será executado o processo Java. Por fim, acessaremos novamente `apache-tomcat-9.0.11-prod` e veremos que uma nova pasta `gerenciador` foi criada, essa pasta contém todos os arquivos necessários para executar nossa aplicação.

Testaremos no navegador todas as modificações que realizamos. No navegador, escreveremos `localhost:8080` e verificaremos que o Tomcat é executado, estaremos acessando a uma tela padrão do Tomcat, já que o iniciamos fora do Eclipse. Agora abriremos nosso gerenciador, então, no navegador escreveremos `localhost:8080/gerenciador/`, e teremos a seguinte mensagem exibida na tela:

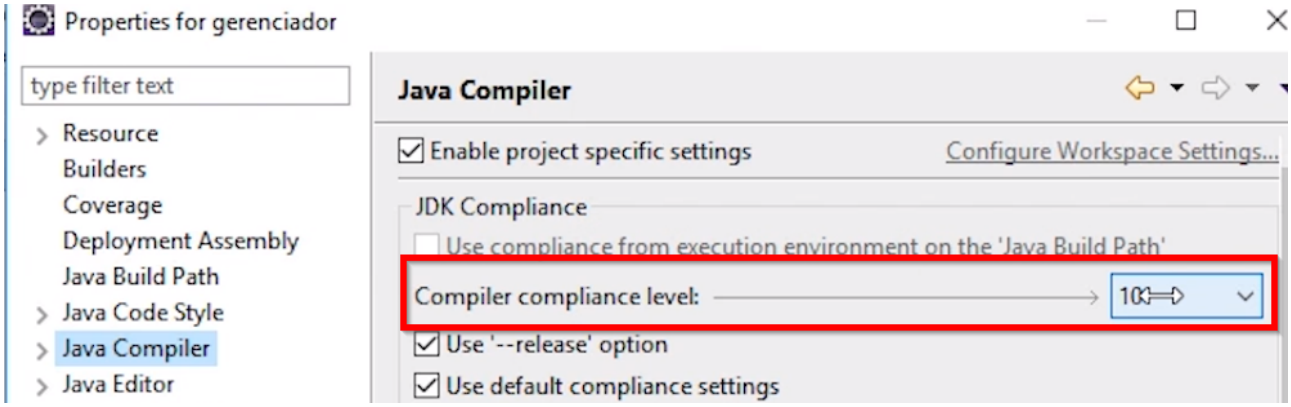
```
Bem-vindo no curso de Servlets da Alura!
```

COPIAR CÓDIGO

O html inicial foi exibido, como esperávamos. Tentaremos agora listar as nossas empresas acessando `localhost:8080/gerenciador/listaEmpresas/`. E teremos uma mensagem de erro `HTTP Status 500 - Internal Server Error`, a *root cause* é `UnsupportedClassVersionError`, relacionada a `class file version 54.0`. Isso ocorreu porque compilamos as classes com uma versão mais recente daquela que usamos para executar o Tomcat.

No Eclipse, podemos consultar qual a versão do Java que estamos utilizando para a compilação: clicaremos com o botão direito sobre a pasta "gerenciador" e selecionaremos a opção "Properties", na caixa de diálogo, procuraremos po

"Java Compiler". Verificaremos que a versão utilizada é a mais recente (`10`) disponível no momento de gravação deste curso.



Ao acessarmos o console do Tomcat veremos que a versão executada é `1.8` . Poderíamos realizar a recompilação no Eclipse utilizando uma outra versão, mas ao invés disso instruiremos o Tomcat a utilizar a versão mais recente.

Vamos entender o funcionamento do Tomcat para que possamos entender o que deve ser feito. No Explorer, clicaremos com o botão direito sobre "Este Computador" e selecionaremos a opção "Propriedades". Na nova caixa de diálogo, clicaremos sobre "Configurações avançadas do Sistema", em seguida "Variáveis de Ambiente". Caso você esteja utilizando outro sistema operacional, haverá outros percursos da janela para acessar as variáveis de ambiente, mas o mesmo conceito existe.

Há uma variável de ambiente que o Tomcat utiliza: `JAVA_HOME` , localizada em `C:\Program Files (x86)\Java\jre1.8.0_171\bin` , que define qual máquina virtual será utilizada. Clicaremos sobre a opção "Editar" e configuraremos o destino `C:\Program Files\Java\jre-10.0.2` .

Teremos mais uma variável de ambiente que pode nos atrapalhar, `Path` , configurando em diversas pastas de diferentes e que por sua vez define quais executáveis serão visíveis na linha de comando. Uma das pastas é `C:\Program Files\Java\jre-10.0.2\bin` , ela está correta, afinal o executável do Java está armazenado dentro de uma pasta `bin` .

Abriremos novamente o terminal e acessaremos a pasta do Tomcat, acionaremos o arquivo de inicialização `startup.bat` :

```
D:\apache-tomcat-9.0.11-prod\bin>startup.bat
```

COPIAR CÓDIGO

Dessa forma, teremos a versão correta do Java sendo executada. Acessaremos no navegador a nossa lista de empresas via `localhost:8080/gerenciador/listaEmpresas` , e teremos o conteúdo esperado exibido na tela:

```
Lista de empresas:

. Alura - 03/09/2018  edita remove
. Caelum - 03/09/2018  edita remove
```

COPIAR CÓDIGO

Podemos, inclusive, clicar em "edita" e realizar alterações nas empresas que constam na lista, bem como removê-las. Dessa forma, conseguimos executar a aplicação em outro Tomcat, com o ambiente de desenvolvimento na mesma versão que o ambiente de produção.

Nossa aplicação está executando, e fizemos esse processo de maneira manual para entendermos melhor seu funcionamento interno. Em um projeto real, todas essas etapas seriam automatizadas.

Veremos um último item: no console do Tomcat, todas as saídas `System.out` estão visíveis. Se quisermos realizar alguma alteração, precisaremos trabalhar no código fonte, isto é, precisaríamos criar o war novamente. A plataforma de ensino **Alura** opera por meio de um Tomcat instalado na nuvem, mas ela não é executada na porta `8080` , como isso se dá?

Acessaremos o Tomcat de produção, afinal a porta é algo definido pelo servidor. Na pasta `apache-tomcat-9.0.11-prod` teremos a pasta "conf", em que encontraremos o arquivo `server.xml`, que precisa ser editado. É importante utilizarmos um editor previsível, para que nenhuma alteração fora do nosso controle seja realizada, neste caso usaremos o Notepad++.

Com o arquivo `server.xml` aberto no Notepad++, utilizaremos o atalho "Ctrl + F" para buscar a porta "8080", e encontremos o trecho de código correspondente:

```
<Connector port="8080" protocol="HTTP/1.1"
        connectionTimeout="20000"
        redirectPort-"8443" />
```

COPIAR CÓDIGO

Modificaremos a porta `8080` simplesmente para `80`, porque essa é a porta padrão do protocolo http e é esse valor assumido quando não especificamos a porta. Assim feito, iremos reiniciar o Tomcat e acessaremos nosso projeto no navegador digitando simplesmente `localhost/gerenciador`, sem qualquer problema.