05

Inversão de controle

Transcrição

Nas últimas aulas comentamos rapidamente sobre o `web.xml`, e aprendemos como configurar as anotações nesse formato, apesar de ser um processo trabalhoso. Uma vantagem do XML é termos todas as configurações em um único lugar, ao invés de dispersas em várias classes, que é o que acontece quando utilizamos anotações.

Como normalmente o mapeamento não é modificado, as anotações são mais usadas. Além disso, elas são mais fáceis de usar e são compiladas pelo computador.

Agora retomaremos o conteúdo principal do curso: **o que é um Servlet?**

Servlet é um objeto que pode ser acionado por meio de uma requisição do protocolo HTTP. Essa interação é possibilitada pelo Tomcat, e precisa seguir algumas regras, como estender, sobrescrever os métodos `doGet()`, `doPost()` e `service()`, e fazer o mapeamento para indicar a URL.

Como o Servlet é um objeto, devemos criar uma instância da classe `OiMundoServlet.java`. Repare que em nenhum local do projeto temos o método `main()`. Quem participou dos cursos básicos do Java sabe que sempre criávamos esse método e, a partir dele, criávamos a nossa aplicação, instanciando objetos, chamando métodos etc. Veja o exemplo a seguir:

```
public static void main(String[] args) {
    OiMundoServlet servlet = new OiMundoServlet();
```

```
        servlet.service(req, resp);
    }
```

COPIAR CÓDIGO

Não fizemos esse procedimento, mas o objeto da classe `OiMundoServlet` precisa ser criado... e quem faz isso é o Tomcat, nosso ***servlet container***. Isso ocorre porque é o Tomcat que receberá a requisição HTTP, e depois deve chamar o método necessário.

Ou seja, o Tomcat realiza o papel intermediário entre o navegador e objeto, e por isso também é conhecido como *middleware*.

Para provarmos esse ponto, criaremos um construtor sem argumentos em `oiMundoServlet`, e usaremos o `System.out.println()` para imprimir uma mensagem:

```
    public oiMundoServlet() {
        System.out.println("Criando Oi Mundo Servlet");
    }
```

COPIAR CÓDIGO

Para testar, acessaremos o endereço http://localhost:8080/gerenciador/ola (http://localhost:8080/gerenciador/ola) no navegador. Será impressa na tela a mensagem "*oi, parabens vc escreveu o primeiro servlets*". Já no console, receberemos a mensagem "*Criando Oi Mundo Servlet*...

Ou seja, o Tomcat criou o Servlet. Porém, reiniciarmos o servidor, notaremos que ele não criará o Servlet de forma automática, isto é, ele não instanciará todos os Servlets de uma vez. No momento em que alguém chama o mapeamento associado com esse Servlet, ele é criado. O Tomcat só criará o objeto quando ele for completamente necessário.

Para demonstrarmos melhor esse comportamento do Tomcat, na página http://localhost:8080/gerenciador/ola (http://localhost:8080/gerenciador/ola),

repetiremos a requisição diversas vezes pressionando o atalho "F5".

No console apesar das diversas requisições, o Tomcat terá criado apenas uma instância do Servlet, chamando uma única vez o construtor. O objeto sempre fica em memória no Tomcat, e esse objeto é reaproveitado nas próximas requisições.

```
Criando Oi Mundo Servlet

o servlet OiMundoServlet foi chamado

o servlet OiMundoServlet foi chamado

o servlet OiMundoServlet foi chamado

o servlet OiMundoServlet foi chamado

o servlet OiMundoServlet foi chamado
```

No Tomcat teremos apenas um Servlet, `oiMundoServlet` ou `ListaEmpresaServlet` , e isso se eles forem chamados. Por isso o Servlet é chamado de ***Singleton***, um escopo, que sobrevive no projeto por tempo indeterminado enquanto o Tomcat estiver no ar, sem nunca recriá-lo.

Esse processo poderia ser diferente, isto é, o Tomcat poderia recriar um Servlet a cada nova requisição. Existem outras bibliotecas ou frameworks que recriam objetos a cada nova requisição recebida pelo servidor. O escopo é aquilo que determina quanto tempo vive um objeto, e por padrão ele é *Singleton*.

Esse assunto faz parte de um tópico que chamamos de **inversão de controle**, em inglês **IOC** (***-Inversion Of Control***). Isso significa que o método `main()` é quem instancia o objeto, mas no caso do nosso projeto em realiza esse processo é o Tomcat, nós apenas criamos as classes.

O Tomcat trabalha com o protocolo HTTP e nos envia os dados de forma que não precisemos nos preocupar com o *parsing* dos cabeçalhos e assim por diante.

Nós poderíamos desenvolver nosso próprio servidor, mas geralmente não é assim que se ganha dinheiro. A atividade lucrativa neste caso é criar modelos de negócio para empresas, com suas devidas diferenças e especificidades.

O Tomcat poderia ser mais complexo e permitir, por exemplo, a criação e a configuração de um Servlet para cada requisição, ou ainda, que ele pudesse gerenciar transações com o banco de dados, verificar regras de segurança antes de fazer a chamada para o Servlet e assim por diante.

Esses são assuntos para *containers* mais sofisticados. Um deles é o ***SpringMVC***, que possui uma infraestrutura muito maior que o Tomcat. Ele realiza configurações de segurança, cash, escopo, entre outras, e nós apenas criamos as classes.

O mais importante dessa explicação é entendermos o conceito da *inversão de controle*, que é realizada pelo servidor.