

Java Enterprise Edition - JEE

05. Servlets com Annotations



Esp. Márcio Palheta
gtalk: marcio.palheta@gmail.com



Agenda

- O que são servlets
- Pacote javax.servlet
- Ciclo de vida
- Mapeamento
- Exercícios



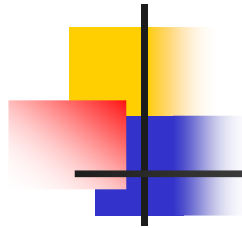
Páginas dinâmicas

- No princípio, páginas estáticas – HTML;
- Páginas HTML geradas dinamicamente;
- Explosão do conteúdo dinâmico;
 - CGI, PHP, ASP e companhia;
- A primeira tecnologia JAVA a gerar páginas dinâmicas: **Servlets**;



Servlets

- Classes JAVA que podem ser acessadas a partir da WEB;
- Têm a capacidade de gerar HTML;
- Recebem **requisições HTTP**, executam um determinado processamento e devolvem a **resposta** ao cliente;



Servlets - conceitos

- Uma **servlet** funciona como um pequeno servidor que recebe chamadas de diversos clientes.
- Recebe requisições (**request**) e retorna algo (**response**), como por exemplo uma página html ou uma imagem do formato jpg.



O pacote

- `javax.servlet.HttpServlet`
- a mesma instância de uma servlet (o mesmo objeto) pode ser chamada mais de uma vez para diferentes requisições ao mesmo tempo;
- Uma página JSP compilada gera uma Servlet;



Ciclo de vida

- **sua inicialização** – momento da criação do objeto;
- **chamadas a métodos de serviço**, essas chamadas passam dois argumentos para o método **service**, a requisição que o cliente faz e a resposta que permite enviar dados para o mesmo:
 - `void service(HttpServletRequest req, HttpServletResponse res);`
- **finalização** – destruição do objeto



Ciclo de vida

```
@Override
public void init(ServletConfig config) throws ServletException {
    super.init(config);
    System.out.println("Início da servlet");
}

@Override
public void service(ServletRequest request,
    ServletResponse response)
    throws ServletException, IOException {
    System.out.println("Método de requisições da servlet");
}


@Override
public void destroy() {
    System.out.println("Fim da servlet");
}
```




Exercícios

- Criação de um novo projeto web chamado **servleittest**;
- Publique o novo projeto no **Tomcat**
- Criação da servlet chamada **MinhaServlet**;
- **Mapeamento** da nova servlet;
- Criação do método **service()**;
- Teste a aplicação gerada;

Novo projeto web



Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: 1

Project location

☒ Use default location

Location: Browse...

Target runtime

2 New Runtime...

Dynamic web module version

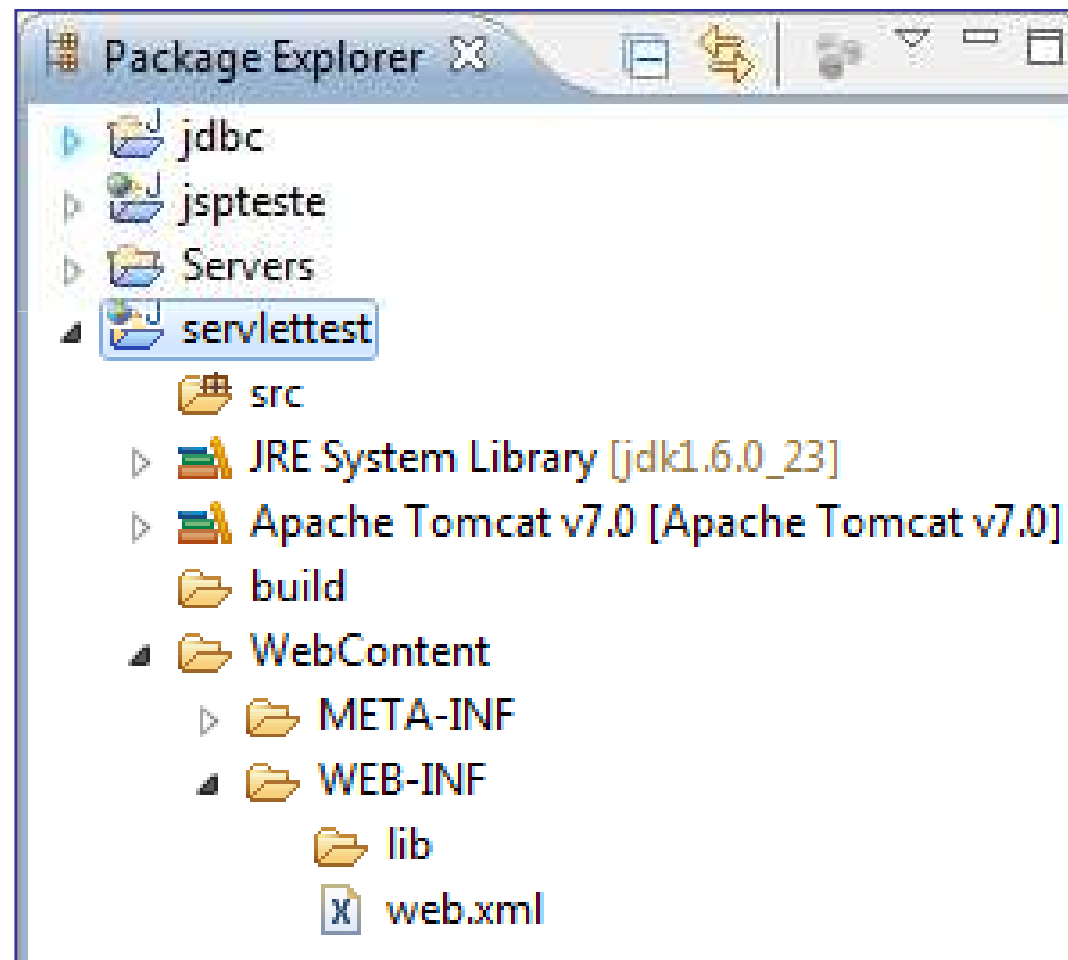
3

Configuration

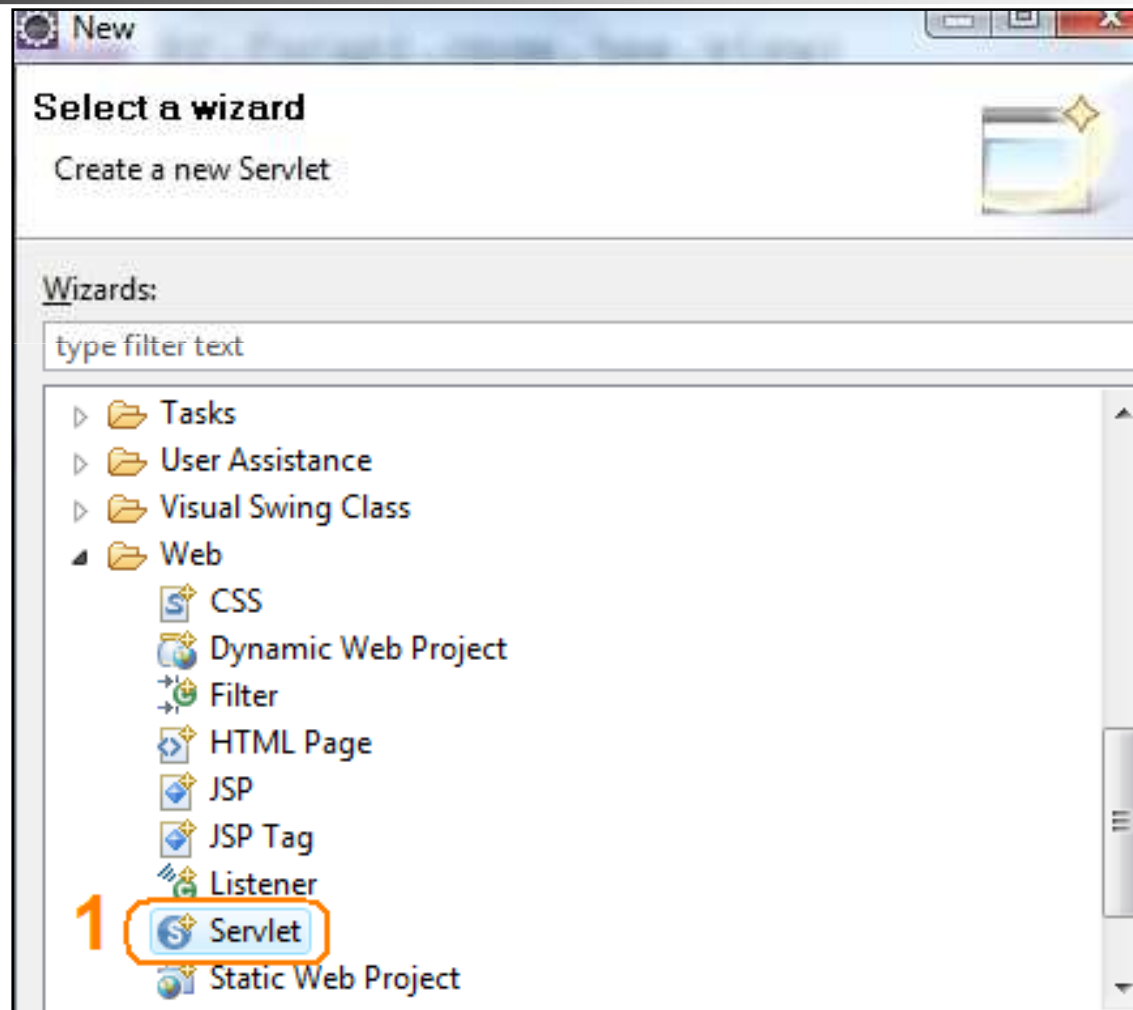
Modify...

? < Back Next > **Finish** 4 Cancel

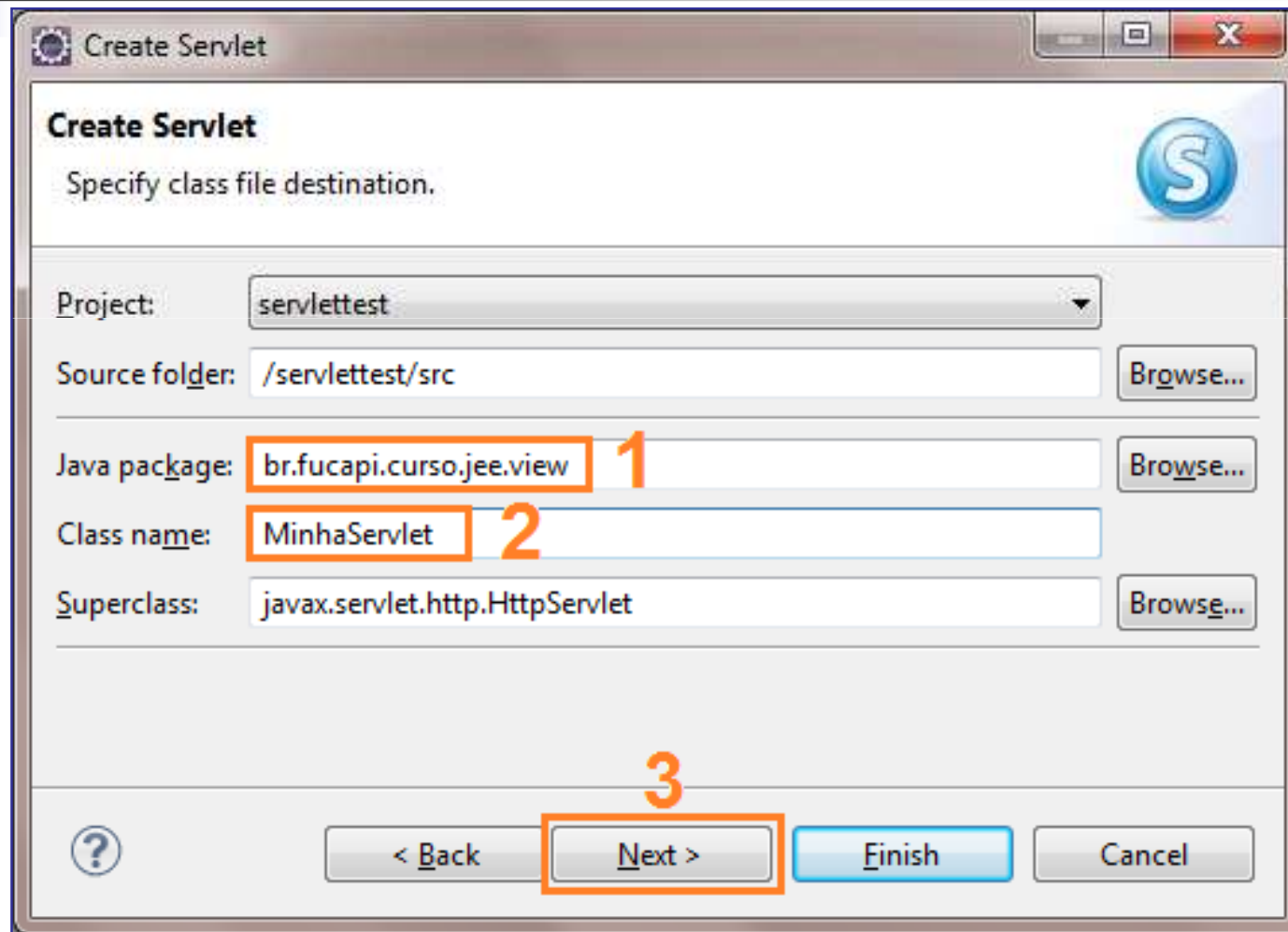
Estrutura criada



Minha primeira Servlet: New/Other/Web/Servlet



Pacote e nomenclatura



The image shows a 'Create Servlet' dialog box from an IDE. It contains several input fields and buttons. Three orange boxes with numbers 1, 2, and 3 highlight specific elements: box 1 highlights the 'Java package' field containing 'br.fucapi.curso.jee.view'; box 2 highlights the 'Class name' field containing 'MinhaServlet'; and box 3 highlights the 'Next >' button at the bottom.

Create Servlet
Specify class file destination.

Project: servlettest

Source folder: /servlettest/src [Browse...](#)

Java package: br.fucapi.curso.jee.view **1** [Browse...](#)

Class name: MinhaServlet **2**

Superclass: javax.servlet.http.HttpServlet [Browse...](#)

[?](#) [< Back](#) **3** [Next >](#) [Finish](#) [Cancel](#)

Parâmetros e mapeamentos

Create Servlet

Enter servlet deployment descriptor specific information.

Name: 1

Initialization parameters:

Name	Value	Description

Add...
Edit...

URL mappings:

2

Add...
Edit...

3

? < Back **Next >** Finish Cancel

Definição de métodos

Create Servlet
Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: ☒ public ☐ abstract ☐ final

Interfaces:

Which method stubs would you like to create?

☐ Constructors from superclass

☒ Inherited abstract methods **1**

<input type="checkbox"/> init	<input type="checkbox"/> destroy	<input type="checkbox"/> getServletConfig
<input type="checkbox"/> getServletInfo	<input checked="" type="checkbox"/> service 2	<input type="checkbox"/> doGet
<input type="checkbox"/> doPost	<input type="checkbox"/> doPut	<input type="checkbox"/> doDelete
<input type="checkbox"/> doHead	<input type="checkbox"/> doOptions	<input type="checkbox"/> doTrace

3



MinhaServlet.java

```
package br.fucapi.curso.jee.view;
import java.io.IOException;
//Mapeamento da classe MinhaServlet para "/Teste" 1
@WebServlet("/Teste")
public class MinhaServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    2 protected void service(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        3 // Recupera o objeto que escreve a resposta
        PrintWriter out = response.getWriter();

        4 //Codigo HTML a ser gerado
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Minha primeira servlet</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```


Mapeamento da servlet, antes da versão 3.0

- Se não usar a annotation `@WebServlet` para mapear sua Servlet, inclua as seguintes tags `<servlet>` e `<servlet-mapping>` em `web.xml`:

```
<servlet>
  <servlet-name>Minha Primeira Servlet</servlet-name>
  <servlet-class>br.fucapi.curso.jee.view.MinhaServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Minha Primeira Servlet</servlet-name>
  <url-pattern>/TesteServlet</url-pattern>
</servlet-mapping>
</web-app>
```

Teste da servlet

- <http://localhost:8080/servlettest/Teste>

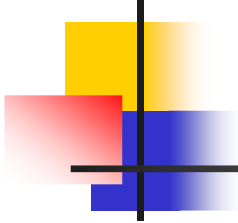




Opções de código:

```
<servlet>
  <description>Meu primeiro servlet</description>
  <servlet-name>MeuServlet</servlet-name>
  <servlet-class>br.fucapi.cpge.jee.view.MeuServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>MeuServlet</servlet-name>
  <url-pattern>/Teste</url-pattern>
</servlet-mapping>
```



Envio de Parâmetros na requisição - <form>

- Na web, a interação com o usuário é, na maioria das vezes, baseada no uso de formulários;
- O usuário preenche os campos de um formulário e os envia ao servidor como parâmetros de uma requisição;
- O servidor recupera os parâmetros e executa seu processamento;

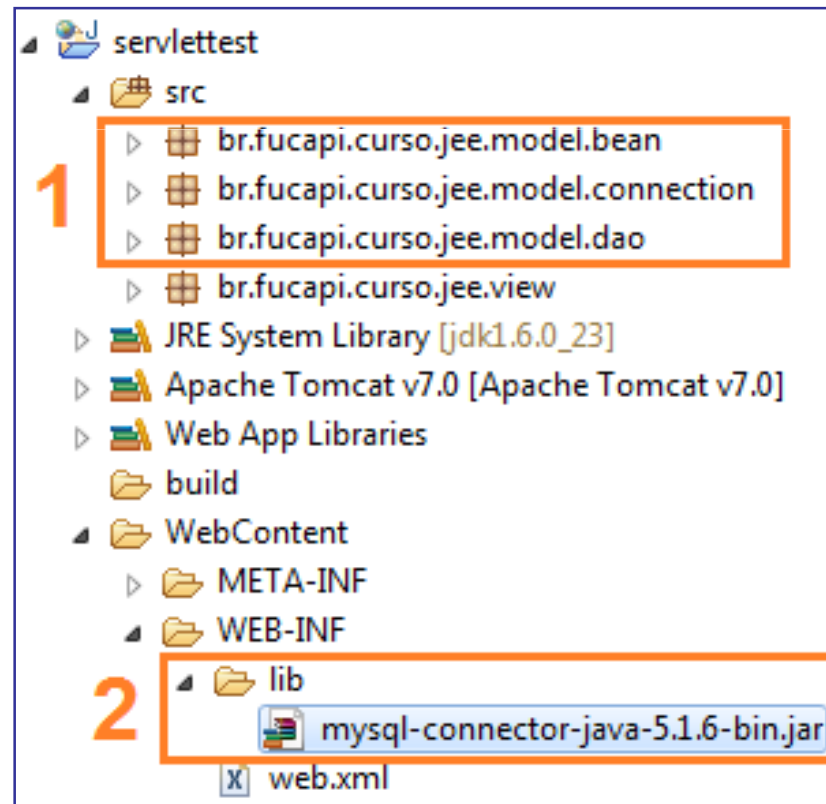


Exercícios

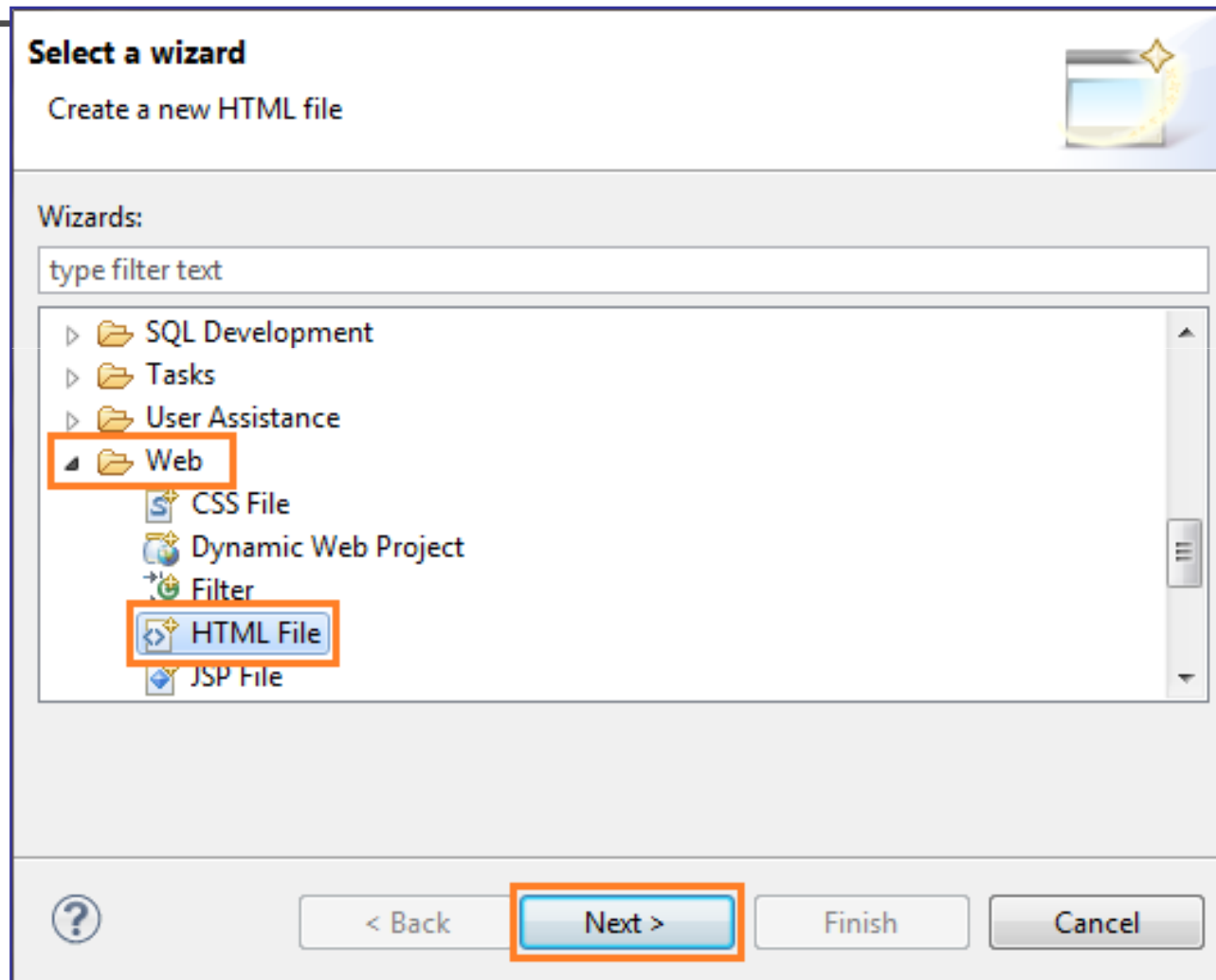
- Copie os pacotes de **persistência** criados no capítulo de JDBC, para a pasta **src** do projeto **servlettest**;
- Crie uma página HTML para cadastro de empresas: **WebContent/cadastro.html**;
- Cria uma nova servlet:
 - CadastrarEmpresaServlet.java;
- Teste o cadastro de empresas

Camada de modelo

- Importe as classes de modelo e o **driver jdbc** de acesso ao MySQL:



Nova página HTML – Ctrl+N



Definição do nome da página

HTML
Create a new HTML file.

Enter or select the parent folder:
servlettest/WebContent

Servers
servlettest
 .settings
 build
 src
 WebContent

File name: cadastro.html

Advanced >>

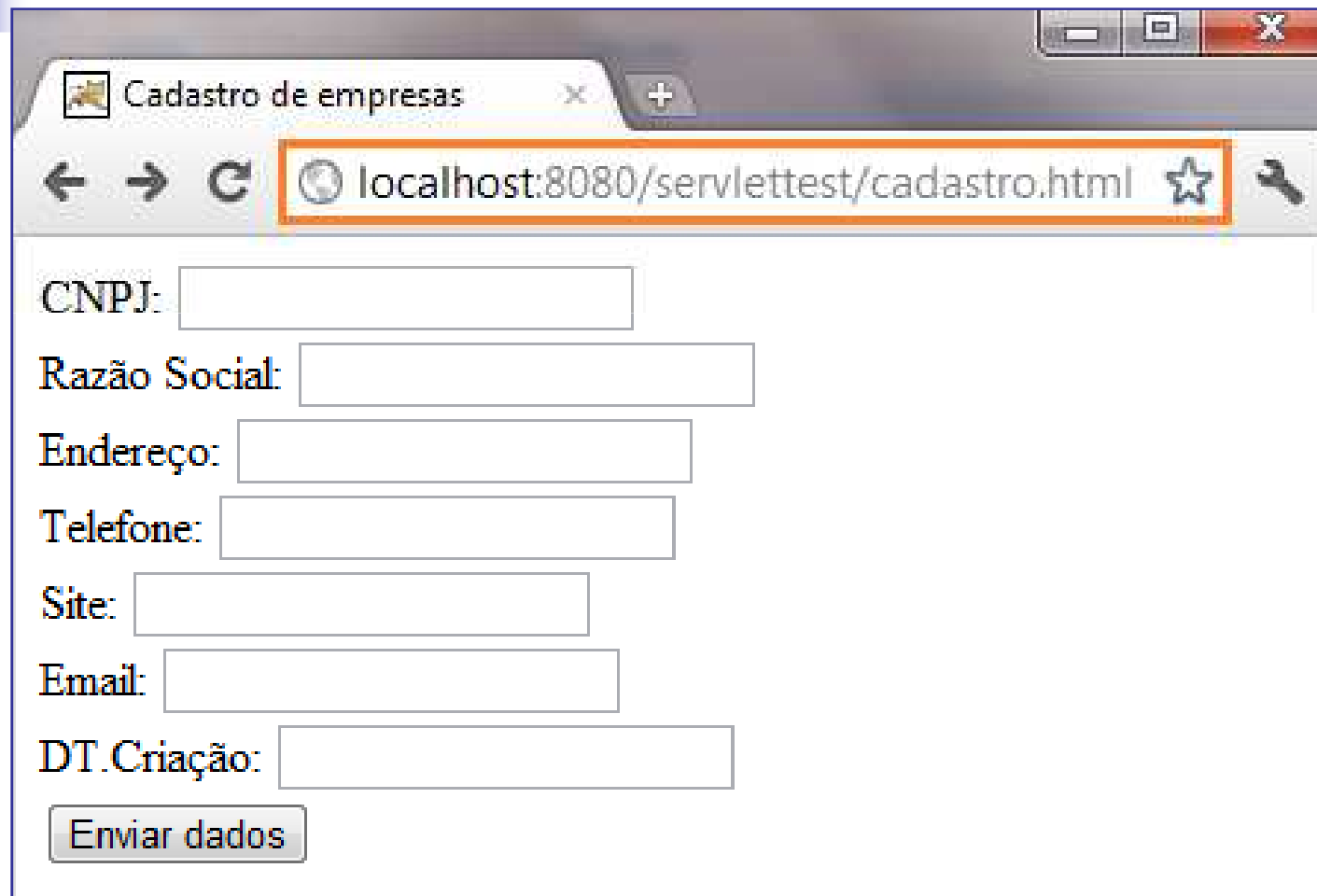
< Back Next > **Finish** Cancel



Página cadastro.html

```
<html>
<head><title>Cadastro de empresas</title></head>
<body>
1 <form action="adicionarEmpresa">
    CNPJ: <input type="text" name="cnpj"/><br>
    Razão Social: <input type="text" name="razaoSocial"/><br>
    Endereço: <input type="text" name="endereco"/><br>
    Telefone: <input type="text" name="telefone"/><br>
    Site: <input type="text" name="site"/><br>
    Email: <input type="text" name="email"/><br>
    DT.Criação: <input type="text" name="dataCriacao"/><br>
2 <input type="button" value="Enviar dados"/>
</form>
</body>
</html>
```

Pagina renderizada



A screenshot of a web browser window displaying a company registration form. The browser's address bar shows the URL `localhost:8080/servlettest/cadastro.html`, which is highlighted with an orange border. The form contains several input fields for company information, followed by a submit button.

Cadastro de empresas

← → ↻ `localhost:8080/servlettest/cadastro.html` ☆ 🔧

CNPJ:

Razão Social:

Endereço:

Telefone:

Site:

Email:

DT.Criação:



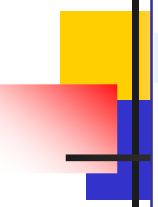
Servlet p/ adicionar empresa

```
package br.fucapi.curso.jee.view;
import java.io.IOException;
@WebServlet("/adicionarEmpresa")
public class AdicionarEmpresaServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void service(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        PrintWriter out = response.getWriter();
        //Carga do onjeto empresa, a partir dos campos da requisicao
        Empresa empresa = new Empresa();
        empresa.setCnpj(request.getParameter("cnpj"));
        empresa.setRazaoSocial(request.getParameter("razaoSocial"));
        empresa.setEndereco(request.getParameter("endereco"));
        empresa.setTelefone(request.getParameter("telefone"));
        empresa.setSite(request.getParameter("site"));
        empresa.setEmail(request.getParameter("email"));

        //Continuacao...
```

Continuação...



```
//Continuacao...
//Conversao da data, de String para Calendar
String dataEmTexto = request.getParameter("dataCriacao");
try {
    Date date =
        new SimpleDateFormat("dd/MM/yyyy").parse(dataEmTexto);
    empresa.setDataCriacao(Calendar.getInstance());
    empresa.getDataCriacao().setTime(date);
} catch (ParseException e) {
    out.println("Erro de conversão da data");
    return;
}
//Salva os dados da empresa
EmpresaDAO dao = new EmpresaDAO();
dao.cadastrar(empresa);
//Imprime os dados da empresa cadastrada
out.println("<html>");
out.println("<body>");
out.println("Empresa <h1>" + empresa.getRazaoSocial() +
    "</h1> adicionada com sucesso");
out.println("</body>");
out.println("</html>");
}
}
```



Atividades

- Reinicie o servidor;
- Acesse a página de cadastro;
- Informe os dados da empresa e clique em enviar;
- Verifique a mensagem de sucesso;
- Crie um botão para retornar à tela anterior;



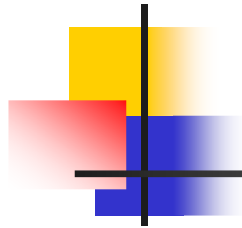
Métodos doGet() e doPost()

- O método **service()** é o ponto de partida e redireciona a requisição para o método adequado: **GET** ou **POST**;
- As implementações de doGet() e doPost() **não** são obrigatórias quando sua classe estende de **HttpServlet**;



Exercício 01

- Crie uma servlet chamada NovaServlet;
- Crie o método service(), conforme slide 08;
- Altere o código de service() para que seja impresso na tela o parâmetro “matricula” da variável request;
- Faça o mapeamento da NovaServlet no arquivo web.xml;
- Realize testes de chamada à NovaServlet



Tratamento de parâmetros

- Toda requisição pode possuir um conjunto de parâmetros;
- No método GET é comum ter uma URL que termine com “?parametro=valor”
- no método POST podemos enviar todos os parâmetros através de um formulário ou simplesmente escondidos da URL.



Parâmetros

- Independente do método chamado, os valores dos parâmetros podem ser lidos com o seguinte código:
- `request.getParameter("nome_parametro");`
- O comando acima devolve um objeto do tipo `String`;
- Ex: `algumaCoisa.jsp?matricula=001`
- `request.getParameter("matricula");`



Exercício 02

- Crie uma página chamada testeGet.jsp;
- Crie um link para acessar a Servlet criada no exercício 01;
- No link criado, informe o valor do parâmetro matricula;
- Teste a execução da página criada;



Exercício 03

- Crie uma página chamada testePost.jsp;
- Crie um formulário para que o usuário informe o valor da matrícula;
- O formulário deve enviar os dados para a servlet criada no exercício 01;
- Teste a execução da página criada;



Exercício 04

- Crie uma servlet que conte a quantidade de requisições de acesso que recebe;
- Sempre que solicitada, a servlet deve incrementar o contador de acesso e exibir a quantidade atual;



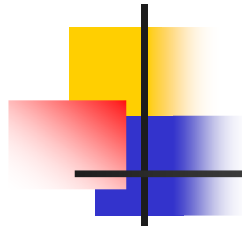
Exercício 05

- Crie uma servlet que recebe como parâmetro os dados de uma empresa e envia os dados para o banco de dados usando a classe EmpresaDAO;



Exercício 06 - persistência

- Crie uma servlet para listar todos os registros de empresas armazenados no banco de dados, a partir da classe EmpresaDAO;



Código HTML na Servlet

- Foi fácil o uso de código HTML na servlet, usando `out.println()`?
- Como será o trabalho de designer, caso seja necessária mudança no layout?
- O código html passa a atrapalhar o código java;
- Como resolver?



Uso de páginas JSP

- O web container interpreta o arquivo JSP, o compila e transforma em uma servlet!
- Quando da 1ª chamada ao arquivo JSP, é criada uma servlet correspondente.
- Não colocamos código HTML em uma classe JAVA;
- Facilita a manutenção do designer;



Referências

- www.caelum.com.br
- Hall, Marty, "Core Servlets and Java Server Pages", Janeiro 2002, Sun Microsystems Press;
- <http://java.sun.com/j2ee/1.6/docs/tutorial/doc/index.html>
- <http://java.sun.com/products/jndi/docs.html>
- <http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html>

Java Enterprise Edition - JEE

05. Servlets com Annotations



Esp. Márcio Palheta
gtalk: marcio.palheta@gmail.com