Programação Web com Servelets, JSP e JSTL

Prof: Rafael L. Bernardes Lima

rafaelbernardes0@gmail.com

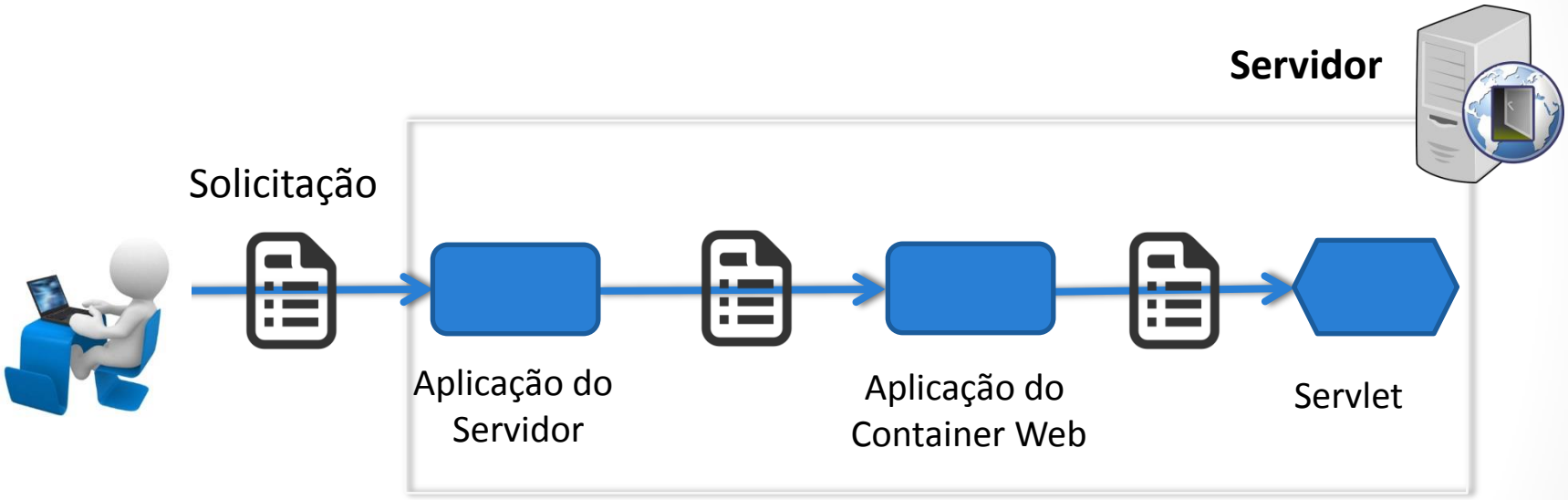
Centro Universitário do Triângulo
Pós-Graduação Java Unitri

Unitri

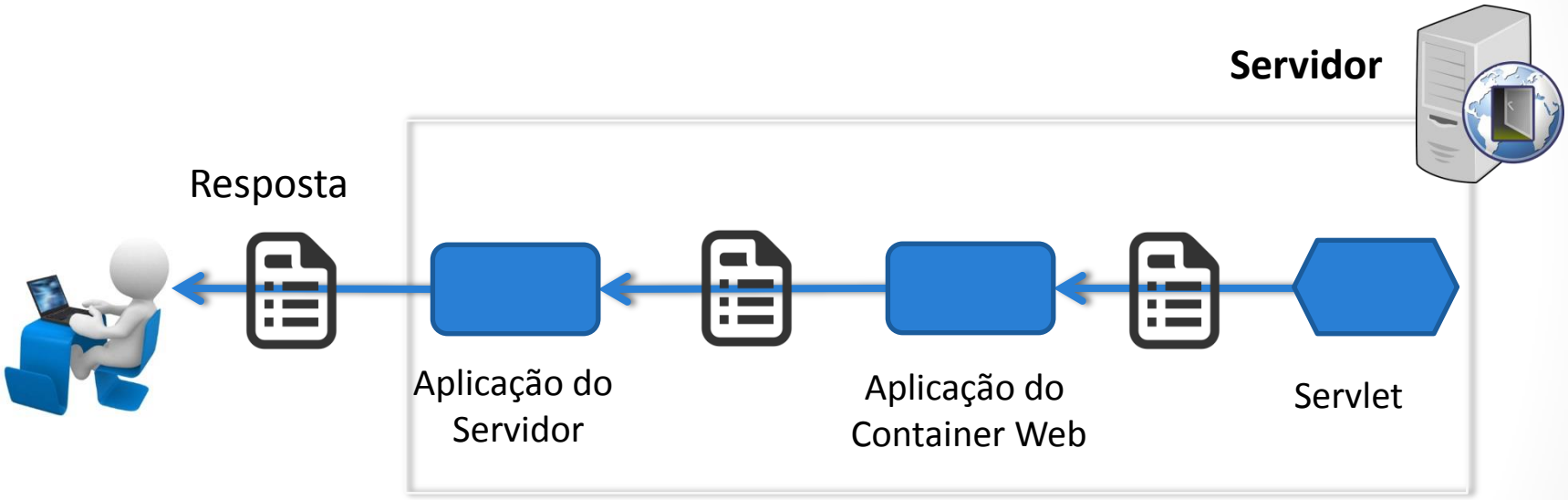
Agenda de Hoje

- Container
- Servlet
  - Deployment Descriptor
- MVC

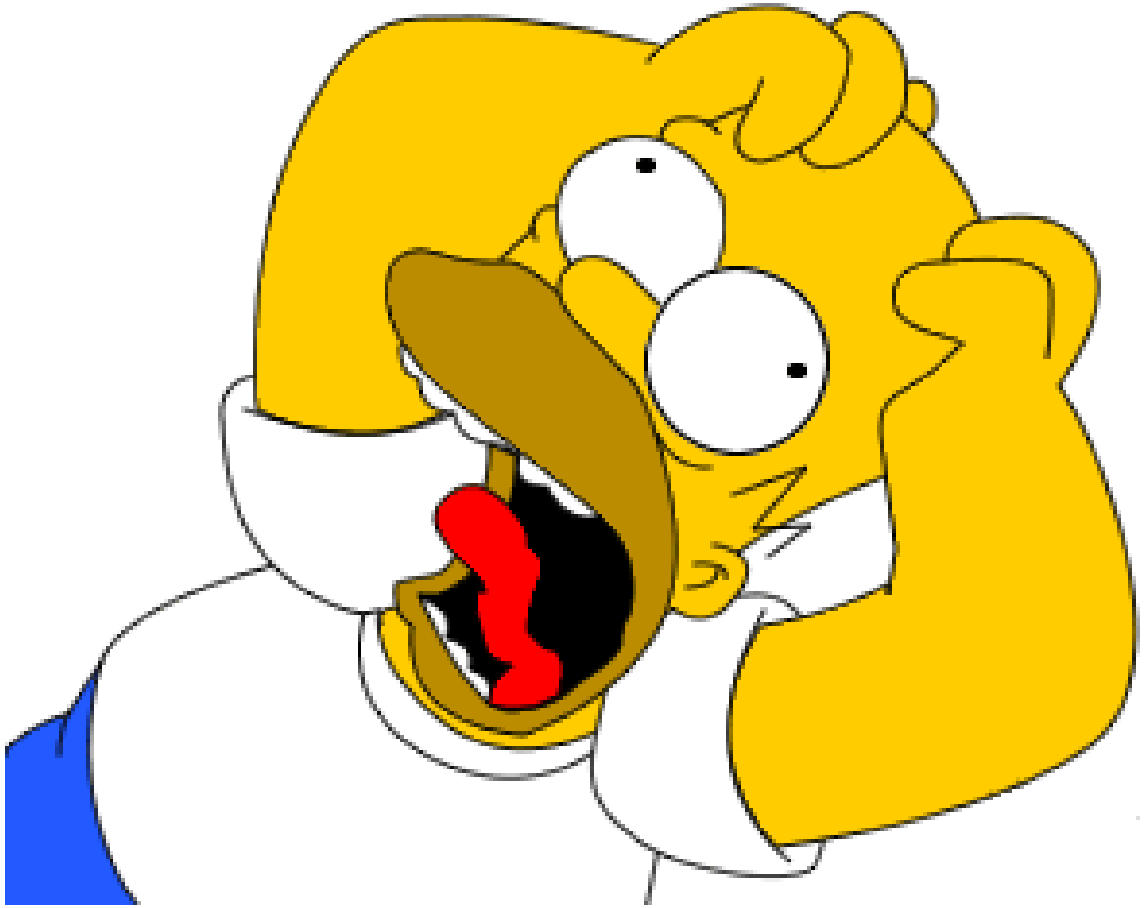
O que é container?



O que é container?



E se não existisse um Container?



O que é um Container?

O que é um Container?

- **Servlets não possuem método main(). Eles estão sobre o controle de outra aplicação Java chamada Container.**
  - Tomcat é um exemplo de container.
- Quando sua aplicação web recebe uma solicitação para o *Servlet*, o servidor entrega a solicitação para o *container* no qual o *servlet* é distribuído.
- É o container que entrega ao *servlet* a *request* e *response* HTTP, e chama os métodos do *servlet* (como o *doGet()* ou o *doPost()*).

O que é um Container?

- **É um software que disponibiliza *serviços*, tais como:**
  - Comunicação de rede que permite **receber e enviar requisições e respostas HTTP**.
  - **Transforma** mensagens HTTP em objetos Java e vice-versa
  - **Suporte** para comunicação
  - Controla todo o **ciclo de vida** dos *servlets*
  - Gerenciamento de ***threads*** para os *servlets*
  - Gerenciamento de **segurança**
  - Suporte ao **JSP**

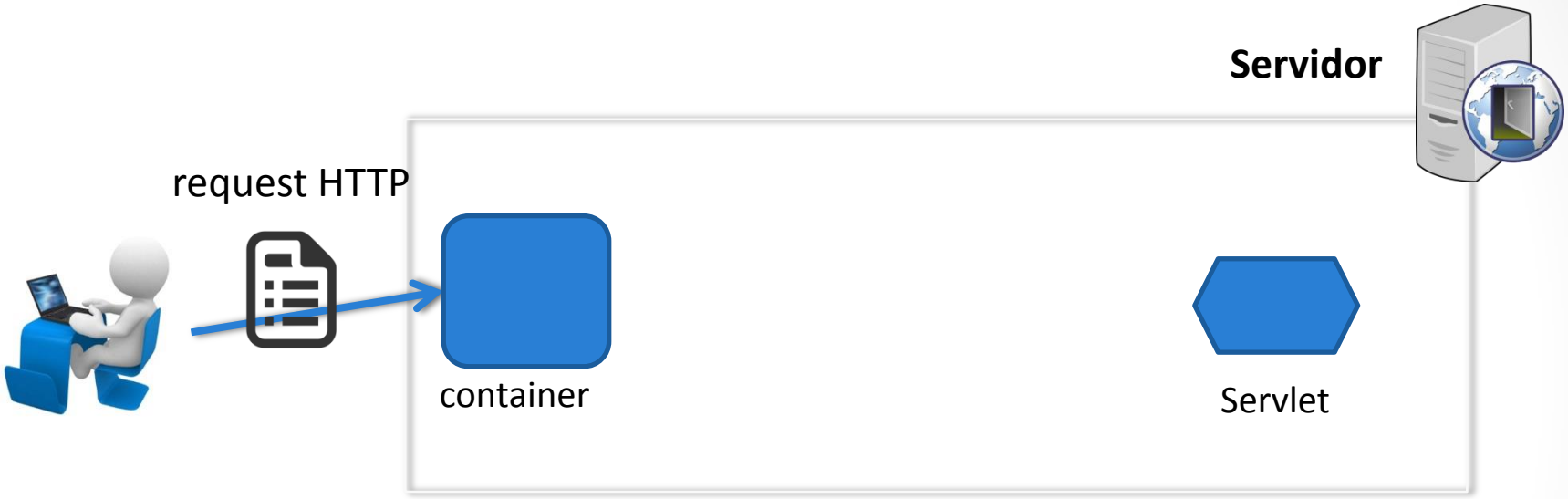
Alguns Containers Servlets

- Apache Jakarta TomCat
- Borland Enterprise Server
- Sybase EAServer
- Sun Java System Application Server
- Jetty
- Macromedia Jrun
- Oracle Application Server
- Resin
- WebLogic Application Server
- WebSphere

Como o Container trata uma solicitação?

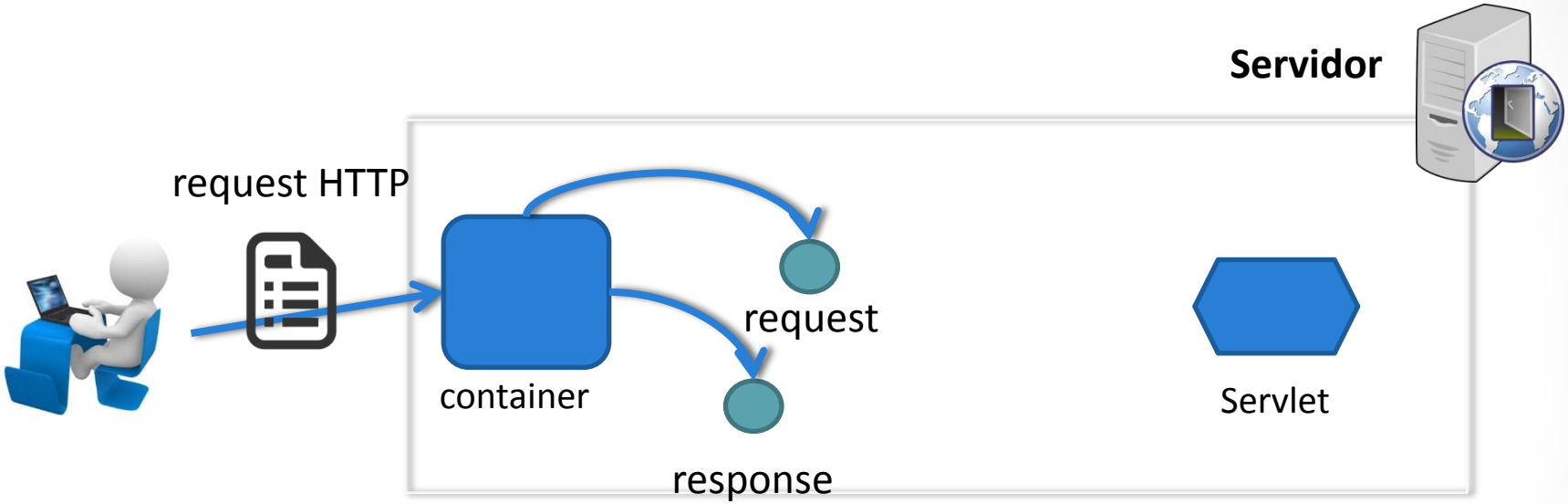
- O usuário **solicita**, através de uma página estática, um ***servlet***;
- O container "**vê**" que a solicitação é para um servlet e então **cria** dois objetos:
  - HttpServletResponse
  - HttpServletRequest
- O container **encontra** o servlet, **cria** um thread para a solicitação, e passa os objetos "**solicitação**" e "**resposta**" para o thread do servlet;

O que é container?



O usuário clica em um link que contém uma URL para um *servlet,* em vez de uma página estática

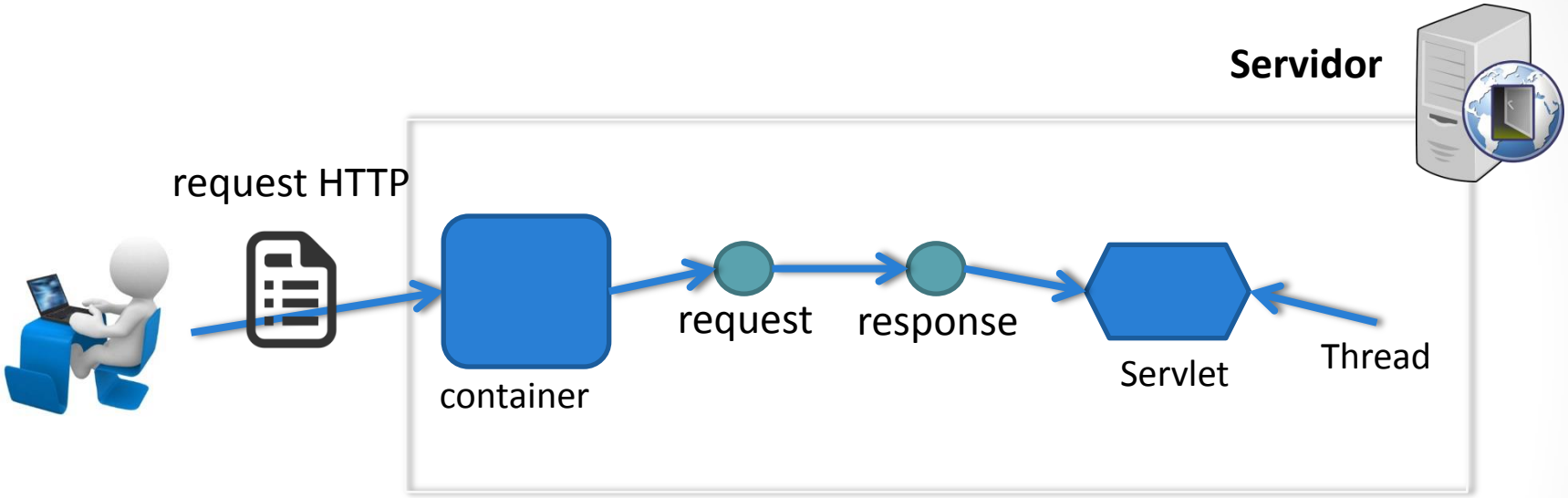
O que é container?



O container "**vê**" que a *request* é para um *servlet* e então ele cria dois objetos

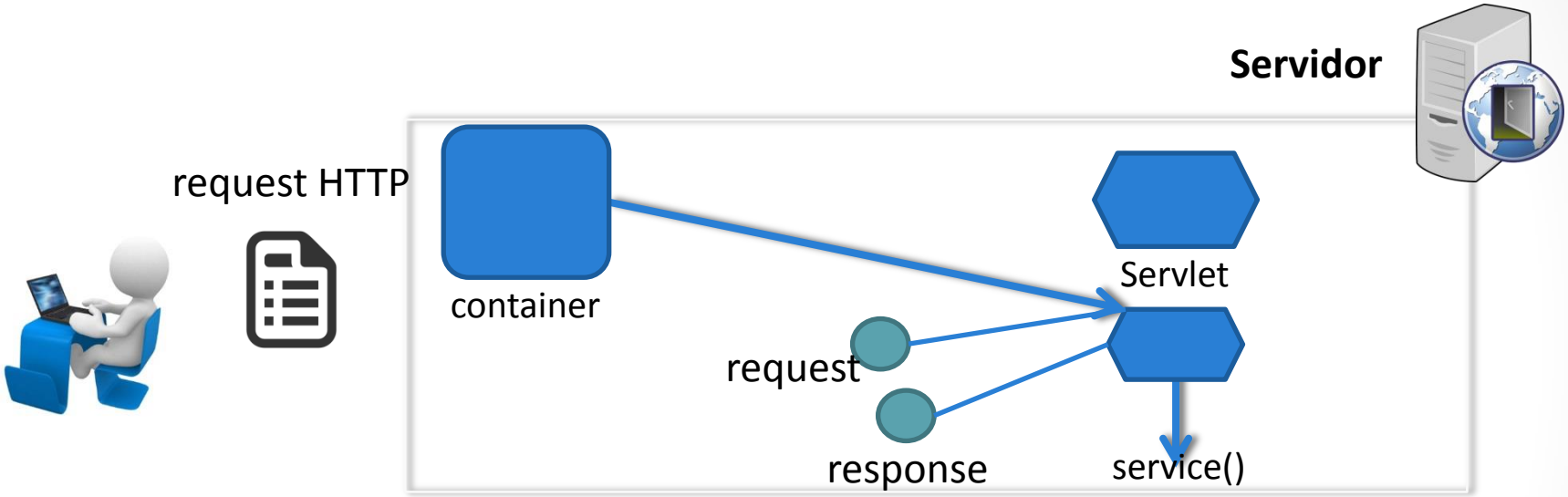
- HttpServletResponse
- HttpServletRequest

O que é container?



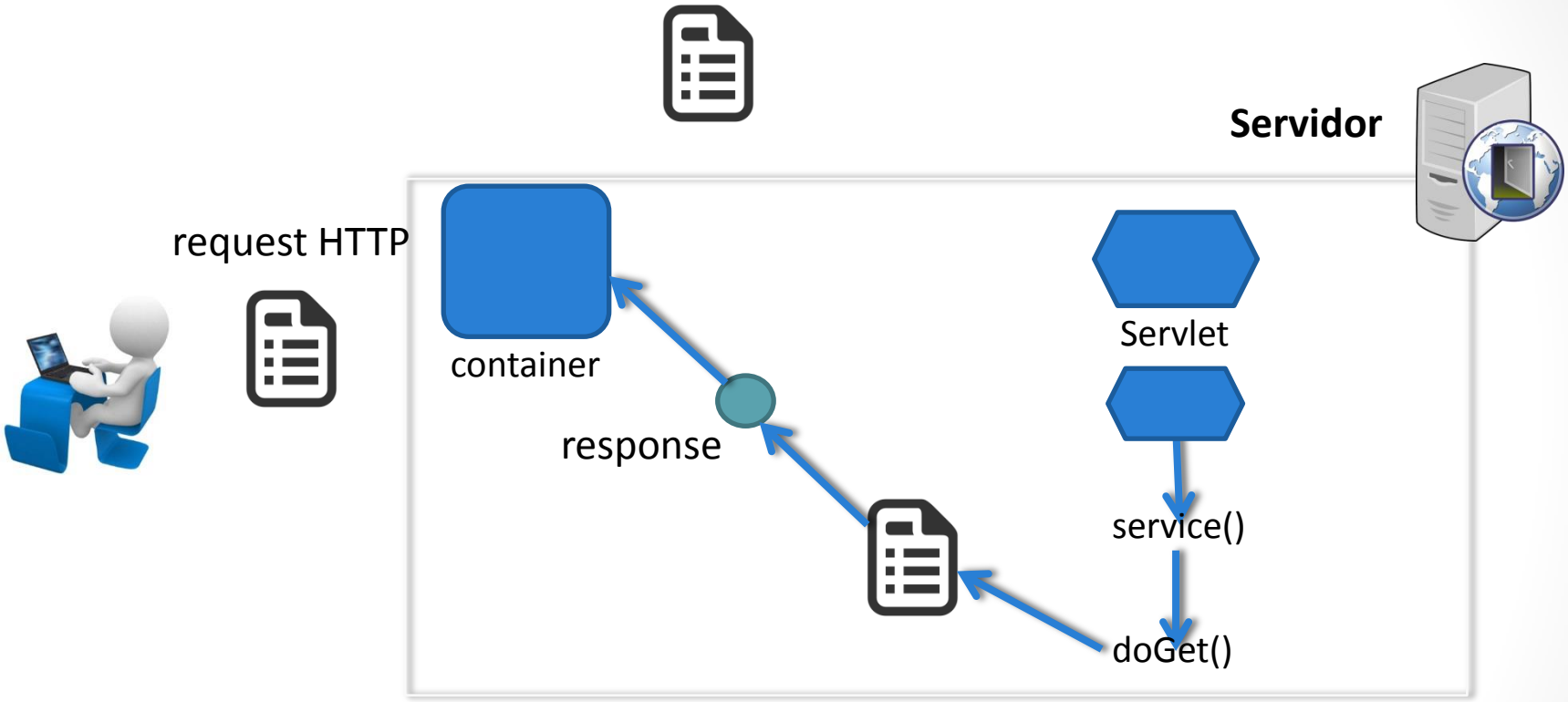
O container encontra o *servlet* correto baseado na URL da *request*, cria ou aloca uma thread para essa *request*, e passa os objetos *request* e response para a thread do *servlet.*

O que é container?



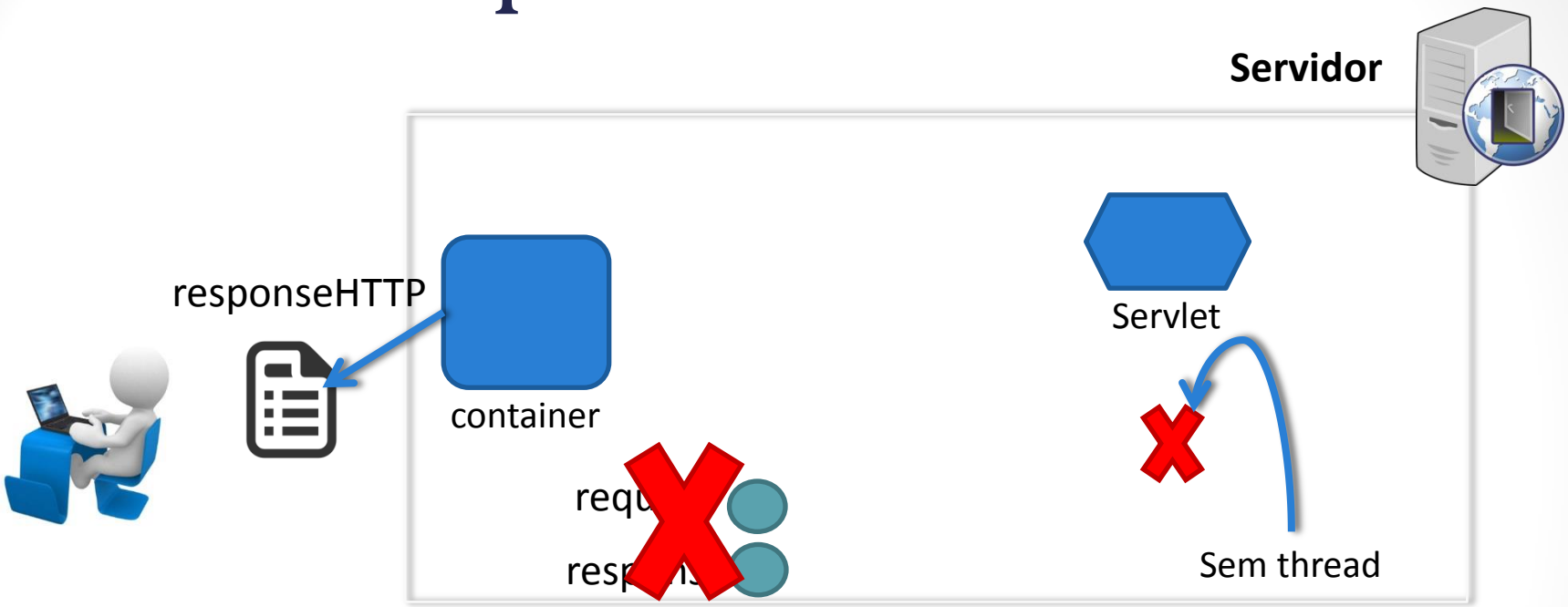
O container chama o método service() do servlet. Dependendo do tipo de request, o método service() chama ou o método doGet(), ou o método doPost(). Para este exemplo, consideraremos que a request foi um HTTP GET

O que é container?



O método doGet gera uma página dinâmica e a insere no objeto response. Lembre-se, o container ainda tem uma referência do objeto response

O que é container?



O thread termina, o container converte o objeto response em uma response HTTP, envia de volta ao cliente e apaga os objetos request e response.

O que é um *Servlet*

- É um **objeto** Java
- **Estende** as funcionalidades de um servidor Web
- **Gera** conteúdo web dinamicamente
- Possui **ciclo** de vida, gerenciado por um **container**
- Roda dentro da Java Virtual Machine (seguro e independente de plataforma e de servidor)

O que é um *Servlet*

- Toda a API Java é suportada
- Não tem método main
- Não exige suporte para Java no browser
- Eficiente e escalonável
- Especificação homologada pela JCP – Java Community Process

O que é um Container *Servlet*

- É um software que disponibiliza serviços, tais como:
  - Comunicação de rede que permite receber e enviar requisições e respostas HTTP
  - Transforma mensagens HTTP em objetos Java e vice – versa
  - Controla todo o ciclo de vida dos servlets
  - Gerenciamento de *threads* para os servlets
  - Gerenciamento de segurança

O código de um *Servlet*

No mundo real, 99,9% de todos os sevlets possuem ou o método doGet() ou o doPost()

99,9999% de todos os sevlets são HttpServlets.

Repare ... Nenhum método main(). Os métodos do ciclo de vida do servlet (como doGet) são Chamados pelo container

É aqui que seu servlet Consegue as referências dos Objetos request e response que O container cria

Você pode conseguir um PrintWriter do objeto response que o seu Servlet recebe do container. Utilize o PrintWriter para escrever texto HTML no objeto response.

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class MeuServlet extends HttpServlet
{
        public void doGet( HttpServletRequest request, HttpServletResponse
      response)
        throws ServletException, IOException {
        resposta.setContentType("text/html");
        PrintWriter saida = resposta.getWriter();
        java.util.Date today = new java.util.Date();
        saida.println("<html>");
        saida.println("<head><title>Meu primeiro servlet</title></head>");
        saida.println("<body>");
        saida.println("<b>Alo Mundo, hoje é: </b>" + today);
        saida.println("</body></html>");
  }
}
```

Como encontrar um servlet?

- Nome da URL conhecido pelo cliente (nome falso)
- Nome interno secreto conhecido pelo distribuidor (nome falso do arquivo e URL)
- Nome do arquivo verdadeiro.

Como encontrar um servlet?

- “projeto/MeuServet”
- “MeuServletSecreto”
- “MeuServlet.class”

Porque devo mapear meu Servlet?

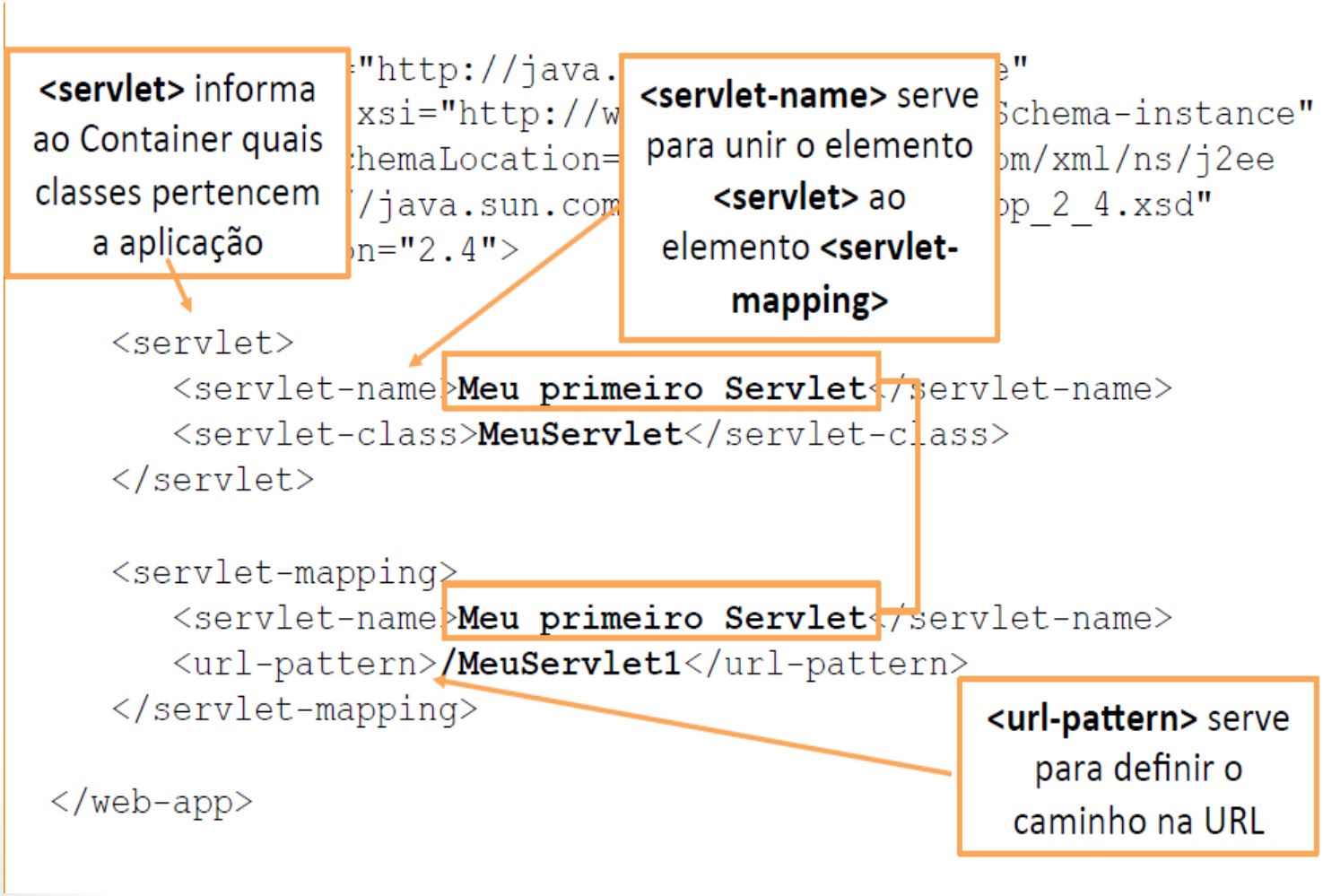
Porque devo mapear meu Servlet?

- **Flexibilidade**
  - Facilidade em mover arquivos de diretórios para organizar uma nova estrutura
- **Segurança**
  - evitar que os clientes acessem diretamente o servlet podendo gerar uma requisição inválida

Usando o Deployment Descriptor (DD) para mapear

- **<servlet>**
  - mapeia o nome interno para o nome da classe completamente qualificado
- **<servlet-mapping>**
  - mapeia o nome interno para o nome público de URL

Deployment Descriptor



Benefícios do DD

- Reduz a necessidade de alteração do código - fonte que foi testado;
- Permite que você ajustes os recursos da aplicação, mesmo que você não possua o código - fonte;
- Permite que você adapte sua aplicação de acordo com diferentes recursos sem ter necessidade de recompilar e testar nenhum código;
- Facilita a manutenção das informações dinâmicas sobre segurança como: lista de controle de acessos e funções de segurança;
- Permite que aqueles que não sejam programadores modifiquem e distribuam suas aplicações.

Prontos para criar uma aplicação?

- Servlet e JSP
  - O que irá conter?
  - O que irá fazer?

E onde fica a regra de negócio?

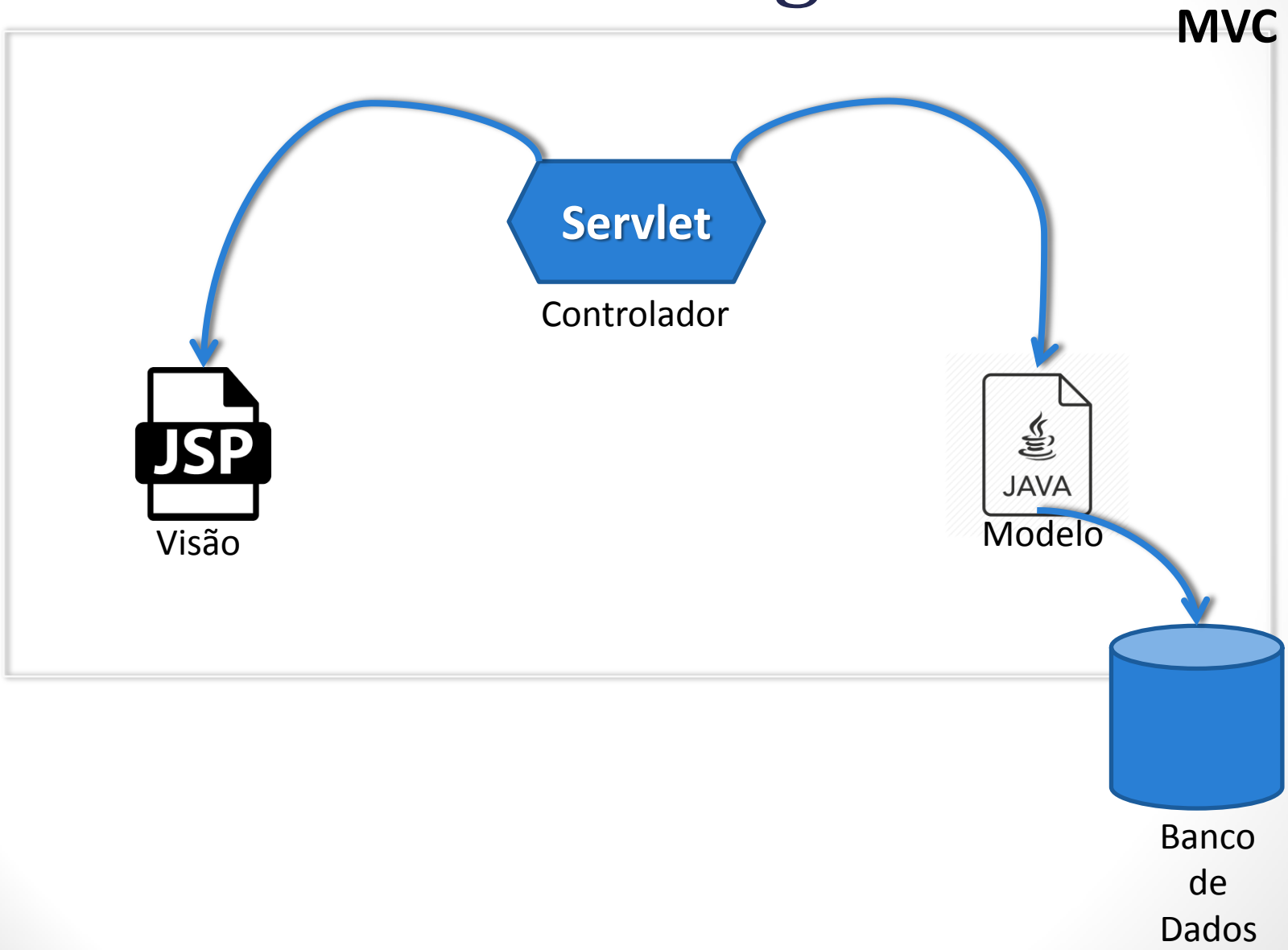
Regra de Negócio

- Devemos separar a “regra de negócio” da aplicação para darmos mais flexibilidade.
- Uma aplicação servlet/JSP não será acessada, exclusivamente, via web.
- Podemos também ter acesso por uma aplicação GUI (Graphical User Interface) Swing.

Padrão de Design MVC

- O MVC (Model – View – Controller) retira a lógica de negócio fora do servlet e a coloca em um "**modelo**";
- O modelo é a combinação dos dados de negócio e os métodos que operam nesses dados.

Padrão de Design MVC



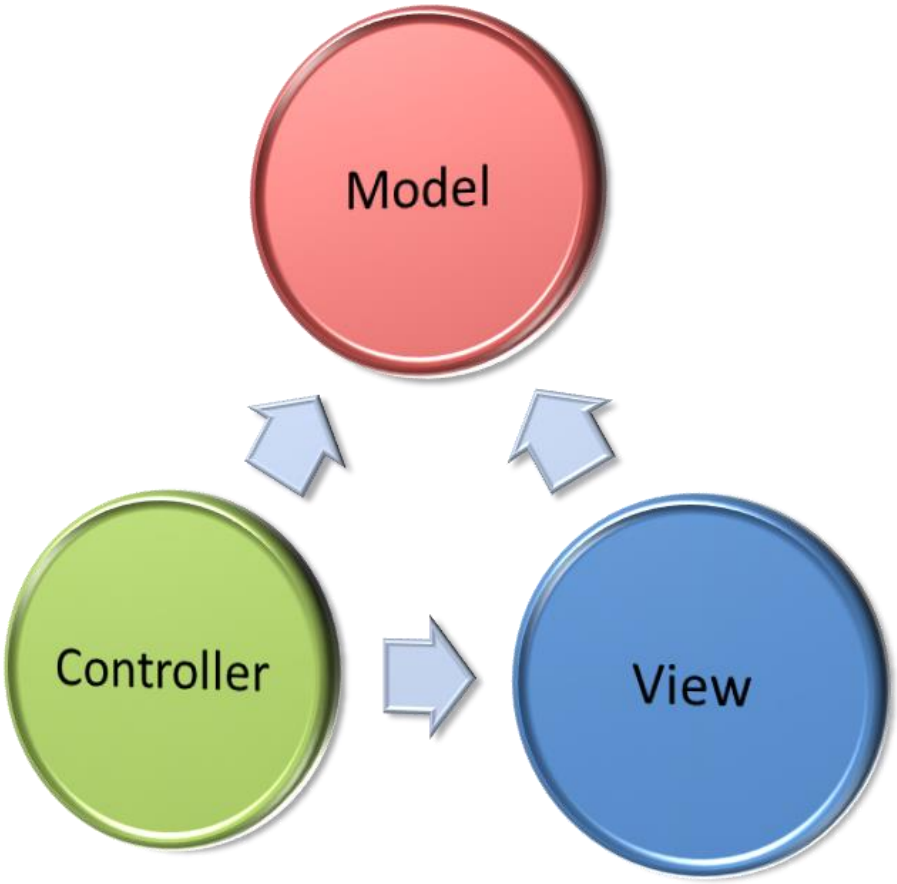
Exercício

- Quem é responsável?
  - Preencha a tabela, indicando se o responsável é o servidor, container ou servlet.
  - Em alguns casos, mais de uma opção é verdadeira. Preencha justificando a resposta.

Exercício

TAREFA	SERVIDOR	CONTAINER	SERVLET
Cria os objetos, solicitação e resposta			
Chama o método service()			
Inicia uma nova thread para tratar as solicitações			
Converte um objeto resposta em uma resposta HTTP			
Conhece HTTP			
Acrescenta o HTML ao objeto resposta			
Tem uma referência para o objeto resposta			
Encontra a URL no DD			
Apaga os objetos "solicitação" e "resposta"			
Coordena o preparo de conteúdos dinâmicos			
Gerencia os clicos de vida			
Tem um nome que coincide com o elemento <servlet-class> no DD			

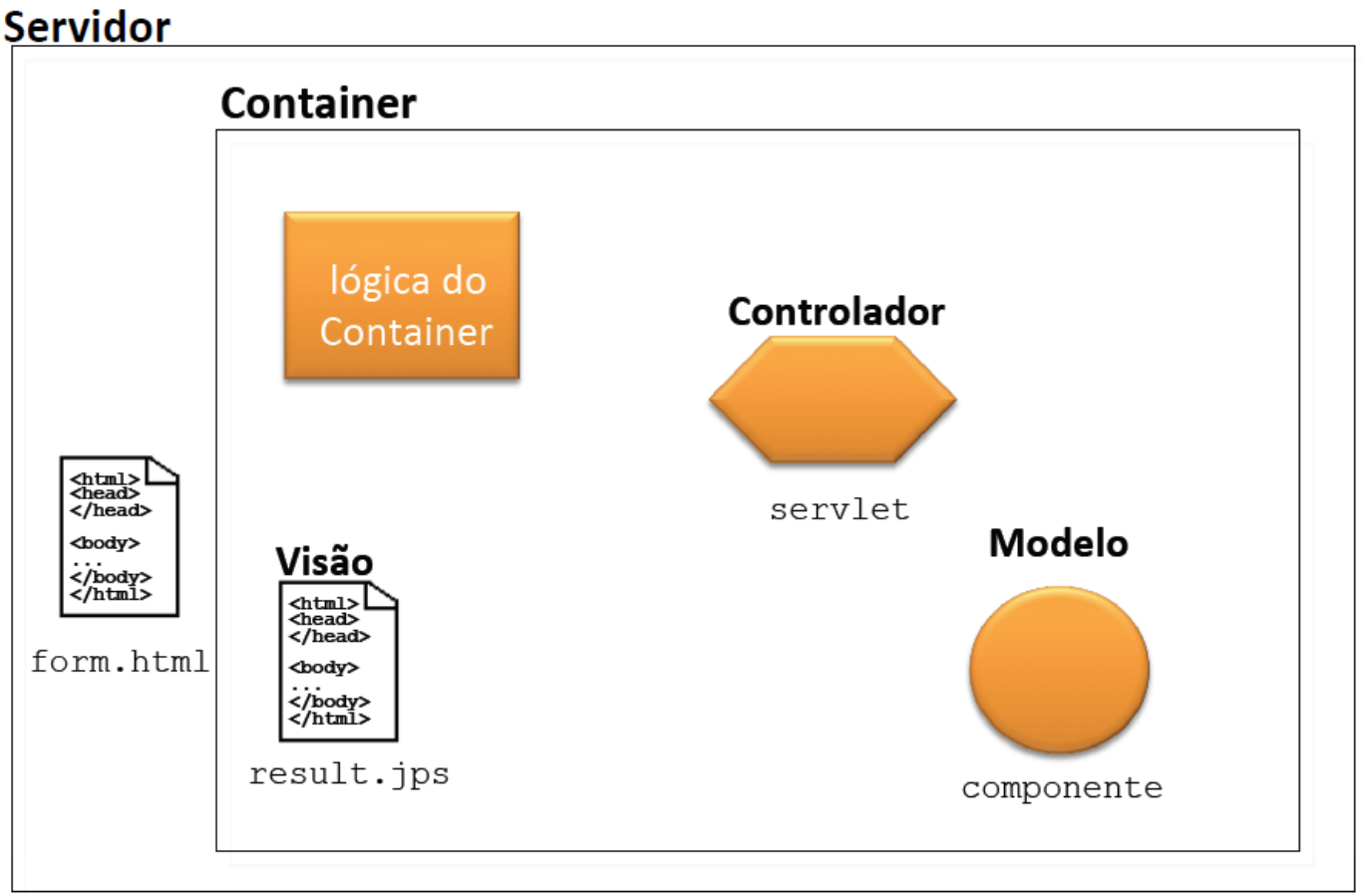
Construindo uma aplicação no modelo MVC



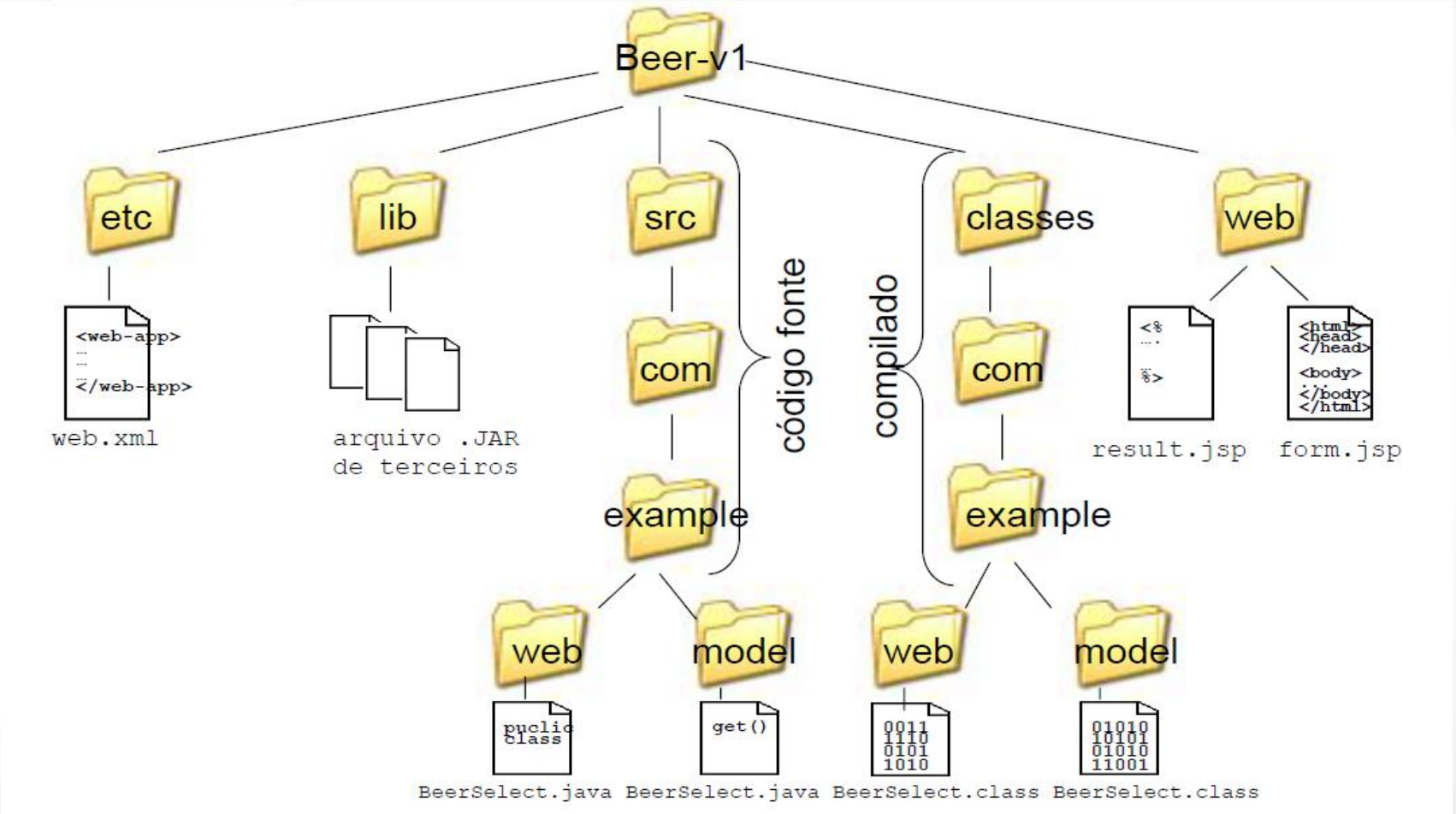
Etapas

- Criar as ***visões*** **do usuário**;
- Criar o **ambiente de *desenvolvimento***;
- Criar o **ambiente de *distribuição;***
- ***Desenvolver e testar***.

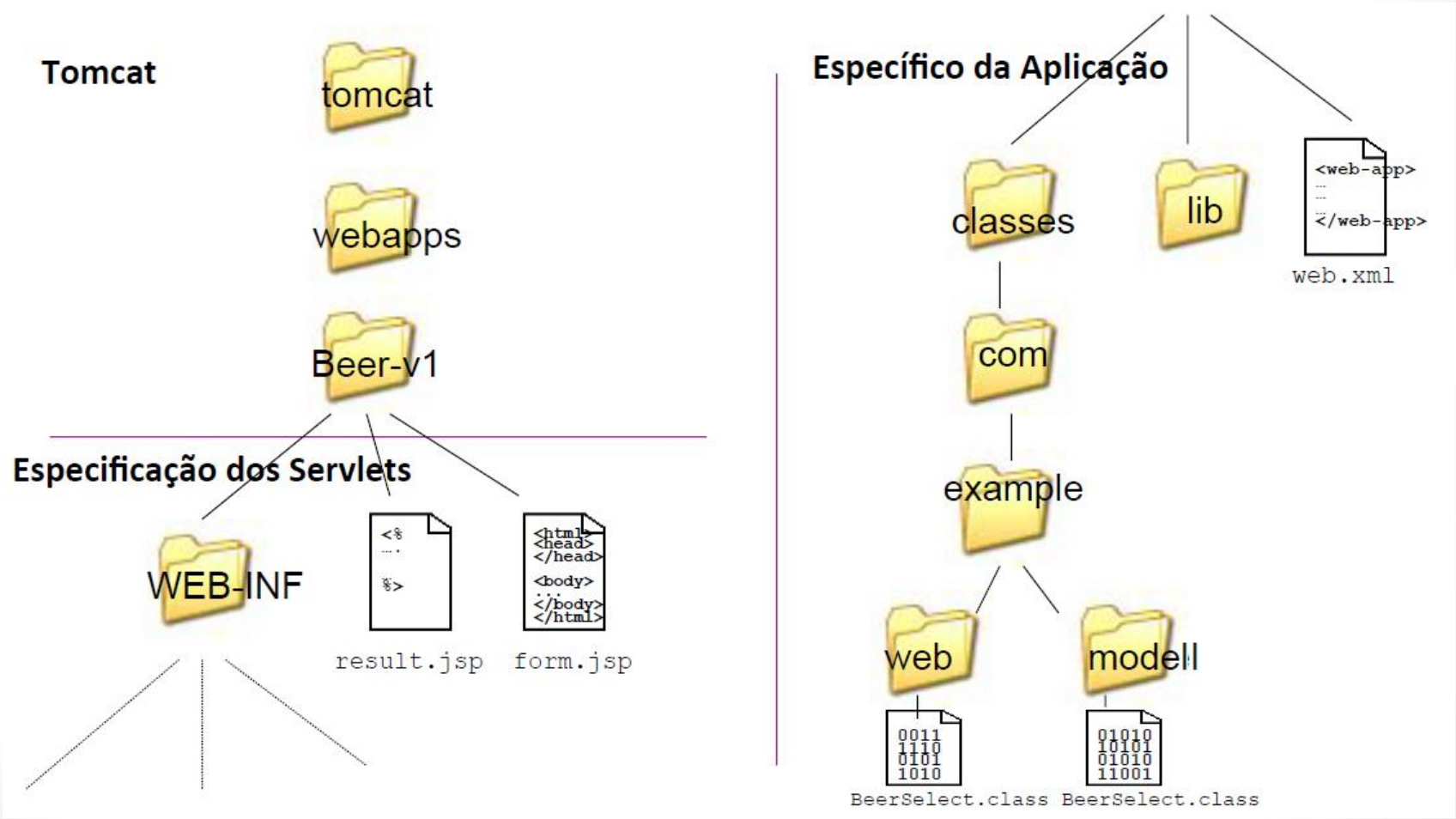
Arquitetura



Ambiente de Desenvolvimento



Ambiente de Desenvolvimento



Como gerar os .class

- Via *prompt* Entre na pasta onde seu projeto esta
  - cd MeusProjetos/ExemploProjeto
- Execute o comando **javac –classpath /users/joao/aplicacoes/tomcat/lib/servlet-api.jar;classes:. –d caminho pasta classes src/br/com/exemplo/web/MeuServlet.java**
- Acesse a pasta bin do tomcat e rode o arquivo startup.bat

Difícil?

Para nossa alegria

Existem as ferramentas de desenvolvimento

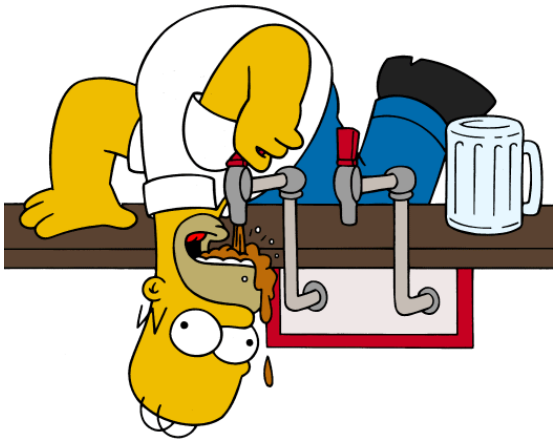
- A partir das ferramentas de desenvolvimento é possível gerar os arquivos.war, esses arquivos podem ser adicionados na pasta webapp do tomcat.
- No netbeens basta clicar com o botão direito sobre a pasta do projeto, selecionar a opção exportar e selecionar o tipo war.

Ainda não está satisfeito?

Existe um outra forma

- Como vimos no final da aula anterior é possível criar um servidor tomcat no eclipse adicionar o projeto a esse servidor e rodar o servidor, ele gera os arquivos necessários na pasta webapp do tomcat.

Aplicação



Especialista em Cervejas

Aplicação

- Nossa aplicação será uma especialista em cervejas;
- Os usuários poderão navegar em nossa aplicação, responder perguntas e receber conselhos valiosíssimos sobre cervejas.

Criar a Aplicação

- **Construir e testar o formulário HTML** que o usuário irá solicitar primeiro;
- **Construir e testar a versão 1 do servlet controlador** com o formulário HTML. Esta versão é invocada via formulário HTML e exibe o parâmetro que ela recebe.
- **Construir uma classe teste** para a classe modelo/expert, além de construir e testar a classe modelo/expert em si.
- **Atualizar o servlet para a versão 2**. Está versão nos dá a capacidade de chamar a classe modelo a fim de obter os conselhos sobre cerveja.
- **Construir o JSP e atualizar o servlet para a versão 3** (que agrega a capacidade de entregar para o JSP), e testar toda a aplicação.

Exercício

Mãos a obra!


That's All Folks