26

Servlets

OBJETIVOS

- **Neste capítulo, você aprenderá:**
- **Como servlets podem ser utilizados para estender as funcionalidades de um servidor Web.**
- **O ciclo de vida dos servlets.**
- **Como executar servlets com o servidor Apache Tomcat.**
- **Como ser capaz de responder a solicitações de HTTP a partir de um `HttpServlet`.**
- **Como ser capaz de redirecionar solicitações a recursos Web estáticos e dinâmicos.**
- **Como utilizar JDBC a partir de um servlet.**

Sumário

26.1 Introdução

- **Capacidades de rede do Java:**
  - **Comunicações baseadas em sockets e baseadas em pacotes:**
    - **Pacote `java.net`.**
  - **Remote Method Invocation (RMI):**
    - **Pacote `java.rmi`.**
  - **Common Object Request Broker Architecture (CORBA):**
    - **Pacote `org.omg`.**
  - **Remote Method Invocation Over the Internet Inter-Orb Protocol (RMI-IIOP).**

26.1 Introdução (*Continuação*)

- **Relacionamento cliente-servidor:**
  - **Servlets e JavaServer Pages (JSP):**
    - **Modelo de solicitação-resposta.**
    - **Pacotes `javax.servlet`:**
      - `javax.servlet.http`
        `javaxjavax.servlet.jsp`
        `javaxjavax.servlet.tagext`
      - **Implementação comum do modelo de solicitação-resposta.**
      - **Navegadores Web e servidores Web formam os componentes Web do J2EE.**

26.1 Introdução (*Continuação*)

- **Clientes magros:**
  - **Fornecem a apresentação.**
  - **Não processam dados.**
  - **Requerem menos recursos de computação.**

26.1 Introdução (*Continuação*)

- **Apache Jakarta Project e o Tomcat Server.**
- **Tomcat:**
  - **Jakarta Project.**
  - **Implementação oficial de referência dos padrões de servlets e JSP do Jakarta.**

26.2 Visão geral e arquitetura de servlets

- **Servlet:**
  - **Pequena parte do conteúdo é texto estático ou marcação.**
  - **Não produz conteúdo.**
  - **Realiza uma tarefa em nome do cliente.**
- **JavaServer Pages:**
  - **Extensão da tecnologia de servlet.**
  - **A maioria do conteúdo é texto estático ou marcação.**
  - **Pequena parte do conteúdo é gerada dinamicamente.**
- **Contêiner de servlet (mecanismo de servlet):**
  - **Servidor que executa um servlet.**

26.2 Visão geral e arquitetura de servlets (*Continuação*)

- **Servidores Web e servidores de aplicativo:**
  - **Sun Java System Application Server.**
  - **Internet Information Server (IIS), da Microsoft.**
  - **Apache HTTP Server**
  - **Servidor de aplicativo WebLogic, da BEA.**
  - **Servidor de aplicativo WebSphere, da IBM.**
  - **Servidor Web Jigsaw do World Wide Web Consortium.**

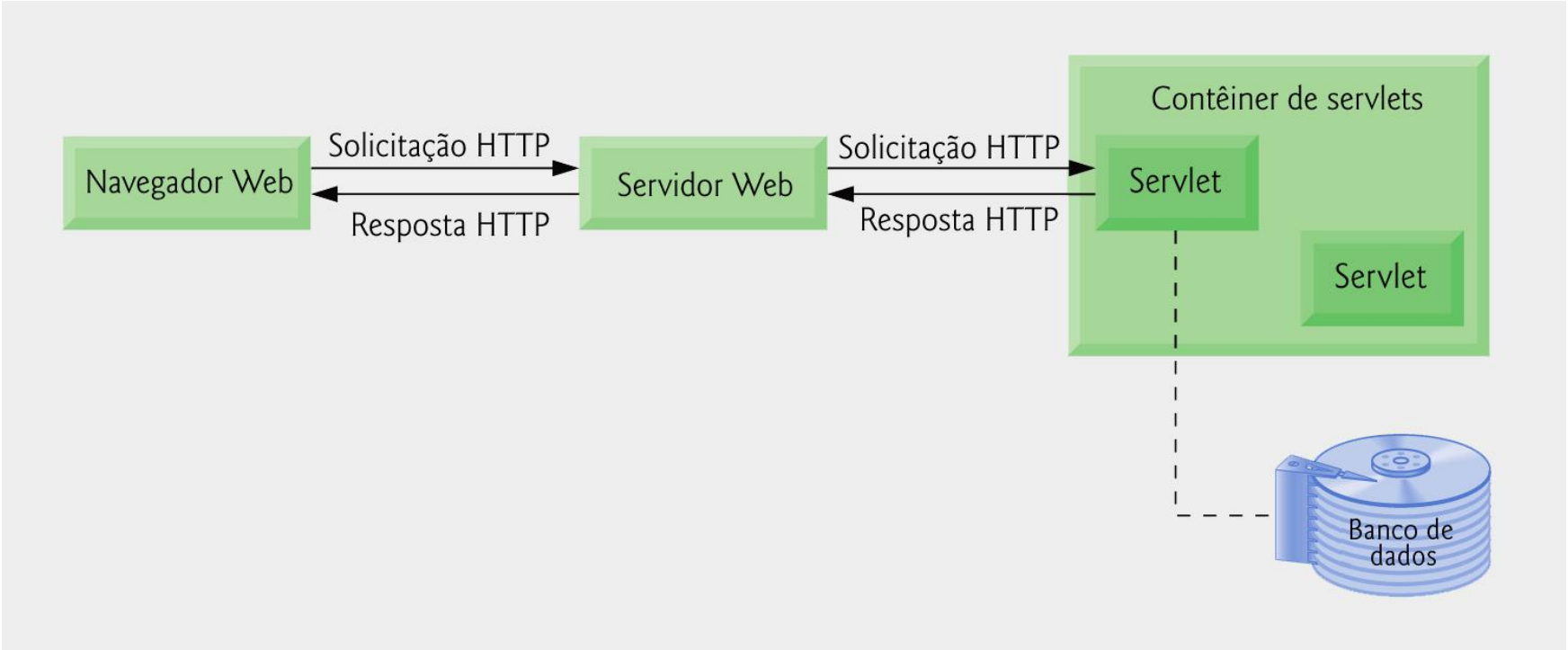


Figura 26.1 | Arquitetura de servlet.

26.2.1 Interface `Servlet` e o ciclo de vida de um servlet

- **Interface `Servlet`:**
  - **Todos os servlets devem implementar essa interface.**
  - **Todos os métodos da interface `Servlet` são invocados pelo contêiner de servlets.**
- **Ciclo de vida de um servlet:**
  - **O contêiner de servlet invoca o método `init` do servlet.**
  - **O método `service` do servlet trata solicitações.**
  - **O método `destroy` do servlet libera recursos de servlet quando o contêiner de servlet termina o servlet.**
- **Implementação de servlet:**
  - `GenericServlet`
  - `HttpServlet`

Observação de engenharia de software 26.1

Os servlets implementam a interface `Servlet` do pacote `javax.servlet`.

Método	Descrição
`void init( ServletConfig config )`	**O contêiner de servlets chama esse método uma vez durante o ciclo de execução de um servlet para inicializar o servlet. O argumento `ServletConfig` é fornecido pelo contêiner de servlets que executa o servlet.**
`ServletConfig getServletConfig()`	**Esse método retorna uma referência para um objeto que implementa a interface ServletConfig. Esse objeto fornece acesso às informações de configuração do servlet, como seus parâmetros de inicialização e `ServletContext`, que fornece ao servlet acesso ao seu ambiente (isto é, o contêiner de servlets em que o servlet executa).**
`String getServletInfo()`	**Esse método é definido por um programador de servlet para retornar uma string que contém informações do servlet como o autor e a versão do servlet.**
`void service( ServletRequest request, ServletResponse response )`	**O contêiner de servlets chama esse método para responder a uma solicitação do cliente para o servlet.**
`void destroy()`	**Esse método de 'limpeza' é chamado quando um servlet é terminado pelo seu contêiner de servlets. Os recursos utilizados pelo servlet, como abrir arquivos ou abrir conexões ao banco de dados, devem ser desalocados aqui.**

Figura 26.2 | Métodos de interface de `Servlet`.

26.2.1 Interface `Servlet` e o ciclo de vida de um servlet (*Continuação*)

- **Implementação da interface `Servlet`:**
  - `GenericServlet`:
    - **Classe abstrata.**
    - **Pacote `javax.servlet`.**
    - **Servlet independente de protocolo.**
  - `HttpServlet`:
    - **Classe abstrata.**
    - **Pacote `javax.servlet.http`.**
    - **Utiliza o protocolo HTTP para trocar informações.**
    - **Método chave `service` :**
      - `ServletRequest` e `ServletResponse`.

26.2.2 Classe `HttpServlet`

- Sobrescreve o método `service`.
- Dois tipos de solicitações HTTP mais comuns:
  - solicitações `get`:
    - Obtêm/recuperam informações no servidor.
  - solicitações `post`:
    - Postam/enviam dados ao servidor.
- O método `doGet` responde a solicitações `get`.
- O método `doPost` responde a solicitações `post`.
- Objetos `HttpServletRequest` e `HttpServletResponse`.

Método	Descrição
`doDelete`	Chamado em resposta a uma solicitação de HTTP `delete`. Essa solicitação é normalmente utilizada para excluir um arquivo de um servidor. Isso talvez não esteja disponível em alguns servidores por causa dos riscos inerentes de segurança (por exemplo, o cliente poderia excluir um arquivo que é crítico para a execução do servidor ou para um aplicativo).
`doHead`	Chamado em resposta a uma solicitação de HTTP `head`. Normalmente, essa solicitação é utilizada quando o cliente quer somente os cabeçalhos da resposta, como o tipo do seu conteúdo e comprimento do conteúdo. Sobrescrevendo esse método, o servlet não calcula o corpo da resposta, melhorando assim o desempenho.
`doOptions`	Chamado em resposta a uma solicitação de HTTP `options`. Isto retorna as informações ao cliente indicando as opções de HTTP suportadas pelo servidor, como a versão do HTTP (1.0 ou 1.1) e os métodos de solicitação que o servidor suporta.
`doPut`	Chamado em resposta a uma solicitação de HTTP `put`. Essa solicitação é normalmente utilizada para armazenar um arquivo no servidor. Isso talvez não esteja disponível em alguns servidores por causa dos riscos inerentes de segurança (por exemplo, o cliente poderia colocar um aplicativo executável no servidor, que, se executado, danificaria o servidor — talvez excluindo arquivos críticos ou ocupando os recursos).
`doTrace`	Chamado em resposta a uma solicitação de HTTP `trace`. Essa solicitação é normalmente utilizada para depuração. A implementação desse método retorna automaticamente um documento de HTML para o cliente contendo as informações de cabeçalho da solicitação (dados enviados pelo navegador como parte da solicitação).

Figura 26.3 | Outros métodos da classe `HttpServlet`.

Observação de engenharia de software 26.2

Não sobrescreva o método `service` em uma subclasse de `HttpServlet`. Fazer isso impede que o servlet faça uma distinção entre os tipos de solicitação.

26.2.3 Interface `HttpServletRequest`

- **Contêiner de servlets:**
  - **Cria um objeto `HttpServletRequest`.**
  - **Passa-o para o método `service` do servlet.**
- **Objeto `HttpServletRequest`:**
  - **Contém a solicitação do cliente.**
  - **Fornece os métodos que permitem ao servlet processar a solicitação.**

Método	Descrição
`String getParameter( String name )`	Obtém o valor de um parâmetro enviado ao servlet como parte de uma solicitação `get` ou `post`. O argumento name representa o nome de parâmetro.
`Enumeration getParameterNames()`	Retorna os nomes de todos os parâmetros enviados para o servlet como parte de uma solicitação `post`.
`String[] getParameterValues( String name )`	Para um parâmetro com múltiplos valores, esse método retorna um array de strings contendo os valores para um parâmetro especificado de servlet.
`Cookie[] getCookies()`	Retorna um array de objetos `Cookie` armazenados no cliente pelo servidor. Objetos `Cookie` podem ser utilizados para identificar unicamente os clientes para o servlet.
`HttpSession getSession( boolean create )`	Retorna um objeto `HttpSession` associado com a atual sessão de navegação do cliente. Esse método pode criar um objeto `HttpSession` (argumento true) se um ainda não existir para o cliente. Objetos `HttpSession` e `Cookies` são utilizados de maneiras semelhantes para clientes unicamente identificados.
`String getLocalName()`	Obtém o nome de host em que a solicitação foi recebida.
`String getLocalAddr()`	Obtém o endereço IP (Internet Protocol) em que a solicitação foi recebida.
`int getLocalPort()`	Obtém o número de porta do Internet Protocol (IP) em que a solicitação foi recebida.

Figura 26.4 | Métodos `HttpServletRequest`.

26.2.4 Interface HttpServletResponse

- **Contêiner de servlets:**
  - **Cria um objeto `HttpServletResponse`.**
  - **Passa-o para o método `service` do servlet.**
- **Objeto `HttpServletResponse`:**
  - **Fornece os métodos que permitem ao servlet formular a resposta ao cliente.**

Método	Descrição
`void addCookie( Cookie cookie )`	Utilizado para adicionar um `Cookie` ao cabeçalho da resposta para o cliente. A idade máxima do `Cookie` e se `Cookies` estão ativados no cliente determina se `Cookies` são armazenados no cliente.
`ServletOutputStream getOutputStream()`	Obtém um fluxo de saída baseado e bytes para enviar dados binários ao cliente.
`PrintWriter getWriter()`	Obtém um fluxo de saída baseado em caracteres para enviar dados de texto (normalmente, texto formatado em HTML) ao cliente.
`void setContentType( String type )`	Especifica o tipo de conteúdo da resposta para o navegador. O tipo de conteúdo ajuda o navegador a determinar como exibir os dados (ou possivelmente que outro aplicativo executar para processar os dados). O tipo de conteúdo também é conhecido como o tipo de dados MIME (*Multipurpose Internet Mail Extensions*). Por exemplo, o tipo do conteúdo `"text/html"` indica que a resposta é um documento HTML, portanto o navegador exibe a página HTML; o tipo do conteúdo `"image/gif"` indica que a resposta é uma imagem, portanto o navegador exibe a imagem. Para uma lista completa de tipos de conteúdo, visite `www.isi.edu/in-notes/iana/assignments/media-types/media-types`.
`String getContentType()`	Obtém o tipo de conteúdo da resposta.

Figura 26.5 | Métodos `HttpServletResponse`.

26.3 Configurando o servidor Apache Tomcat

- **Faça o download do Tomcat (versão 5.0.25):**
  - `apache.towardex.com/jakarta/tomcat-5/v5.0.25/bin`
- **Defina as variáveis de ambiente:**
  - **JAVA_HOME**
    - `C:\Program Files\Java\jdk1.5.0`
  - **CATALINA_HOME**
    - `C:\jakarta-tomcat-5.-.25`

Dica de prevenção de erro 26.1

Em algumas plataformas, talvez seja necessário reiniciar seu computador para que as novas variáveis de ambiente tenham efeito.

26.3 Configurando o servidor Apache Tomcat (*Continuação*)

- **Inicie o servidor Tomcat:**
  - `startup`
- **Carregue o servidor Tomcat:**
  - **http://localhost:8080/**
- **Desative o servidor Tomcat:**
  - `shutdown`

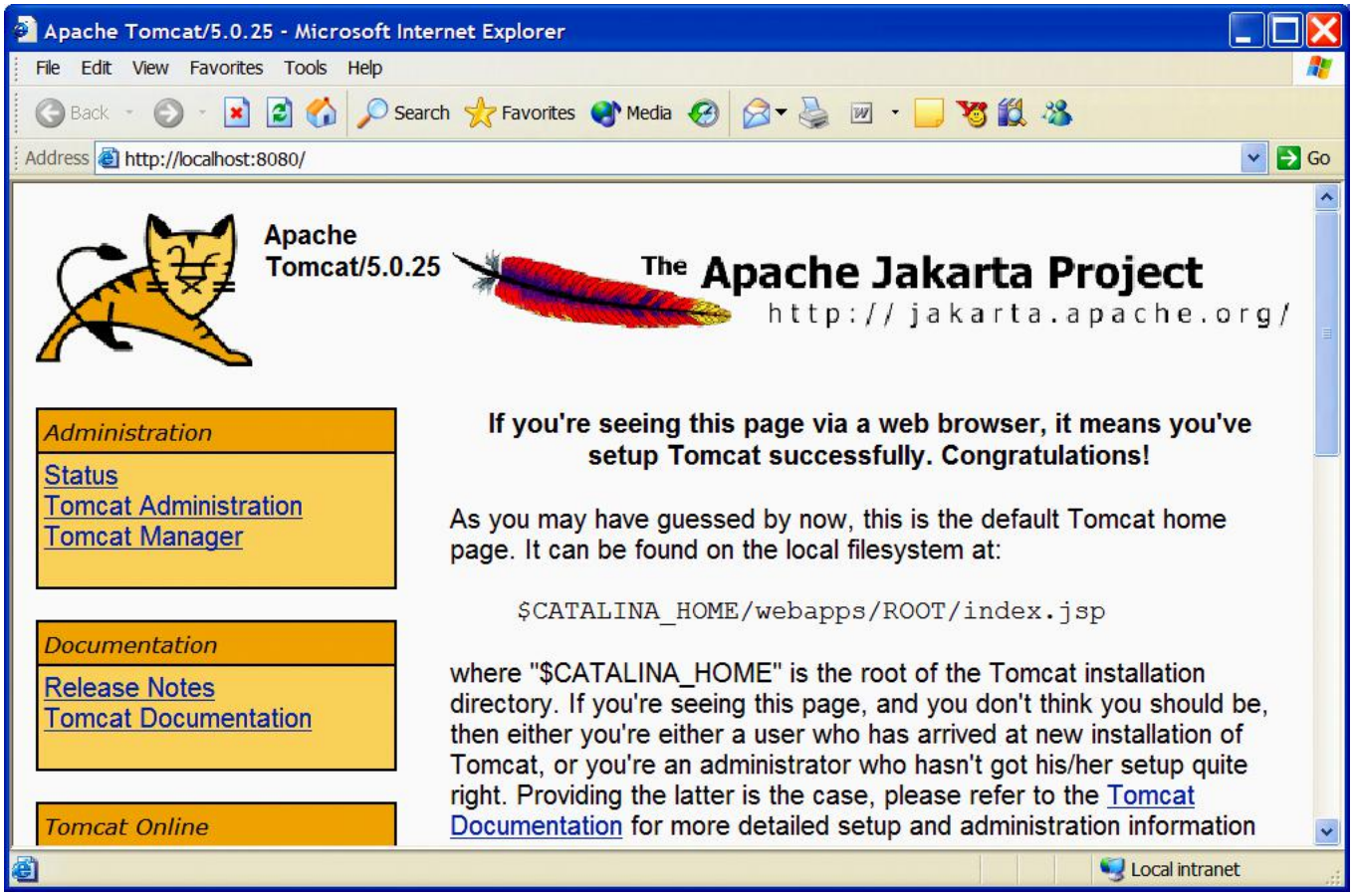


Figura 26.6 | A home page da documentação do Tomcat. Copyright © 2000-2004 The Apache Software Foundation (`http://www.apache.org/`). Todos os direitos reservados.

Dica de prevenção de erro 26.2

Se o nome de host `localhost` não funcionar no seu computador, substitua-o pelo endereço IP `127.0.0.1`.

26.4 Tratando solicitações `get` de HTTP

- **Solicitação `get`:**
  - **Recupera o conteúdo de um URL.**
- **Exemplo: `WelcomeServlet`:**
  - **O servlet trata solicitações `get` de HTTP.**

Resumo

```
1  // Fig. 26.7: WelcomeServlet.java
2  // Um servlet simples para processar solicitações get.
3  import javax.servlet.ServletException;
4  import javax.servlet.http.HttpServlet;
5  import javax.servlet.http.HttpServletRequest;
6  import javax.servlet.http.HttpServletResponse;
7  import java.io.IOException;
8  import java.io.PrintWriter;
9
10 public class WelcomeServlet extends HttpServlet
11 {
12    // processa solicitações "get" dos clientes
13    protected void doGet( HttpServletRequest request,
14       HttpServletResponse response )
15          throws ServletException, IOException
16    {
17       response.setContentType( "text/html" );
18       PrintWriter out = response.getWriter();
19
20       // envia página de XHTML para o cliente
21
22       // inicia documento XHTML
23       out.println( "<?xml version = \"1.0\"?>" );
24
25       out.printf( "%s%s%s", "<!DOCTYPE html PUBLIC",
26          " \"-//W3C//DTD XHTML 1.0 Strict//EN\"",
27          " \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">\n" );
28
29       out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
30
```

Importa as classes e interfaces nos pacotes **javax.servlet** e **javax.servlet.http**

WelcomeServlet.java

Estende **HttpServlet** para tratar solicitações HTTP **get** e solicitações HTTP **post**

Sobrescreve o método **doGet** para fornecer processamento de solicitações **get** personalizado

Utiliza o método **setContentType** do objeto **response** para especificar o tipo MIME da resposta

Utiliza o método **getWriter** do objeto **response** para obter uma referência ao objeto **PrintWriter** que permite ao servlet enviar o conteúdo ao cliente.

Cria o documento XHTML gravando strings com o método **println** do objeto **out**

Linha 23

Resumo

WelcomeServlet.java

(2 de 2)

Linha 43

```
31       // seção da cabeça do documento
32       out.println( "<head>" );
33       out.println( "<title>A Simple Servlet Example</title>" );
34       out.println( "</head>" );
35
36       // seção do corpo do documento
37       out.println( "<body>" );
38       out.println( "<h1>Welcome to Servlets!</h1>" );
39       out.println( "</body>" );
40
41       // fim do documento XHTML
42       out.println( "</html>" );
43       out.close();  // fecha o fluxo para completar a página
44    } // fim do método doGet
45 } // fim da classe WelcomeServlet
```

Fecha o fluxo de saída, esvazia o buffer de saída e envia as informações ao cliente

Resumo

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 26.7: WelcomeServlet.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Handling an HTTP
10 </head>
11
12 <body>
13    <form action = "/jhtp6/welcome1" method = "get">
14       <p><label>Click the button to invoke the servlet
15          <input type = "submit" value = "Get HTML Document" />
16       </label></p>
17    </form>
18 </body>
19 </html>
```

WelcomeServlet.htm
l

O atributo **action** do fo
(/jhtp6/welcome1) e
caminho do URL que inv

O atributo **method** do **form** indica que o navegador envia uma solicitação **get** ao servidor, o que resulta em uma chamada ao método **doGet** do servlet

Linha 15

Cria um botão; quando clicado, a ação do formulário é realizada

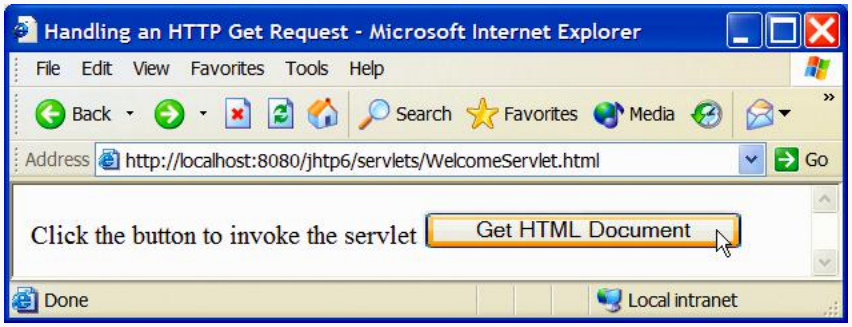
Linha 15

Resumo

`WelcomeServlet.html`

(2 de 2)

Saída do programa



Observação de engenharia de software 26.3

A documentação do Tomcat especifica como integrar o Tomcat a aplicativos populares de servidor Web como o servidor HTTP Apache e o IIS da Microsoft.

Erro comum de programação 26.1

Utilizar ‘servlet’ ou ‘servlets’ como uma raiz de contexto talvez impeça que um servlet funcione corretamente em alguns servidores.

26.4.1 Implementando um aplicativo Web

- **Aplicativos Web:**
  - **JSPs, servlets e seus arquivos de suporte.**
- **Implementando um aplicativo Web:**
  - **Estrutura de diretório:**
    - **Raiz de contexto.**
  - **Arquivo WAR (repositório de arquivos de aplicativos Web).**
  - **Descritor de implantação:**
    - `web.xml`

Diretório	Descrição
raiz de contexto	Esse é o diretório-raiz para o aplicativo Web. Todos os JSPs, documentos HTML, servlets e arquivos de suporte como imagens e arquivos de classe residem nesse diretório ou nos seus subdiretórios. O nome desse diretório é especificado pelo criador do aplicativo Web. Para fornecer estrutura a um aplicativo Web, os subdiretórios podem ser colocados na raiz de contexto. Por exemplo, se seu aplicativo utilizar muitas imagens, você poderia criar um subdiretório images nesse diretório. Os exemplos deste capítulo utilizam o `jhtp6` como a raiz de contexto.
`WEB-INF`	Esse diretório contém o descritor de implantação de aplicativo Web (`web.xml`).
`WEB-INF/classes`	Esse diretório contém os arquivos da classe de servlet e outros arquivos de suporte de classe utilizados em um aplicativo Web. Se as classes fossem parte de um pacote, a estrutura completa de diretórios dos pacotes iniciaria aqui.
`WEB-INF/lib`	Esse diretório contém arquivos (JAR) Java archive. Os arquivos JAR podem conter arquivos de classe de servlet e outros arquivos de suporte de classe utilizados em um aplicativo Web.

Figura 26.9 | Diretórios-padrão de aplicativos Web.

Resumo

```
1  <web-app xmlns="http://java.sun.com/xml/ns/j2ee"
2     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
4        http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
5     version="2.4">
6
7     <!-- Descrição geral do se
8     <display-name>
9        Java How to Program JSP
10       and Servlet Chapter Exa
11    </display-name>
12
13    <description>
14       This is the Web applic
15       demonstrate our JSP an
16    </description>
17
18    <!-Definições de servlet -->
19    <servlet>
20       <servlet-name>we
21
22       <description>
23          A simple servlet th
24       </description>
25
26       <servlet-class>
27          WelcomeServlet
28       </servlet-class>
29    </servlet>
30
```

O elemento **web-app** define a configuração de cada servlet no aplicativo Web e o mapeamento de servlets para cada servlet.

Linhas 1-37

O elemento **display** (linhas 811) especifica pode ser exibido para o administrador do servi

O elemento **description** especifica uma descrição do aplicativo Web que seria exibida para o administrador do servidor

Linhas 8-11

Linhas 13-16

O elemento **servlet** descre um servlet

O elemento **servlet-name** é o nome do servlet

has 19-29

O elemento **description** especifica uma descrição para esse servlet particular

Linha 20

O elemento **servlet-class** especifica o nome de classe completamente qualificado do servlet compilado

Linhas 22-24

Linhas 26-28

Resumo

```
31    <!-- Mapeamentos de Servlet -->
32    <servlet-mapping>
33       <servlet-name>welcome1</servlet-name>
34       <url-pattern>/welcome1</url-pattern>
35    </servlet-mapping>
36
37 </web-app>
```

O elemento **`servlet-mapping`** especifica os el… e …

O elemento **`url-pattern`** ajuda o servidor a determinar quais solicitações são enviadas ao servlet

(2 de 2)

Linhas 32-35

Linha 34

26.4.1 Implementando um aplicativo Web (*Continuação*)

- **Invoca o exemplo de `WelcomeServlet`:**
  - **/jhtp6/welcome1**
    - **`/jhtp6` especifica a raiz do contexto.**
    - **`/welcome1` especifica o padrão do URL.**
- **Formatos do padrão de URL:**
  - **Correspondência exata:**
    - **`/jhtp5/welcome1`**
  - **Mapeamentos de caminho:**
    - **`/jhtp5/example/*`**
  - **Mapeamentos de extensão:**
    - ***.jsp**
  - **Servlet padrão:**
    - **/**

O diretório de aplicativos Web e a estrutura dos arquivos para `WelcomeServlet`

```
jhtp6
   servlets
      WelcomeServlet.html
   WEB-INF
      web.xml
      classes
         WelcomeServlet.class
```

Figura 26.11 | O diretório de aplicativos Web e a estrutura dos arquivos para `WelcomeServlet`.

Erro comum de programação 26.2

Não colocar um servlet ou outros arquivos de classe na estrutura de diretório apropriada impede que o servidor localize essas classes adequadamente. Isso resulta em uma resposta de erro ao navegador Web do cliente.

Dica de prevenção de erro 26.3

Você pode testar um servlet que trata solicitações `get` de HTTP digitando o URL que invoca o servlet diretamente no campo `Endereço` ou `Local` do seu navegador porque `get` é o método-padrão de HTTP ao navegar.

26.5 Tratando solicitações `get` de HTTP contendo dados

- **Servlet `WelcomeServlet2`:**
  - **Responde a uma solicitação `get` que contém dados.**

Resumo

WelcomeServlet2.java

(1 de 2)

Linha 17

```
1  // Fig. 26.12: WelcomeServlet2.java
2  // Processando solicitações get de HTTP contendo dados.
3  import javax.servlet.ServletException;
4  import javax.servlet.http.HttpServlet;
5  import javax.servlet.http.HttpServletRequest;
6  import javax.servlet.http.HttpServletResponse;
7  import java.io.IOException;
8  import java.io.PrintWriter;
9
10 public class WelcomeServlet2 extends HttpServlet
11 {
12    // processa a solicitação "get" do cliente
13    protected void doGet( HttpServletRequest request,
14       HttpServletResponse response )
15          throws ServletException, IOException
16    {
17       String firstName = request.getParameter( "firstname" );
18
19       response.setContentType( "text/html" );
20       PrintWriter out = response.getWriter();
21
22       // envia o documento XHTML ao cliente
23
24       // inicia o documento XHTML
25       out.println( "<?xml version = \"1.0\"?>" );
26
27       out.printf( "%s%s%s", "<!DOCTYPE html PUBLIC",
28          " \"-//W3C//DTD XHTML 1.0 Strict//EN\"",
29          " \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">\n" );
```

O método **getParameter** do objeto **request** recebe o nome do parâmetro e retorna o valor de **String** correspondente

Resumo

WelcomeServlet2.java

(2 de 2)

```
30
31       out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
32
33       // seção da cabeça do documento
34       out.println( "<head>" );
35       out.println(
36          "<title>Processing get requests with data</title>" );
37       out.println( "</head>" );
38
39       // seção do corpo do documento
40       out.println( "<body>" );
41       out.println( "<h1>Hello " + firstName + ",<br />" );
42       out.println( "Welcome to Servlets!</h1>" );
43       out.println( "</body>" );
44
45       // fim do documento XHTML
46       out.println( "</html>" );
47       out.close();  // fecha o fluxo para completar a página
48    } // fim do método doGet
49 } // fim da classe WelcomeServlet2
```

Utiliza o resultado da linha 17 como parte da resposta ao cliente

Resumo

WelcomeServlet2.html

(1 de 2)

Linha 16

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 26.13: WelcomeServlet2.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Processing get requests with data</title>
10 </head>
11
12 <body>
13    <form action = "/jhtp6/welcome2" method = "get">
14       <p><label>
15          Type your first name and press the Submit button
16          <br /><input type = "text" name = "firstname" />
17          <input type = "submit" value = "Submit" />
18       </p></label>
19    </form>
20 </body>
21 </html>
```

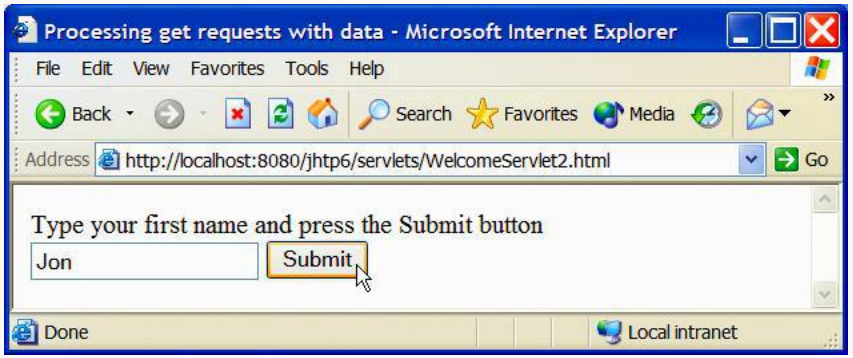
Obtém o primeiro nome do usuário.

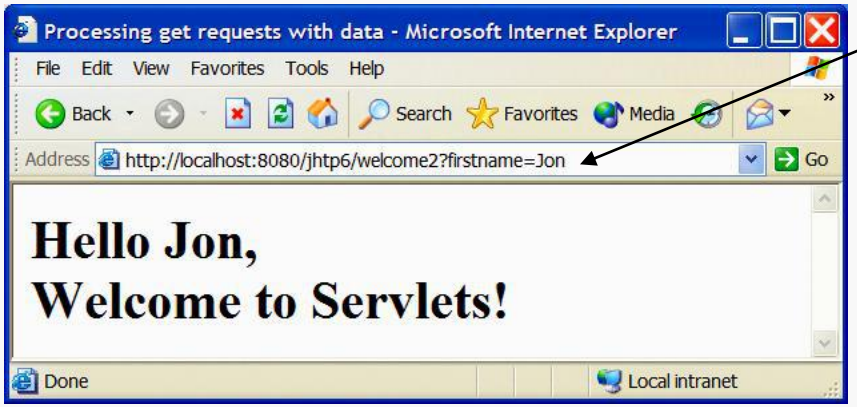
Resumo

`WelcomeServlet2.html`

(2 de 2)

`Saída do programa`





Dados do formulário especificados na string de consulta do URL como parte de uma solicitação `get`

Elemento de descritor	Valor
elemento `servlet`	
`servlet-name`	`welcome2`
`description`	`Handling HTTP get requests with data.`
`servlet-class`	`WelcomeServlet2`
elemento `servlet-mapping`	
`servlet-name`	`welcome2`
`url-pattern`	`/welcome2`

Figura 26.14 | Informações do descritor de implantação para o servlet `WelcomeServlet2`.

Dica de prevenção de erro 26.4

Se um erro ocorrer durante a invocação do servlet, os arquivos de log no diretório `logs` da instalação do Tomcat podem ajudá-lo a determinar o erro e depurar o problema.

Observação de engenharia de software 26.4

Uma solicitação `get` está limitada a caracteres-padrão, o que significa que você não pode submeter nenhum caractere especial via uma solicitação `get`. O comprimento do URL em uma solicitação `get` é limitado. Por exemplo, o comprimento máximo de URL no Internet Explorer é de 2.083 caracteres. Alguns servidores Web talvez restrinjam ainda mais esse comprimento.

Boa prática de programação 26.1

Uma solicitação `get` não deve ser utilizada para enviar dados sigilosos (por exemplo, uma senha) porque os dados no formulário são colocados em uma string de consulta que é acrescentada ao URL de solicitação como texto simples e pode ser interceptada.

26.6 Tratando solicitações `post` de HTTP

- **Solicitação HTTP `post`:**
  - **Posta os dados em um formulário HTML para um handler de formulário do lado do servidor.**
  - **Procura páginas Web em cache.**
- **Servlet `WelcomeServlet3`:**
  - **Responde a uma solicitação `post` que contém dados.**

Resumo

WelcomeServlet3.java

(1 de 2)

```
1  // Fig. 26.15: WelcomeServlet3.java
2  // Processando solicitações post contendo dados.
3  import javax.servlet.ServletException;
4  import javax.servlet.http.HttpServlet;
5  import javax.servlet.http.HttpServletRequest;
6  import javax.servlet.http.HttpServletResponse;
7  import java.io.IOException;
8  import java.io.PrintWriter;
9
10 public class WelcomeServlet3 extends HttpServlet
11 {
12    // processa uma solicitação "post" do cliente
13    protected void doPost( HttpServletRequest request,
14       HttpServletResponse response )
15          throws ServletException, IOException
16    {
17       String firstName = request.getParameter( "firstname" );
18
19       response.setContentType( "text/html" );
20       PrintWriter out = response.getWriter();
21
22       // envia página XHTML para o cliente
23
24       // inicia o documento XHTML
25       out.println( "<?xml version = \"1.0\"?>" );
26
27       out.printf( "%s%s%s", "<!DOCTYPE html PUBLIC",
28          " \"-//W3C//DTD XHTML 1.0 Strict//EN\"",
29          " \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">\n" );
30
```

Declara um método **doPost** para responder a solicitações post

-48

O método **getParameter** do objeto **request** recebe o nome do parâmetro e retorna o valor de **String** correspondente

```
31       out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
32
33       // seção da cabeça do documento
34       out.println( "<head>" );
35       out.println(
36          "<title>Processing post requests with data</title>" );
37       out.println( "</head>" );
38
39       // seção do corpo do documento
40       out.println( "<body>" );
41       out.println( "<h1>Hello " + firstName + ",<br />" );
42       out.println( "Welcome to Servlets!</h1>" );
43       out.println( "</body>" );
44
45       // fim do documento XHTML
46       out.println( "</html>" );
47       out.close(); // fecha o fluxo para completar a página
48    } // fim do método doPost
49 } // fim da classe WelcomeServlet3
```

Resumo

WelcomeServlet3.java

(2 de 2)

Resumo

WelcomeServlet3.html

(1 de 2)

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 26.16: WelcomeServlet3.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Handling an HTTP Post Request with Data</title>
10 </head>
11
12 <body>
13    <form action = "/jhtp6/welcome3" method = "post">
14       <p><label>
15          Type your first name and press the Submit button
16          <br /><input type = "text" name = "firstname" />
17          <input type = "submit" value = "Submit" />
18       </label></p>
19    </form>
20 </body>
21 </html>
```

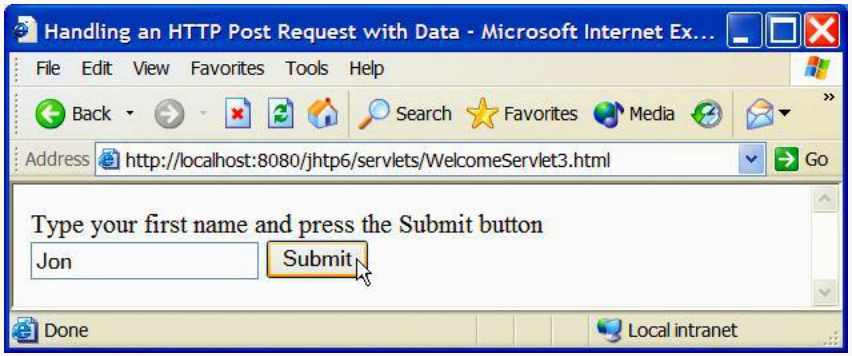
Fornece um **form** em que o usuário pode inserir um nome no elemento de entrada **text firstname** e então clicar no botão **Submit** para invocar **WelcomeServlet3**

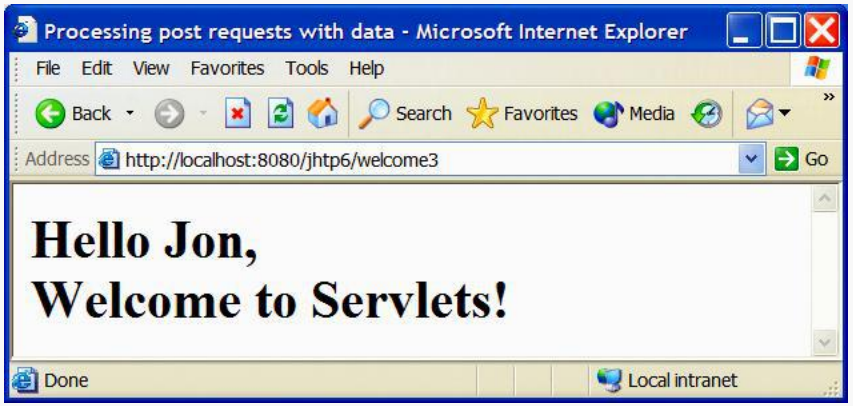
Resumo

WelcomeServlet3.html

(2 de 2)

Saída do programa





Elemento de descritor	Valor
elemento `servlet`	
`servlet-name`	`welcome3`
`description`	`Handling HTTP post requests with data.`
`servlet-class`	`WelcomeServlet3`
elemento `servlet-mapping`	
`servlet-name`	`welcome3`
`url-pattern`	`/welcome3`

Figura 26.17 | Informações do descritor de implantação para o servlet `WelcomeServlet3`.

26.7 Redirecionando solicitações para outros recursos

- Servlet `RedirectServlet`:
  - Redireciona a solicitação para um recurso diferente.

Resumo

RedirectServlet.Java

(1 de 2)

Linha 17

21 e 23

Linha 22

```
1  // Fig. 26.18: RedirectServlet.java
2  // Redirecionando um usuário para uma página Web diferente.
3  import javax.servlet.ServletException;
4  import javax.servlet.http.HttpServlet;
5  import javax.servlet.http.HttpServletRequest;
6  import javax.servlet.http.HttpServletResponse;
7  import java.io.IOException;
8  import java.io.PrintWriter;
9
10 public class RedirectServlet extends HttpServlet
11 {
12    // processa a solicitação "get" do cliente
13    protected void doGet( HttpServletRequest request,
14       HttpServletResponse response )
15          throws ServletException, IOException
16    {
17       String location = request.getParameter( "page" );
18
19       if ( location != null )
20       {
21          if ( location.equals( "deitel" ) )
22             response.sendRedirect( "http://www.deitel
23          else if ( location.equals( "welcome1" ) )
24             response.sendRedirect( "welcome1" );
25       } // fim do if
26
27       // código que executa somente se esse servlet
28       // não redirecionar o usuário para uma outra página
29       response.setContentType( "text/html" );
30       PrintWriter out = response.getWriter();
```

Obtém o parâmetro **page** da solicitação.

Determina se o valor é '**deitel**' ou '**w**

Redireciona a

Redireciona a solicitação ao servlet **WelcomeServlet**

```
31
32       // inicia o documento XHTML
33       out.println( "<?xml version = \"1.0\"?>" );
34
35       out.printf( "%s%s%s", "<!DOCTYPE html PUBLIC",
36          " \"-//W3C//DTD XHTML 1.0 Strict//EN\"",
37          " \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">\n" );
38
39       out.println(
40          "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
41
42       // seção da cabeça do documento
43       out.println( "<head>" );
44       out.println( "<title>Invalid page</title>" );
45       out.println( "</head>" );
46
47       // seção do corpo do documento
48       out.println( "<body>" );
49       out.println( "<h1>Invalid page requested</h1>" );
50       out.println( "<p><a href = " +
51          "\"servlets/RedirectServlet.html\">" );
52       out.println( "Click here to choose again</a></p>" );
53       out.println( "</body>" );
54
55       // fim do documento XHTML
56       out.println( "</html>" );
57       out.close();  // fecha o fluxo para completar a página
58    } // fim do método doGet
59 } // fim da classe RedirectServlet
```

Resumo

RedirectServlet.Java

(2 de 2)

Observação de engenharia de software 26.5

Utilizar caminhos relativos para fazer referência a recursos na mesma raiz de contexto torna seu aplicativo Web mais flexível. Por exemplo, você pode alterar a raiz de contexto sem fazer alterações nos recursos estáticos e dinâmicos no aplicativo.

Resumo

RedirectServlet.html

(1 de 2)

Linhas 15-16 e 17-18

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 26.19: RedirectServlet.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Redirecting a Request to Another Site</title>
10 </head>
11
12 <body>
13    <p>Click a link to be redirected to the appropriate page</p>
14    <p>
15    <a href = "/jhtp6/redirect?page=deitel">
16       www.deitel.com</a><br />
17    <a href = "/jhtp6/redirect?page=welcome1">
18       Welcome servlet</a>
19    </p>
20 </body>
21 </html>
```

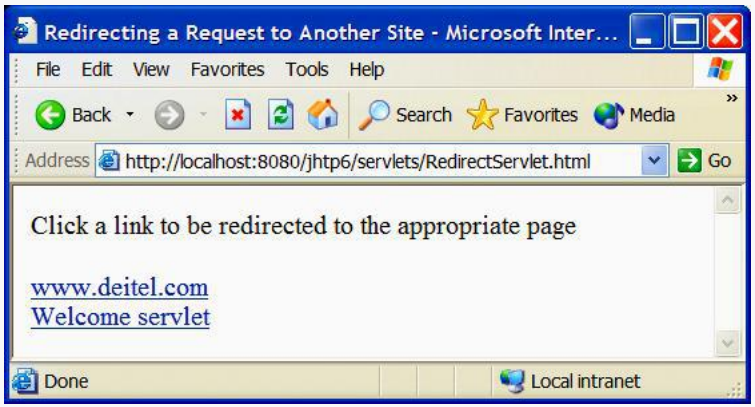
Fornecem os links que permitem ao usuário invocar o servlet **RedirectServlet**

Resumo

RedirectServlet.html

(2 de 2)

Saída do programa



Elemento de descritor	Valor
elemento `servlet`	
`servlet-name`	`redirect`
`description`	`Redirecting to static Web pages and other servlets.`
`servlet-class`	`RedirectServlet`
elemento `servlet-mapping`	
`servlet-name`	`redirect`
`url-pattern`	`/redirect`

Figura 26.20 | Informações do descritor de implantação para o servlet `RedirectServlet`.

26.8 Aplicativos de múltiplas camadas: Utilizando JDBC a partir de um servlet

- **Aplicativos distribuídos de três camadas:**
  - **Interface com o usuário.**
  - **Lógica do negócio.**
  - **Banco de dados.**
- **Servidores Web freqüentemente representam a camada intermediária.**
- **Exemplo de aplicativo distribuído de três camadas:**
  - `SurveyServlet`
  - `Survey.html`
  - **Banco de dados MySQL**

Resumo

SurveyServlet.java

(1 de 6)

Linhas 7-11

Linha 21

Linha 22

```
1  // Fig. 26.21: SurveyServlet.java
2  // Uma pesquisa baseada na Web que utiliza o JDBC a partir de um servlet.
3  package com.deitel.jhtp6.servlets;
4
5  import java.io.PrintWriter;
6  import java.io.IOException;
7  import java.sql.Connection;
8  import java.sql.DriverManager;
9  import java.sql.Statement;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12 import javax.servlet.ServletConfig;
13 import javax.servlet.ServletException;
14 import javax.servlet.UnavailableException;
15 import javax.servlet.http.HttpServlet;
16 import javax.servlet.http.HttpServletRequest;
17 import javax.servlet.http.HttpServletResponse;
18
19 public class SurveyServlet extends HttpServlet
20 {
21    private Connection connection;
22    private Statement statement;
23
```

Importa interfaces e classes para manipulação de banco de dados

Declara uma `Connection` para

Declara uma `Statement` para atualizar a contagem de votos, totalizando todos os votos e obtendo o resultado completo da enquete

Resumo

```
24    // configura a conexão de banco de dados e cria a instrução de SQL
25    public void init( ServletConfig config ) throws ServletException
26    {
27       // tenta uma conexão ao banco de dados e cria instruções
28       try
29       {
30          Class.forName( config.getInitParameter( "databaseDriver" ) );
31          connection = DriverManager.getConnection(
32             config.getInitParameter( "databaseName" ) );
33             config.getInitParameter( "username" ),
34             config.getInitParameter( "password" ) );
35
36          // cria Statement para consultar banco de dados
37          statement = connection.createStatement();
38       } // fim do try
39       // para qualquer exceção lança uma UnavailableExcepti
40       // indica que o servlet não está atualmente disponível
41       catch ( Exception exception )
42       {
43          exception.printStackTrace();
44          throw new UnavailableException(exception.getMessage());
45       } // fim do catch
46    } // fim do método init
47
```

Carrega o driver do banco de dados, que é

Tenta abrir uma conexão ao banco de dados **animalsurvey**; o nome do banco de dados, nome de usuário e senha são parâmetros

Cria **Statement** para consultar o banco de dados

Linha 37

Resumo

SurveyServlet.java

(3 de 6)

Linhas 71-72

```
48    // processa as respostas da pesquisa
49    protected void doPost( HttpServletRequest request,
50       HttpServletResponse response )
51          throws ServletException, IOException
52    {
53       // configura a respostas para o cliente
54       response.setContentType( "text/html" );
55       PrintWriter out = response.getWriter();
56
57       // inicia o documento XHTML
58       out.println( "<?xml version = \"1.0\"?>" );
59
60       out.printf( "%s%s%s", "<!DOCTYPE html PUBLIC",
61          " \"-//W3C//DTD XHTML 1.0 Strict//EN\"",
62          " \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">\n" );
63
64       out.println(
65          "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
66
67       // seção da cabeça do documento
68       out.println( "<head>" );
69
70       // lê a resposta de pesquisa atual
71       int value =
72          Integer.parseInt( request.getParameter( "animal" ) );
73       String sql;
74
```

Obtém a resposta da pesquisa

Resumo

```
75    // tenta processar um voto e exibe os resultados atuais
76    try
77    {
78       // atualiza o total para as respostas atuais da pesquisa
79       sql = "UPDATE surveyresults SET votes = votes + 1 " +
80             "WHERE id = " + value;
81       statement.executeUpdate( sql );
82
83       // obtém o total de todas as respostas da p
84       sql = "SELECT sum( votes ) FROM surveyresults";
85       ResultSet totalRS = statement.executeQuery( sql );
86       totalRS.next(); // position to first record
87       int total = totalRS.getInt( 1 );
88
89       // obtém os resultados
90       sql = "SELECT surveyoption, votes, id FROM surveyresults " +
91          "ORDER BY id";
92       ResultSet resultsRS = statement.executeQuery( sql );
93       out.println( "<title>Thank you!</title>" );
94       out.println( "</head>" );
95
96       out.println( "<body>" );
97       out.println( "<p>Thank you for participating." );
98       out.println( "<br />Results:</p><pre>" );
99
```

Cria SQL para atualizar o total para a resposta da

Executa a instrução SQL para atualizar o total para a resposta da enquete atual

Cria a consulta para obter o

Executa a consulta para obter o total de todas as respostas da enquete

Linha 81

Cria a consulta para obter os resultados da

Executa a consulta para obter os resultados da enquete

Linhas 90-91

Linha 92

Resumo

```
100       // processa os resultados
101       int votes;
102
103       while ( resultsRS.next() )
104       {
105          out.print( resultsRS.getString(
106          out.print( ": " );
107          votes = resultsRS.getInt( 2 );
108          out.printf( "%.2f", ( double ) votes / to
109          out.print( "%  responses: " );
110          out.println( votes );
111       } // fim do while
112
113       resultsRS.close();
114
115       out.print( "Total responses: " );
116       out.print( total );
117
118       // fim do documento XHTML
119       out.println( "</pre></body></html>" );
120       out.close();
121    } // fim do try
122    // se ocorrer uma exceção de banco de dados, retorna a página de erro
123    catch ( SQLException sqlException )
124    {
125       sqlException.printStackTrace();
126       out.println( "<title>Error</title>" );
127       out.println( "</head>" );
128       out.println( "<body><p>Database error occurred. " );
129       out.println( "Try again later.</p></body></html>" );
```

Faz um loop por todos os registros em **resultsRS**

Obtém o valor da primeira

Obtém o valor da segunda coluna no registro atual

vlet.java

(5 de 6)

Linhas 103-111

Linha 105

Linha 107

```
130         out.close();
131      } // // fim do catch
132   } // fim do método doPost
133
134   // fecha instruções de SQL e banco de dados quando servlet termina
135   public void destroy()
136   {
137      // tenta fechar instruções e conexão do banco de dados
138      try
139      {
140         statement.close();
141         connection.close();
142      } // fim do try
143      // trata exceções de banco de dados retornando um erro ao cliente
144      catch ( SQLException sqlException )
145      {
146         sqlException.printStackTrace();
147      } // fim do catch
148   } // fim do método destroy
149} // fim da classe SurveyServlet
```

Fecha **`Statement`** e a conexão de banco de dados

Resumo

`SurveyServlet.java`

(6 de 6)

Linhas 140-141

Resumo

Survey.html

(1 de 2)

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 26.22: Survey.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Survey</title>
10 </head>
11
12 <body>
13 <form method = "post" action = "/jhtp6/animalsurvey">
14    <p>What is your favorite pet?</p>
15    <p>
16       <input type = "radio" name = "animal"
17          value = "1" />Dog<br />
18       <input type = "radio" name = "animal"
19          value = "2" />Cat<br />
20       <input type = "radio" name = "animal"
21          value = "3" />Bird<br />
22       <input type = "radio" name = "animal"
23          value = "4" />Snake<br />
24       <input type = "radio" name = "animal"
25          value = "5" checked = "checked" />None
26    </p>
27    <p><input type = "submit" value = "Submit" /></p>
28 </form>
29 </body>
30 </html>
```

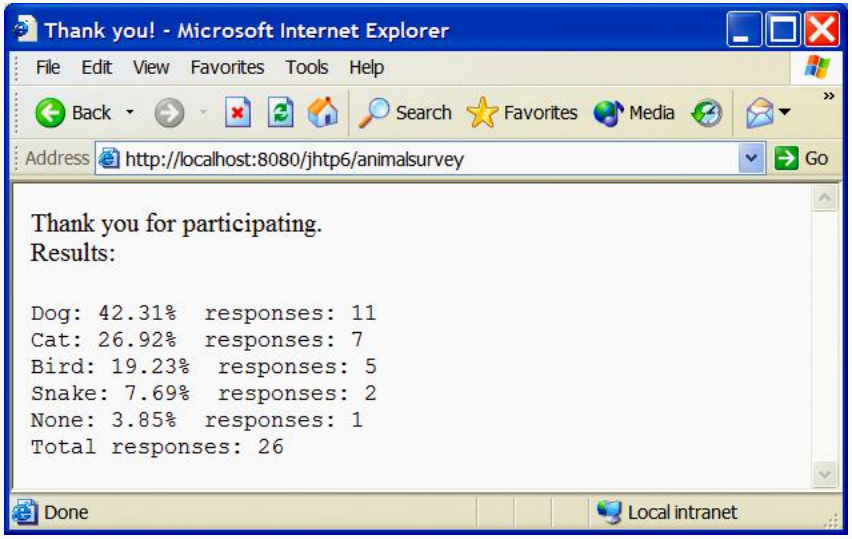
Fornece um **form** em que o usuário pode selecionar um animal em uma lista de botões de opção e, então, clicar no botão **Submit** para invocar o **animalsurvey**

Resumo

`Survey.html`

(2 de 2)

Saída do programa



Elemento de descritor	Valor
elemento servlet	
servlet-name	animalsurvey
description	Connecting to a database from a servlet.
servlet-class	com.deitel.jhtp6.servlets.SurveyServlet
init-param	
param-name	databaseDriver
param-value	com.mysql.jdbc.Driver
init-param	
param-name	databaseName
param-value	jdbc:mysql://localhost/animalsurvey
init-param	
param-name	username
param-value	jhtp6
init-param	
param-name	password
param-value	jhtp6
elemento servlet-mapping	
servlet-name	animalsurvey
url-pattern	/animalsurvey

Figura 26.23 | Informações do descritor de implantação para o servlet `SurveyServlet`.

26.9 Arquivos welcome

- **Arquivos welcome.**
  - **Lista ordenada dos arquivos:**
    - **Documentos JSP, HTML.**
  - **Carregados quando o URL de solicitação não é mapeado para um servlet.**
  - **Definidos utilizando o elemento `welcome-file-list`**
    - **Contêm um ou mais elementos `welcome-file`.**
      - **Especificam o URL parcial de um arquivo `welcome`.**
        - **Sem / no início ou no fim.**
    - **Por exemplo, para especificar `index.html` e `index.htm` como arquivos:**

```
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
</welcome-file-list>
```

Resumo

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3
4
5  <!-- Fig. 26.24: index.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Welcome File</title>
10 </head>
11
12 <body>
13    <p>Click a link to test each example demonstrated in this chapter</p>
14    <p>
15    <a href = "/jhtp6/servlets/WelcomeServlet.html">
16       WelcomeServlet</a><br />
17    <a href = "/jhtp6/servlets/WelcomeServlet2.html">
18       WelcomeServlet2</a><br />
19    <a href = "/jhtp6/servlets/WelcomeServlet3.html">
20       WelcomeServlet3</a><br />
21    <a href = "/jhtp6/servlets/RedirectServlet.html">
22       RedirectServlet</a><br />
23    <a href = "/jhtp6/servlets/Survey.html">
24       SurveyServlet</a><br />
25    </p>
26 </body>
27 </html>
```

index.html

(1 de 2)

Linhas 15-24

Fornecem links para testar todos os exemplos demonstrados neste capítulo

Resumo

index.html

(2 de 2)

Saída do programa