📐 Arquitetura MVC (Model-View-Controller)

O padrão **MVC** separa uma aplicação em três camadas principais:

Camada	Função
Model	Representa os **dados** e **regras de negócio** da aplicação. Ex: classes `Aluno`, `Curso`.
View	Interface com o **usuário**, ou seja, aquilo que é exibido no navegador.
Controller	**Controla a lógica** da aplicação. Recebe requisições do usuário e coordena as interações entre Model e View.

🎯 Vantagens do MVC

- Separação clara de responsabilidades.
- Facilita manutenção e testes.
- Permite que back-end e front-end evoluam de forma independente.

🏛️ Modelos de Arquitetura MVC em Web Java

📄 Model 1 (Page-Centric)

- JSP controla tudo: interface e lógica.
- Não há separação real entre Camadas.
- Difícil de manter em sistemas maiores.

📌 Muito usado no início dos anos 2000. Exemplo típico de aplicação com JSP + scriptlets.

📦 Model 2 (Servlet-Centric)

- O **Controller** é um Servlet central.
- Ele decide qual **View (JSP)** será exibida.

- O JSP mostra dados vindos do **Model**.
- Alta **separação de responsabilidades**.

📌 Padrão consagrado em frameworks como **Struts**, **Spring MVC** e **JSF**.

🧠 Como funciona o MVC no JSF

1. FacesServlet (Controller)

- Roteia todas as requisições `.xhtml`.
- Controla o **ciclo de vida** dos componentes da interface.
- Trata eventos, validações e conversões de dados.

2. Managed Beans (Model)

- Contêm dados e lógica de negócio.
- Os componentes de interface se conectam diretamente aos beans com `#{}` (EL).
- Exemplo:

```
@ManagedBean
@RequestScoped
public class AlunoBean {
   private String nome;
   // getter/setter + regras de negócio
}
```

3. Arquivos XHTML (View)

- Responsáveis pela interface gráfica.
- Utilizam **componentes JSF**, como `<h:form>`, `<h:inputText>`, etc.
- Ligam-se ao bean via EL:

```
<h:inputText value="#{alunoBean.nome}" />
```

🔁 Ciclo de vida de uma requisição JSF

1. **Requisição** chega ao `FacesServlet`.
2. JSF **constrói a árvore de componentes** (UIComponent Tree).
3. JSF **aplica validações**, conversões e executa ações.
4. Bean manipula os dados (**Model**).
5. JSF seleciona a **View** correta.
6. A View é **renderizada como HTML**.
7. O navegador do usuário **exibe o resultado**.

📌 Conclusão

O JSF aplica de forma clara o padrão MVC:

- `Model`: Beans + Regras de Negócio.
- `View`: Páginas XHTML com tags JSF.
- `Controller`: FacesServlet que cuida do ciclo de vida da requisição e das interações entre Model e View.

Exemplo de Aplicação JSF - Simulação de Login

Para compreender o funcionamento do **JSF (JavaServer Faces)**, apresentamos uma aplicação simples de login com três páginas:

- `index.xhtml`: tela de login.
- `sucesso.xhtml`: exibida quando o login é válido.
- `erro.xhtml`: exibida quando o login falha.

A aplicação utiliza **Facelets** (páginas `.xhtml`), que substituíram o uso de `.jsp` nas versões JSF 2.x em diante.

---

1. Página de Login (`index.xhtml`)

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
  <h:head>
    <title>Login</title>
  </h:head>
  <h:body>
    <f:view>
      <h:form>
        <h3>Por favor, informe seu usuário e senha:</h3>
        <table>
          <tr>
            <td>Nome:</td>
            <td><h:inputText value="#{usuarios.nome}"/></td>
          </tr>
          <tr>
            <td>Senha:</td>
            <td><h:inputSecret value="#{usuarios.senha}"/></td>
          </tr>
        </table>
        <p>
          <h:commandButton value="Login" action="#{usuarios.validarUsuario}"/>
        </p>
      </h:form>
    </f:view>
  </h:body>
</html>
```

Destaques:

- **`<f:view>`**: define o contexto JSF da página.
- **`<h:form>`**: substitui o `<form>` tradicional do HTML.
- **Componentes de entrada**: `<h:inputText>` e `<h:inputSecret>` ligados ao bean `usuarios`.
- **Ação de login**: `<h:commandButton>` chama o método `validarUsuario()` no bean.

---

2. Bean Gerenciado (`UsuariosBean.java`)

```
@ManagedBean(name = "usuarios")
@SessionScoped
public class UsuariosBean {
   private String nome;
   private String senha;

   // Getters e Setters omitidos por brevidade

   public String validarUsuario() {
      if ("user".equals(nome) && "123".equals(senha)) {
         return "/sucesso";
      } else {
         return "/erro";
      }
   }
}
```

Explicação:

- **@ManagedBean**: registra a classe como bean JSF acessível via `#{usuarios}`.
- **@SessionScoped**: mantém os dados do bean enquanto a sessão do usuário estiver ativa.
- **`validarUsuario()`**: valida as credenciais e redireciona para a página correspondente.

---

3. Páginas de Resposta

Sucesso (`sucesso.xhtml`)
<h3>Seja bem-vindo, <h:outputText value="#{usuarios.nome}"/></h3>

Exibe o nome do usuário autenticado.

Erro (`erro.xhtml`)
<h3>Usuário ou senha inválidos!!</h3>

Informa erro na autenticação.

Conceitos Importantes de JSF

Namespaces utilizados

- `http://java.sun.com/jsf/html`: para componentes de interface como `<h:form>`, `<h:inputText>`, etc.
- `http://java.sun.com/jsf/core`: para componentes estruturais como `<f:view>` e manipulação avançada.

Categorias de Tags JSF

1. **Tags Fundamentais (`jsf/core`)**:
    - Controlam fluxo da página, recursos, mensagens e visões.
2. **Tags de Interface (`jsf/html`)**:
    - Componentes visuais como formulários, inputs, botões, mensagens e tabelas.

✅ Conclusão

Esse exemplo de login com JSF demonstra de forma prática os principais conceitos:

- Criação de **páginas com Facelets**.

- Uso de **beans gerenciados** com escopo de sessão.
- Integração entre **interface e lógica de negócio** via EL (`#{}`).
- Navegação **baseada em retorno de métodos** do bean.

Esse tipo de estrutura é comum em aplicações web com JSF e serve como base para projetos maiores com controle de sessão, autenticação e fluxo de navegação.