

PROGRAMAÇÃO WEB

Java EE com Java Server Faces

Instrutor

Marcos Henrique Muniz {vexxile@gmail.com}

- ▣ Mestre em Ciência da Computação pela UFMG.
- ▣ Bacharel em Ciência da Computação pelo UniBH.
- ▣ Experiência em análise, desenho e implementação de aplicações Java EE para web.
- ▣ Arquiteto de Software e Sistemas pela Stefanini – maior empresa brasileira de TI.
- ▣ Professor dos cursos de Ciência da Computação, Engenharia de Alimentos e Engenharia Civil pelo UniBH.

Pré-requisitos

- Noções básicas de TI;
- Conhecimento em Java SE;
- Proficiência na língua inglesa;
- **Experiência preliminar em Java e Orientação a Objetos.**

Ementa

- Arquitetura Java EE: Conceitos;
 - ▣ Padrões arquiteturais, padrões de desenho;
 - ▣ Especificações Java EE.
- Servlets e JSP: Conceitos e implementação;
- Arcabouços de desenvolvimento para a web: Estado da arte;
- JSF:
 - ▣ Conceitos, Configuração, JBoss Seam, JSTL, Facelets;

Java EE

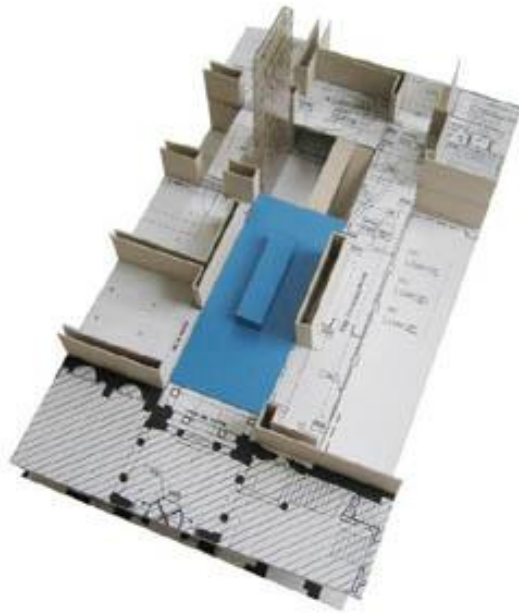
- Ambiente de desenvolvimento:
 - É o conjunto de ferramentas em uso durante o desenvolvimento de um sistema de software.
- JDK (*Java Development Kit*):
 - API (*Application Programming Interface*) base para o desenvolvimento Java.
 - Neste curso: **Java SE 6 – 32 bits** (<http://java.sun.com>).
- IDE (*Integrated Development Environment*):
 - Ambiente de desenvolvimento integrado.
 - Neste curso: **Eclipse Helios for Java EE Developers** (<http://www.eclipse.org>).
- HSQLDB:
 - *Lightweight* 100% Java SQL SGBD (<http://hsqldb.org>).
- Web Server:
 - Servidor onde será implantada uma aplicação para a web.
 - Neste curso: **Tomcat 7.0.x** (<http://tomcat.apache.org>).

Java EE



Arquitetura Java EE

- O que é Arquitetura Java EE?
 - ▣ O que é **Arquitetura**?
 - ▣ O que é **Java EE**?



Arquitetura

- Na construção civil, o que é definido por uma planta arquitetural?
 - ▣ Divisão de espaços? → Componentes?
 - ▣ Medidas? → Especificações?
 - ▣ Tipos de cômodos? → Divisão funcional?
 - ▣ Acessos aos cômodos? → Integração e Comunicação?
 - ▣ ...

Arquitetura

- Em um projeto arquitetônico de uma casa, ao definir que certas portas entre cômodos serão portas de correr embutidas, existe a preocupação em se definir o processo de fabricação destas portas? Ou ainda, quais os componentes mecânicos são necessários para construir uma porta de correr embutida?
- ▣ **Não**, pois o enfoque do projeto arquitetônico é a **funcionalidade** associada a porta – porta de correr embutida.

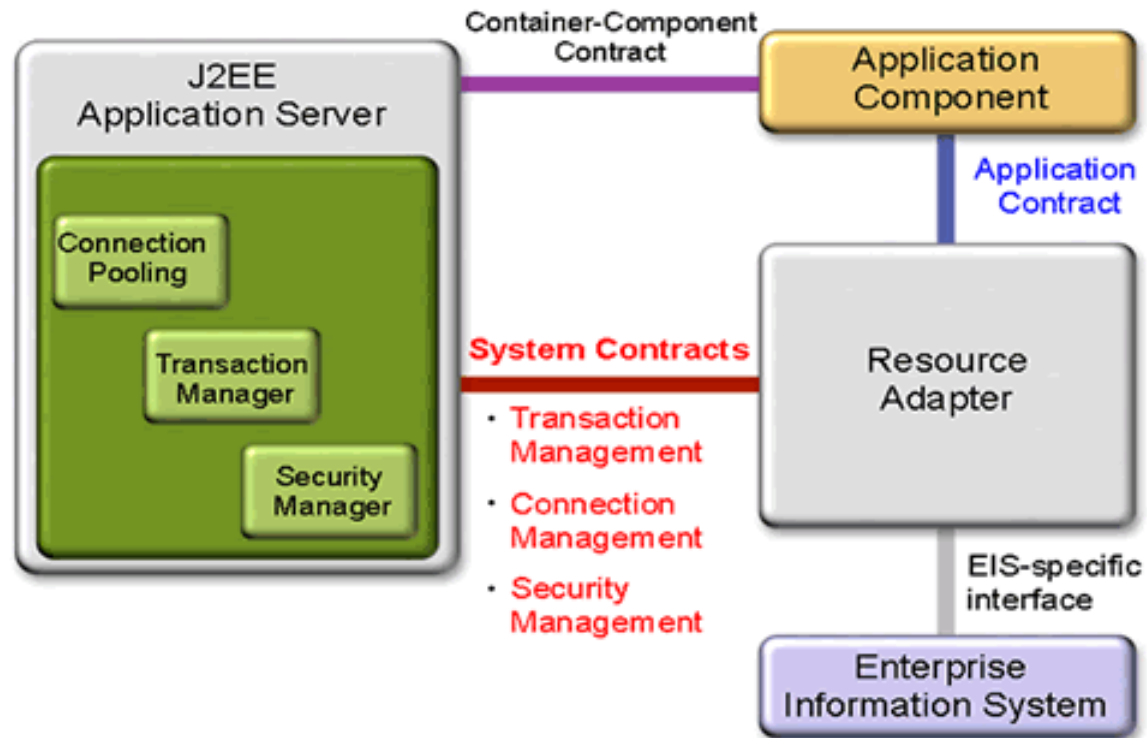
Arquitetura

- Então, o que é Arquitetura?
 - ▣ Arquitetura de software é, ao mesmo tempo, o componente que faz com que as partes de um sistema trabalhem juntas como um todo e com sucesso; e, é o padrão organizacional fundamental de um sistema e seus componentes, seus inter-relacionamentos e relacionamentos com o ambiente e princípios diretores de desenho e evolução [IEEE, 2000].
 - ▣ Arquitetura de software é a estrutura de estruturas de um sistema, cada qual contendo componentes, propriedades externamente visíveis destes componentes, e as suas relações entre si [Clements et al., 2006]. Então, se uma propriedade de um elemento arquitetural não é visível, ou perceptível, a qualquer outro elemento arquitetural, este elemento não é arquitetural.

Arquitetura

- Alguns pontos sobre Arquitetura:
 - ▣ “... o **componente** que faz com que as partes de um sistema trabalhem juntas como um todo e com sucesso...”
- O que seria um componente?
 - ▣ *Component-based development*: ênfase na decomposição de sistemas de software em componentes funcionais ou lógicos com interfaces bem definidas para intercomunicação entre os componentes.
 - ▣ Componentes são considerados um nível de abstração acima ao de objetos em Orientação a Objetos.

Componentes



Arquitetura

- Ainda sobre Arquitetura:
 - “... é o **padrão organizacional** fundamental de um sistema e seus componentes, seus inter-relacionamentos e relacionamentos com o ambiente e princípios diretores de **desenho** e evolução...”
- Está claro o que é padrão organizacional? E desenho?

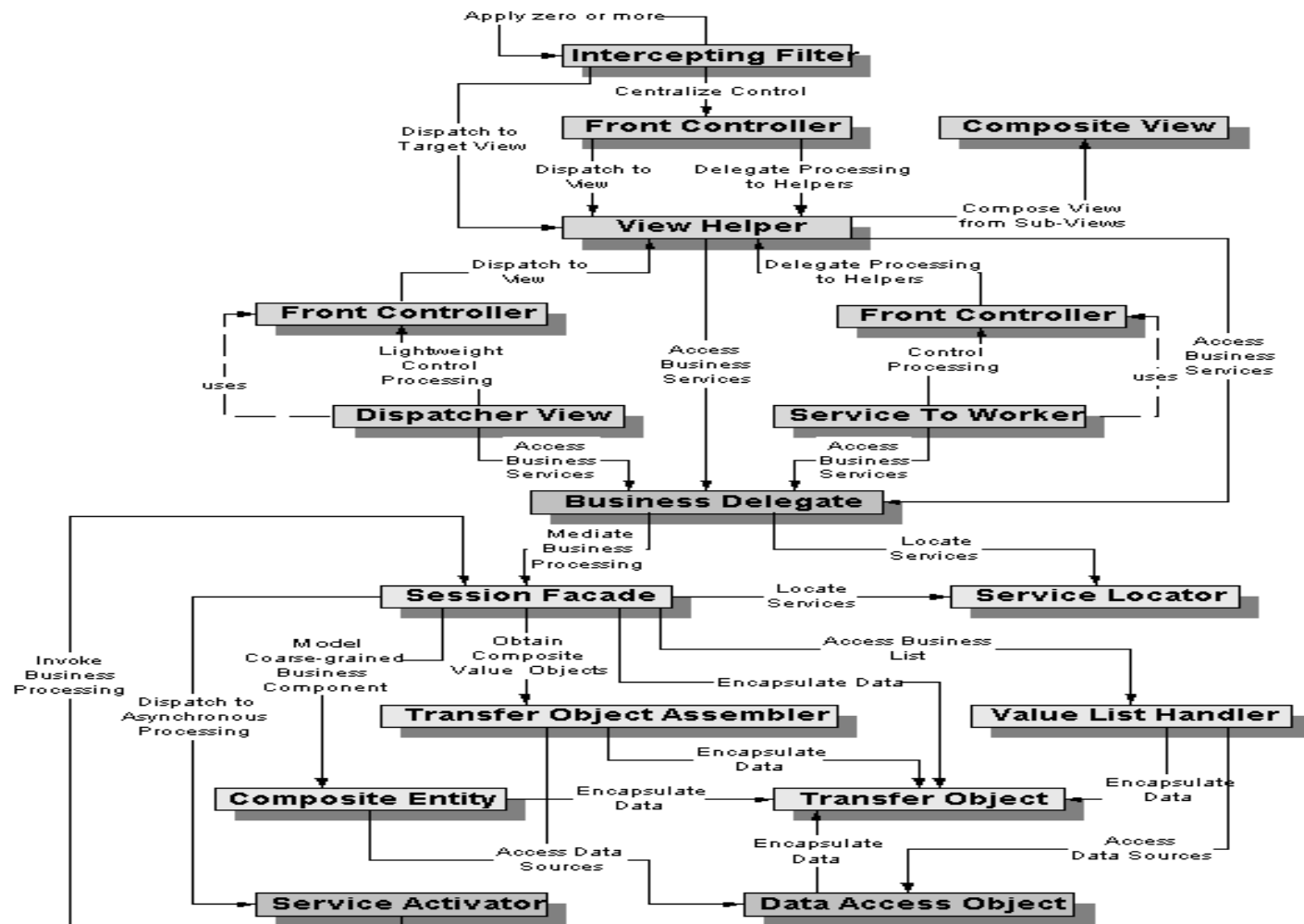
Arquitetura

- O que seria um padrão organizacional em um Arquitetura de software?
 - ▣ Um padrão arquitetônico. Um padrão de **desenho** arquitetônico.
- Padrão de desenho:
 - ▣ “Cada padrão descreve um problema que ocorre inúmeras vezes em nosso ambiente, e então descreve o ponto central da solução para este problema, de maneira que esta solução pode ser usada milhões de vezes durante o passar do tempo, sem que seja empregada da mesma maneira duas vezes” – Christopher Alexander [Alexander et al., 2007].

Arquitetura

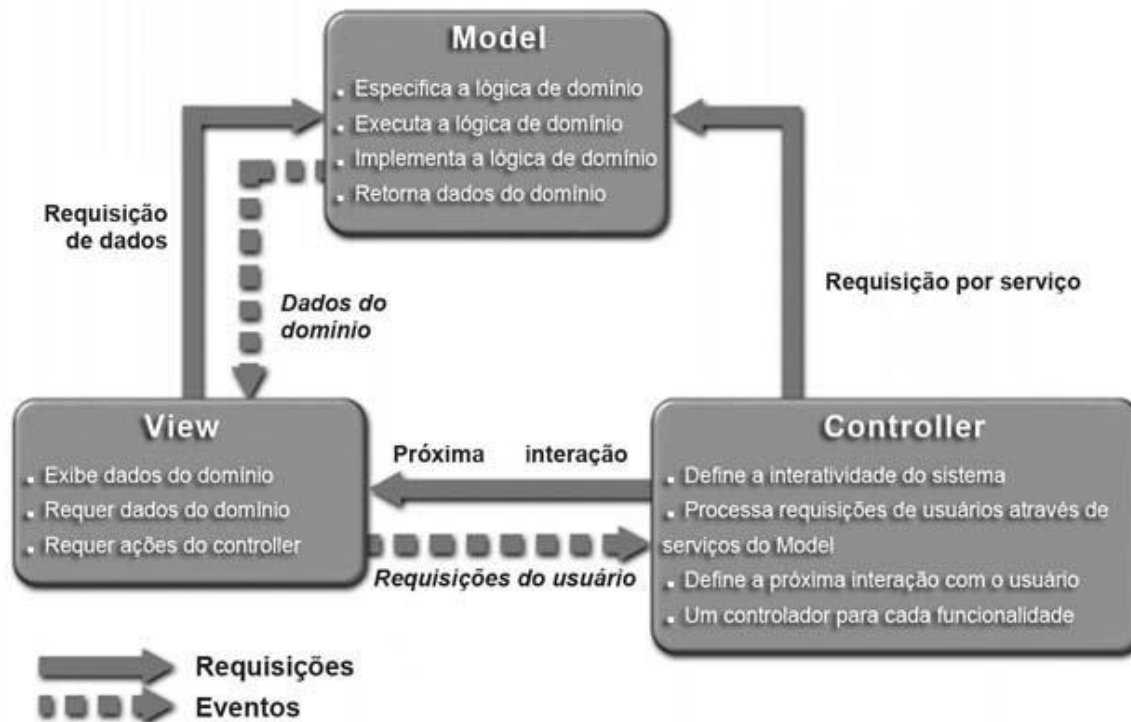
- Em geral padrões de desenho possuem quatro elementos essenciais [Gamma et al., 2003]:
 - ▣ Nome. O nome de um padrão é usado para descrever o problema abordado, sua solução e conseqüências.
 - ▣ Problema. O problema descreve o cenário onde o padrão é aplicável, expondo as questões e contexto envolvidos.
 - ▣ Solução. A solução descreve os elementos de desenho, seus relacionamentos, responsabilidades e colaborações que satisfazem as condições impostas pelo problema. A solução não é uma solução de desenho e implementação em particular, pois um padrão oferece um gabarito aplicável a diferentes situações que retratam o mesmo problema.
 - ▣ Conseqüências. As conseqüências são os resultados e relações de custo-benefício de aplicação do padrão. Essas relações são essenciais para análise de alternativas de desenho e entendimento das conseqüências da aplicação do padrão.

Core Java EE Patterns



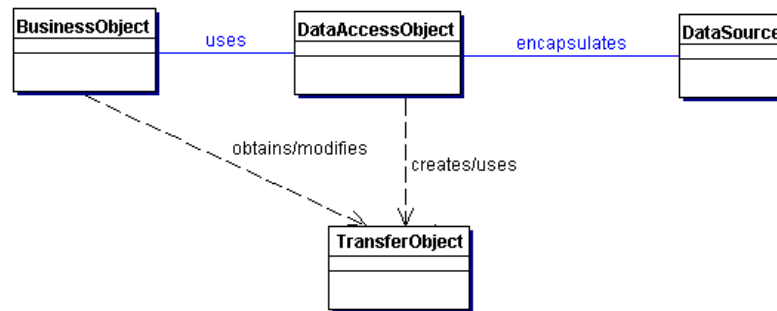
Java EE Patterns

□ Model View Controller (MVC)

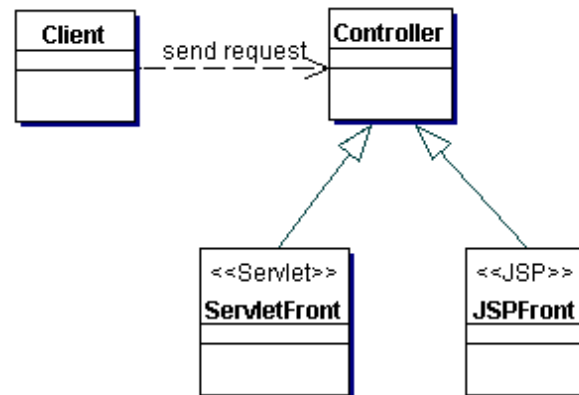


Java EE Patterns

□ Data Access Object



□ Front Controller



Arquitetura

- Conclusão: um padrão de desenho nomeia, cria abstrações e colaborações, e identifica pontos chave a respeito de estruturas de desenho que são úteis durante a criação de desenhos reutilizáveis. O padrão de desenho identifica as classes participantes e suas instâncias, seus papéis e colaborações, e a distribuição de responsabilidades.



- Arquitetura de software é a estrutura de estruturas de um sistema, cada qual contendo componentes, propriedades externamente visíveis destes componentes, e as suas relações entre si [Clements et al., 2006].

Java EE

- E, o que é Java EE?
 - ▣ Plataforma de desenvolvimento Java EE (*Enterprise Edition*).
- Plataforma?
 - ▣ Conjunto de programas correlacionados para o desenvolvimento e execução de programas escritos em Java.
- *Enterprise Edition*?
 - ▣ Conceito associado a funcionalidades como *fault-tolerant*, *distributed applications*, *multi-tier Java software*, *modular components*, executadas em um *application server*.

Especificações Java EE

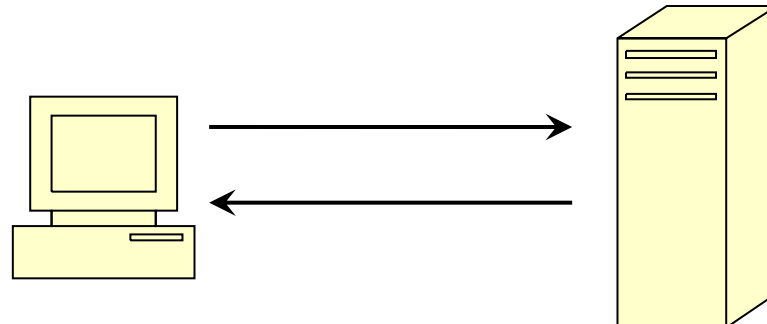
- E as especificações Java EE?
 - ▣ Java EE é definida através de especificações (*specifications, specs*).
- As especificações são escritas sob a ótica de desenvolvimento, determinando o que é esperado de um componente, programa, sistema de software.
- Os fornecedores de produtos Java EE devem atender aos requisitos estabelecidos pelas especificações Java EE para se dizer que suas soluções são aderentes a Java EE (*Java EE compliant*).

Arquitetura Java EE



Servlets e JSP

- Em sistemas para web, o cliente faz uma requisição **HTTP** ao servidor através de um navegador. O servidor recebe a requisição, processa a requisição (executando a lógica de negócio da aplicação), e envia uma resposta (uma página HTML) ao cliente.



HTTP

- O que é HTTP?
 - ▣ Protocolo formal de comunicação entre computadores em rede.
 - ▣ Basicamente em formato textual.
 - ▣ Dos sete métodos de solicitação, os mais usados são o GET e o POST.

HTTP

- ❑ Não mantém estado.
- ❑ Reescrita de URL e cookies são utilizados para controlar os usuários entre as solicitações.
- ❑ HTTP não trafega conteúdo dinâmico.

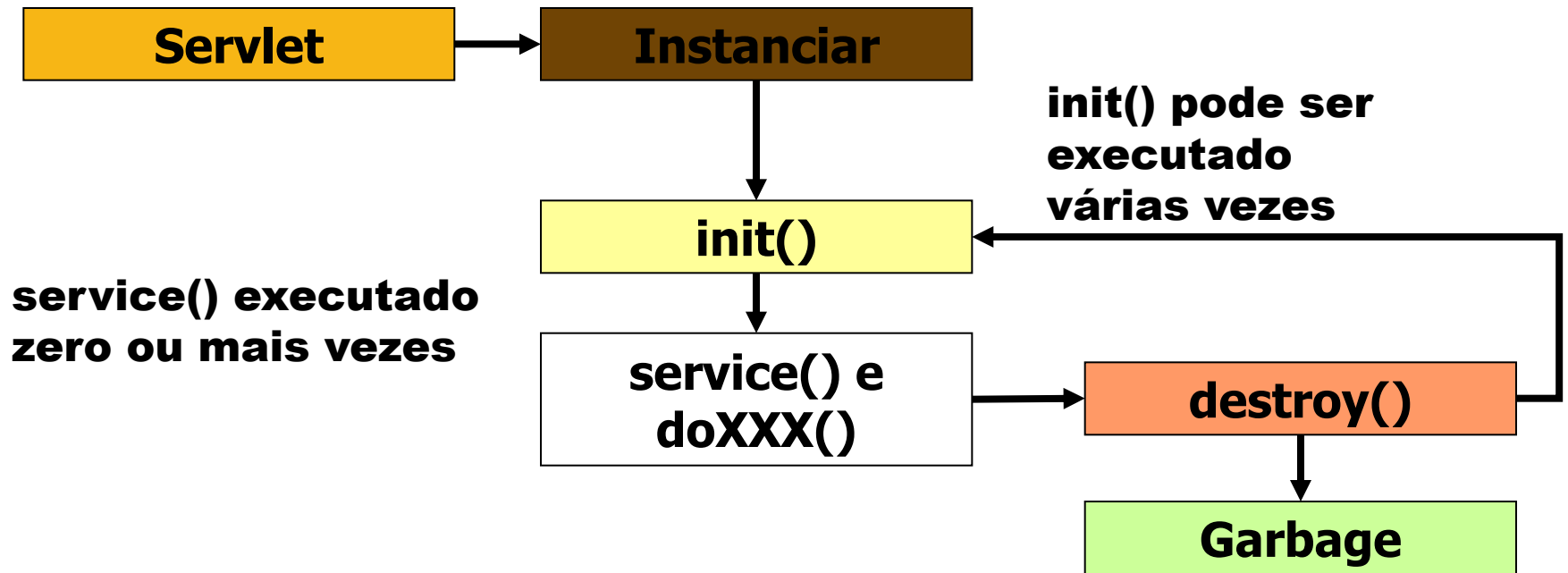
Servlets

- Qual seria o papel de Servlets em aplicações para a web? O que seria um Servlet?
 - ▣ Classes de intermédio entre as requisições de um *browser* e o uso de aplicativos de um servidor.

Servlets

- Então, Servlets são responsáveis por:
 - ▣ Ler dados enviados explicitamente pelo cliente (dados de formulário, parâmetros de requisição);
 - ▣ Ler dados enviados implicitamente pelo cliente (cabeçalhos de requisição, campos ocultos);
 - ▣ Gerar a resposta adequada;
 - ▣ Enviar explicitamente dados de retorno ao cliente (HTML);
 - ▣ Enviar implicitamente dados ao cliente (códigos de estado e cabeçalhos de resposta).

Servlets



Servlets e JSP

- A especificação de Servlets define uma classe base (abstrata) para a implementação de Servlets:
 - ▣ **`javax.servlet.http.HttpServlet`**
- Versão atual da especificação de Servlet:
 - ▣ 3.0.
- Versão atual da especificação de JSP:
 - ▣ 2.1.

Servlets e JSP

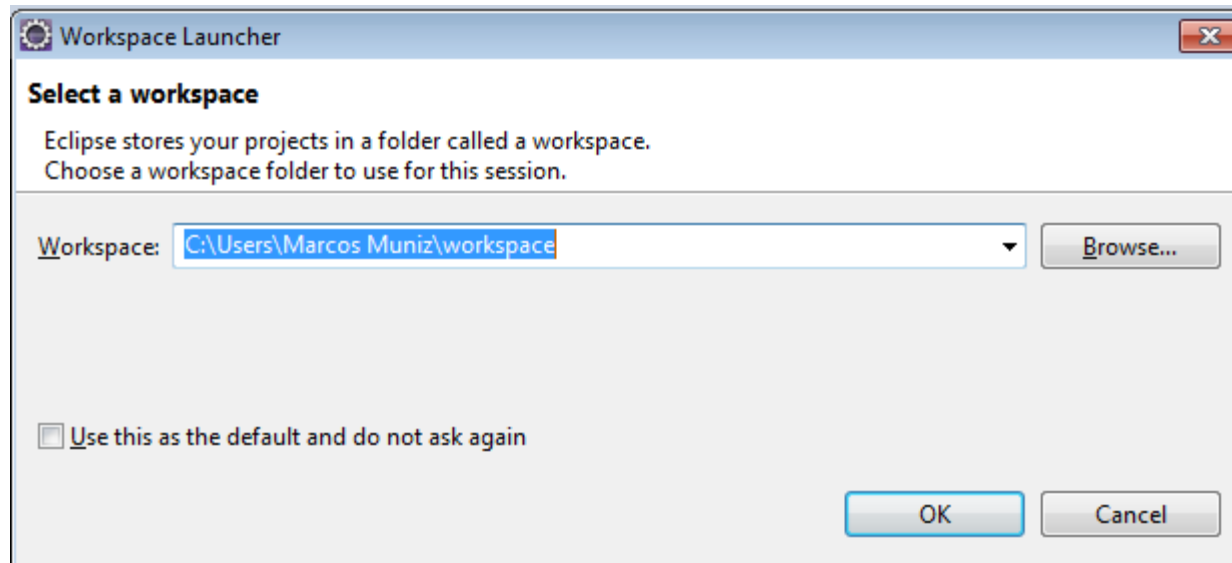
- Conforme visto anteriormente, provedores de soluções Java EE devem atender aos requisitos estabelecidos pelas especificações Java EE para se dizer que suas soluções são aderentes a Java EE (*Java EE compliant*).
- Para o contexto de Servlet 3.0 e JSP 2.1, neste curso, será adotado o seguinte *web server*:
 - ▣ Tomcat v7.0.27 (<http://tomcat.apache.org>).

Prática 01

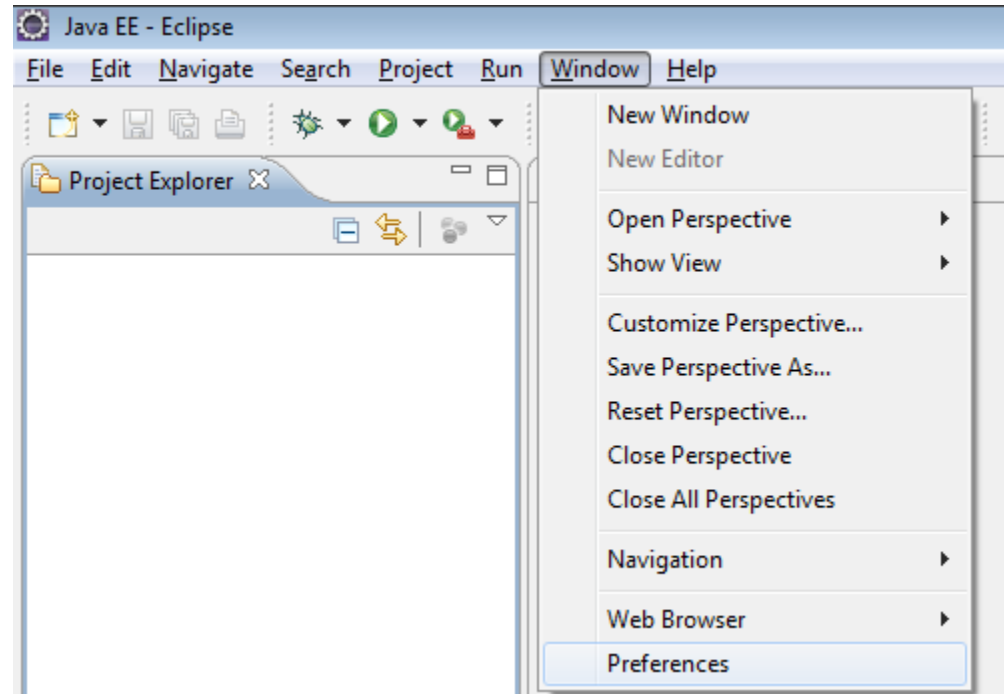
- Criando um Servlet, atividades:
 1. Abrir o Eclipse;
 2. Criar e configurar um servidor web;
 3. Criar e configurar um projeto web;
 4. Criar um Servlet;
 5. Configurar um Servlet e a URL de acesso;
 6. Executar o projeto web;
 7. Acessar o Servlet através de sua URL.

Prática 01

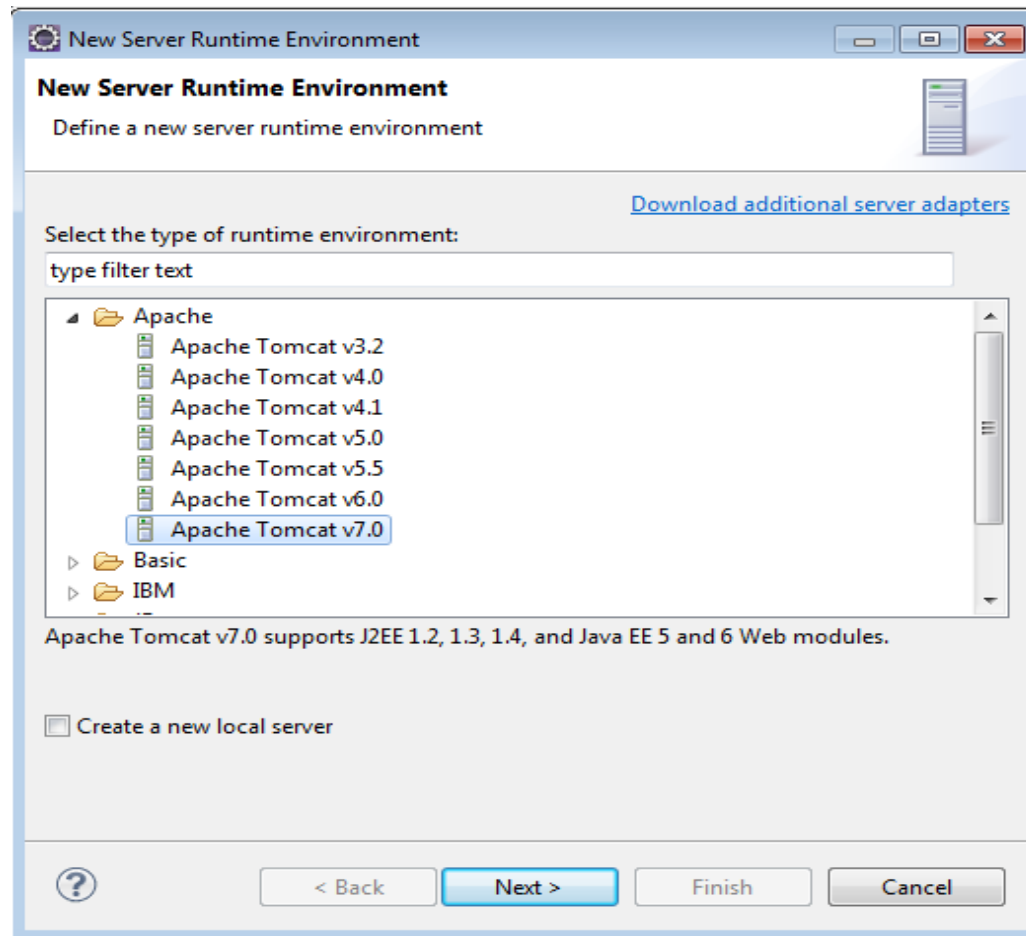
□ Atividade 1:



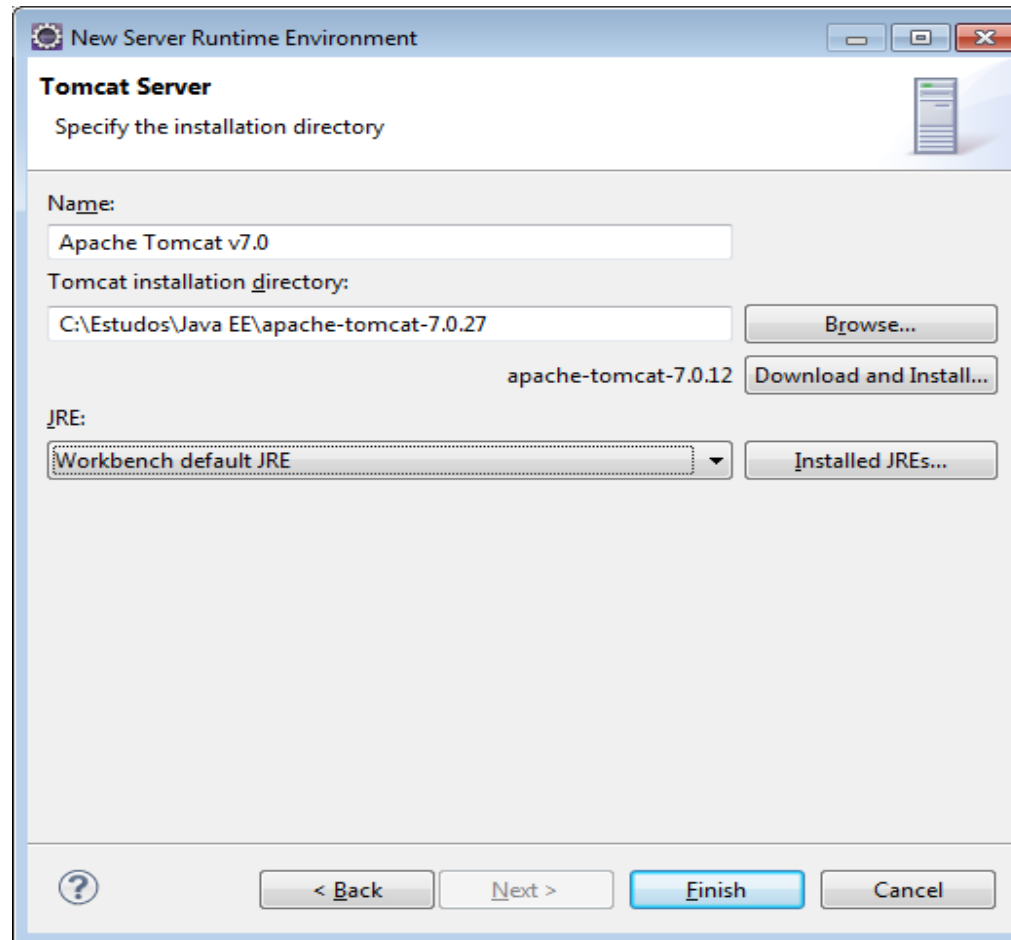
Prática 01



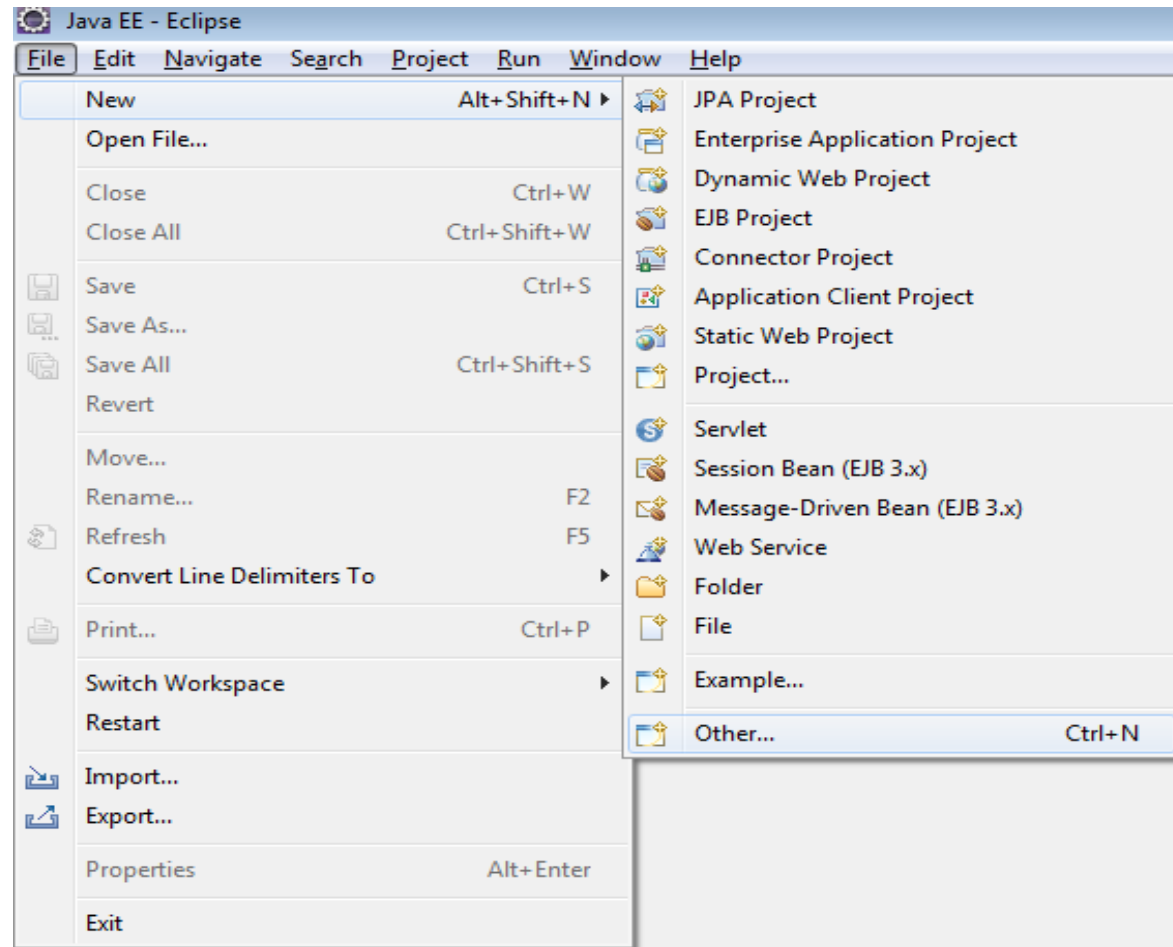
Prática 01



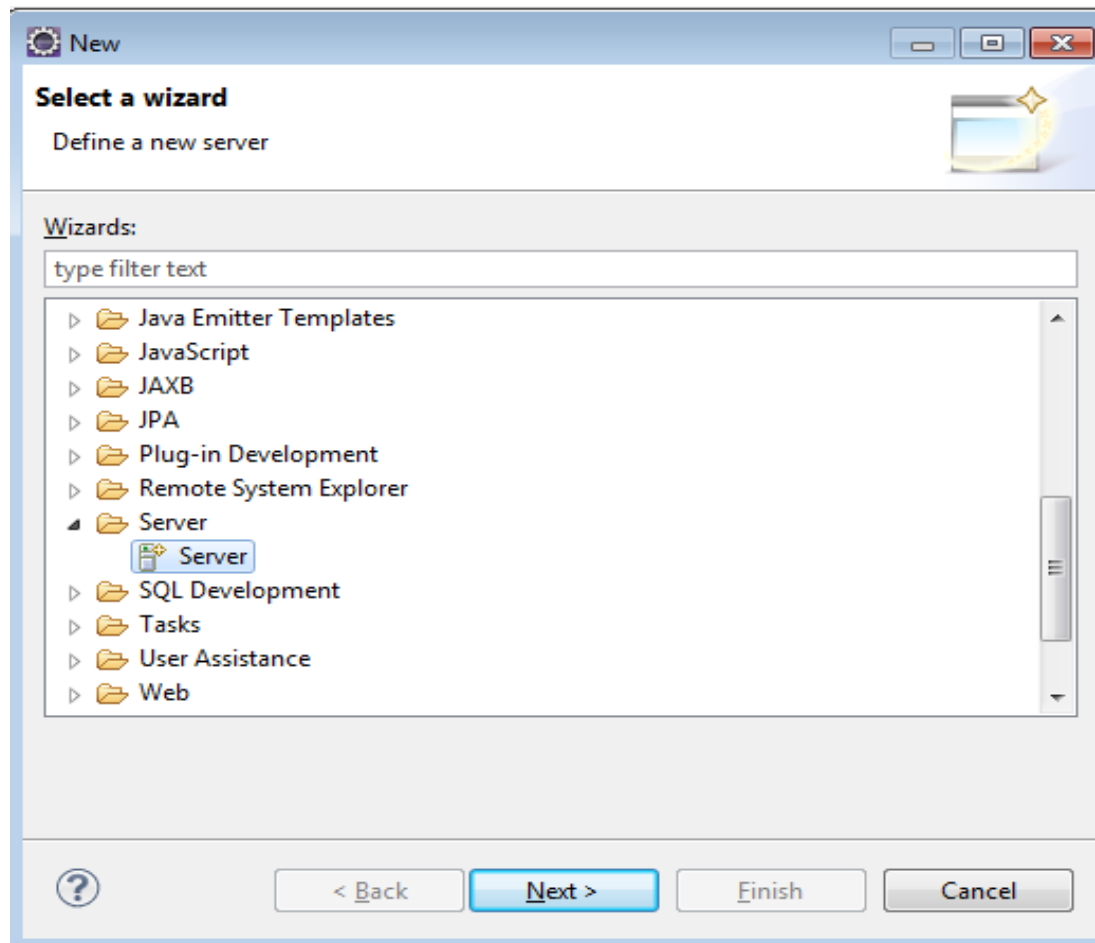
Prática 01



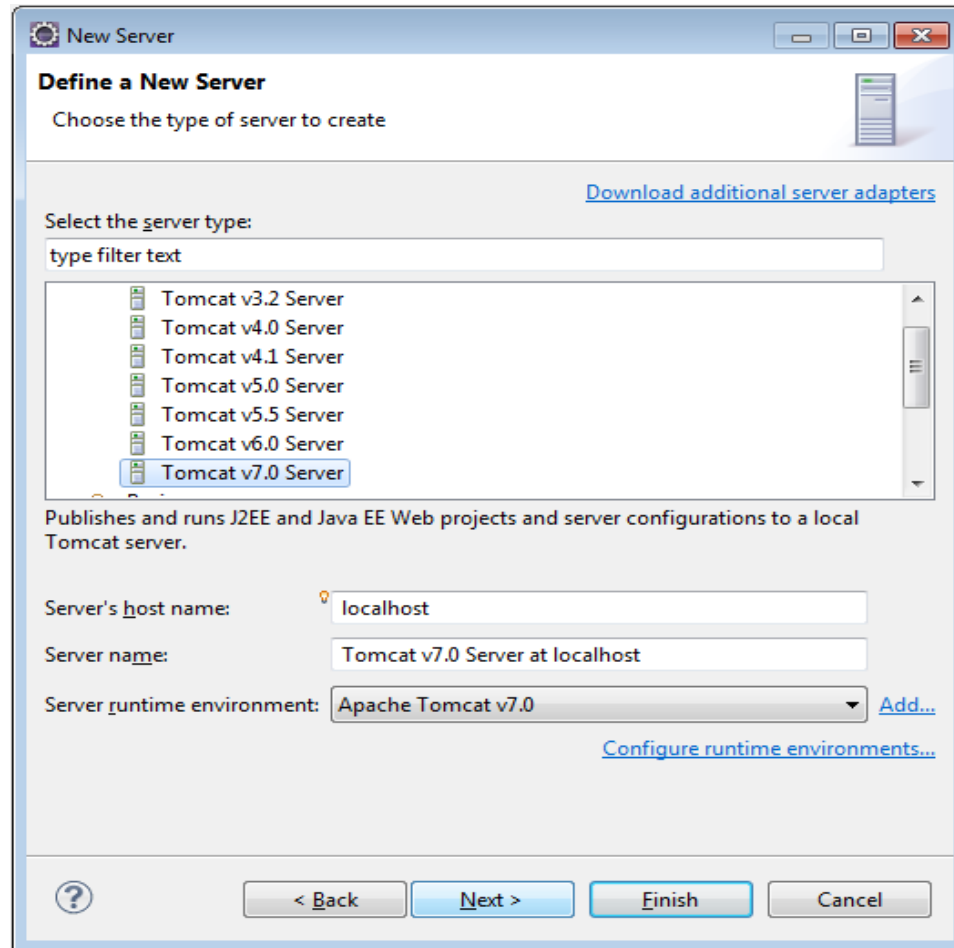
Prática 01



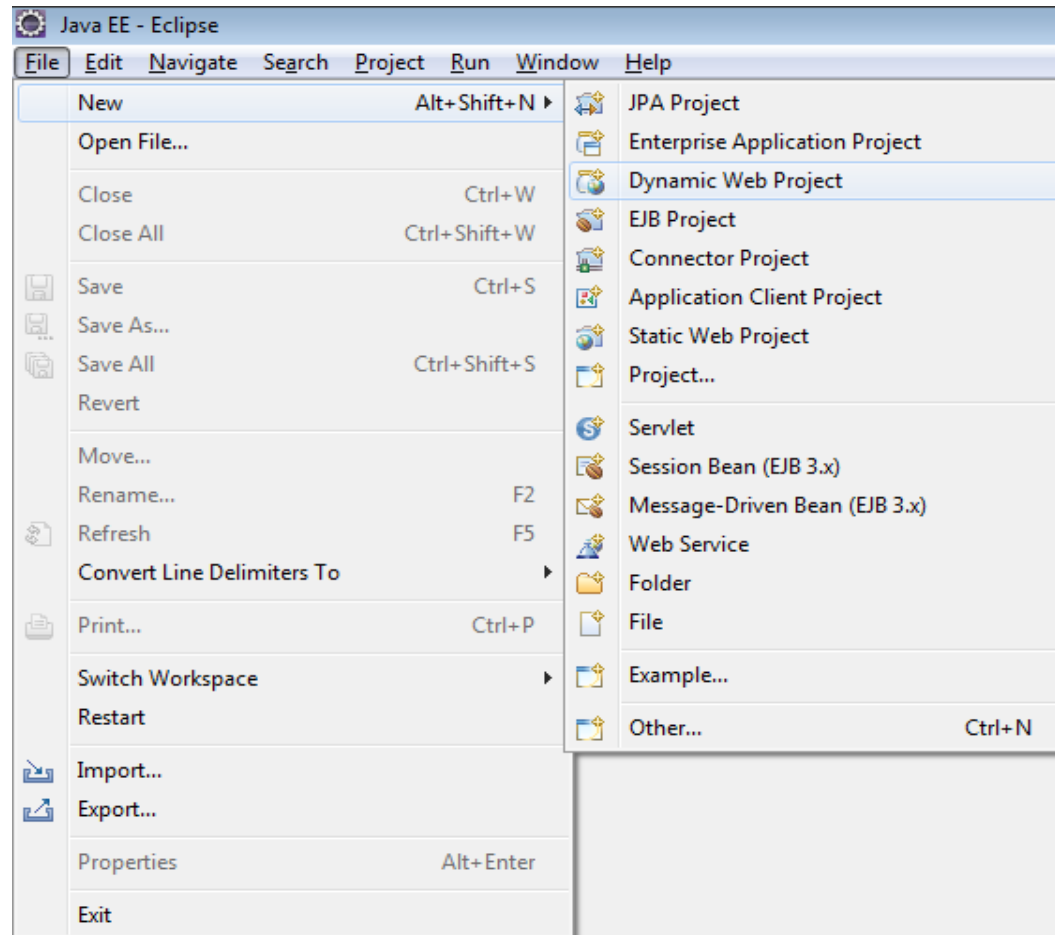
Prática 01



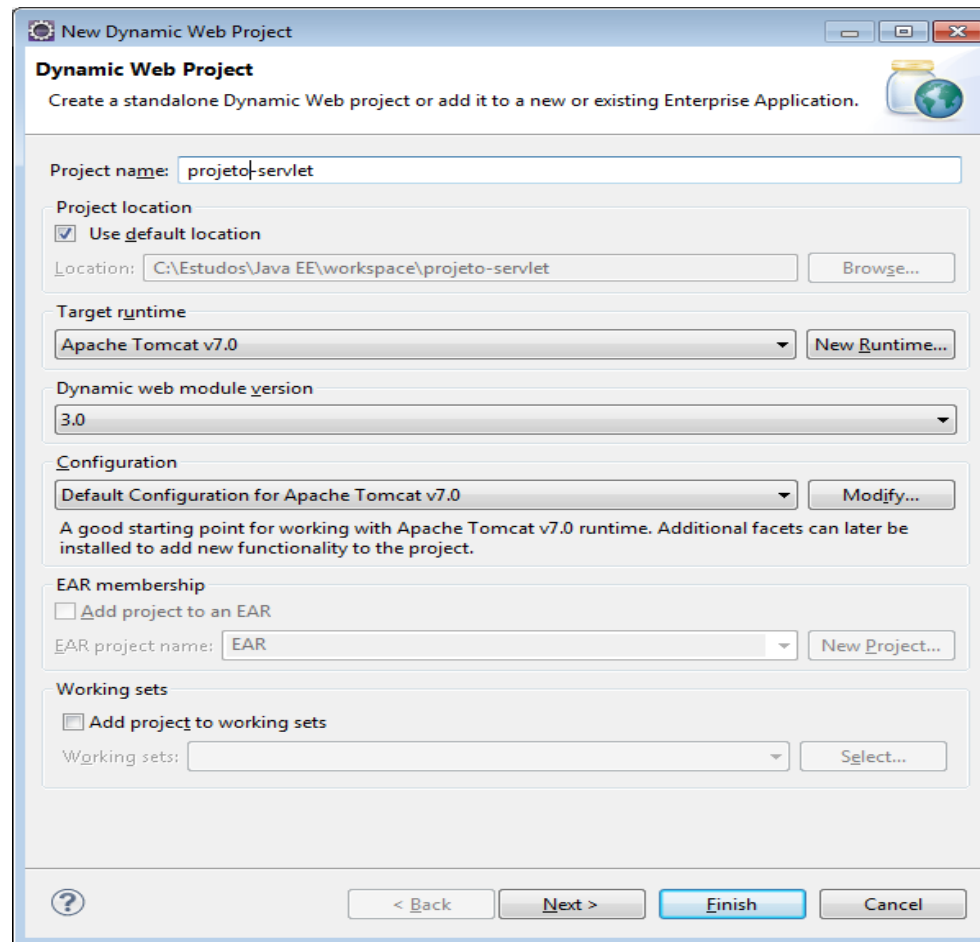
Prática 01



Prática 01



Prática 01



The screenshot shows the 'New Dynamic Web Project' wizard. The title bar reads 'New Dynamic Web Project'. The main heading is 'Dynamic Web Project' with a subtitle 'Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.' and a small globe icon.

Project name: projeto-servlet

Project location
☒ Use default location
Location: C:\Estudos\Java EE\workspace\projeto-servlet [Browse...]

Target runtime
Apache Tomcat v7.0 [New Runtime...]

Dynamic web module version
3.0

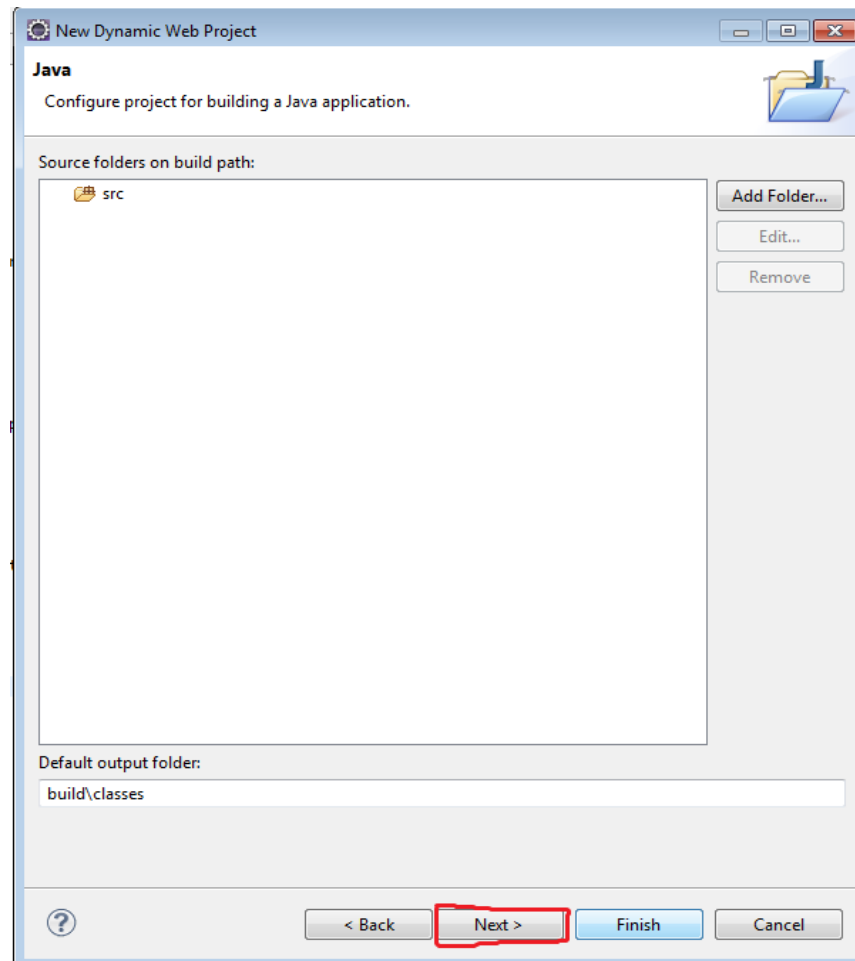
Configuration
Default Configuration for Apache Tomcat v7.0 [Modify...]
A good starting point for working with Apache Tomcat v7.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership
☐ Add project to an EAR
EAR project name: EAR [New Project...]

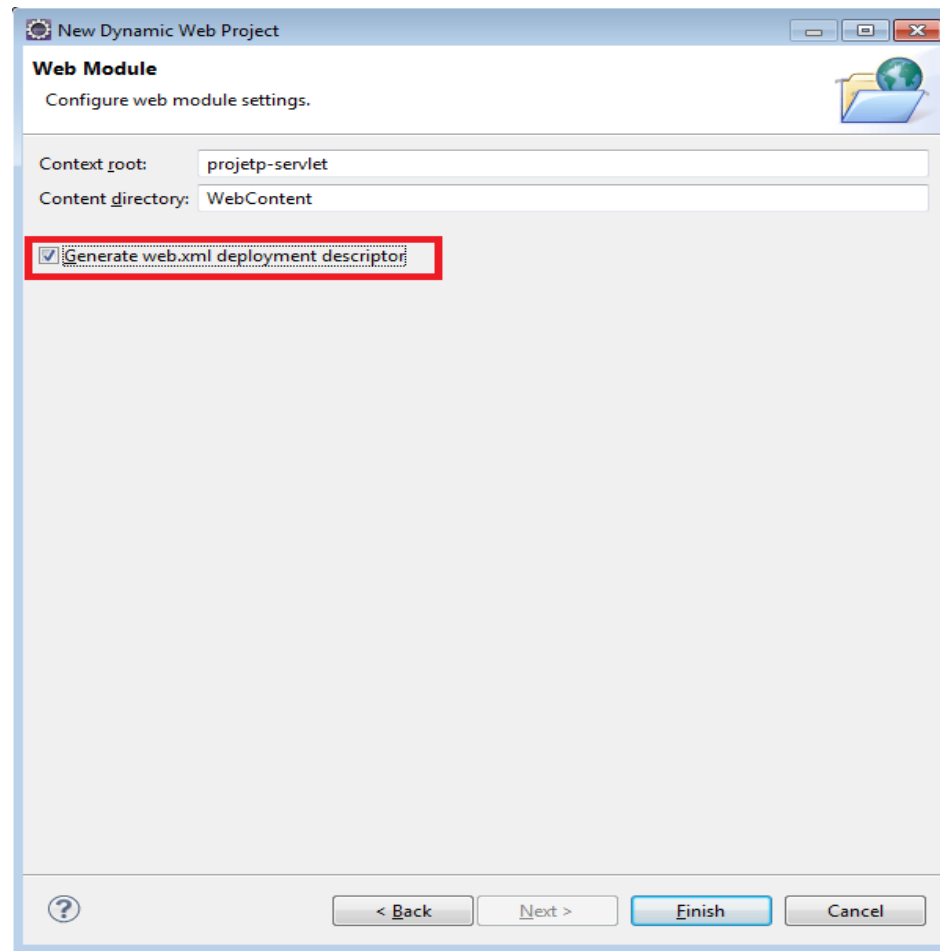
Working sets
☐ Add project to working sets
Working sets: [Select...]

At the bottom, there are navigation buttons: '< Back', 'Next >', 'Finish' (highlighted in blue), and 'Cancel'. A help icon (?) is also present.

Prática 01

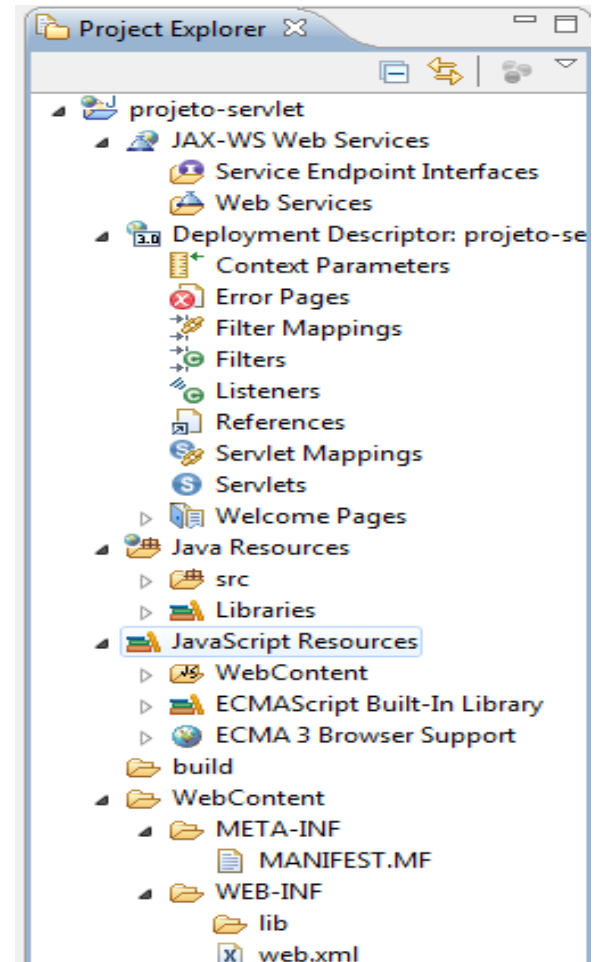


Prática 01



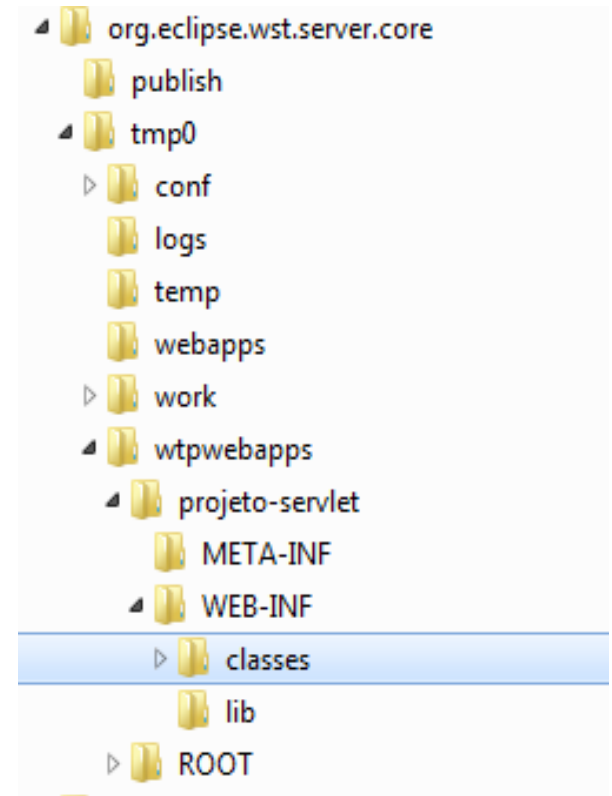
Prática 01

- A estrutura de um projeto web, segundo as especificações Java EE, segue uma mesma definição de diretórios e sub-diretórios principais.
- WebContent = Context root: diretório raiz do projeto web (<http://localhost:8080/projeto-servlet/>).
- WEB-INF: diretório de *ressources* e *sources* da aplicação web (não é acessível via URL).

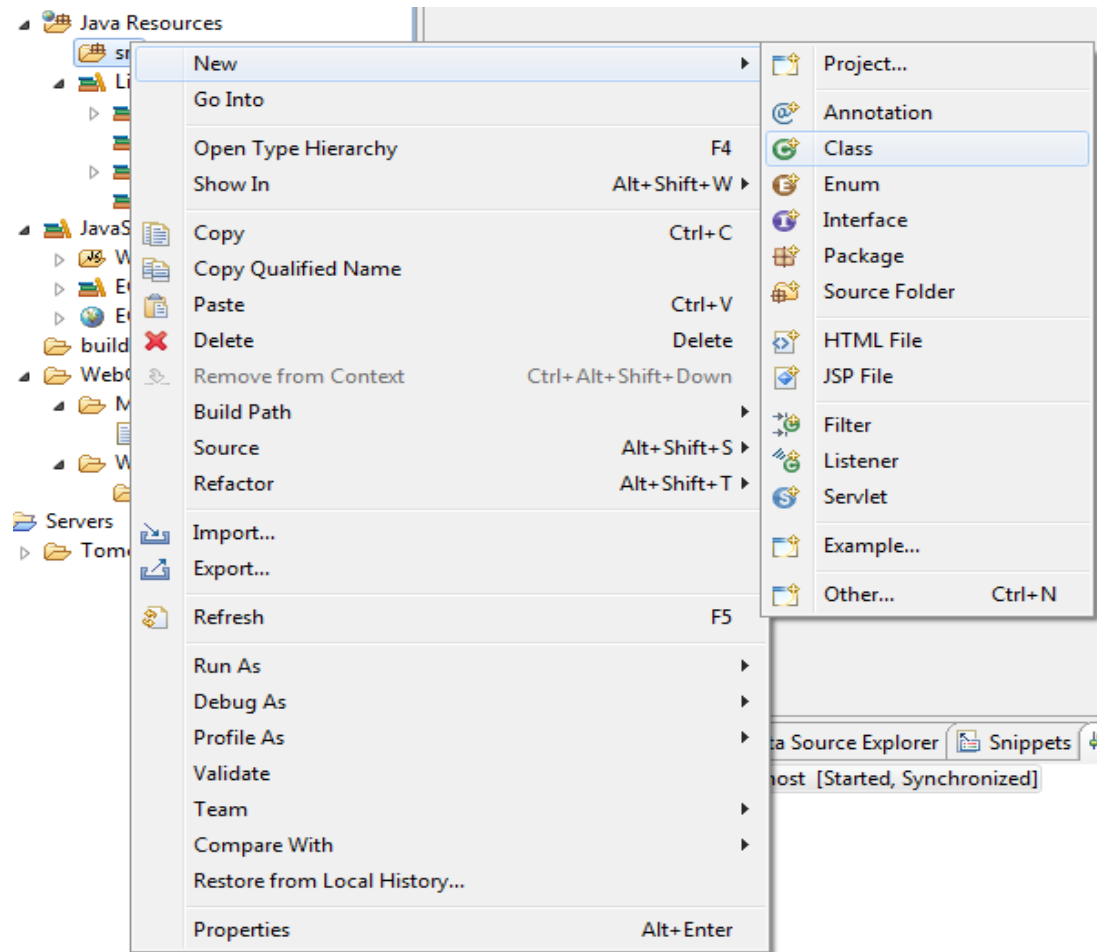


Prática 01

- Ao executar um projeto web, este projeto será implantado (*deployment*) em um servidor web.
- O Eclipse faz o *deployment* automaticamente, obedecendo a estrutura de diretórios de implantação definida pelo servidor web em uso.



Prática 01



Prática 01

New Java Class

Create a new Java class.

Source folder: projeto-servlet/src **Browse...**

Package: cuso.servlet **Browse...**

☐ Enclosing type: **Browse...**

Name: SimpleHelloWorldServlet

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: javax.servlet.http.HttpServlet **Browse...**

Interfaces: **Add...**
Remove

Which method stubs would you like to create?

- ☐ public static void main(String[] args)
- ☐ Constructors from superclass
- ☒ Inherited abstract methods

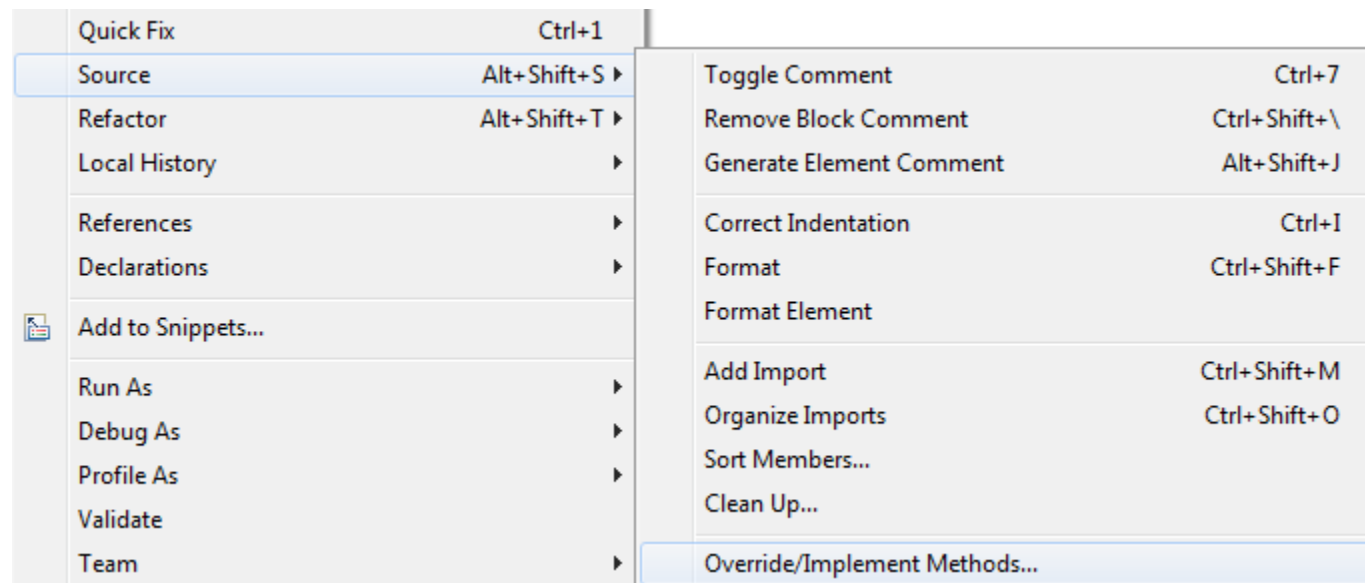
Do you want to add comments? (Configure templates and default value [here](#))

- ☐ Generate comments

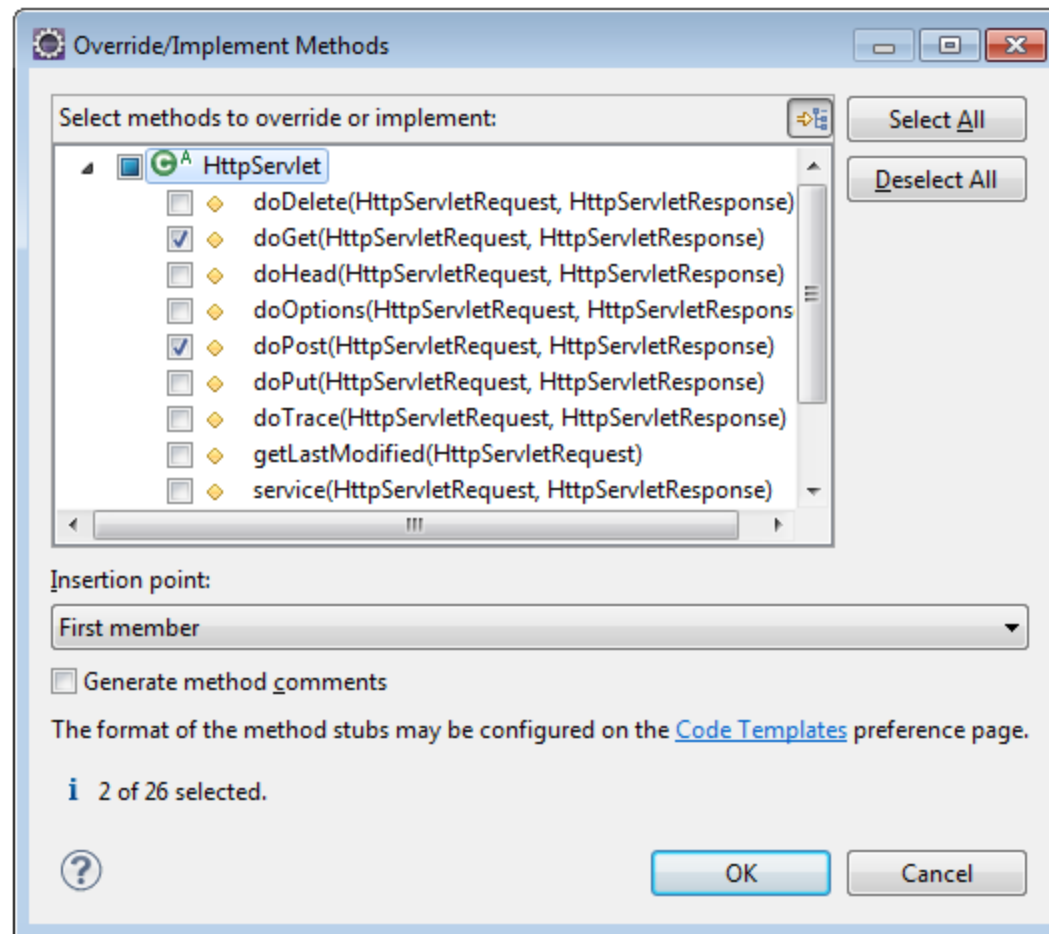
Finish **Cancel**

Prática 01

- Vamos sobrescrever os métodos `doGet()` e `doPost()` definidos pela classe base `HttpServlet`.



Prática 01



Prática 01

```
public class SimplesHelloWorldServlet extends HttpServlet {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

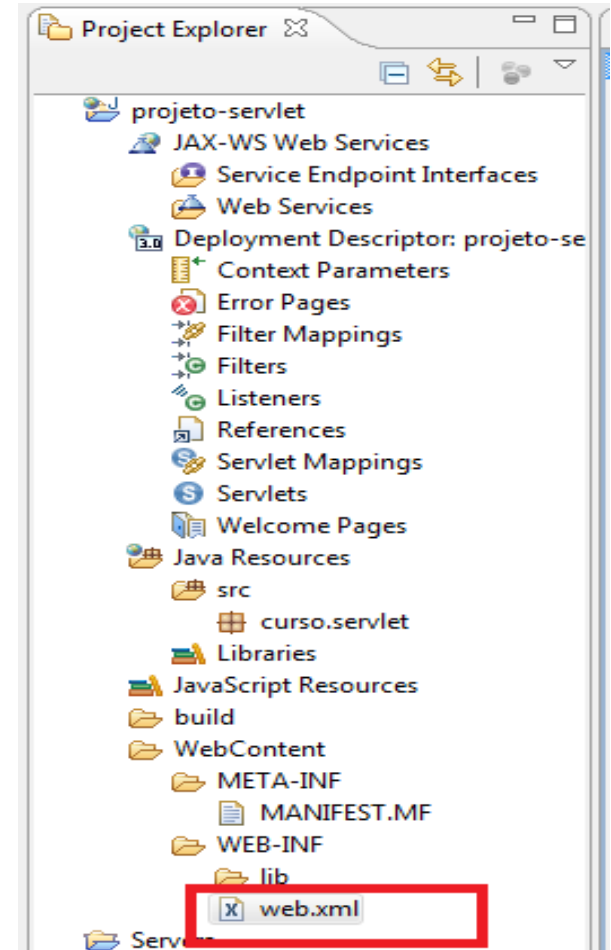
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        execute(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        execute(req, resp);
    }

    protected void execute(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        // Cabeçalho da resposta
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
        out.print("<html>" +
            "<head>" +
            "<title> Simples Hello World </title>" +
            "</head>" +
            "<body> <h1>Simples Hello World </h1></body>" +
            "</html>");
    }
}
```

Prática 01

- Abra o arquivo web.xml.
- Declare a definição do servlet.

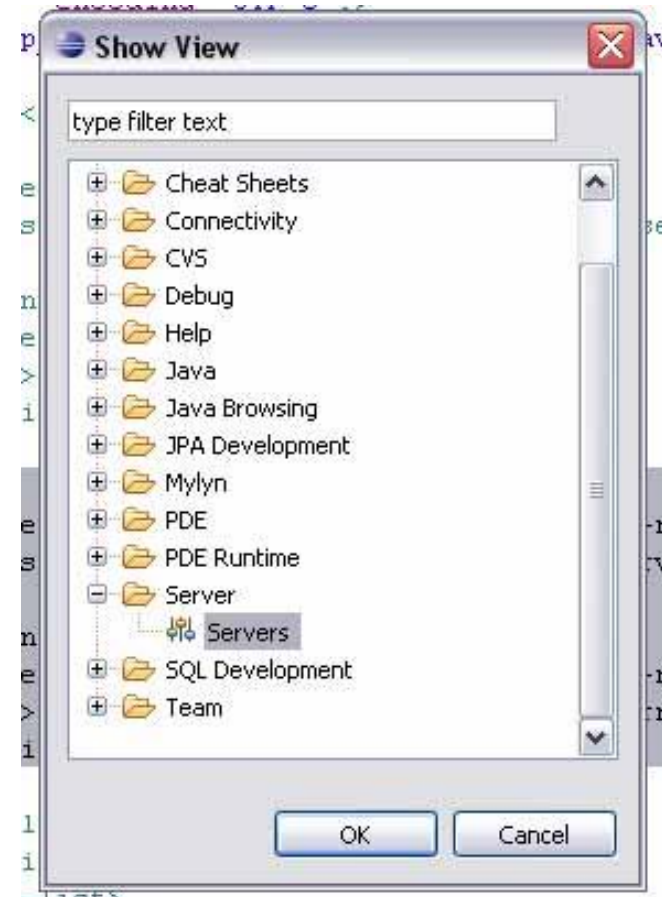
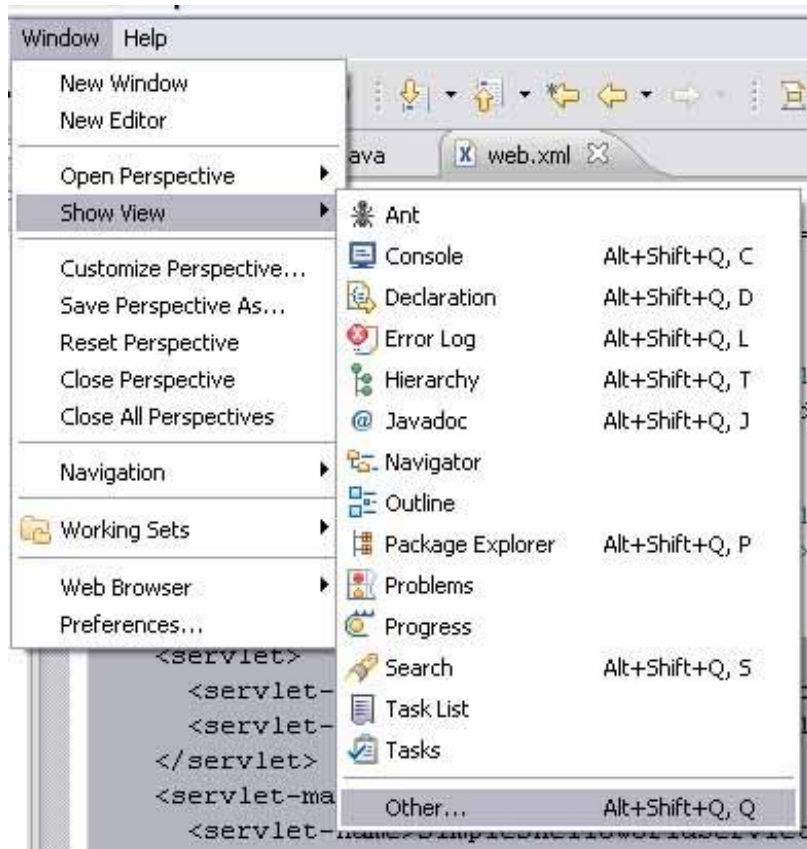


Prática 01

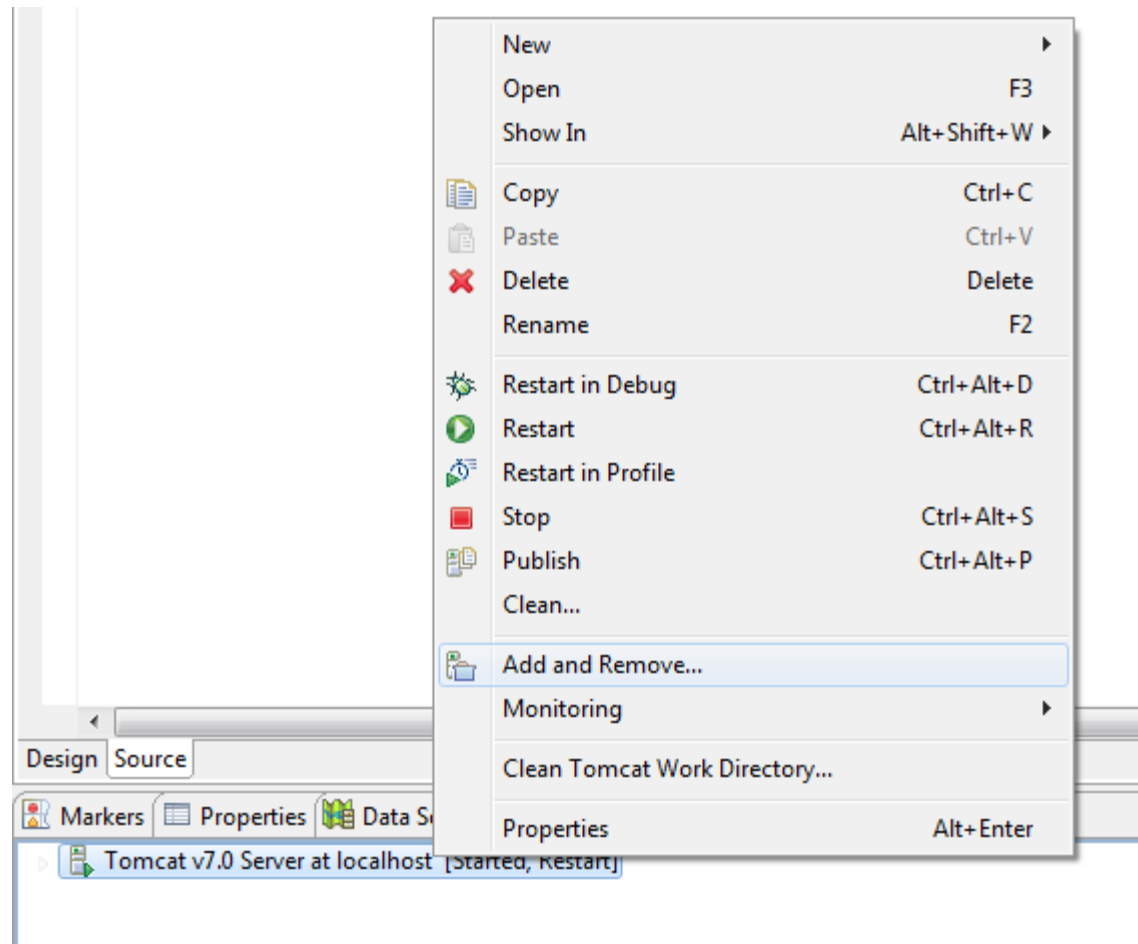


```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://j
  <display-name>projeto-servlet</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>simplesHelloWorldServlet</servlet-name>
    <servlet-class>curso.servlet.SimplesHelloWorldServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>simplesHelloWorldServlet</servlet-name>
    <url-pattern>/simplesHelloWorldServlet.html</url-pattern>
  </servlet-mapping>
</web-app>
```

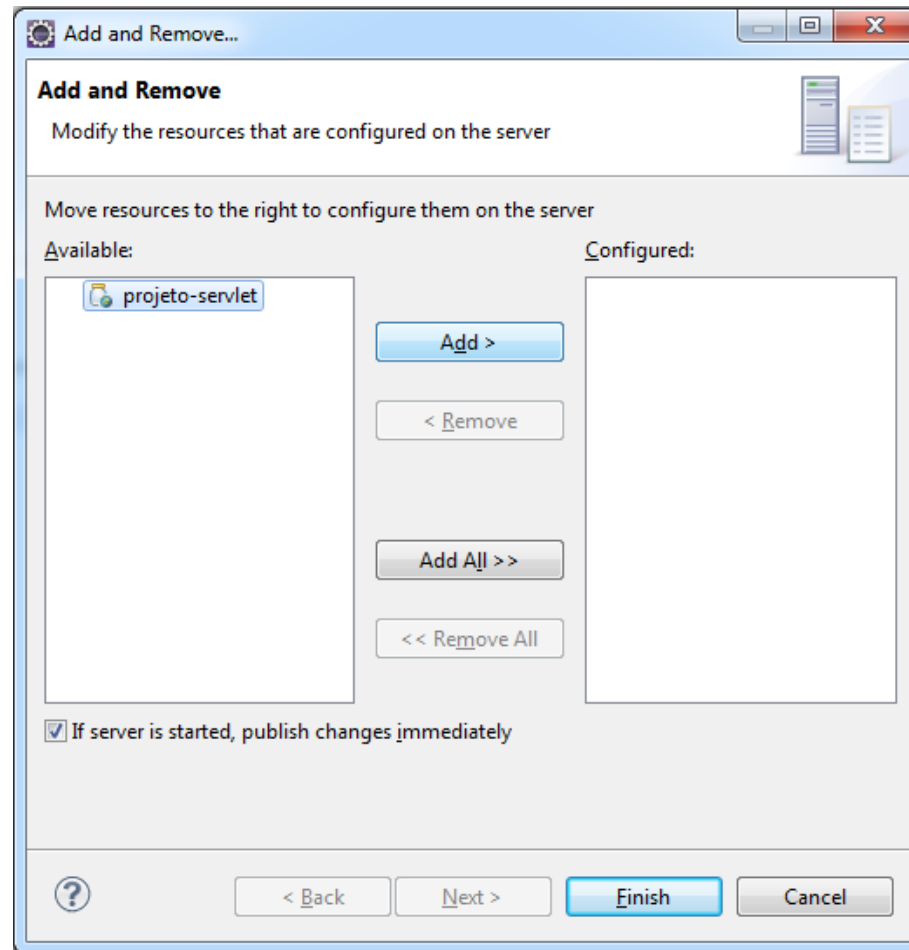
Prática 01



Prática 01

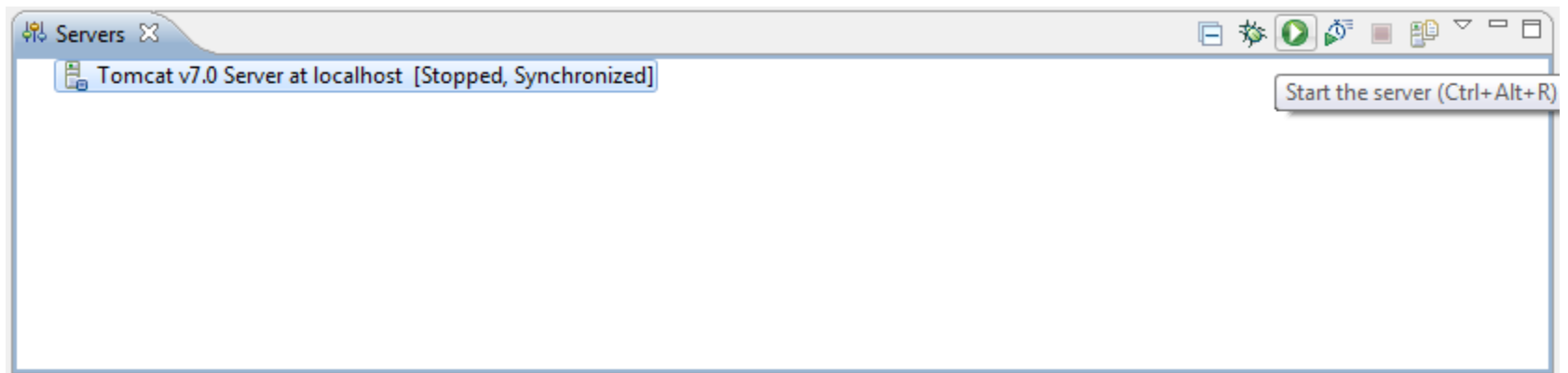


Prática 01



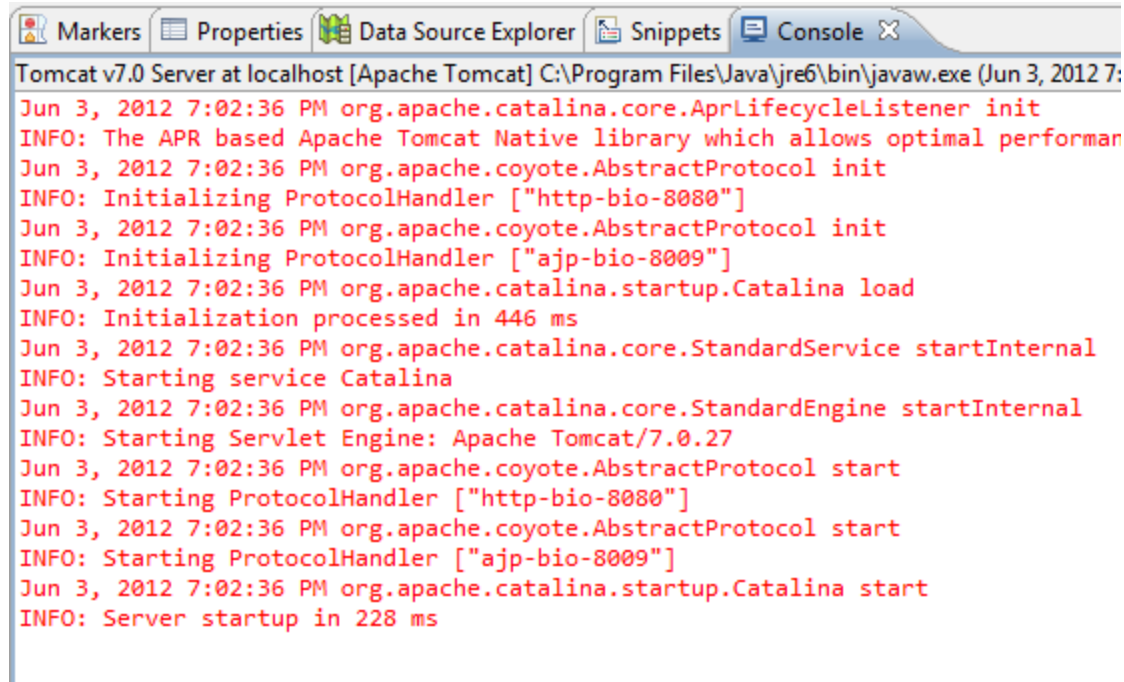
Prática 01

□ Inicie o servidor:



Prática 01

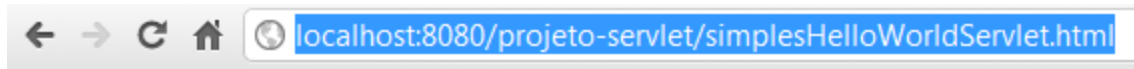
- Verifique na view console (Window – Show view - Console) se o servidor foi iniciado corretamente:



```
Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre6\bin\javaw.exe (Jun 3, 2012 7:
Jun 3, 2012 7:02:36 PM org.apache.catalina.core.AprLifecycleListener init
INFO: The APR based Apache Tomcat Native library which allows optimal performan
Jun 3, 2012 7:02:36 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-bio-8080"]
Jun 3, 2012 7:02:36 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["ajp-bio-8009"]
Jun 3, 2012 7:02:36 PM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 446 ms
Jun 3, 2012 7:02:36 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting service Catalina
Jun 3, 2012 7:02:36 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.27
Jun 3, 2012 7:02:36 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
Jun 3, 2012 7:02:36 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Jun 3, 2012 7:02:36 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 228 ms
```

Prática 01

- Vá até o navegador web e digite o endereço:
- <http://localhost:8080/projeto-servlet/simplesHelloWorldServlet.html>



Simples Hello World

Prática 01

- Dúvidas, perguntas, reclamações?



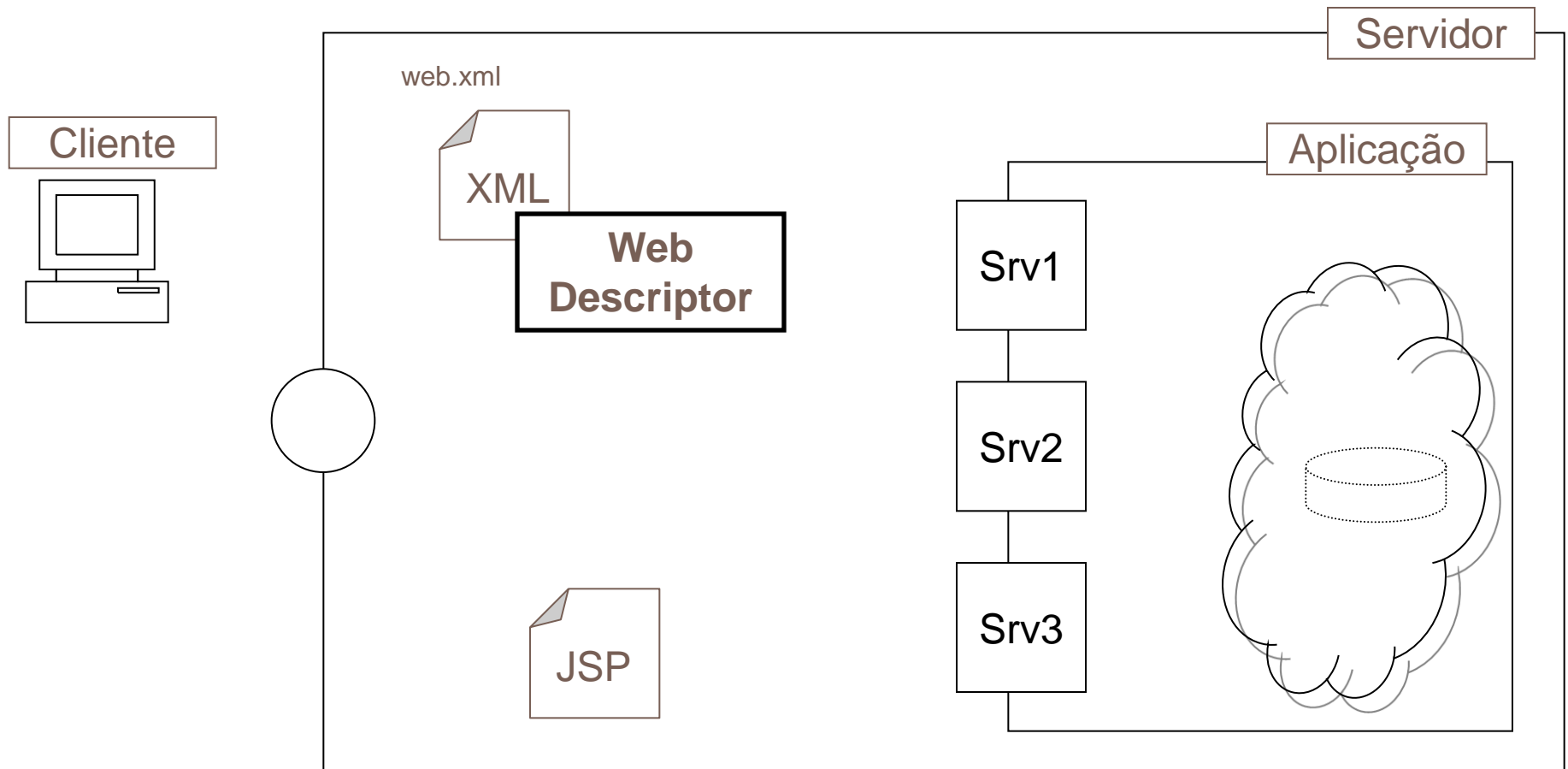
JSP

- JSP (*Java Server Pages*):
 - ▣ Documentos JSP incluem tanto partes estáticas quanto “dinâmicas”.
 - ▣ *Servlets* são classes especiais que implementam alguns serviços pré-definidos e que rodam em *containers* JSP.
 - ▣ Documentos JSP são convertidos em *servlets* quando são requisitados pela primeira vez.

JSP

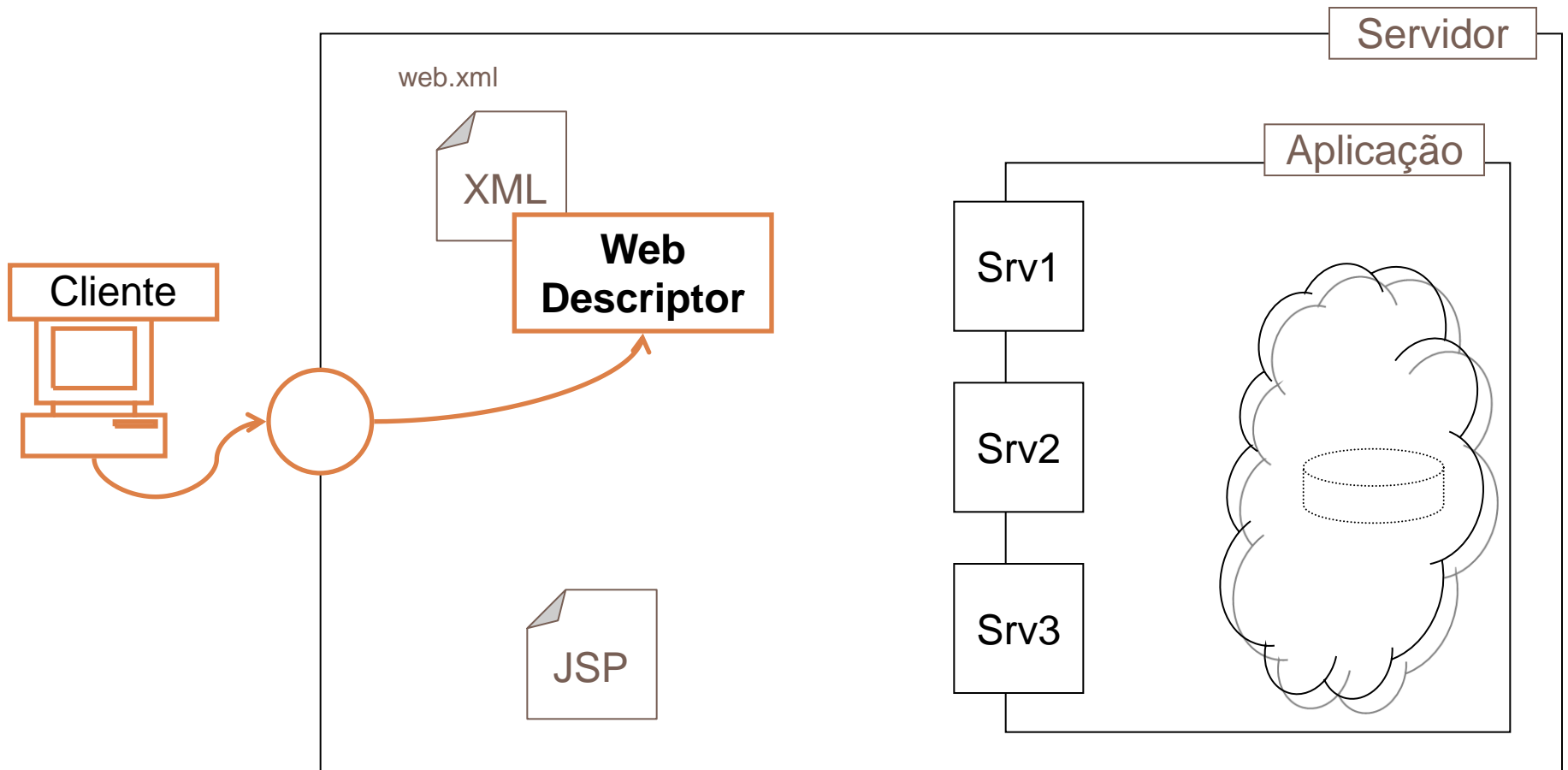
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<%@ page language="java" contentType="text/html" %>
<%@ taglib prefix="c" uri="/WEB-INF/jstl-core.tld" %>
<html>
<head>
    <title>Exemplo de JSP</title>
</head>
<body>
Exemplo de conteúdo estático e dinâmico.
O texto é estático porém a data é dinâmica - data atual do
sistema.
<%=new java.util.Date().toString()%>, é dinâmico!
</body>
</html>
```

Servlets e JSP



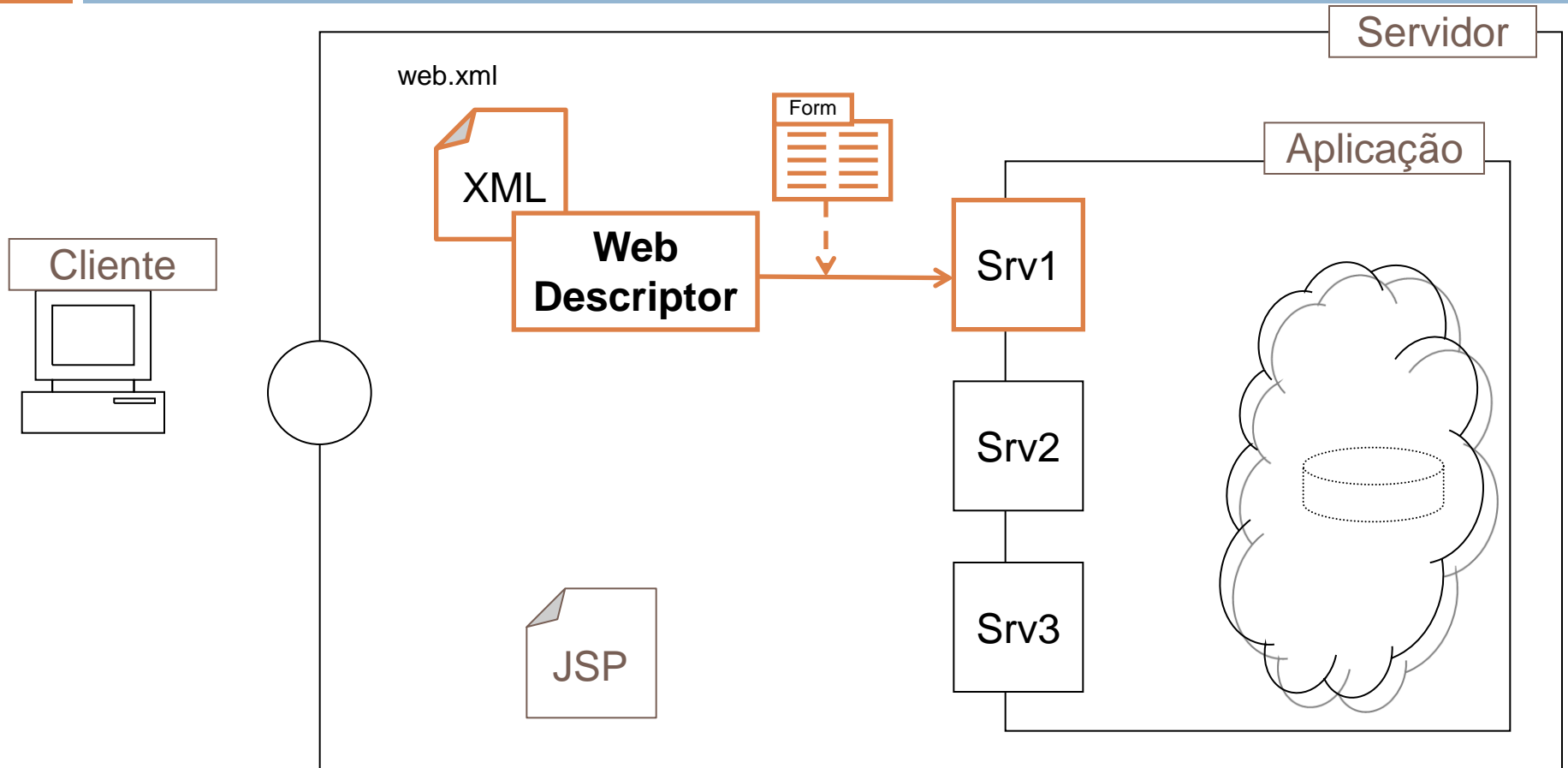
Ambiente Web.

Servlets e JSP



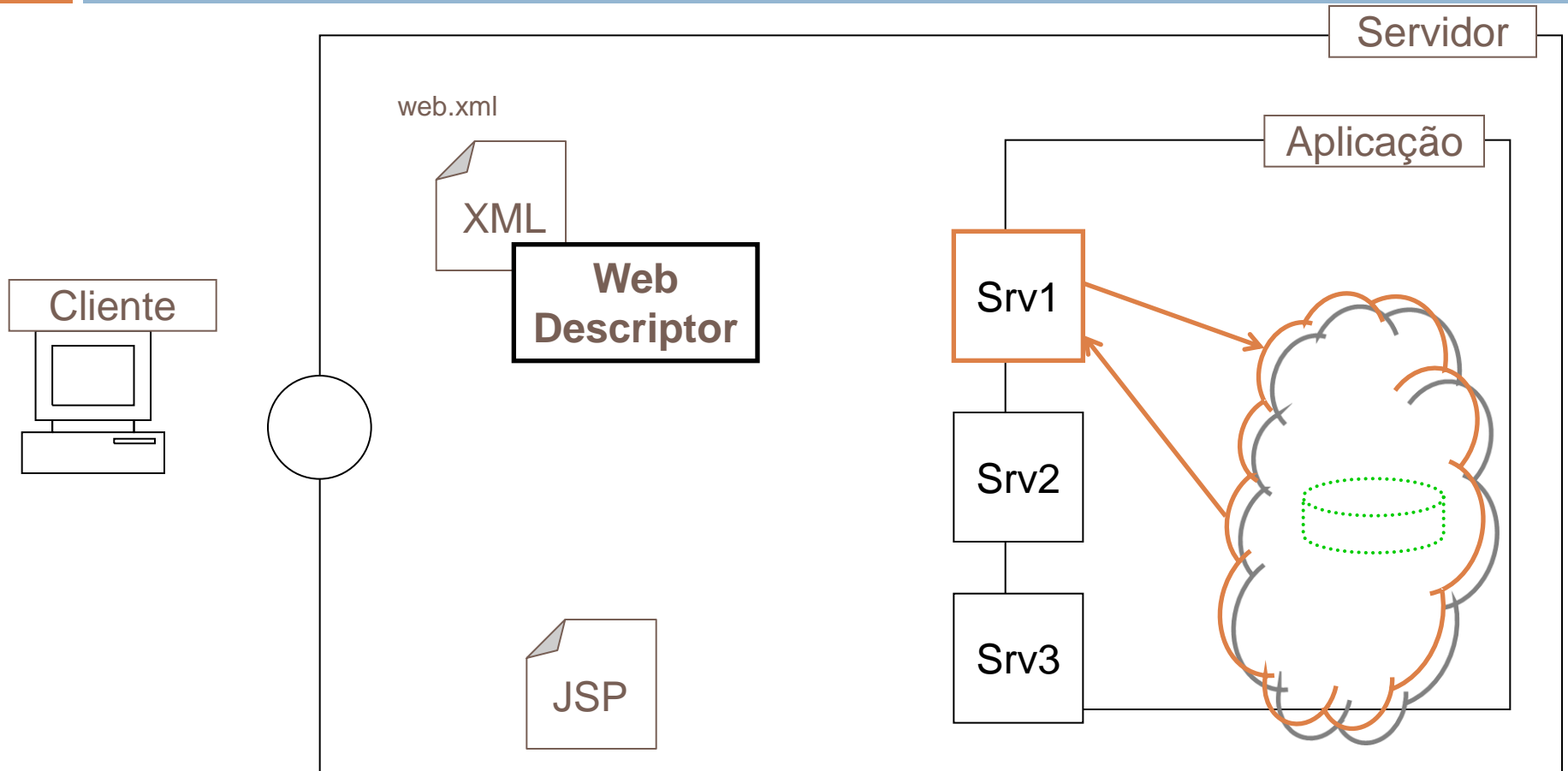
O navegador do cliente faz uma requisição para o servidor acionando um link ou digitando o caminho, por exemplo.

Servlets e JSP



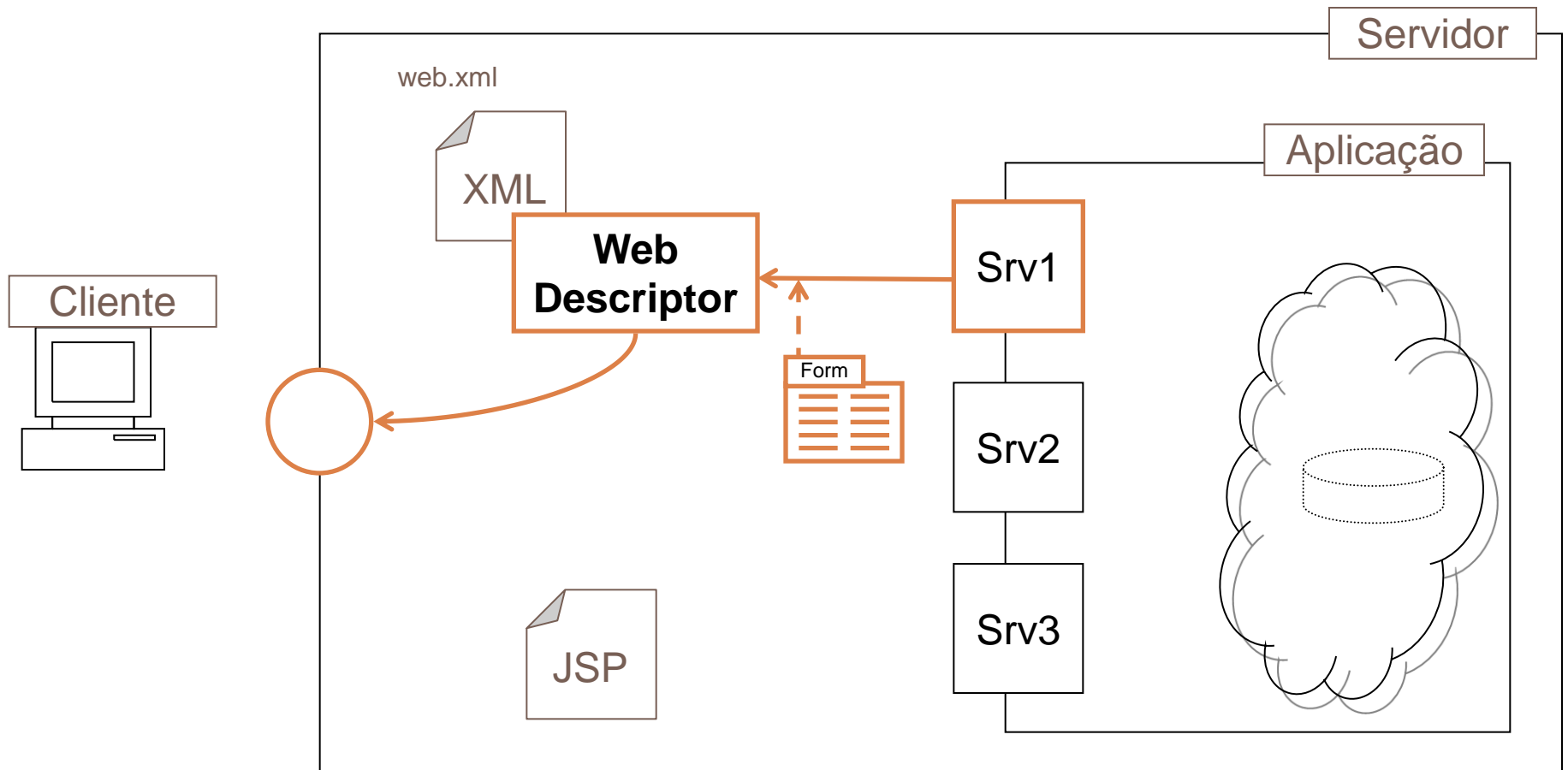
O servidor recebe a requisição, analisa seu formato e repassa para o servlet responsável por tratá-la.

Servlets e JSP



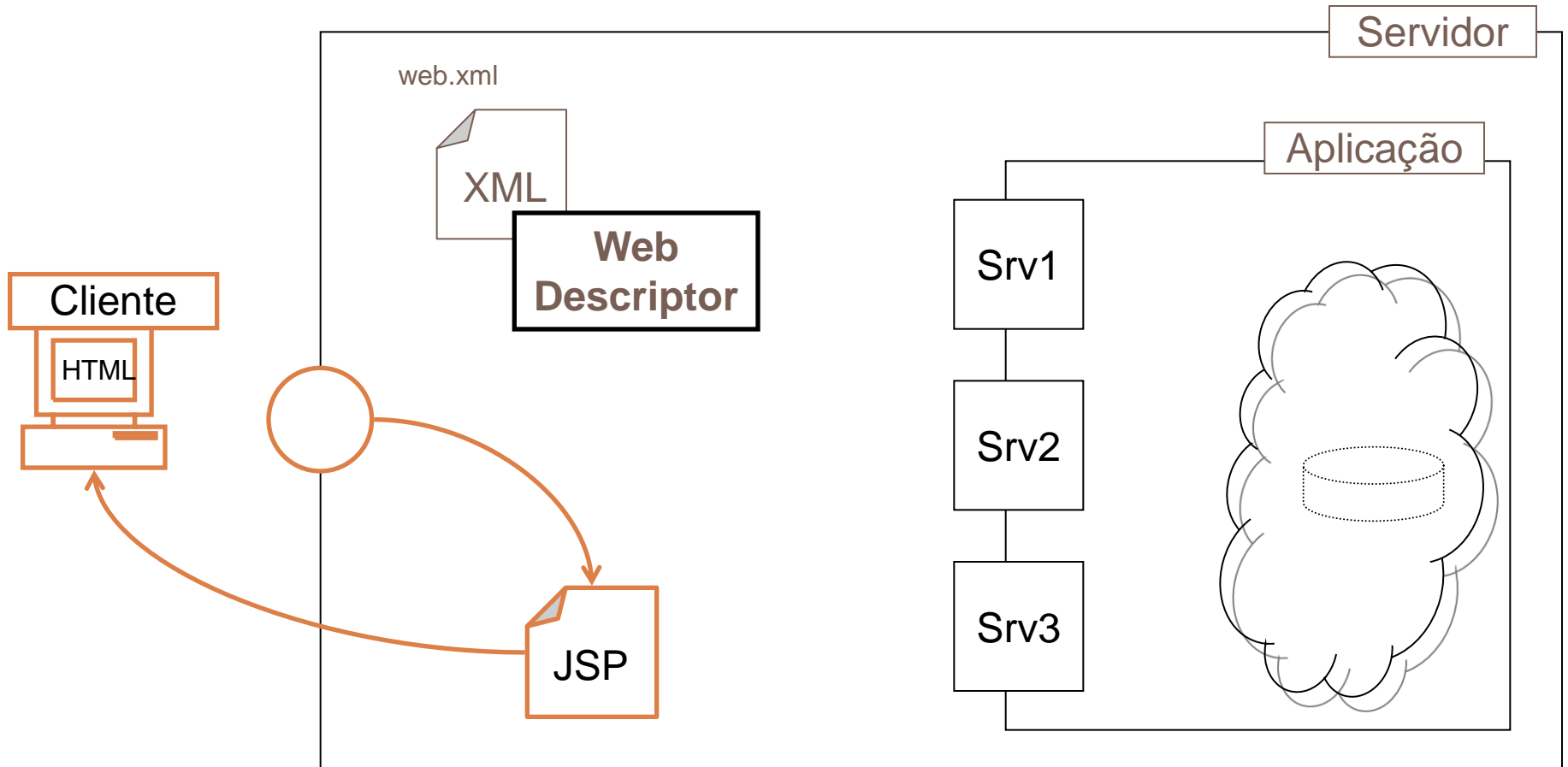
As outras camadas da aplicação são acessadas. Ao final, o processamento retorna para o Servlet em execução.

Servlets e JSP



A execução do Servlet é finalizada com a geração de uma uma resposta enviada ao cliente.

Servlets e JSP



A resposta a ser enviada ao usuário pode ser gerada através do uso de um JSP.

Servlets e JSP

- Ainda sobre Servlets, lembrando, dentre as funções temos:
 - ▣ Ler dados enviados implicitamente pelo cliente (cabeçalhos de requisição, campos ocultos);
 - ▣ Enviar implicitamente dados ao cliente (códigos de estado e cabeçalhos de resposta).

Cabeçalhos HTTP

- Cabeçalhos da requisição:
 - ▣ Dados enviados ao servidor pelo browser.
- Cabeçalhos da resposta:
 - ▣ Contém informações do documento enviado como resposta a uma requisição.

```
GET / HTTP/1.1
Accept: image/gif, image/x-
bitmap, image/jpeg,
image/pjpeg, application/x-
shockwave-flash, */*
Accept-Language: pt
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
(compatible; MSIE 6.0; Windows
NT 5.0)
Host: www.globo.com
Connection: Keep-Alive
Cookie: RMID=c8a5201b3f95aa90
```

```
HTTP/1.1 200 OK
Age: 2
Date: Fri, 13 Feb 2004 13:59:57 GMT
Content-Length: 40827
Content-Type: text/html
Cache-Control: max-age=30
Server: ""
<html><head>
<title>Globo.com</title>
<meta name="editoria"
content="Globo.com">
...
</html>
```

Códigos HTTP de estado

- 200 → OK.
- 400 → Bad request.
- 401 → Unauthorized.
- ...
- [RFC 2616 Fielding, et al.]

Servlets e JSP

- E mais, Servlets são responsáveis por:
 - ▣ Ler dados enviados explicitamente pelo cliente (dados de formulário, parâmetros de requisição);
 - ▣ Gerar a resposta adequada;

Request

- A variável **request** é responsável por permitir manipular elementos da requisição no JSP.
- Instância da classe **HttpServletRequest**.
- Para manipular parâmetros de uma requisição HTTP: método **getParameter()**.
- Para manipular cabeçalhos: **getHeader()**.
- Outros métodos: consultar API da classe

```
<%-- login.jsp --%>
<%
    String login = request.getParameter("login");
    int codigo = 0;
    try {
        codigo = Integer.parseInt(request.getParameter("codigo"));
    }
    catch (NumberFormatException nfe) {}
    out.println("Bem-vindo, " + login);
%>
```

Response

- A variável **response** é responsável por permitir manipular a resposta do servidor no JSP.
- Instância da classe **HttpServletResponse**.

```
<% // Impedindo o uso de cache ...  
response.setHeader("Pragma", "no-cache");  
response.setHeader("Cache-Control", "no-cache");  
...  
// Redirecionamento ...  
response.sendRedirect("formNovo.jsp");  
  
// Encaminhamento ...  
pageContext.forward("documento.jsp");  
request.getRequestDispatcher  
("documento.jsp").forward(request, response);
```

JSP e HTTP

- O protocolo HTTP é *stateless*, ou seja, não mantém estado entre requisições.
- Um dos principais objetivos do JSP é garantir o dinamismo das informações.
- Via especificação, foram definidos objetos implícitos ao JSP para permitir o trânsito de informação sob dois aspectos:
 - ▣ Escopo e tempo de vida.

page	request
session	application

Prática 02

- ❑ Criar um novo Servlet cuja função será redirecionar para um JSP.
- ❑ O JSP deve exibir a mensagem “Hello JSP!”.

Prática 03

- ❑ Criar um Servlet que será responsável por obter da requisição o nome do usuário, informado via formulário HTML, e redirecioná-lo para uma página de boas vindas.
- ❑ A página de boas vindas deverá ser um JSP que exiba a mensagem “Bem vindo <nome>!”.

Prática 04

- Criar um Servlet responsável por somar dois valores, informados pelo usuário via formulário, exibindo ao usuário o resultado desta soma.
- Pergunta: você pretende tratar erros de entrada de dados pelo usuário? Como seria?

Arcabouços para web

 **Struts**

Struts²

Spring
2.0

///Stripes



tapestry

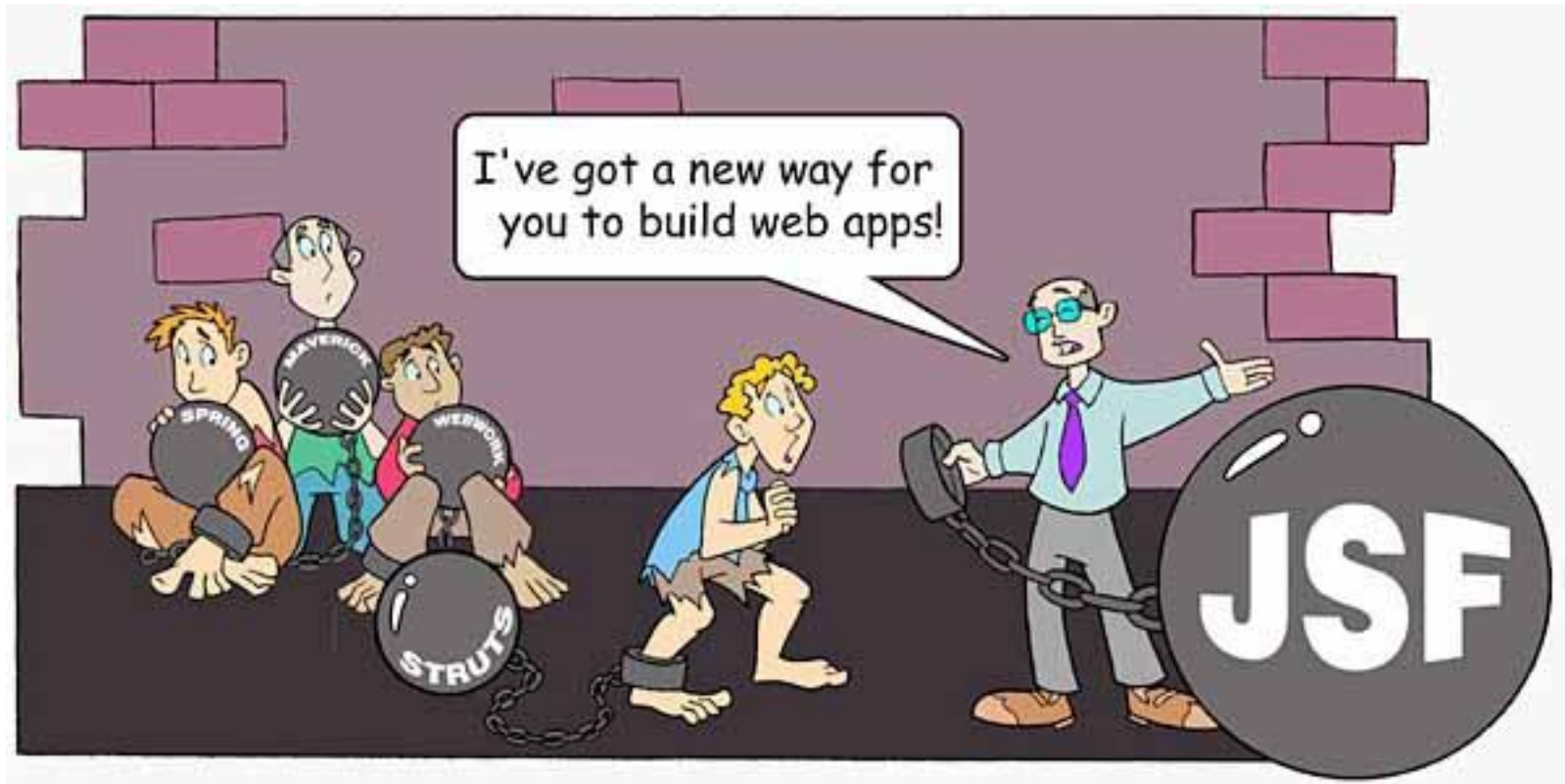


My Faces



WICKET

Arcabouços para web



JSF

- *Java Server Faces*
- Prós:
 - ▣ Atual padrão Java EE.
 - ▣ Fácil e rápido para uso iniciante.
 - ▣ Muitos componentes de interface de usuário já implementados.
- Contra:
 - ▣ Sopa de tags para uso em JSP.
 - ▣ Não trata muito bem REST e segurança.
 - ▣ Não há uma única fonte (versão).

Struts 2

- Struts 2
- Prós:
 - ▣ Arquitetura mais simples com relação ao Struts 1.
 - ▣ Melhor personalização de taglibs com FreeMarker e Velocity.
 - ▣ Navegação via *Controller-based* ou *page-based*.
- Contra:
 - ▣ Documentação escassa e mal organizada.
 - ▣ Excessiva concentração em novas funcionalidades (e os bugs?).
 - ▣ Uso de documentação do Struts 1.x como referência.

Tapestry

- Tapestry
- Prós:
 - ▣ Alta produtividade uma vez que você o aprenda.
 - ▣ Os templates são HTML – ótimo para designers.
 - ▣ Várias inovações entre versões.
- Contra:
 - ▣ A documentação é muito conceitual e pouco prática.
 - ▣ Longa curva de aprendizado.
 - ▣ Longos ciclos de versionamento – a maior parte das atualizações ocorrem anualmente.

Java EE



Referências

- [Alexander et al., 2007] Alexander, C.; Ishikawa, S.; Silverstein, M.; Jacobson, M.; Fiksdahl, I. e Angel, S. (2007). A Pattern Language. Oxford University.
- [Clements et al., 2006] Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R. e Staord, J. (2006). Documenting Software Architectures. Addison Wesley, 8 edição.
- [Gamma et al., 2003] Gamma, E.; Helm, R.; Johson, R. e Vlissides, J. (2003). Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley.
- [IEEE, 2000] IEEE (2000). Standard for Modeling and Simulation. IEEE, Disponível em: <http://shop.ieee.org/store>.