Java Web com Servlets e JSPs

Servlets

Softblue
cursos online

1

Tópicos Abordados

- Introdução aos servlets
- Ciclo de vida de um servlet
- Criando servlets
- Requisições GET e POST
- As interfaces HttpServletRequest e HttpServletResponse
- Configurando servlets
- Inicialização automática
- Passando parâmetros para servlets
- Enviando uma resposta ao cliente

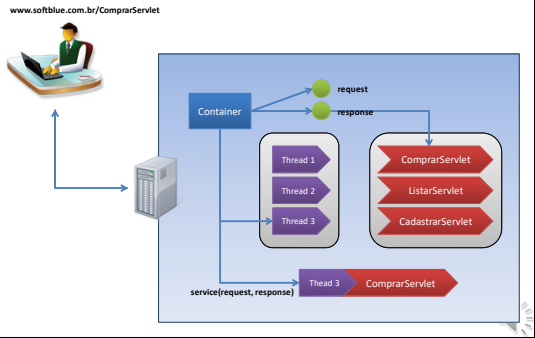
2

Introdução aos Servlets

- Componente Java EE
- Seu objetivo é atender requisições
- Reside no servidor, dentro do web container
- É controlado pelo container

3

Requisição a um Servlet



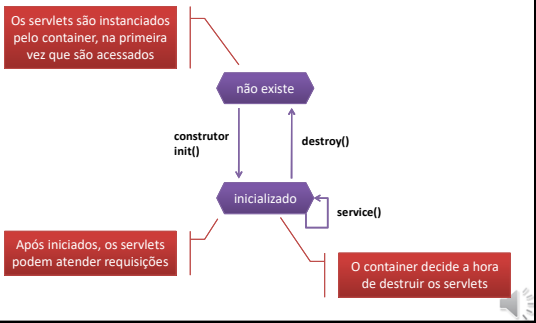
4

Informações Importantes

- O objeto do servlet é único na aplicação
- Para atender requisições simultâneas, são usadas threads
  - Tomar cuidado, pois o mesmo objeto de servlet é compartilhado entre várias threads
- Quando uma requisição ao servlet é feita, ele já foi previamente inicializado

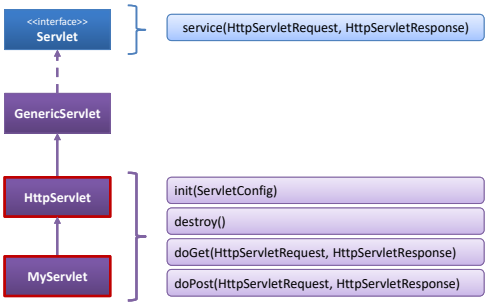
5

Ciclo de Vida de um Servlet



6

Criando Servlets



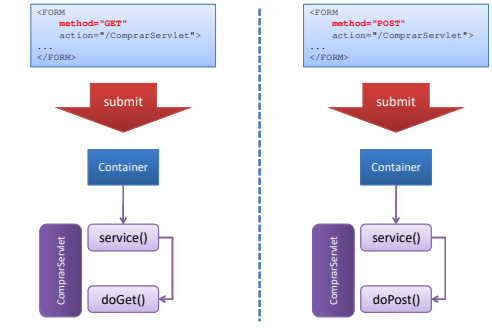
7

Criando Servlets

```
public class MyServlet extends HttpServlet {

  public void init(ServletConfig config) throws ServletException {
    //inicialização do servlet
  }

  public void destroy() {
    //destruição do servlet
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //atendimento de requisição do tipo GET
  }

  public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //atendimento de requisição do tipo POST
  }
}
```

8

Requisições GET e POST



9

Tratando GET e POST Igualmente

Softblue

```
public class MyServlet extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    doIt(request, response);
  }

  public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    doIt(request, response);
  }

  private void doIt(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    //atendimento de requisição GET ou POST
  }
}
```

Basta criar um método e chamá-lo para requisições GET ou POST

10

Outros Tipos de Requisições HTTP

Softblue

- Além de GET e POST, existem outros tipos, mas dificilmente são usados

HTTP Method	Servlet Method
HEAD	doHead()
TRACE	doTrace()
PUT	doPut()
DELETE	doDelete()
OPTIONS	doOptions()
CONNECT	-

11

A Interface HttpServletRequest

Softblue

- Representa a requisição feita pelo usuário
- Alguns métodos importantes
  - Obter dados do HTTP request header
    - getHeaderNames()
    - getHeader(String)
  - Obtém informações submetidas de um formulário HTML (por GET ou por POST)
    - getParameter(String)

12

A Interface HttpServletResponse

- Representa a resposta que será enviada de volta ao cliente
- Alguns métodos importantes
  - Colocar dados no HTTP response header
    - addHeader(String, String)
  - Definir o tipo de resposta
    - setContentType(String)
  - Obter referência ao canal de saída dos dados
    - getWriter()
    - getOutputStream()

13

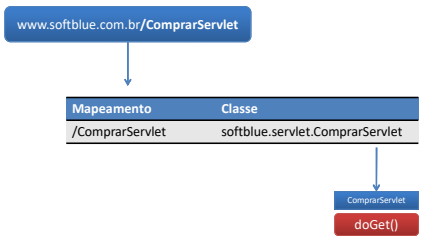
Configurando um Servlet

- O arquivo **web.xml** é utilizado para configurar um servlet

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">

  <servlet>
    <servlet-name>Comprar</servlet-name>
    <servlet-class>softblue.servlet.ComprarServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Comprar</servlet-name>
    <url-pattern>/ComprarServlet</url-pattern>
  </servlet-mapping>

</web-app>
```

14

Como Funciona o Mapeamento



15

Inicialização Automática

- É possível definir quais servlets devem ser inicializados junto com a aplicação
- Ao ser inicializado, o **init()** é chamado

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">

  <servlet>
    <servlet-name>Comprar</servlet-name>
    <servlet-class>softblue.servlet.ComprarServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

</web-app>
```

16

Inicialização Automática

- A tag *load-on-startup* permite ordenar a inicialização dos servlets

```
<servlet>
  <servlet-name>Comprar</servlet-name>
  <servlet-class>softblue.servlet.ComprarServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
  <servlet-name>Listar</servlet-name>
  <servlet-class>softblue.servlet.ListarServlet</servlet-class>
  <load-on-startup>2</load-on-startup>
</servlet>

<servlet>
  <servlet-name>Cadastrar</servlet-name>
  <servlet-class>softblue.servlet.CadastrarServlet</servlet-class>
  <load-on-startup>3</load-on-startup>
</servlet>
```

17

Passando Parâmetros para Servlets

- Ao inicializar os servlets, é possível passar parâmetros a eles através do arquivo *web.xml*

```
<servlet>
  <servlet-name>Comprar</servlet-name>
  <servlet-class>softblue.servlet.ComprarServlet</servlet-class>
  <init-param>
    <param-name>moeda</param-name>
    <param-value>R$</param-value>
  </init-param>
</servlet>
```

- É possível fornecer mais de uma parâmetro usando a tag init-param várias vezes

18

Recuperando os Parâmetros

- A recuperação dos parâmetros pode ser feita no código do servlet

```
ServletConfig config = getServletConfig();
String moeda = config.getInitParameter("moeda");
```

```
String moeda = getInitParameter("moeda");
```

- Caso o init() seja sobrescrito, é preciso invocar o método da superclasse

```
public void init(ServletConfig config)
    throws ServletException {

    super.init(config);
    //...
}
```

19

Recuperando os Parâmetros

- O método **getInitParameterNames()** permite ler todos os parâmetros do servlet

```
ServletConfig config = getServletConfig();
Enumeration<String> e = config.getInitParameterNames();

while(e.hasMoreElements()) {
    String param = e.nextElement();
}
```

20

Servlets e Annotations

- Servlets podem ser configurados também via annotations

21

Servlets e Annotations

- O atributo **loadOnStartup** define a inicialização automática

ComprarServlet

```
@WebServlet(value = "/Comprar", loadOnStartup = 1)
public class ComprarServlet extends HttpServlet {
    //...
}
```

22

Servlets e Annotations

- O atributo **initParams** permite definir parâmetros de inicialização para o servlet
- Ele define um array de parâmetros, onde cada elemento é do tipo **@WebInitParam**

ComprarServlet

```
@WebServlet(value = "/ComprarServlet", initParams = {
    @WebInitParam(name = "moeda", value = "R$"),
    @WebInitParam(name = "pais", value = "Brasil")
})
public class ComprarServlet extends HttpServlet {
    //...
}
```

23

Enviando uma Resposta ao Cliente

- Depois que uma requisição é feita, o cliente (browser) aguarda uma resposta
- Depois de processar a requisição, uma das opções é que o servlet gere a resposta
- Como deve ser a resposta?
  - Normalmente é em formato HTML
  - Pode ser também qualquer outro tipo de formato, seja ele binário ou texto

24

Produzindo uma Resposta

Softblue

- O servlet acessa o canal de envio da resposta através do objeto **HttpServletResponse**

Resposta em formato texto

```
PrintWriter out = response.getWriter();
```

out.print() é usado para enviar dados

Resposta em formato binário

```
OutputStream out = response.getOutputStream();
```

out.write() é usado para enviar dados

25

Definindo um Content-Type

Softblue

- É preciso avisar ao cliente (browser) sobre o tipo de resposta que está sendo enviado
- Isto é feito através do content-type, definido no HTTP response header

```
response.setContentType("text/html");
```

Define que a resposta será no formato HTML

26

Tipos de Content-Type

Softblue

Content Type	Extensão Típica	Formato
text/html	.htm .html	HTML
text/plain	.txt	Texto sem formatação
image/gif	.gif	Imagem GIF
image/jpeg	.jpg	Imagem JPEG
image/x-png	.png	Imagem PNG
application/pdf	.pdf	Documento PDF
application/zip	.zip	Conteúdo compactado no formato ZIP

formato texto
formato binário

27

Exemplo de Reposta HTML

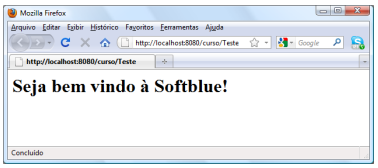
Softblue

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    response.setContentType("text/html");

    PrintWriter out = null;

    try {
        out = response.getWriter();
        out.print("<HTML><BODY>");
        out.print("<HEAD>");
        out.print("<META http-equiv=\"Content-Type\" " +
            "content=\"text/html; charset=ISO-8859-1\" />");
        out.print("</HEAD>");
        out.print("<H1>Seja bem vindo à Softblue!</H1>");
        out.print("</BODY></HTML>");
    } finally {
        if (out != null) {
            out.close();
        }
    }
}
```

28

Exemplo de Resposta HTML

Softblue



29

Exemplo de Resposta Binária

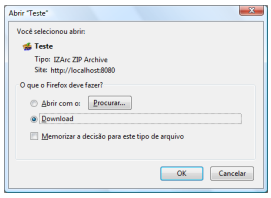
Softblue

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    response.setContentType("application/zip");
    OutputStream out = null;
    InputStream in = null;

    try {
        out = response.getOutputStream();
        in = new FileInputStream("D:/Temp/java.zip");

        byte[] buffer = new byte[1024];
        int numBytes;
        while((numBytes = in.read(buffer, 0, buffer.length)) > -1) {
            out.write(buffer, 0, numBytes);
        }
    } finally {
        if (in != null) {
            in.close();
        }
        if (out != null) {
            out.close();
        }
    }
}
```

30

Exemplo de Resposta Binária

Softblue



31