Programação para Web I
Aula 8

Profa. Silvia Bertagnolli

Servlets

Introdução

Páginas Web

Páginas estáticas: uma página estática é aquela que, independentemente da situação sempre exibirá o mesmo conteúdo

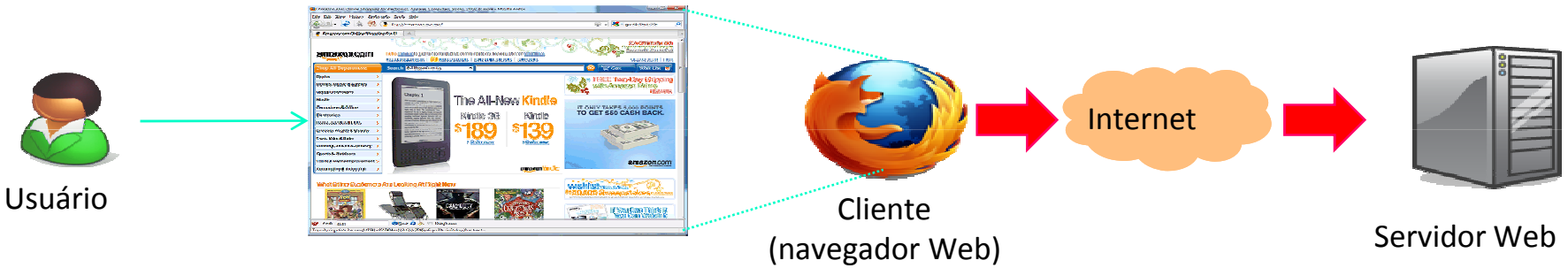
Note que a formatação pode variar conforme modificamos os estilos vinculados (CSS)

Já o conteúdo é fixo ele só muda se editarmos a página

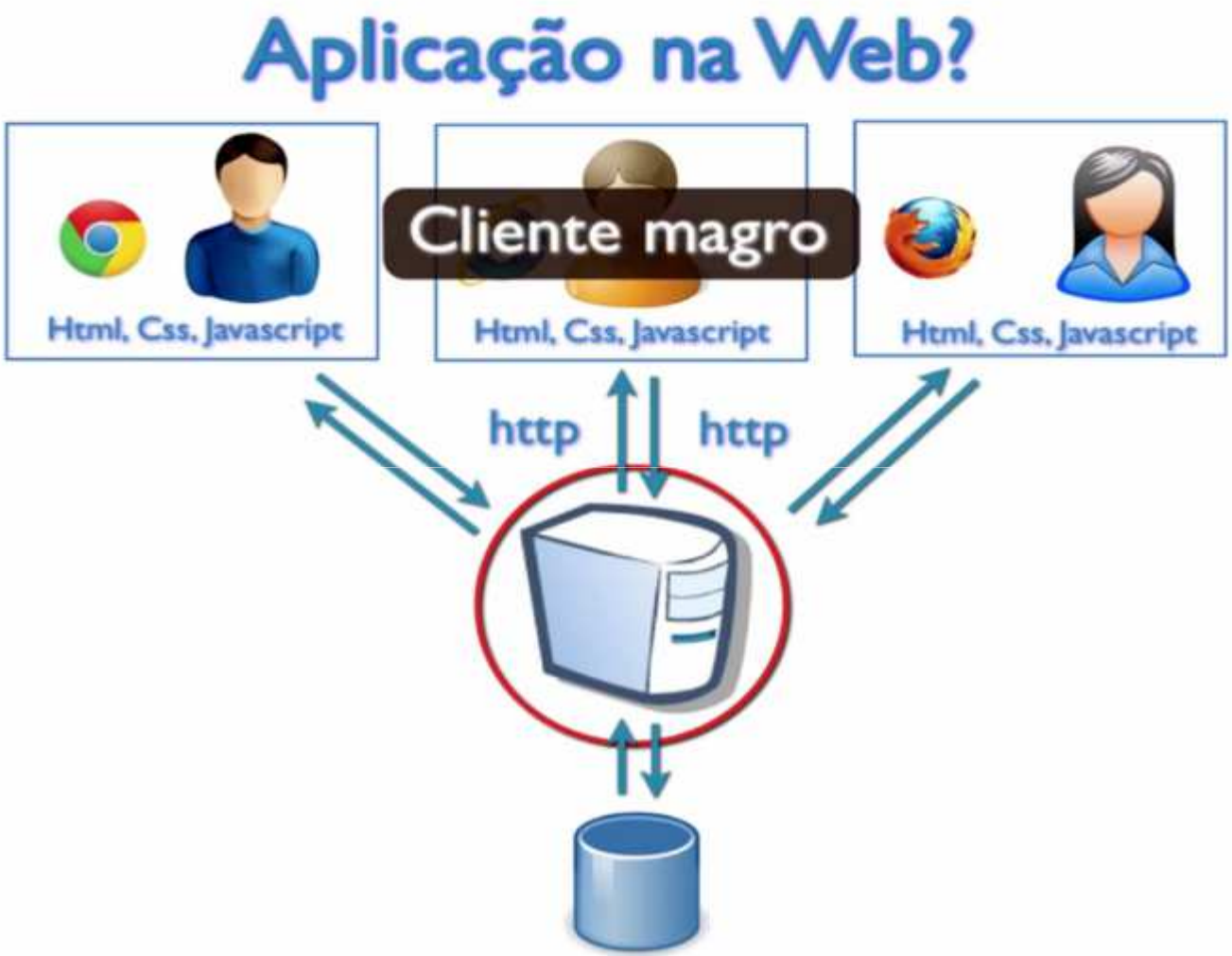
Páginas Web: estáticas

Página dinâmica: uma página é dinâmica quando o seu conteúdo sofre alterações, em um servidor, no momento em que a página é “ativada” no navegador

Arquitetura Web



Cliente Magro



Tecnologias para Páginas Dinâmicas

CGI

Plataforma Microsoft: ASP

Plataforma Java: Servlets, JSP, JSF, ...

PHP

Perl

Entre outras

Servlet: o que é?

São componentes, pequenos programas em Java, que geram páginas visualizadas pelo navegador

São classes Java que são executadas em um Servlet Contêiner:

Tomcat

GlassFish

JBoss

Outros

Vantagens Servlets

Baixo consumo de memória

Uma instância atende diversas requisições

Possibilita reutilização de recursos caros

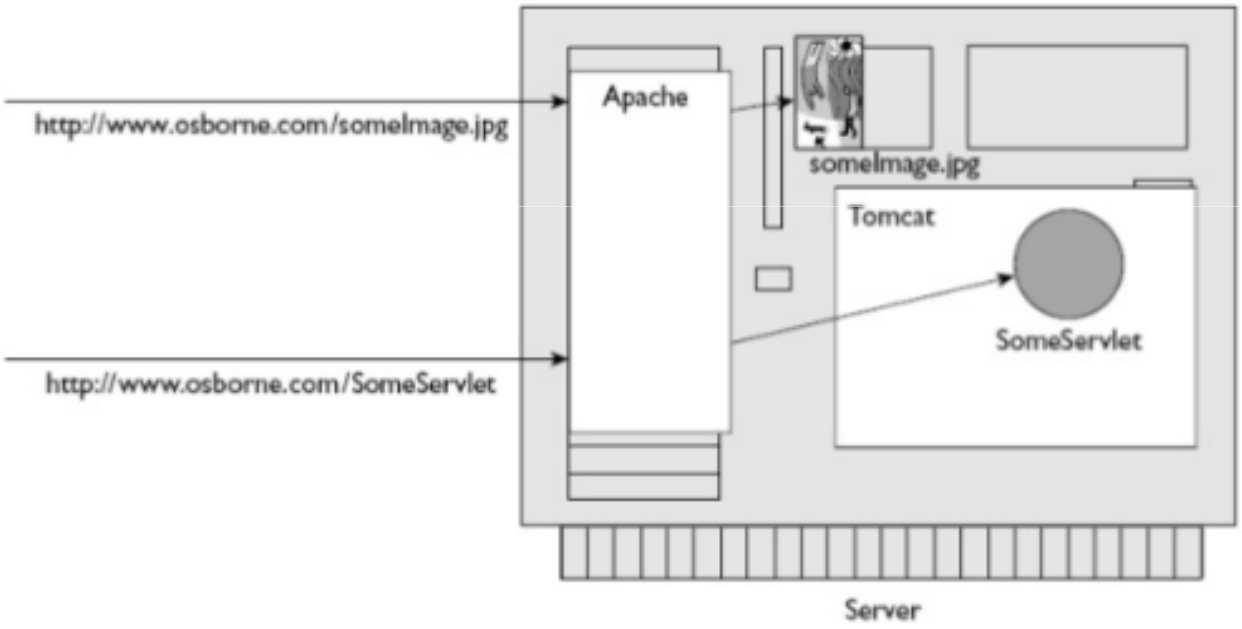
Grande velocidade de execução

Independente de navegador

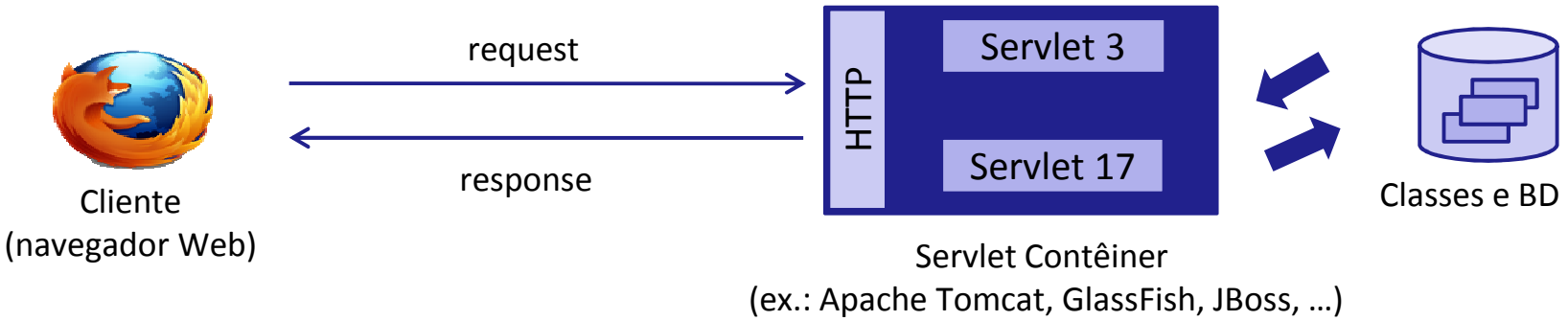
Gerenciamento de sessão

Robustez, segurança, multiplataforma, possui praticamente toda a plataforma Java disponível

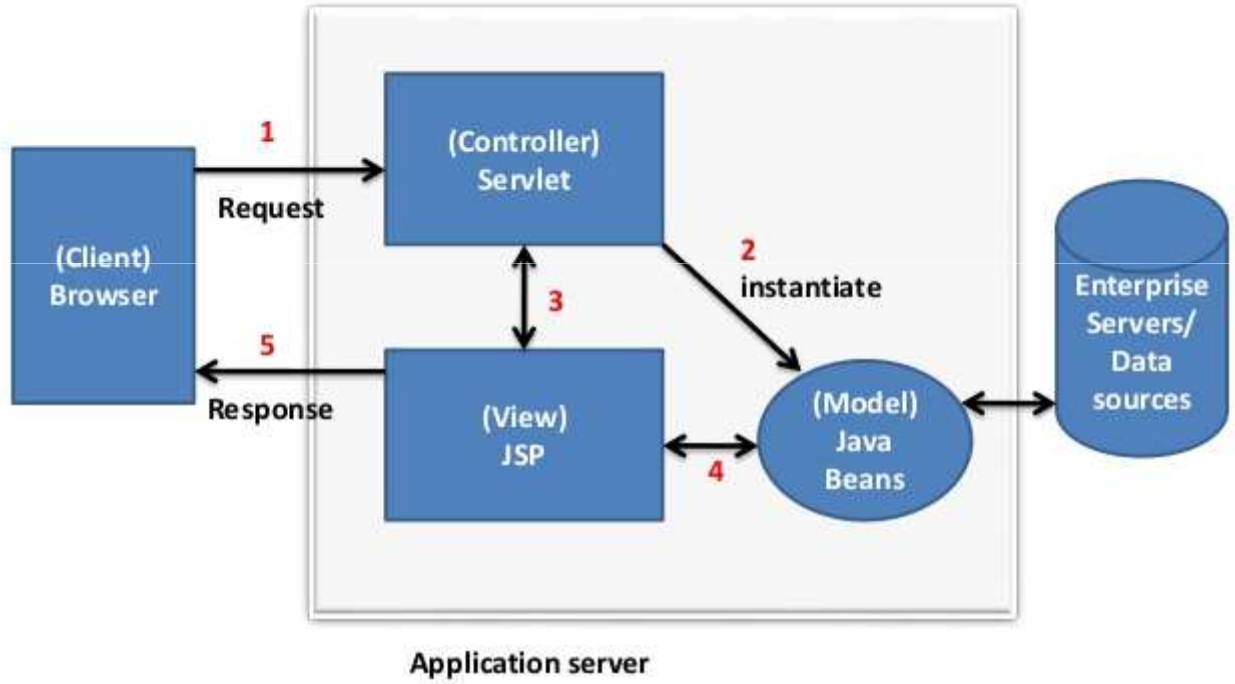
WebServer x Servlet Container



Modelo requisição/resposta



MVC com Servlets



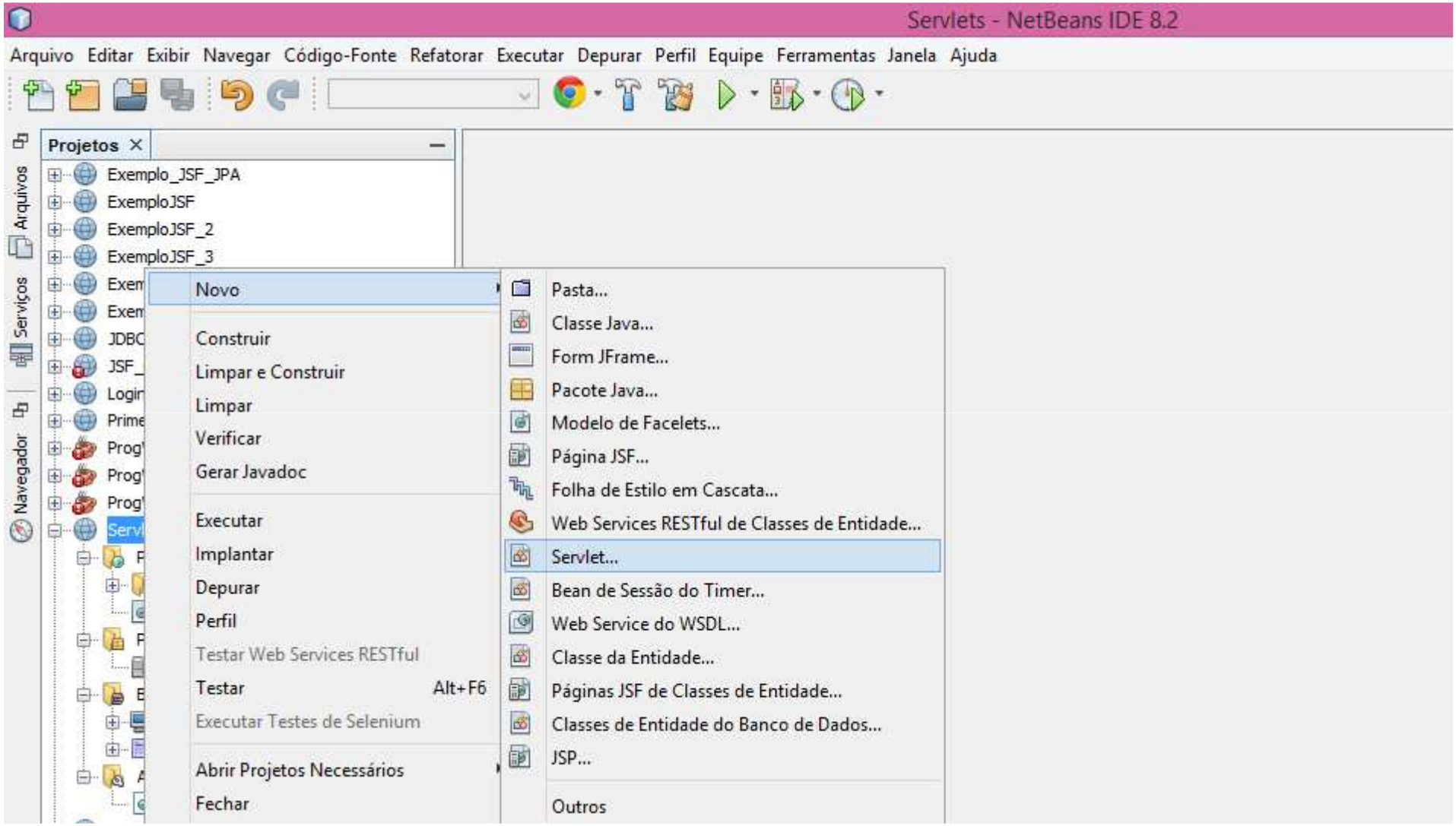
Configurando

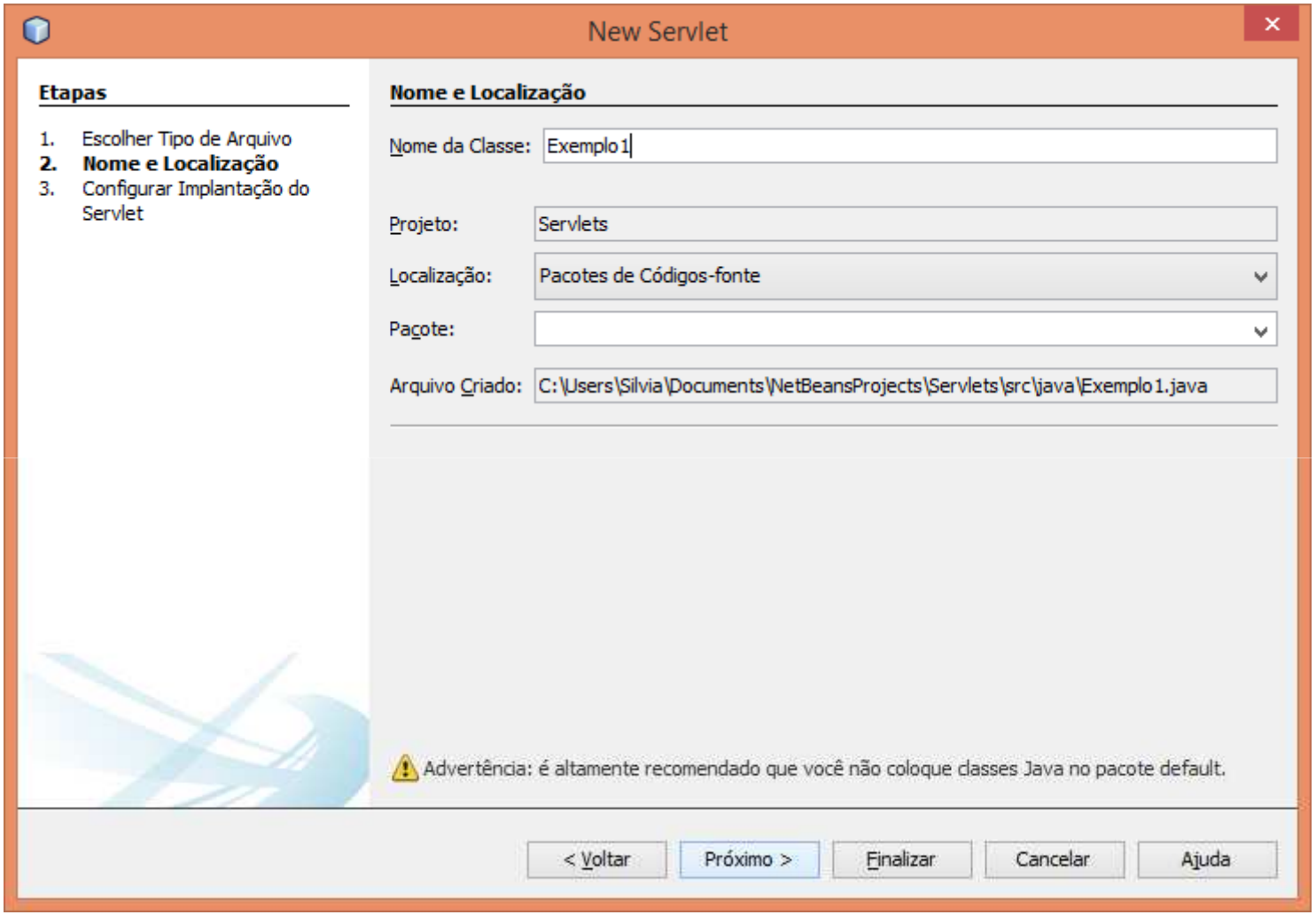
Exemplo 1

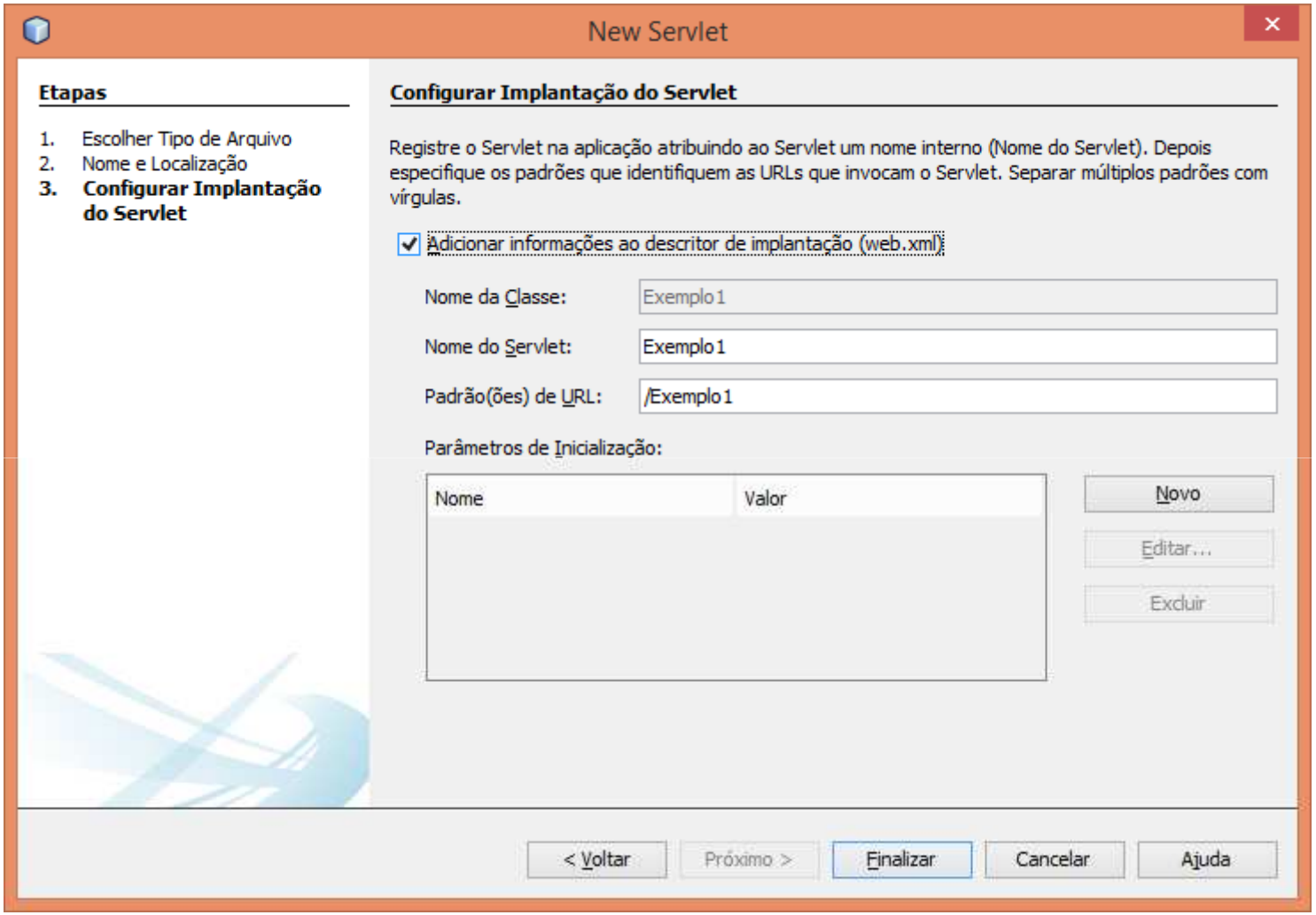
Criar um projeto Java Web -> Aplicação Web com o nome: Servlets

Criar um servlet com o nome: Exemplo1

Observe que ao criar esse servlet o NetBeans criou o arquivo web.xml automaticamente


Servlets - NetBeans IDE 8.2
Arquivo Editar Exibir Navegar Código-Fonte Refatorar Executar Depurar Perfil Equipe Ferramentas Janela Ajuda
Projetos
Arquivos
Serviços
Navegador
Exemplo_JSF_JPA
ExemploJSF
ExemploJSF_2
ExemploJSF_3
Novo
Construir
Limpar e Construir
Limpar
Verificar
Gerar Javadoc
Executar
Implantar
Depurar
Perfil
Testar Web Services RESTful
Testar Alt+F6
Executar Testes de Selenium
Abrir Projetos Necessários
Fechar
Pasta...
Classe Java...
Form JFrame...
Pacote Java...
Modelo de Facelets...
Página JSF...
Folha de Estilo em Cascata...
Web Services RESTful de Classes de Entidade...
Servlet...
Bean de Sessão do Timer...
Web Service do WSDL...
Classe da Entidade...
Páginas JSF de Classes de Entidade...
Classes de Entidade do Banco de Dados...
JSP...
Outros


New Servlet
Etapas
1. Escolher Tipo de Arquivo
2. Nome e Localização
3. Configurar Implantação do Servlet
Nome e Localização
Nome da Classe: Exemplo1
Projeto: Servlets
Localização: Pacotes de Códigos-fonte
Pacote:
Arquivo Criado: C:\Users\Silvia\Documents\NetBeansProjects\Servlets\src\java\Exemplo1.java
Advertência: é altamente recomendado que você não coloque classes Java no pacote default.
< Voltar
Próximo >
Finalizar
Cancelar
Ajuda


New Servlet
Etapas
1. Escolher Tipo de Arquivo
2. Nome e Localização
3. Configurar Implantação do Servlet
Configurar Implantação do Servlet
Registre o Servlet na aplicação atribuindo ao Servlet um nome interno (Nome do Servlet). Depois especifique os padrões que identifiquem as URLs que invocam o Servlet. Separar múltiplos padrões com vírgulas.
Adicionar informações ao descritor de implantação (web.xml)
Nome da Classe:
Exemplo1
Nome do Servlet:
Exemplo1
Padrão(ões) de URL:
/Exemplo1
Parâmetros de Inicialização:
Nome
Valor
Novo
Editar...
Excluir
< Voltar
Próximo >
Finalizar
Cancelar
Ajuda

Arquivo web.xml

Arquivo que faz o mapeamento entre URLs e Servlets

Define configurações de segurança, eventos, filtros, entre outras

Arquivo também é chamado de: deployment descriptor

Arquivo web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <servlet>
        <servlet-name>Exemplo1</servlet-name>
        <servlet-class>Exemplo1</servlet-class>
    </servlet>
```

Arquivo web.xml

```
<servlet-mapping>
        <servlet-name>Exemplo1</servlet-name>
        <url-pattern>/Exemplo1</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

Arquivo web.xml

```
<servlet>
        <servlet-name>Exemplo1</servlet-name>
        <servlet-class>Exemplo1</servlet-class>
</servlet>
```

Nomenclatura: Como servlets são classes Java recomenda-se que seus nomes sigam a convenção de nomes de classes

<servlet>: configura uma instância de um servlet

<servlet-name> e <servlet-class>: são elementos obrigatórios

Arquivo web.xml

```
<servlet-mapping>
        <servlet-name>Exemplo1</servlet-name>
        <url-pattern>/Exemplo1</url-pattern>
</servlet-mapping>
```

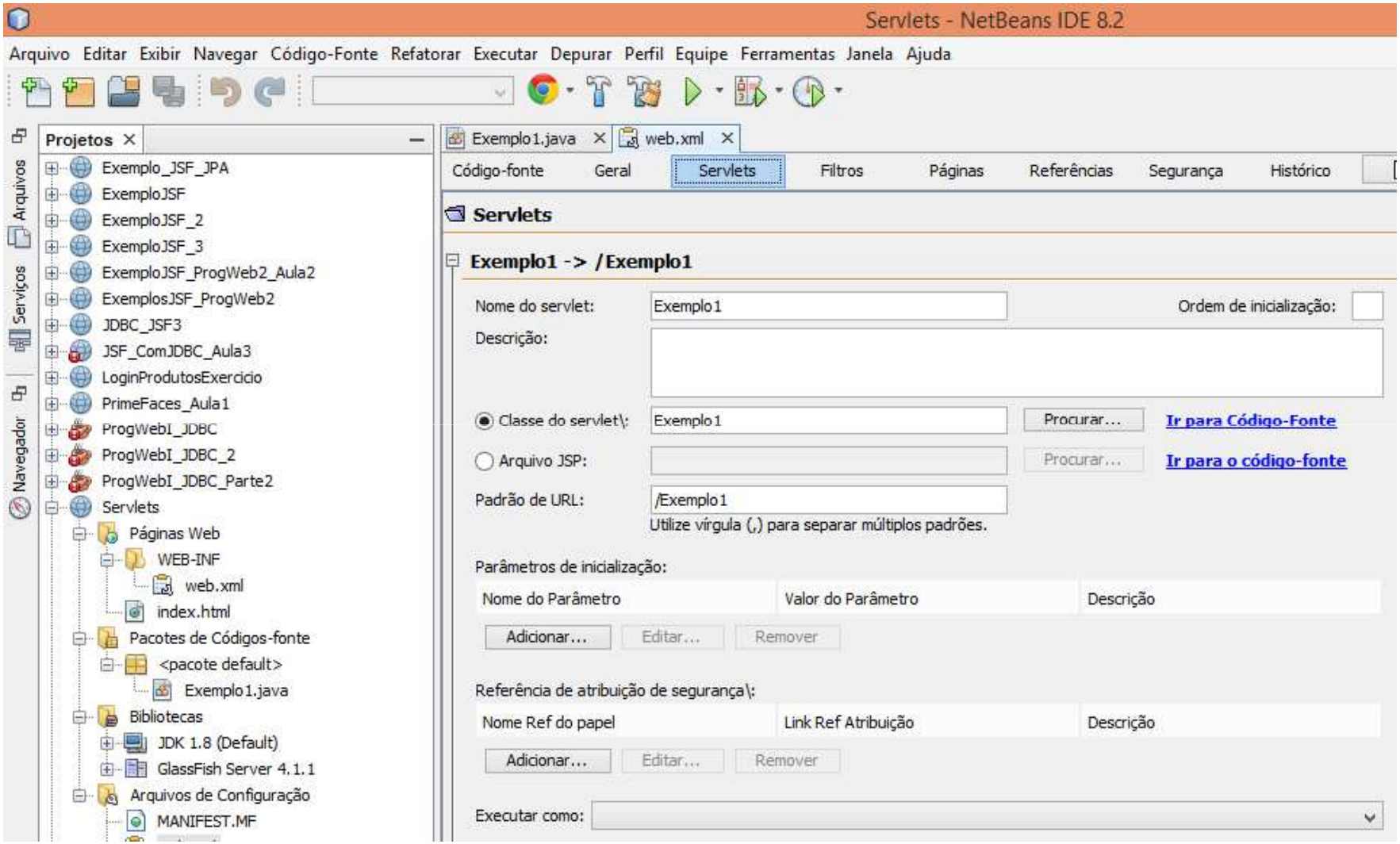
<servlet-mapping>: associa o nome do servlet a um padrão de URL relativo ao contexto

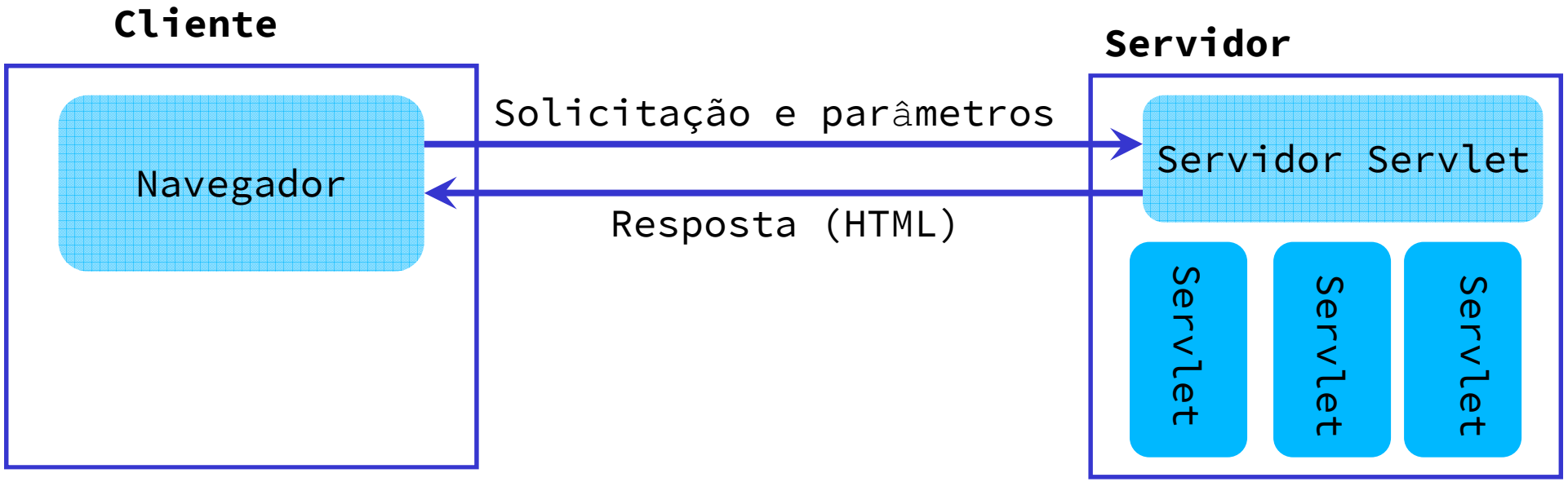
A URL pode ser:
- Um caminho relativo ao contexto iniciando por /
- uma extensão de arquivo, por exemplo, *.jsp. Neste caso, os arquivos com a extensão serão redirecionados ao servlet

NetBeans: arquivo web.xml

Abrindo o arquivo ele irá mostrar uma janela com várias abas superiores

Uma destas abas é a Servlets, nela você pode informar o nome do Servlet que iremos criar


Servlets - NetBeans IDE 8.2
Arquivo Editar Exibir Navegar Código-Fonte Refatorar Executar Depurar Perfil Equipe Ferramentas Janela Ajuda
Projetos
Exemplo_JSF_JPA
ExemploJSF
ExemploJSF_2
ExemploJSF_3
ExemploJSF_ProgWeb2_Aula2
ExemplosJSF_ProgWeb2
JDBC_JSF3
JSF_ComJDBC_Aula3
LoginProdutosExercicio
PrimeFaces_Aula1
ProgWebI_JDBC
ProgWebI_JDBC_2
ProgWebI_JDBC_Parte2
Servlets
Páginas Web
WEB-INF
web.xml
index.html
Pacotes de Códigos-fonte
<pacote default>
Exemplo1.java
Bibliotecas
JDK 1.8 (Default)
GlassFish Server 4.1.1
Arquivos de Configuração
MANIFEST.MF
Exemplo1.java
web.xml
Código-fonte
Geral
Servlets
Filtros
Páginas
Referências
Segurança
Histórico
Servlets
Exemplo1 -> /Exemplo1
Nome do servlet:
Exemplo1
Ordem de inicialização:
Descrição:
Classe do servlet\:
Exemplo1
Procurar...
Ir para Código-Fonte
Arquivo JSP:
Procurar...
Ir para o código-fonte
Padrão de URL:
/Exemplo1
Utilize vírgula (,) para separar múltiplos padrões.
Parâmetros de inicialização:
Nome do Parâmetro
Valor do Parâmetro
Descrição
Adicionar...
Editar...
Remover
Referência de atribuição de segurança\:
Nome Ref do papel
Link Ref Atribuição
Descrição
Adicionar...
Editar...
Remover
Executar como:


Cliente
Navegador
Solicitação e parâmetros
Resposta (HTML)
Servidor
Servidor Servlet
Servlet
Servlet
Servlet

Ciclo de Vida

Ciclo de vida

O ciclo de vida de um servlet é controlado pelo contêiner

Quando o servidor recebe uma requisição, ela é repassada para o contêiner que a delega a um servlet

O contêiner realiza as etapas abaixo:

- Carrega a classe na memória
- Cria uma instância da classe do servlet
- Inicializa a instância chamando o método init()

Ciclo de vida

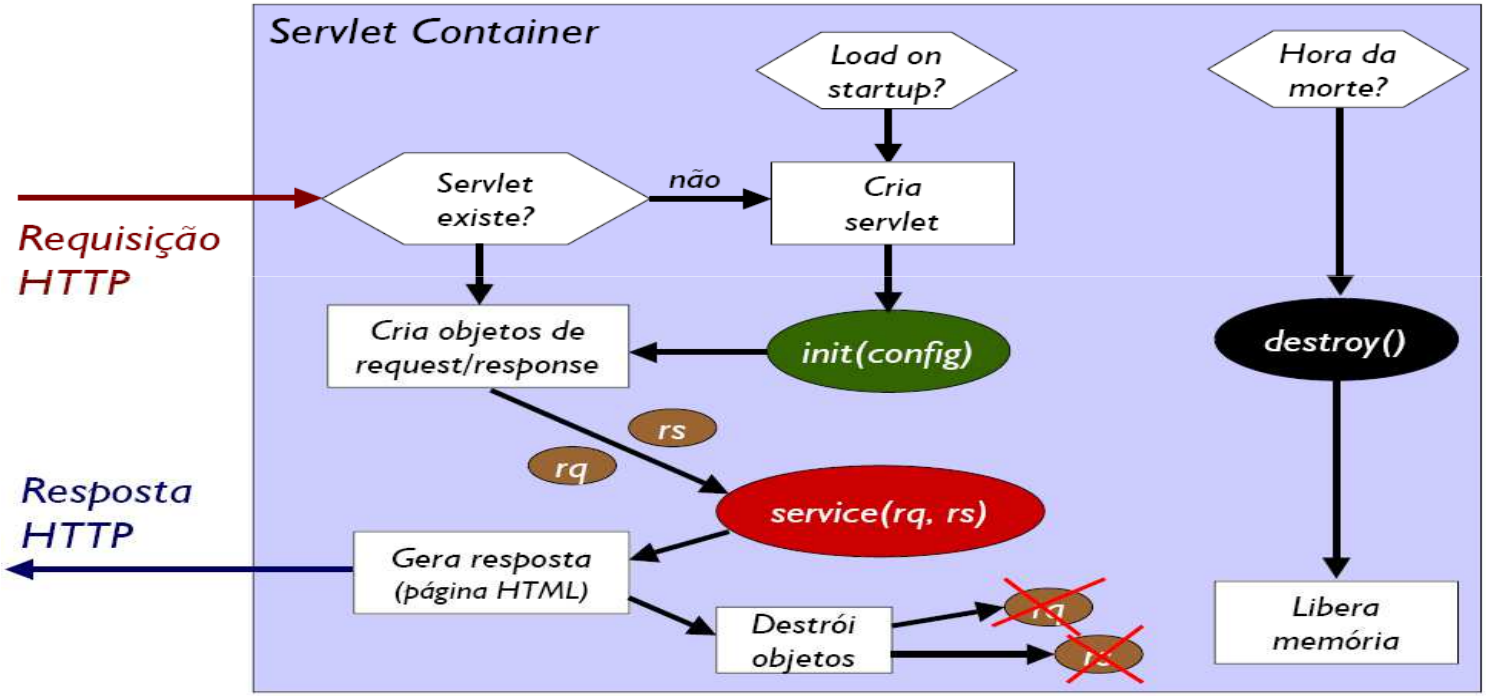
Depois que o servlet foi inicializado, cada requisição é executada em um método service()

O contêiner cria um objeto de requisição (ServletRequest) e de resposta (ServletResponse) e depois chama service() passando os objetos como parâmetros

Quando a resposta é enviada, os objetos são destruídos

Quando o contêiner decide remover o servlet da memória, ele o finaliza chamando destroy()

Ciclo de vida



[Argonavis]

Ciclo de Vida

Método	Descrição
void init(ServletConfig)	Chamado pelo contêiner para iniciar o servlet. Usado para carregar parâmetros de inicialização, dados de configuração, etc.
void service(ServletRequest, ServletResponse)	Invocado pelo contêiner para atender requisições dos clientes. Ele intercepta as chamadas e delega aos métodos correspondentes (doGet, doPost, ...)
void destroy( )	Invocado pelo container quando decide descarregar o servlet.

API Servlets

API: Fundamental

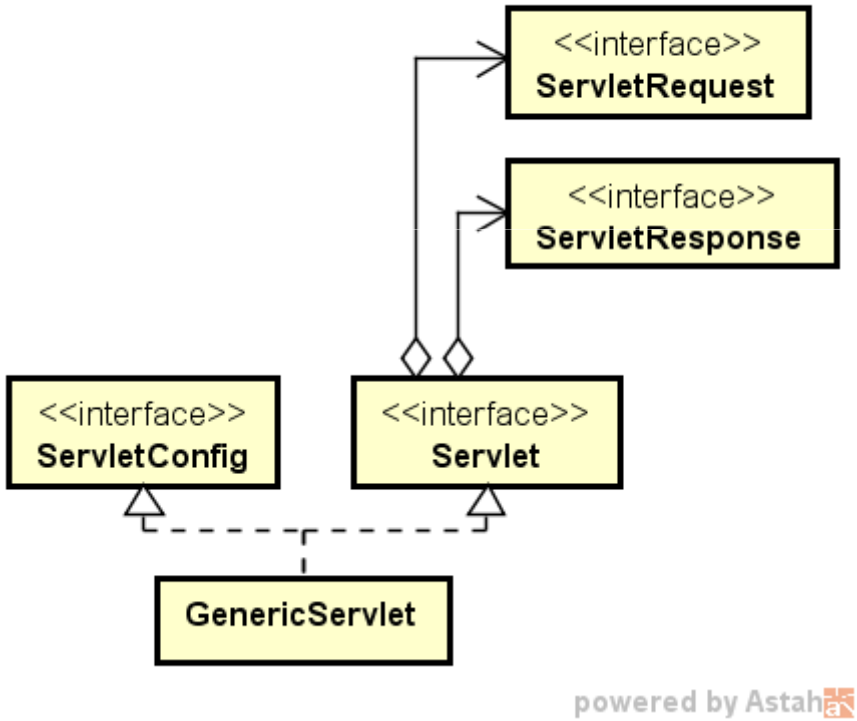
Principais classes e interfaces do pacote **javax.servlet**

Interfaces - Servlet, ServletConfig, ServletContext, Filter, FilterChain, FilterConfig, ServletRequest, ServletResponse, SingleThreadModel RequestDispatcher

Classe Abstrata – GenericServlet

Classes Concretas - ServletException, UnavailableException, ServletInputStream e ServletOutputStream

API: Fundamental



Interfaces e Funcionalidades

Interface	Descrição
RequestDispatcher	Cria um objeto que após receber uma solicitação do usuário a envia para um outro componente da aplicação como: outro servlet ou página JSP
Servlet	Define os métodos que um servlet irá implementar
ServletConfig	Define um objeto que receberá do contêiner informações que serão usadas para inicialização
ServletContext	Define métodos que o servlet usa para troca de informações com o contêiner
ServletRequest	Implementa um objeto que receberá as solicitações do usuário
ServletResponse	Implementa um objeto que enviará as respostas das solicitações do usuário

Classes e Funcionalidades

Classe	Descrição
GenericServlet	Define métodos genéricos para a implementação de servlets
ServletException	Classe de exceção genérica usada quando o servlet encontrar problemas
UnavailableException	Classe de exceção que é ativada quando o servlet estiver indisponível

Classes e Métodos

Classe	Método	Descrição
Servlet	getServletConfig	Retorna os parâmetros de inicialização do servlet
Servlet	getServletInfo	Retorna informações sobre o servlet
ServletConfig	getServletContext	Retorna o contexto do servlet
ServletConfig	getServletName	Retorna o nome de instância do servlet
ServletContext	getServerInfo	Retorna o nome e a versão do contêiner servlet
ServletContext	getServletContextName	Retorna o nome da aplicação na qual o servlet é executado

Exercícios

Exercícios

Faça as questões 1 a 3 da lista de exercícios.

ServletConfig

ServletConfig

A interface ServletConfig possibilita o acesso do servlet às informações que estão no web.xml

Todo servlet implementa ServletConfig e tem acesso aos seus métodos

Os métodos dessa interface devem ser chamados no método init() do servlet

Arquivo web.xml

```
<servlet>
    <servlet-name>Exemplo1</servlet-name>
    <servlet-class>Exemplo1</servlet-class>
    <init-param>
       <param-name>urlBD</param-name>
       <param-value>jdbc:mysql://localhost:3306/bd</param-
value>
    </init-param>
</servlet>
```

Parâmetros de inicialização possuem 2 sub-elementos que definem o nome do atributo e o seu valor

Obtendo dados do Web.xml

```
public void init(ServletConfig config) throws ServletException {
    urlBd = config.getInitParameter(“urlBD");
    if (urlBD == null) {
        throw new UnavailableException(
             "Configuração incorreta!");
    }
    ...
}
```

Exemplo 2

```
out.println("<h2> Parâmetro urlBD:
       "+getServletConfig().getInitParameter("urlBD")+"</h2>");


<servlet>
    <servlet-name>Exemplo2</servlet-name>
    <servlet-class>Exemplo2</servlet-class>
      <init-param>
        <param-name>urlBD</param-name>
        <param-value>jdbc:mysql://localhost:3306/bd</param-
value>
      </init-param>
</servlet>
```

Exercícios

Exercícios

Faça a questão 4 da lista de exercícios.

Servlet HTTP

API: Servlet HTTP

São extensões para servidores Web

Estendem javax.servlet.http.HttpServlet

Abordam características típicas do HTTP:

- Métodos GET e POST
- Cookies
- Sessões

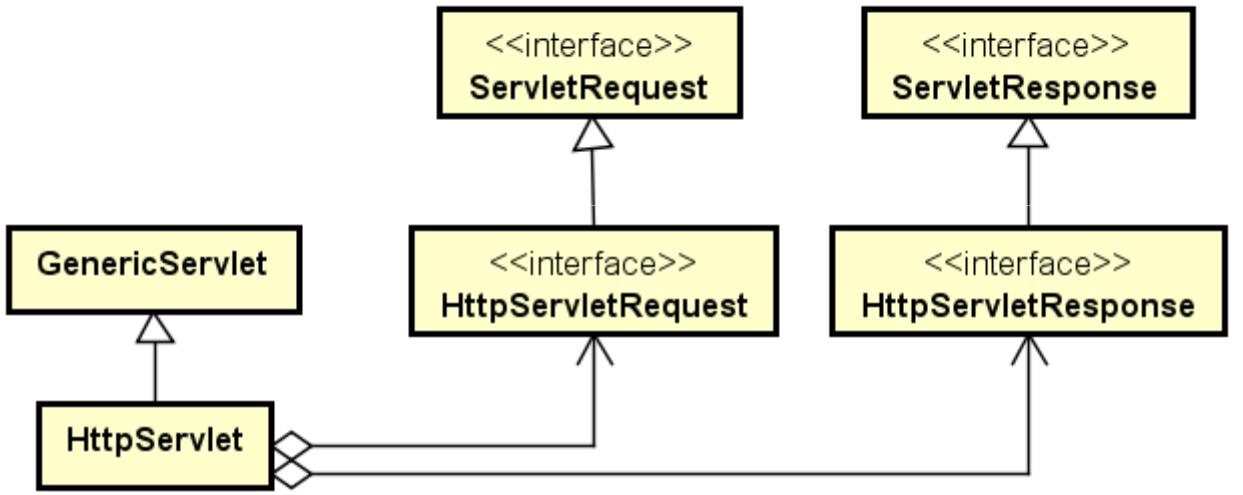
API: Servlet HTTP

Interfaces: HttpServletRequest, HttpServletResponse e HttpSession

Classes abstratas: HttpServlet

Classes concretas: Cookie

API: Servlet HTTP



API: Servlet HTTP

Para criar um servlet HTTP é necessário que a classe seja subclasse de HttpServlet e que implemente um dos métodos doPost() ou doGet()

- doGet(HttpServletRequest, HttpServletResponse)

- doPost(HttpServletRequest, HttpServletResponse)

A inicialização desse tipo de servlet deve ser realizada no método init()

Exemplo1

Exemplo 1

```
public class Exemplo1 extends HttpServlet {
protected void processRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    response.setContentType("text/html;charset=UTF-8");
   try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>"); out.println("<head>");
            out.println("<title>Servlet Exemplo1</title>");
            out.println("</head>"); out.println("<body>");
            out.println("<h1>Servlet Exemplo1 at " +
                        request.getContextPath() + "</h1>");
            out.println("</body>"); out.println("</html>");
        }
```

Exemplo 1

Classe é um servlet

```
public class Exemplo1 extends HttpServlet {
protected void processRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    response.setContentType("text/html;charset=UTF-8");
   try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>"); out.println("<head>");
            out.println("<title>Servlet Exemplo1</title>");
            out.println("</head>"); out.println("<body>");
            out.println("<h1>Servlet Exemplo1 at " +
                        request.getContextPath() + "</h1>");
            out.println("</body>"); out.println("</html>");
        }
```

Exemplo 1

Requisição e Resposta são objetos

```
public class Exemplo1 extends HttpServlet {
protected void processRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    response.setContentType("text/html;charset=UTF-8");
   try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>"); out.println("<head>");
            out.println("<title>Servlet Exemplo1</title>");
            out.println("</head>"); out.println("<body>");
            out.println("<h1>Servlet Exemplo1 at " +
                        request.getContextPath() + "</h1>");
            out.println("</body>"); out.println("</html>");
        }
```

Exemplo 1

Obtém objeto para escrita - usando response

```
public class Exemplo1 extends HttpSer      {
protected void processRequest(HttpSer    tRequest request,
HttpServletResponse response) throws     vletException,
IOException {
    response.setContentType("text/html;charset=UTF-8");
   try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>"); out.println("<head>");
            out.println("<title>Servlet Exemplo1</title>");
            out.println("</head>"); out.println("<body>");
            out.println("<h1>Servlet Exemplo1 at " +
                        request.getContextPath() + "</h1>");
            out.println("</body>"); out.println("</html>");
        }
```

Exemplo 1

```
public class Exemplo1 extends HttpServlet {
protected void processRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    response.setContentType("text/html;charset=UTF-8");
   try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>"); out.println("<head>");
            out.println("<title>Servlet Exemplo1</title>");
            out.println("</head>"); out.println("<body>");
            out.println("<h1>Servlet Exemplo1 at " +
                        request.getContextPath() + "</h1>");
            out.println("</body>"); out.println("</html>");
        }
```

Pega dados do objeto request

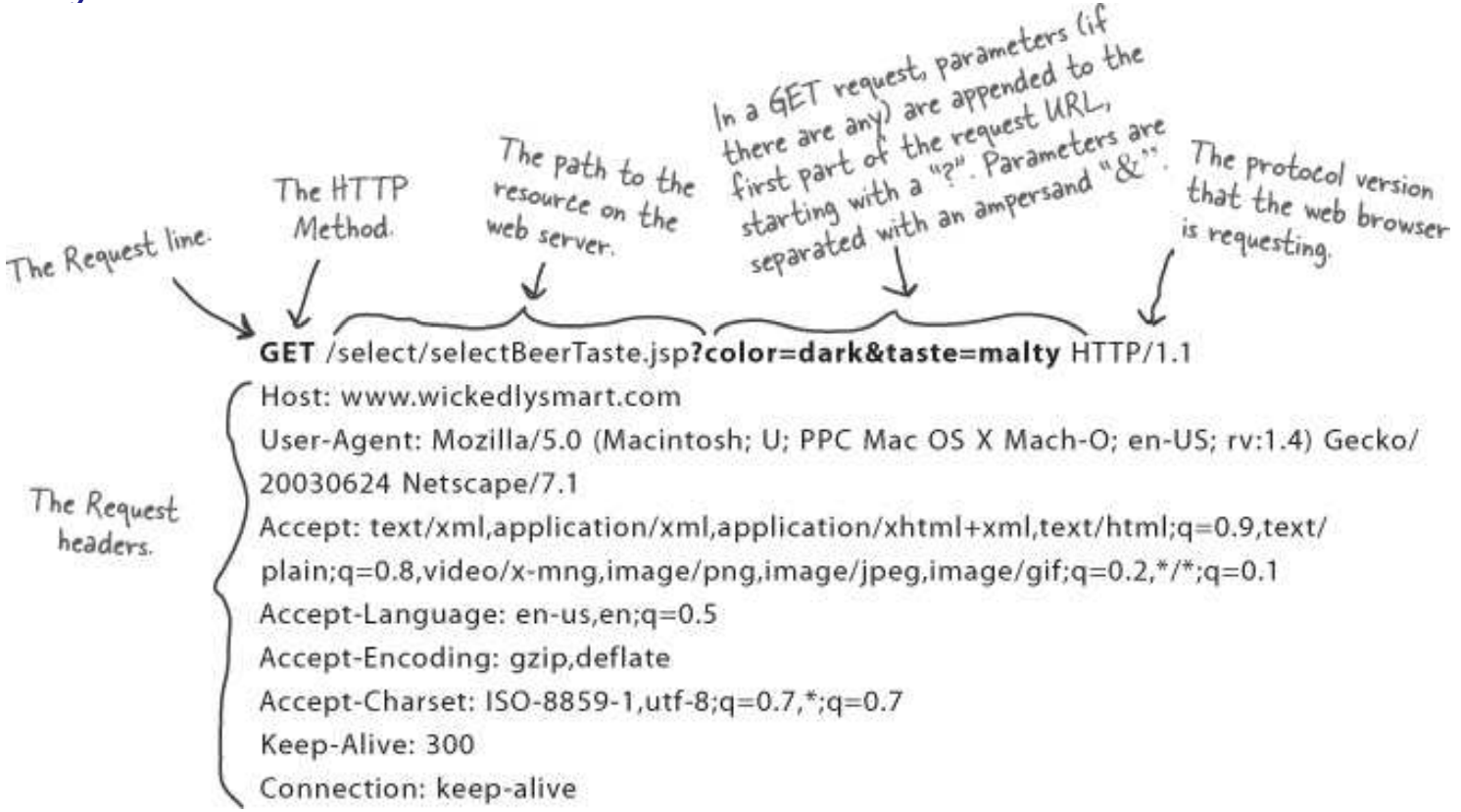
HttpServletRequest

Requisição HTTP

Uma requisição HTTP feita pelo browser

Ela contém vários cabeçalhos e informações, que podem ser extraídas usando os métodos da classe HttpServletRequest

Requisição HTTP



HttpServletRequest: Métodos

Método	Descrição
getParameter(String param)	Retorna o valor do parâmetro informado, se não existir o parâmetro retorna null
getParameterNames	Retorna o nome de todos os parâmetros
getParameterValues	Retorna todos os parâmetros do parâmetro informado
getServerName	Retorna o nome do host que recebe a solicitação
getServerPort	Retorna o número da porta que recebe a solicitação

HttpServletRequest: Métodos

Método	Descrição
getRemoteAddr	Retorna o endereço IP do usuário
getRemoteUser	Retorna o usuário remoto se autenticado, caso contrário retorna null
getCookies	Retorna os cookies do cliente
getSession	Retorna a sessão
getRequestedSessionId()	Retorna o ID da sessão de um usuário
isRequestedSessionIdValid()	Verifica se o ID da sessão ainda é válido
getQueryString()	Retorna o conteúdo da queryString após o endereço
getMethod	Retorna o nome do método HTTP empregado na solicitação

Exercícios

Exercícios

Faça a questão 5 da lista de exercícios.

doGet() e doPost()

Como implementar doGet() e doPost()

```
public class Exemplo1 extends HttpServlet {
protected void processRequest(HttpServletRequest
                  request, HttpServletResponse response)
                  throws ServletException, IOException {//...}
protected void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
```

Método GET

Os parâmetros são passados em uma única linha no query string, que estende a URL após uma “?”

Parâmetros são pares nome=valor que são enviados pelo cliente separados por “&”

Exemplo:

http://localhost:8080/Servlets/Exemplo3?num1=12&num2=13

POST

Os parâmetros são passados como um stream no corpo da mensagem

O cabeçalho Content-length, que existe nas requisições POST informa o tamanho

Exemplo:

```
<form method=“POST” action=“Servlets/Exemplo3”>
<input type=“text” name=“num1”>
<input type=“text” name=“num2”>
<input type=“submit”>
</form>
```

Exemplos: GET x POST

```
int num1 = Integer.parseInt(request.getParameter("num1"));

int num2 = Integer.parseInt(request.getParameter("num2"));
```

O método getParameter() pode ser usado para obter dados que são oriundos de GET ou POST

Exemplo 3

```
out.println("<h2>Num1=“+request.getParameter("num1")+"</h2>");

out.println("<h2>Num2=“+request.getParameter("num2")+”</h2>");
```

Exemplo de URL:

http://localhost:8080/Servlets/Exemplo3?num1=1&num2=3

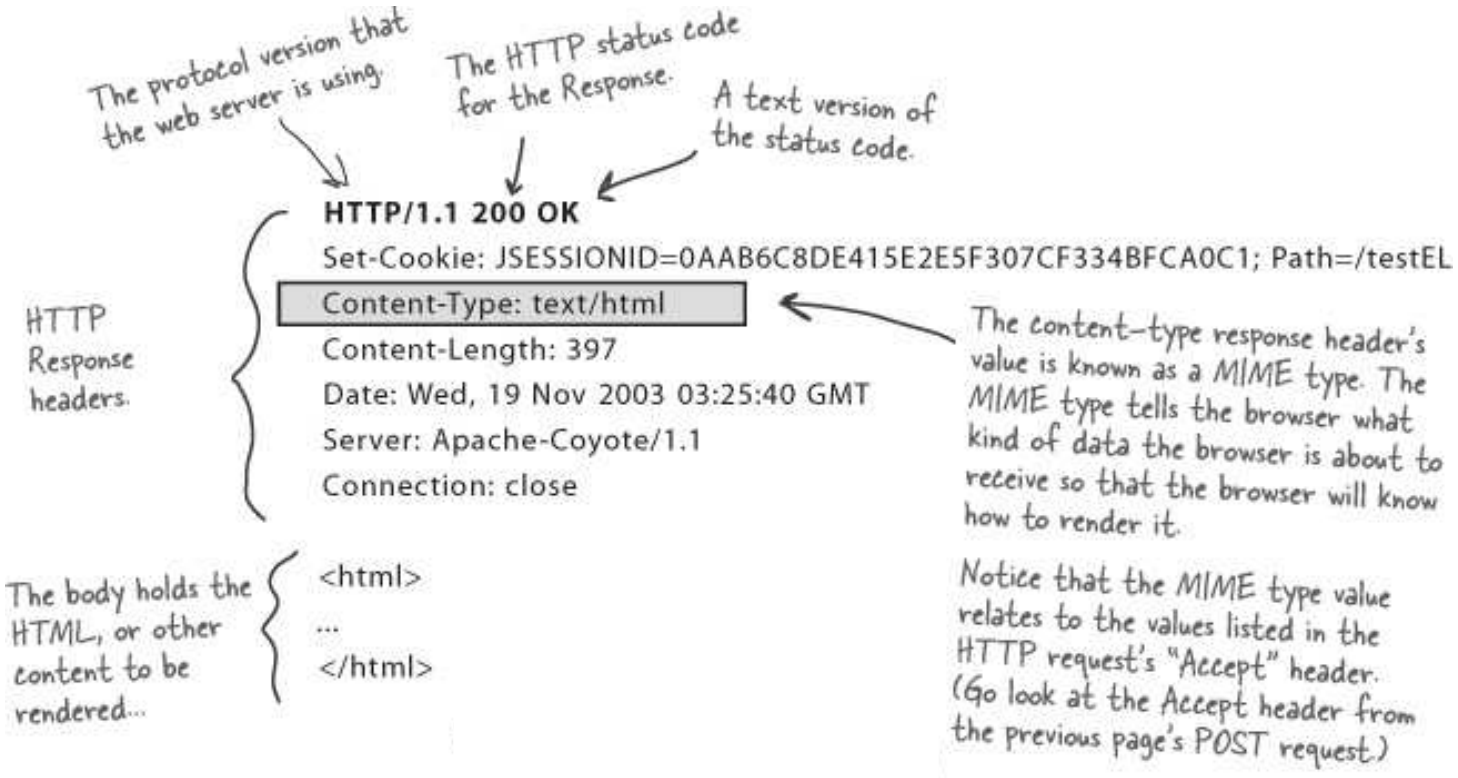
Exercícios

Exercícios

Faça as questões 6 a 8 da lista de exercícios.

HttpServletResponse

HttpServletResponse



Resposta HTTP

Para gerar uma resposta é necessário obter do objeto HttpServletResponse um fluxo de saída

Writer

Fluxo de saída formato caracteres

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

OutputStream

Fluxo de saída formato de bytes

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

Resposta HTTP

Uma resposta HTTP é enviada pelo servidor ao browser

Ela contém vários informações sobre os dados anexados

Os métodos de HttpServletResponse permitem construir um cabeçalho

HttpServletResponse: Métodos

Método	Descrição
addHeader()	Adiciona cabeçalho HTTP
setContentType()	Define o tipo MIME que será usado para gerar a saída (text/html; image/gif, etc.)
sendRedirect()	Envia informação de redirecionamento para o cliente
addCookie()	Adiciona novo cookie
encodeURL()	Encia como anexo da URL a informação de identificador de sessão (sessionid)

Redirecionando Páginas

Redirecionando...

No formCadastro2.html:

```
<form id="formcad" action="Exemplo5" method="POST">
    <p>Informe o seu nome:
       <input type="submit" value="Abrir outra página"/>
    </p>
</form>
```

No servlet Exemplo5:

```
response.sendRedirect("index.html");
```

Redirecionamento com include e forward

Usando RequestDispatcher o controle não volta ao browser, mas continua em outro servlet se usarmos forward() ou no mesmo servlet se for utilizado include()

forward() - repassa a requisição para um recurso

include() - inclui a saída e processamento de um recurso no servlet

Redirecionando Páginas Exemplo6

```
String caminho = "/index.html";

RequestDispatcher disp =

          getServletContext().getRequestDispatcher(caminho);

disp.forward(request, response);
```

Exercícios

Exercícios

Faça as questões 9 e 10 da lista de exercícios.

Sessões

Sessões : Introdução

Como guardar informações entre requisições de um cliente, visto que o HTTP não mantém estado?

Soluções:

- Campos ocultos em formulário (hidden)
- Cookies
- Reescrita de URLs
- Sessões

Sessão

O HTTP não mantém o estado da sessão, logo a aplicação Web deve mantê-lo

A sessão existe enquanto o navegador, que estabeleceu a primeira requisição, estiver aberto

É possível compartilhar objetos na sessão

Sessão: características

Armazenadas no lado do servidor

Possibilidade de se armazenar objetos Java

Cada sessão é única para cada cliente e persiste através de várias requisições

A sessão é representada por um objeto HttpSession e são obtidas através da requisição

Escopos

Contexto da aplicação - existe enquanto a aplicação estiver na memória

Sessão - dura uma sessão do cliente

Requisição - existe durante o tempo que a requisição existir

HttpSession

Classe que representa uma sessão

Exemplo para obter:

HttpSession s = **request.getSession**(boolean create)

create = true – se a sessão não existir, esta é criada e depois retornada; caso exista é, simplesmente, retornada.

create = false – se a sessão não existir, retorna null; caso exista retorna a sessão existente.

HttpSession : Exemplo

```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                                    throws ServletException{
        HttpSession session = request.getSession(true);
        Usuario user = (User) session.getAttribute("usuario");
        if (user != null){
            //Faça alguma coisa
        }else{
            session.invalidate();
            response.sendRedirect("login.jsp");
        }
}
```

HttpSession: Métodos

Método	Descrição
setAttribute(String alias, Object objeto)	Armazena um objeto na sessão corrente. A princípio, só o cliente vinculado à sessão pode ter acesso ao objeto
getAttribute(String alias)	Retorna o objeto armazenado com o alias definido.
removeAttribute(String alias)	Remove o objeto armazenado com o alias definido
setMaxInactiveInterval()	Define quanto tempo a sessão poderá ficar inativa
getMaxInactiveInterval ()	Retorna quanto tempo a sessão poderá ficar inativa

HttpSession: Métodos

Método	Descrição
isNew()	Retorna se uma sessão é válida
getAttributesNames()	Retorna uma enumeração com todos os nomes dos atributos armazenados na sessão
invalidate ()	Invalida a sessão e remove todos os objetos associados a ela

HttpSession

Pode-se definir o Timeout de uma sessão no arquivo descritor web.xml descriptor da aplicação

```
 <web-app>
   …
   <session-config>
       <session-timeout>30</session-timeout>
   </session-config>
   …
<web-app>
```

ID da Sessão

A sessão não é enviada a cada requisição/resposta, pois isso deixaria a aplicação lenta e insegura

Apenas o ID da sessão é transmitido através de um `cookie` usando o nome JSESSIONID

A API de servlets permite o uso de sessões mesmo que o navegador do usuário não suporte cookies

HttpServletResponse