

Disciplina: Programação Orientada a Objetos

Aula 04- Revisão MVC e Primeiro Servlet

Model-View-Controller (MVC)

- É um padrão de arquitetura de software;
- Com o aumento da complexidade das aplicações desenvolvidas torna-se fundamental a separação entre os dados (**Model**) e os componentes de tela (**View**);
- Alterações realizadas no layout não afetam a manipulação de dados, e estes poderão ser reorganizados sem alterar o layout;
- Separação das tarefas de acesso aos dados e lógica de negócio, lógica de apresentação e de interação com o utilizador, introduzindo um componente entre os dois: o **Controller**.

MVC

■ **Model**

- Representação "domínio" específica da informação em que a aplicação opera. Por exemplo, **aluno, professor e turma** fazem parte do domínio de um sistema acadêmico.

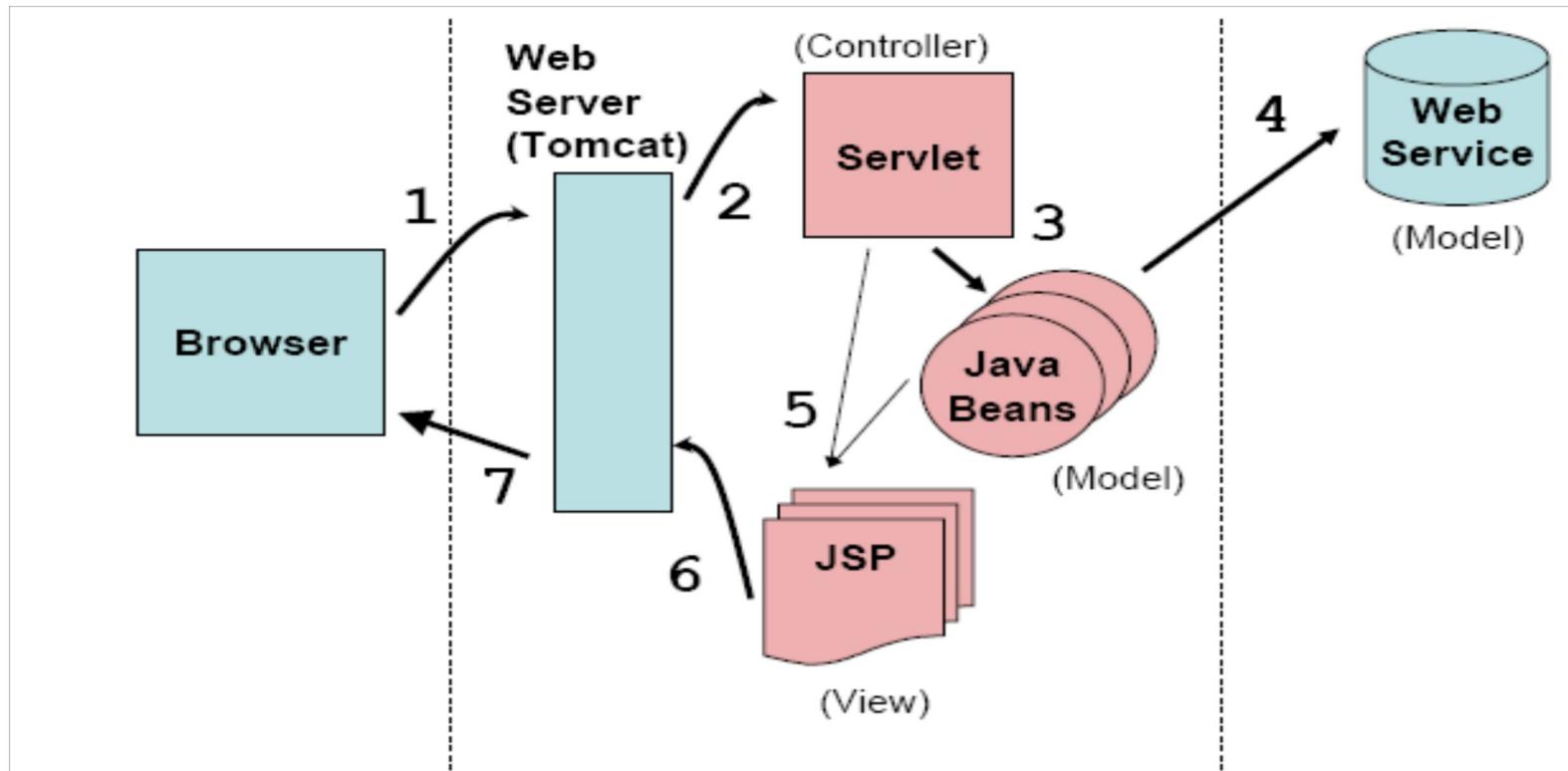
■ **View**

- "Renderiza" o *model* em uma forma específica para a interação, geralmente uma interface de usuário.

■ **Controller**

- Processa e responde a eventos, geralmente ações do usuário, e pode invocar alterações no *Model*. É lá que é realizada a validação dos dados

MVC para Aplicações Web



View - Páginas com resposta em HTML (JSP, ASP, PHP, JavaScript).

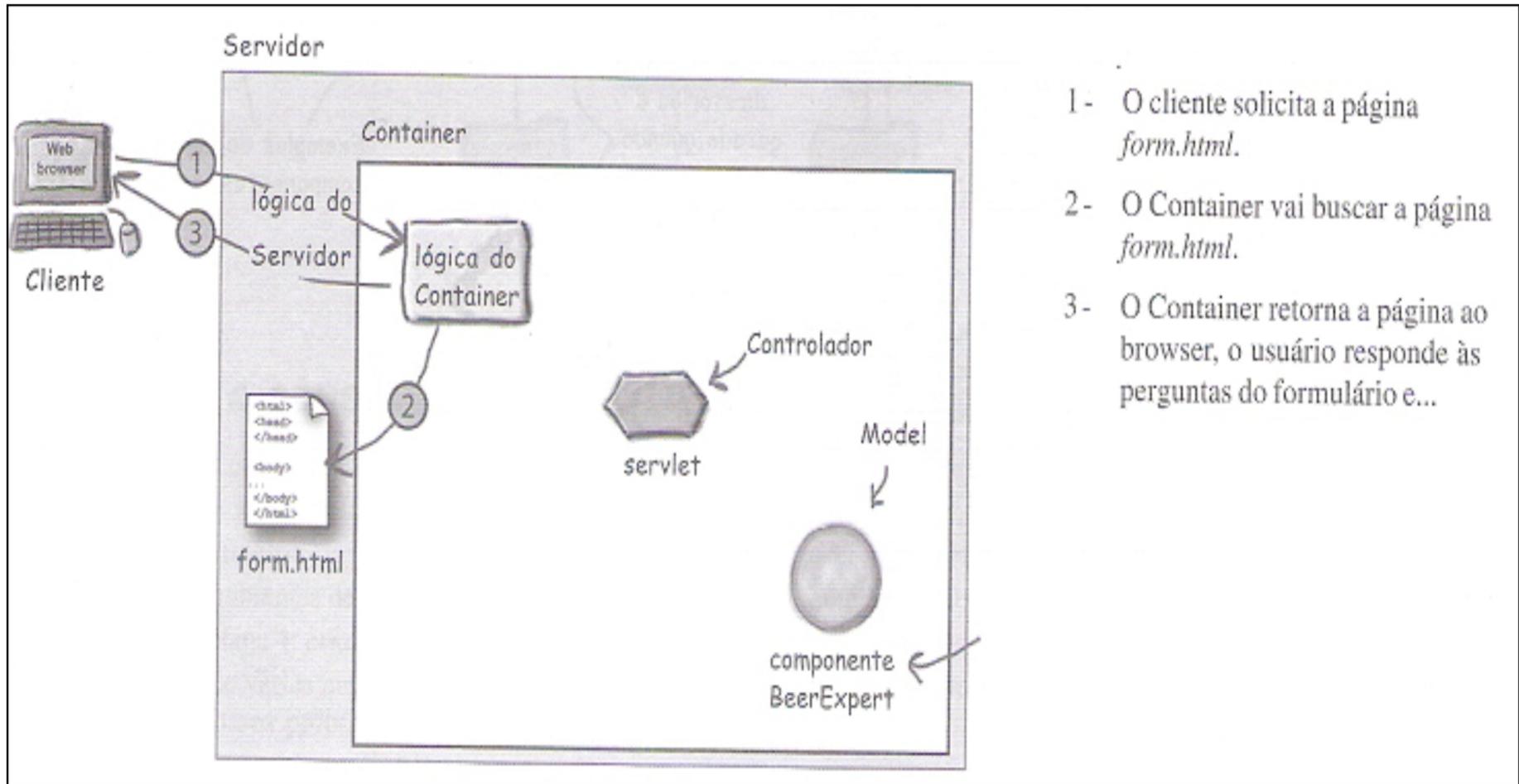
Controller - Controle da comunicação entre a **View** e o **Model**.

Model - Representação dos elementos do domínio e interação com as ferramentas de persistência.

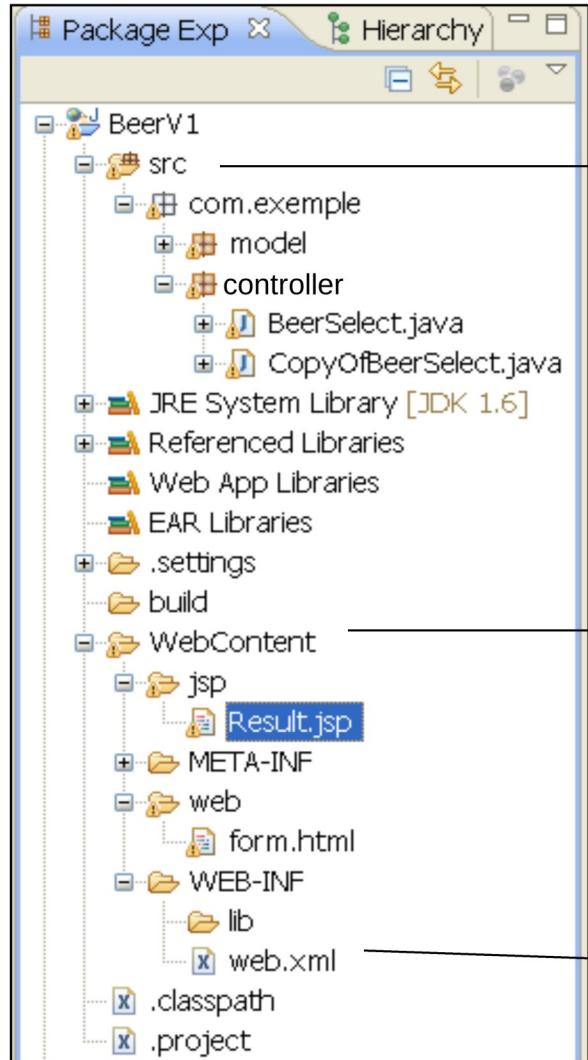
Exemplo MVC

- Exemplo de aplicação: *Beer Advisor* (Especialista em cerveja).
- Os usuários poderão navegar na aplicação, responder perguntas e receber conselhos **valiosíssimos** sobre cervejas.

Arquitetura MVC da Aplicação



Ambiente de Desenvolvimento



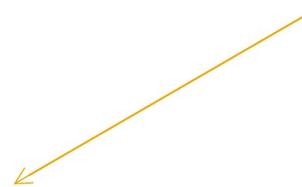
Todo código java

Seus componentes *view* estáticos e dinâmicos ficam aqui.

Arquivo de configuração

1. HTML do Formulário (View)

```
<html>
  <head>
    <title>Beer Expert</title>
  </head>
  <body>
    <form action="selectBeer" method="post">
      Select beer color:
      <select name="color" size="1">
        <option>light</option>
        <option>amber</option>
        <option>brown</option>
        <option>dark</option>
      </select>
      <input type="submit"/>
    </form>
  </body>
</html>
```



O HTML pensa que o servlet tem nome “selectBeer.do”.
Mas não há nada na estrutura de diretórios com esse nome.
Isso é um nome lógico...

Select beer characteristics

Color:

2. Código da *Servlet*

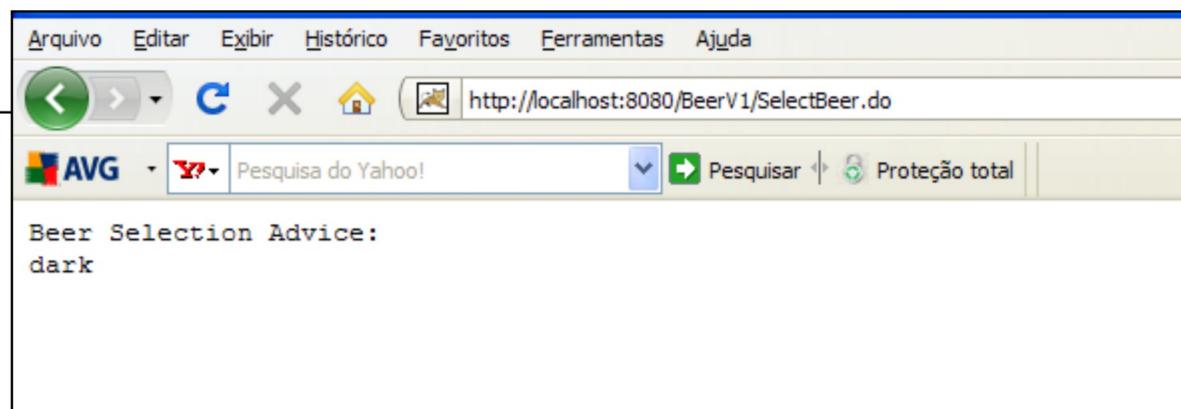
- Seu pacote deve ser:

```
public class BeerSelect extends HttpServlet{

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException{

        PrintWriter out = response.getWriter();
        out.println("Beer Selection Advice:");
        String c = request.getParameter("color");
        out.println(c);

    }
}
```



3. Criar o *Deployment Descriptor*

- Função do DD: definir mapeamento entre o nome lógico que o cliente usa na solicitação e o arquivo de classe *servlet* verdadeiro.

```
<servlet>
    <servlet-name>beer</servlet-name>
    <servlet-class>fatec.lp3.controller.BeerSelect</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>beer</servlet-name>
    <url-pattern>/selectBeer.do</url-pattern>
</servlet-mapping>
```

Este é um nome fictício que você usará APENAS em outras partes do DD.

Verificar se o caminho e nome da classe estão corretos!

Esse é o nome que queremos que o cliente se refira ao *servlet*.

4. Criando o Modelo

- Seu pacote deve ser: **fatec.poo.model**

```
public class BeerExpert {  
  
    public List<String> getBrands(String color) {  
        List<String> brands = new ArrayList<String>();  
        if (color.toLowerCase().equals("amber")) {  
            brands.add("Jack Amber");  
            brands.add("Red Moose");  
        } else {  
            brands.add("Jail Pole Ale");  
            brands.add("Gout Atout");  
        }  
        return brands;  
    }  
}
```

Nova Versão da Classe *BeerSelect*

```
public class BeerSelect extends HttpServlet {  
    @Override  
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)  
    throws ServletException, IOException {  
        PrintWriter out = resp.getWriter();  
        String color = req.getParameter("color");  
  
        BeerExpert beerExpert = new BeerExpert();  
        List<String> brands = beerExpert.getBrands(color);  
  
        out.print("<html><body>");  
        out.print("Try: ");  
  
        for (String brand : brands) {  
            out.print(brand + ", ");  
        }  
  
        out.print("</body></html>");  
    }  
}
```

Instancia a classe *BeerExpert*
e chama o método *getBrands()*