Programação para Web

Servlets

Ivo Calado

Instituto Federal de Educação, Ciência e Tecnologia de Alagoas

22 de Fevereiro de 2016

Roteiro

Observação sobre o conteúdo

Parte deste material é baseada na apostila FJ-21 - Java para Desenvolvimento Web desenvolvido pela Caelum (`www.caelum.com.br`)

O que são?

- As Servlets são a primeira forma que veremos de criar páginas dinâmicas com Java
- Criaremos classes que terão a capacidade de gerar conteúdo HTML!!
- O nome "servlet" vem da ideia de um pequeno servidor (servidorzinho, em inglês) cujo objetivo é receber chamadas HTTP, processá-las e devolver uma resposta ao cliente

A classe **javax.servlet.http.HttpServlet**

- Trata-se da classe principal de uma servlet
- Possui um método genérico (**service**) e diversos métodos como específicos como **doGet, doPost, doDelete, doPost e outros** representando as diversas operações HTTP

A classe **javax.servlet.http.HttpServlet**

- Trata-se da classe principal de uma servlet
- Possui um método genérico (**service**) e diversos métodos como específicos como **doGet, doPost, doDelete, doPost e outros** representando as diversas operações HTTP
- O nosso papel é implementá-las!
- Cada um desses métodos recebem como parâmetro um objeto **HttpServletRequest** e um **HttpServletResponse** representando a requisição e a resposta!

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

O método doGet

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response)
      throws ServletException, IOException {
         ServletOutputStream out = response.
             getOutputStream();
         out.println("<html>");
         out.println("<head>");
         out.println("<title>Minha primeira pagina
             dinamica</title>");
         out.println("</head>");
         out.println("<body>");
         out.println("<h1>Hello World!!!</h1>");
         out.println("</body>");
         out.println("</html>");
}
```

Habilitando a execução da Servlet

- Acabamos de definir uma Servlet, mas como iremos acessá-la pelo navegador?
- Qual o endereço podemos acessar para fazermos com que ela execute?

Habilitando a execução da Servlet

- Acabamos de definir uma Servlet, mas como iremos acessá-la pelo navegador?
- Qual o endereço podemos acessar para fazermos com que ela execute?
- O container não tem como saber essas informações, a não ser que digamos isso para ele
- Para isso, vamos fazer um mapeamento de uma URL específica para uma servlet através do arquivo **web.xml**, que fica dentro do **WEB-INF**

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Web.xml

- No arquivo **web.xml** configuramos todas as propriedades do nosso projeto Java
- A partir da versão 3.0 todas as configurações podem ser realizadas via **annotations**! Porém veremos isso só mais na frente!

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ...>
  <display-name>MeuPrimeiroProjeto3.0</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

web.xml: tags **servlet** e **servlet-mapping**

- Para configurarmos a execução da Servlet criada pelo Container fazemos uso da tag *servlet*
- A tag *servlet-mapping* serve para criar um alias ou um conjunto de aliases para referenciar a servlet

web.xml: exemplos de servlet e servlet-mapping

```
<servlet>
  <servlet-name>minhaPrimeiraServlet</servlet-name>
  <servlet-class>br.edu.ifal.MinhaServlet</servlet-
      class>
</servlet>

<servlet-mapping>
  <servlet-name>minhaPrimeiraServlet</servlet-name>
  <url-pattern>/minha-pagina</url-pattern>
  <url-pattern>/minha-segunda-forma-de-acesso</url-
      pattern>
  <url-pattern>/end/longo/para/minha/pagina/web</url-
      pattern>
</servlet-mapping>
```

A estrutura de diretórios

- Se vocês notarem, o nosso endereço não tem qualquer relação com a estrutura física do projeto
- Trata-se de um referenciamento puramente virtual!

Exercício

- Crie sua primeira servlet. Ela deve implementar os métodos doGet e doPost onde no doGet deve ser retornado uma página com uma mensagem ao usuário e no doPost com a data do dia.
- Deve ser criado uma página HTML estática que faça acesso tanto via GET quanto via POST

Possíveis erros comuns até agora

- Esquecer da barra inicial no URL pattern
- Digitar errado o nome do pacote da sua servlet
- Esquecer de colocar o nome da classe no mapeamento da servlet

Recuperando parâmetros passados

- Até agora vimos como requisitar uma página sem passar qualquer parâmetros
- Para recuperar os parâmetros passados em um formulário usamos os métodos **getParameter**, getParameterMap, getParameterNames, getParameterValues

```
Map<String, String[]> mapa = request.getParameterMap();
PrintWriter out = response.getWriter();
for(String s : mapa.keySet()) {
        out.println("Chave: " + s);
        for(String str : mapa.get(s)) {
                out.println("Valor: " + str);
        }
        out.println();
```

Atividade

- Altere o exemplo anterior para inclur um formulário de inscrição de uma pessoa. No lado servidor, recupere os valores passados e salve em um arquivo XML formando uma agenda de contatos.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Tratando exceções dentro da Servlet

- O que acontece se alguma exceção for lançada na nossa aplicação?

Tratando exceções dentro da Servlet

- O que acontece se alguma exceção for lançada na nossa aplicação?
- a stacktrace da exceção ocorrida será mostrada em uma tela padrão do container
- Qual o problema disso?

Tratando exceções dentro da Servlet

- O que acontece se alguma exceção for lançada na nossa aplicação?
- a stacktrace da exceção ocorrida será mostrada em uma tela padrão do container
- Qual o problema disso?
- O que seria ideal acontecer?

Tratando exceções dentro da Servlet

- O que acontece se alguma exceção for lançada na nossa aplicação?
- a stacktrace da exceção ocorrida será mostrada em uma tela padrão do container
- Qual o problema disso?
- O que seria ideal acontecer? O ideal seria mostrarmos uma página de erro dizendo: "Um erro ocorreu" e com informações de como notificar o administrador
- Para fazermos isso, basta configurarmos nossa aplicação dizendo que, caso aconteça uma Exception no *web.xml*!

Configurando exceções no Web.xml

```
<error-page>
   <exception-type>java.lang.Exception</exception-type>
   <location>/erro.html</location>
</error-page>
```

Tratando códigos de erros

- O que acontece se tentarmos acessar uma página inexistente?

Tratando códigos de erros

- O que acontece se tentarmos acessar uma página inexistente?
- Qual seria a abordagem mais apropriada?

```
<error-page>
   <error-code>404</error-code>
   <location>/404.html</location>
</error-page>
```

Enviando mensagem de erro

Pesquisar sobre o processo de envio de mensagens de erro e como isso se relaciona com ponto anterior! Ver método **response.sendError**

Métodos Init e Destroy

- Toda servlet deve possuir um construtor *default*
- Métodos init e destroy são chamados no início e no fim do ciclo de vida da servlet

Métodos Init e Destroy

- Toda servlet deve possuir um construtor *default*
- Métodos init e destroy são chamados no início e no fim do ciclo de vida da servlet
- Quais seriam um possível uso de tais métodos?

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Recuperando parametros iniciais do servlet

- É possível definir parâmetros iniciais na servlet no web.xml e recuperá-lo no método init

Recuperando parametros iniciais do servlet

- É possível definir parâmetros iniciais na servlet no web.xml e recuperá-lo no método init

```
<servlet>
        <description></description>
        <display-name>Servlet1</display-name>
        <servlet-name>Servlet1</servlet-name>
        <servlet-class>Servlet1</servlet-class>
        <init-param>
            <param-name>Nome</param-name>
            <param-value>Valor</param-value
                >
        </init-param>
</servlet>
```

Redirecionando páginas

Existem duas formas de realizar redirecionamento

- sendRedirect: retorna um código ao browser indicando que ele deve realizar o redirecionamento
- forward: realiza o redirecionamento internamente

Além disso, existe a possibilidade de utilizar o método **include**!!.

sendRedirect e forward

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response)
     throws ServletException, IOException {
        response.sendRedirect("http://www.google.com");
}
```

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response)
throws ServletException, IOException {
        RequestDispatcher dispatcher = request.
            getRequestDispatcher("s2");
        dispatcher.forward(request, response);
}
```

include

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response)
throws ServletException, IOException {
        RequestDispatcher dispatcher = request.
            getRequestDispatcher("s2");
        dispatcher.include(request, response);
}
```

Qual a diferença entre forward e include?

include

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response)
throws ServletException, IOException {
        RequestDispatcher dispatcher = request.
            getRequestDispatcher("s2");
        dispatcher.include(request, response);
}
```

Qual a diferença entre forward e include?

forward: a servlet atual não pode alterar a stream de saída

include: a servlet atual pode alterar (a página redirecionada é "inclusa" na resposta original)

include

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response)
throws ServletException, IOException {
        RequestDispatcher dispatcher = request.
            getRequestDispatcher("s2");
        dispatcher.include(request, response);
}
```

Qual a diferença entre forward e include?

forward: a servlet atual não pode alterar a stream de saída

include: a servlet atual pode alterar (a página redirecionada é "inclusa" na resposta original)

Como inserir conteúdos padrões estáticos?

O que é?

- Uma das características fundamentais do HTTP é não possuir estado. Mas o que viria a ser isso?

O que é?

- Uma das características fundamentais do HTTP é não possuir estado. Mas o que viria a ser isso?
- Isso que dizer que o HTTP não guarda lembrança de requisições anteriores
- Qual o problema disso? Qual seria uma situação onde guardar informações de sessões é importante?
- **Login!**
- Sendo assim, precisamos, implementar em nível de servlet, o controle de sessão!

Quais abordagens existe?

Existem basicamente 4 formas de prover sessão

- Reescrita de URL
- Campos ocultos
- Cookies
- Objetos de sessão

1ª abordagem: Reescrita de URL

- Consiste em adicionar às URLs de envio, identificadores da sessão a ser acessada
- Todos os forms, links etc que façam menção a URL deve ser alterado
- Parâmetros são enviados na URL

2ª abordagem: Campos ocultos

- Semelhante a abordagem anterior, porém o campo é enviado como um atributo *hidden* no HTML
- Não são visíveis na URL mas são facilmente recuperáveis!

3ª abordagem: Cookies

- Cookies são pequenos "pedaços" de informação passado em requisições e respostas HTTP
- Embora seja possível criar um Cookie do lado cliente, ele é gerado principalmente pelo servidor
- Tem a estrutura de chave valor!

```
Cookie c1 = new Cookie("MeuCookie", "MeuValor");
response.addCookie(c1);
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Definido tempo máximo de vida de um cookie

- Nas abordagens anterioes os links e campos ocultos permaneciam disponíveis indefinidamente
- Porém podemos querer definir um tempo limite para que a sessão esteja disponível por um tempo limite de inatividade!
- Cookie possibilita tal recurso a partir da definição da propriedade maxAge

```
Cookie c1 = new Cookie("MeuCookie", "MeuValor");
c1.setMaxAge(1000); // Número de segundos que o cookie
    permanecerá no cliente. Após isso ele é descartado
    e não será mais enviado
response.addCookie(c1);
```

HTTPSession

As abordagens anteriores possuem séries restrições:

- Atributos ocultos e reescrita de url deixam visíveis todos os campos
- Apesar de mais difícei, os cookies tb podem ser visualizados!
- Além disso, temos de ter o trabalho de em todas as solicitações adicionar as informações a serem enviadas de modo a manter o estado

A solução é o HTTPSession!

- Trata-se da maneira mais simples e fácil de se trabalhar com gerenciamento de sessão
- É mantido um mapa com todos os valores que a aplicação precisar armazenar
- Possibilita o armazenamento de valores de tipos complexos (não apenas String)

Exemplo

```
HTTPSession session = request.getSession(true);
session.setAttribute("chave", new String(valor))
```

Exemplo

```
HTTPSession session = request.getSession(true);
session.setAttribute("chave", new String(valor))
```

Mas e se os cookies estiverem desabilitados?

Exemplo

```
HTTPSession session = request.getSession(true);
session.setAttribute("chave", new String(valor))
```

Mas e se os cookies estiverem desabilitados? a sessão cairá no mesmo problema clássico dos cookies. A solução é enviar o id de sessão por reescrita de url.
O objeto session gera um parâmetro jsessionid com o número de sessão.

Filtros

- Suponha que tenhamos nossa aplicação tenha 100 servlets e desejamos adicionar verificação de autenticação em todos eles. Teremos de verificar o id em todos os servlets?