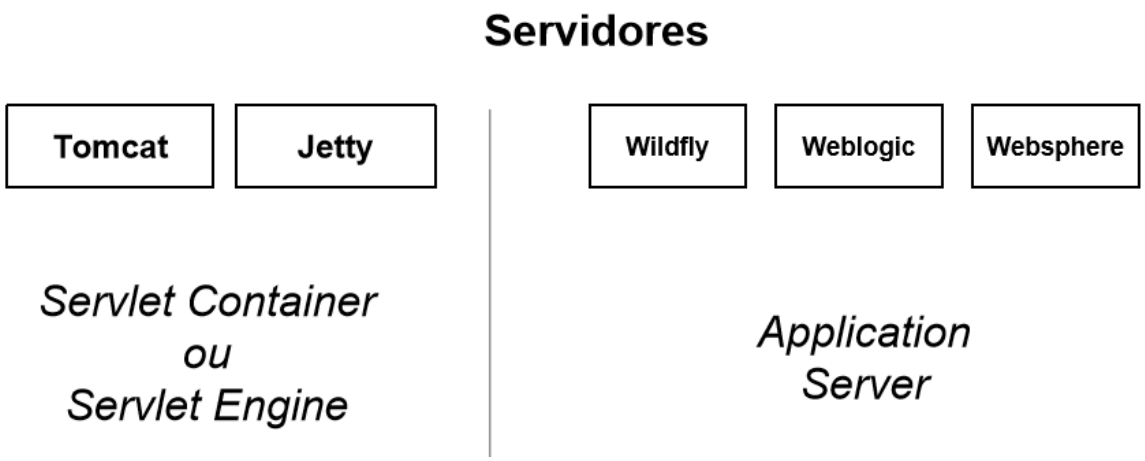
03

Servlet Container Jetty

Transcrição

Olá! Chegamos na última aula desse curso - no caso, mais uma aula sobre **deploy**. O deploy consiste em disponibilizar nossa aplicação dentro do servidor, o que já fizemos com o Tomcat ao final do curso Servlet Parte 1: Fundamentos da programação web Java (https://cursos.alura.com.br/course/servlets-fundamentos-programacao-web-java).

Então por que fazer o deploy novamente? Simples: nós iremos mudar o servidor. Existem vários servidores no mundo Java, e nessa aula iremos apresentar os principais do mercado.



O **Tomcat**, que utilizamos até agora, é um dos mais populares, e inclusive é utilizado por várias aplicações robustas e bem conhecidas, como a própria Alura! O Tomcat é algo chamado **Servlet Container** ou **Servlet Engine**, pois ele faz o mapeamento de uma URL para um Servlet (ou um filtro).

O **Jetty** é a mesma coisa, e daqui a pouco iremos aprender a rodar nossa aplicação dentro do Jetty. Ele não é tão popular quanto o Tomcat, mas também é um servidor muito utilizado.

O Tomcat e o Jetty mapeiam a requisição para a chamada de um objeto, e fazem isso muito bem (com algumas funcionalidades).

Porém, além dos Servlet Containers, existem outros servidores muito maiores, como **Wildfly** (um produto da Red hat, uma famosa empresa de tecnologia), **Weblogic** (da Oracle) e **Websphere** (da IBM), que também podem fazer isso, mas possuem diversos recursos adicionais.

Por exemplo, eles já vêm com bibliotecas embutidas para gerar JSON ou XML, disponibilizar os serviços, trabalhar com banco de dados, oferecer *pool* de conexão, gerenciar transações, etc. - tanto é que o download do Tomcat possui por volta de 16MB, e o Wildfly tem cerca de 80MB.

Enquanto o Weblogic e o Websphere são produtos pagos, o Wildfly é gratuito, e o usuário paga apenas pelo suporte. Como são servidores mais robustos, eles tentam oferecer uma infraestrutura completa para a aplicação, e por isso são chamados de **Application Servers**.

Existem vários outros servidores e variações (como o **Wildfly-swarm**, que é menor que o servidor Wildfly original), mas esses já ocupam a maior parte do mercado. Na área de servidores de aplicações, o Wildfly é o mais famoso.

Para começarmos a usar o Jetty, a primeira coisa que devemos fazer é exportar a nossa aplicação. Como queremos fazer o deploy, precisamos criar um `.war`, que compacta todos os arquivos necessários para rodar nossa aplicação. Porém, antes disso, será necessário fazermos pequenas mudanças nos nossos filtros.

Quando começamos a implementar nossa interface, somente foi necessário implementar um método (`doFilter`), ao invés de três. Ou seja, faltaram dois

`init()`, que recebe `filterConfig`; e `destroy()`. Isso porque nas versões mais recentes do Tomcat e do Java (a partir do Java 8), existe o recurso de oferecer uma implementação padrão, não sendo mais necessário implementar esses outros métodos.

No entanto, o Jetty exige essa implementação. Faremos isso com `@Override` e deixaremos esses métodos vazios, pois realmente não utilizaremos eles:

```
@Override
public void init(FilterConfig filterConfig) throws ServletExce

@Override
public void destroy() {}
```

COPIAR CÓDIGO

Colocaremos esses dois métodos nos filtros `AutorizacaoFilter`, `ControladorFilter` e `MonitoramentoFilter`. Com essas mudanças, nós inclusive garantimos que esse projeto irá rodar em versões mais antigas do Tomcat.

Para exportarmos nossa aplicação, clicaremos com o botão direito em `gerenciador` e em seguida em "Export > Export > Web > WAR file". Na janela seguinte, selecionaremos um local para salvar o `.war`, desmarcaremos a opção "Optimize for a specific server runtime" e clicaremos em "Finish".

O Eclipse irá exportar o nosso `gerenciador.war`, com todas as bibliotecas, classes compiladas, JSPs, etc.

No momento, o Jetty é parte da Eclipse Foundation (ou seja, é um subprojeto do Eclipse). Inclusive, existe um plugin do Jetty para o Eclipse, mas que não testamos ainda. Para facilitar, você pode clicar aqui (https://caelum-online-public.s3.amazonaws.com/1001-servlets-parte2/07/jetty-distribution-9.4.12.v20180830.zip) para fazer o download do Jetty que usaremos nesse cur

Você também pode fazer o download diretamente no site do Jetty (https://www.eclipse.org/jetty/download.html), onde é possível encontrar toda a documentação do servidor e o plugin que comentamos.

Agora precisaremos extrair o Jetty no nosso comportador. Compactado, o arquivo `jetty-distribution-9.4.12.v20180830.zip` possui cerca de 18MB, o que é pouco em comparação com os Application Servers, que são muito maiores.

Repare que, dentro da pasta "jetty-distribution-9.4.12.v20180830" que extraímos, existe uma pasta chamada "webapps" - uma pasta de deploy que também existe no Tomcat. Colocaremos nosso `gerenciador.war` dentro dessa pasta para fazermos o deploy.

Nesse momento, assim como no Tomcat, deveríamos extrair o `gerenciador.war`, começar a ler as configurações `web.xml` e subir nossa aplicação. Porém, ainda temos que subir o Jetty, certo?

O deploy até o momento foi igual ao Tomcat, mas a maneira de subir o servidor é específica de cada um. As instruções para o Jetty podem ser encontradas aqui (https://www.eclipse.org/jetty/documentation/current/quickstart-running-jetty.html).

Nós precisaremos entrar na pasta do Jetty e rodar `java -jar start.jar` na linha de comando, chamando a máquina virtual e passando esse `.jar`.

Através do "Prompt de Comando", vamos acessar a pasta "jetty-distribution-9.4.12.v20180830" (na qual instalamos o jetty). Com o comando `dir`, poderemos ver todo o conteúdo da pasta, incluindo o `start.jar` que precisamos rodar. Faremos isso executando `java -jar start.jar`.

> Dica: Pode ser necessário adicionar o Java às permissões de Firewall do seu computador.

O prompt irá executar o `.jar`, subindo o servidor. Ainda não sabemos para qual diretório o Jetty extraiu nossa aplicação, mas já podemos testá-la acessando http://localhost:8080/ (http://localhost:8080/). O navegador retornará um "Error 404 - Not Found", mas esse erro será mostrado pelo Jetty.

Além disso, o navegador irá mostrar que o Jetty carregou um contexto:

```
/gerenciador ---> o.e.j.w.WebAppContext@7ce69770{gerenciador,/
```

COPIAR CÓDIGO

Ou seja, nosso gerenciador está rodando. Portanto, em teoria, será possível acessar http://localhost:8080/gerenciador/entrada?acao=ListaEmpresas (http://localhost:8080/gerenciador/entrada?acao=ListaEmpresas). Quando fizermos isso, seremos redirecionados para `LoginForm`, e poderemos fazer o login normalmente.

Pronto! Estamos rodando nossa plataforma com outro Servlet Container.

Em teoria, isso também significa que o nosso cliente do WebService também continua funcionando. Para testarmos isso, basta rodarmos o cliente no Eclipse. Nesse caso, tanto a requisição com o cabeçalho JSON quanto com XML funcionarão normalmente. Inclusive, as nossas saídas dos filtros serão mostradas no "Prompt de Comando".

Conseguimos deployar nossa aplicação no Jetty e no Tomcat, e poderíamos até mesmo fazer isso no Wildfly, pois ele também atende o mundo Servlet. Falaremos mais sobre isso no próximo vídeo. Até lá!