Lição 9

MVC Frameworks II – JavaServer Faces

Objetivos

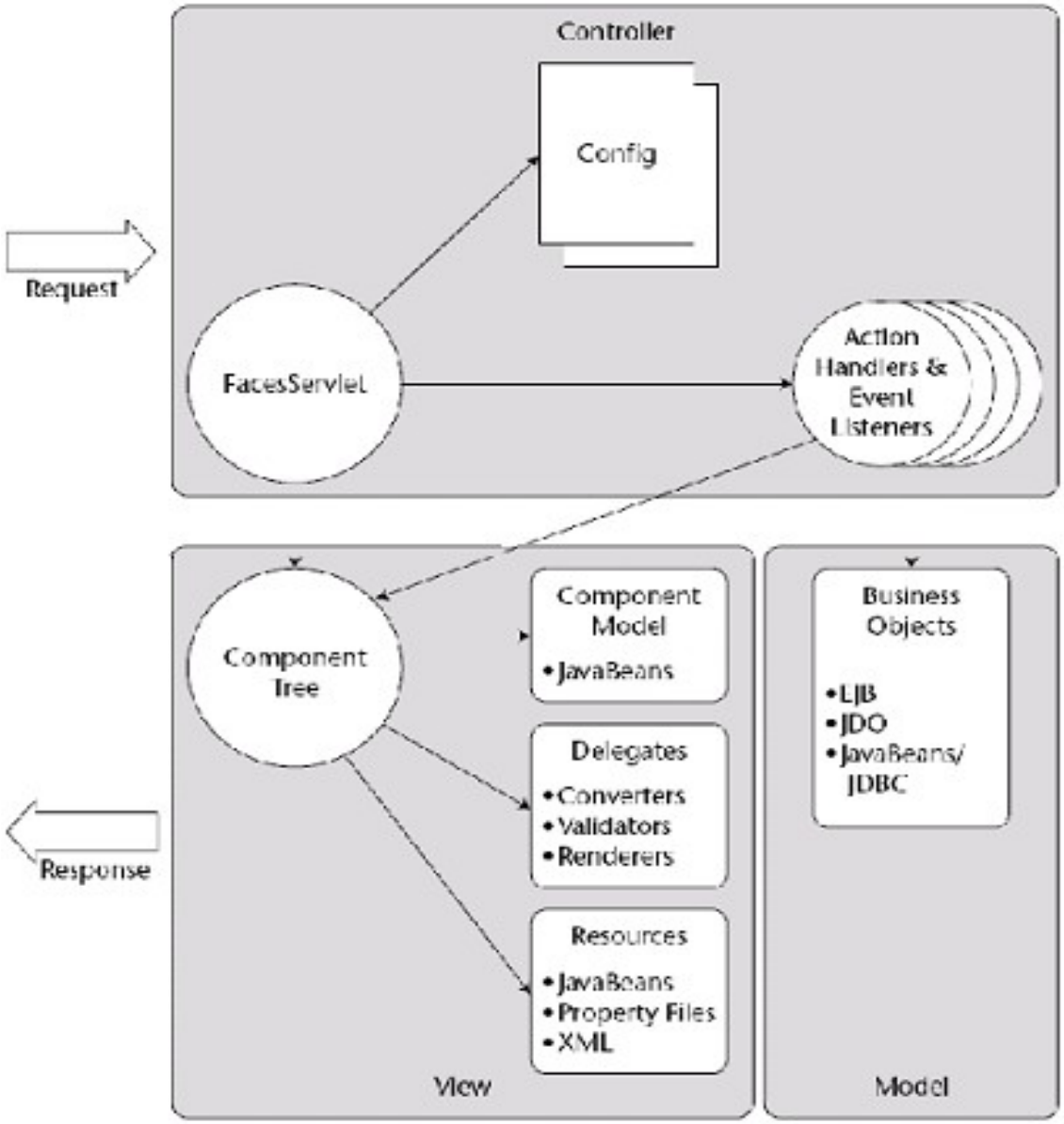
Ao final desta lição, o estudante será capaz de:

- Entender a estrutura MVC para a JSF
- Visão de outros componentes de *tags* JSF

JavaServer Faces

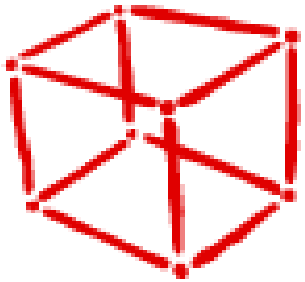
- *Framework* para construção da interface de usuário em aplicações WEB
- Idealizado sobre os conceitos introduzidos pelo *framework Struts*
- Benefícios de uma arquitetura que separa a lógica de apresentação da lógica de negócio e de uma interface padrão de usuário
- Baseada em componente muito similar ao padrão Swing

JavaServer Faces



JavaServer Faces

- Passaremos agora para o NetBeans



built on NetBeans

JSF e Struts

- Semelhanças:
  - Clara separação entre os componentes usados nas camadas
  - JSF também tem um controle frontal, chamado *FacesServlet*
  - Fazem uso de manipuladores de ações separados da *Servet*
- **Diferenças:**
  - *Struts* provê apenas um conjunto de biblioteca de *tags* adicionado em cima da funcionalidade do HTML padrão
  - JSF provê seu próprio conjunto de componentes de interface

Controller

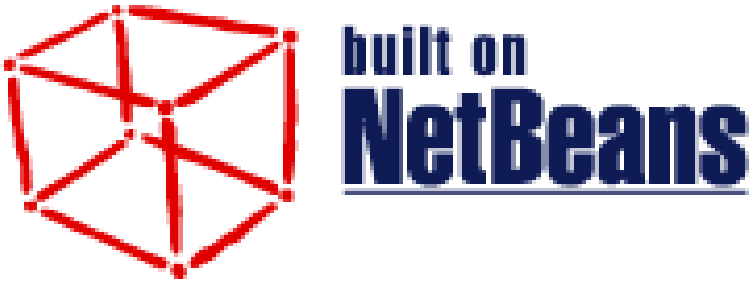
- A camada *controller* do JSF é composta de:
  - *Servlet* de controle (*FacesServlet*)
  - Conjunto de arquivos de configuração em XML
  - Conjunto de manipuladores (*handlers*) de ação

FacesServlet

- Responsável pela recepção das requisições do cliente e pela execução das operações necessárias para produção da resposta
  - Preparar os componentes de UI da requisição
  - Atualizar do estado dos componentes
  - Chamar os manipuladores de ação requisitados
  - Renderizar os componentes UI que são parte da resposta
- É provido pelo *framework* JSF
- Requer configuração do descritor de operação (*deployment)* da aplicação

Configurando o FacesServlet

- Passaremos agora para o NetBeans



Manipuladores de Ação

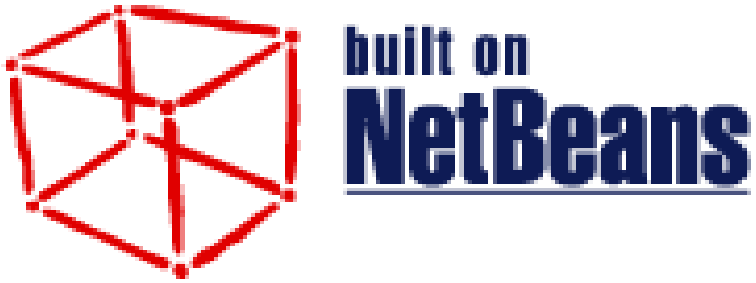
- Formas de criar um manipulador de ação:
  - Ligação (*binding*) de um método de um *JavaBean* para servir como o manipulador de ação
  - Criação de uma instância de uma classe que implementa a interface *ActionListener*

Método *Application*

- Método ligado ao componente UI para servir como seu manipulador de ação
- Regras requeridas para criar um método *application*:
  - Deve ser declarado *public*
  - Não conter parâmetros
  - Tipo de retorno *String*

Método *Application*

- Passaremos agora para o NetBeans



Trabalhando com Objetos de Escopo de Sessão no JSF

- *Framework* JSF possui um método diferente para acessar o escopo de sessão
- Provê acesso para o contexto de sessão (bem como a outros contextos) com o uso do objeto FacesContext

ActionListener

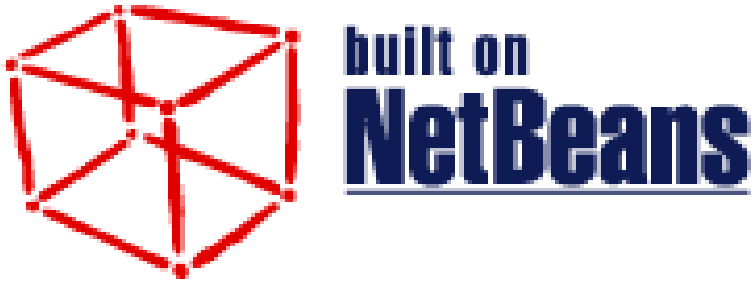
- Outra maneira de implementar um manipulador de ação é criar uma classe que implementa a interface *ActionListener*
- Esta interface define um único método:

```
public void processAction(ActionEvent event)
```

  - Objeto *ActionEvent* : Provê a implementação da classe de acesso ao componente que causou o evento

ActionListener

- Passaremos agora para o NetBeans



Os Métodos *Application* e *ActionListeners*

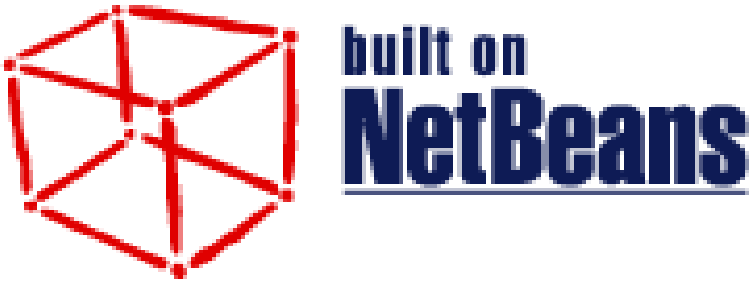
- É mais apropriado usar *application* como manipuladores de ação
  - Localizam na mesma classe a qual serve como o *backing model* de um formulário
  - Permitem agrupar o dado e os métodos que operam sobre ele em uma classe
  - Retornam "saídas" as quais informam ao *FacesServlet* a próxima *view* a ser mostrada
- *ActionListeners* são a escolha mais apropriada embora, para refatorar as funcionalidades comuns deve-se reusar múltiplas fontes de ação

faces-config.xml

- Arquivo de configuração mais importante para a camada de controle do *Framework* JSF
- Contém entradas de configuração para as regra de navegação e para os *JavaBeans* que serão reconhecidos pelo *framework*

faces-config.xml

- Passaremos agora para o NetBeans



Controller

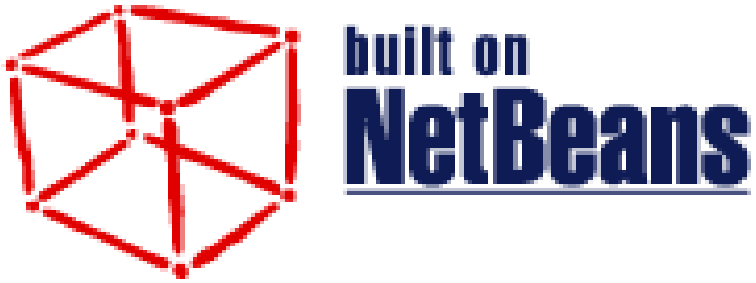
- Para cada *setup*:
  - Configure o *FacesServlet* para uso na sua aplicação.
- Para cada página WEB contendo UI componentes no JSF:
  - Criar uma entrada de configuração para o bean gerenciado que servirá como *backing model* da página
  - Criar regras de navegação as quais definem onde a aplicação poderia possivelmente ir depois da página corrente

Model

- *Backing Model*
  - Classes que armazenarão o estado dos componentes UI em cada página
  - Não existem classes de *model* quando vemos estritamente sob a perspectiva da arquitetura MVC
  - Entretanto, quando pensamos apenas nos componentes UI, faz sentido chamá-los como parte do Modelo
  - Tomar cuidado no desenvolvimento destas classes para que não influenciam nas funcionalidades centrais da aplicação
- Criando um *Backing Model*:
  - Forma idêntica a criar um *JavaBean* com propriedades correspondendo a cada componente na página

Model

- Passaremos agora para o NetBeans

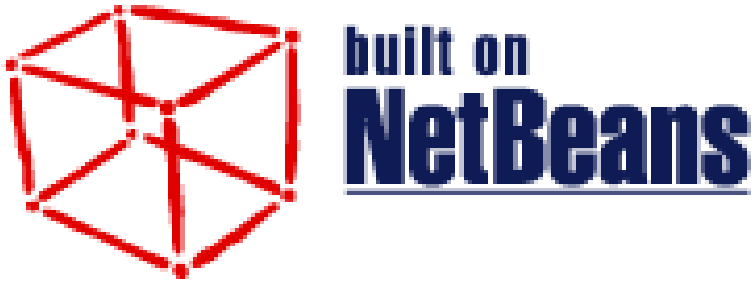


View

- Camada onde o JSF mais deixa sua marca
- Fornece as *tags* customizadas que podem ser usadas para mostrar nossa interface usando JSPs
- Fornece um conjunto de componentes
- Fornece uma API padronizada para acesso e manipulação

Integração JSF-JSP

- Passaremos agora para o NetBeans



Value Binding

- Como ligar nossos componentes ao *backing model*
- Do exemplo anterior:

```
...
<h:inputText id="loginName"
  value="#{loginPage.loginName}"/><br/>
<h:outputLabel for="password">
<h:outputText value="Password : "/>
</h:outputLabel>
<h:inputSecret id="password"
  value="#{loginPage.password}"/><br/>
...
```

- A notação # que serve como o valor para o atributo de valor liga as propriedades do *LoginPageBean* aos componentes UI

Value Binding

- Capaz de associar o identificador *loginPage* a uma instância da classe *LoginPageBean* devido a entrada no *faces-config.xml*
- Desde que o *LoginPageBean* é declarado para ser um *JavaBean* gerenciado no *framework*
- Uma instância do *LoginPageBean* é criada e colocada no escopo configurado (se já não existir) quando a página JSF for avaliada
- Na submissão da página, os valores que o usuário entrar serão automaticamente colocado dentro das propriedades do *JavaBean*

Registrando *Action Handlers* para os Componentes de Visão

- O JSF introduz o conceito de programação baseada em eventos dentro do ambiente WEB
- Alguns dos componentes de UI que o JSF gerarão, dada uma ação ou entrada apropriada do usuário
- Eventos que podem ser processados por *action handlers*

Registrando *Action Handlers* para os Componentes de Visão

- Para "registrar" um *handler* para o component *commandButton*:

```
...
<h:commandButton
  action="#{loginPage.performLogin}"
  value="Login"/>
...
```

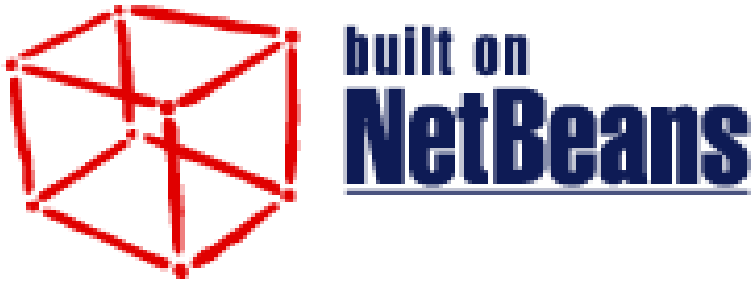
- A notação # liga um método nomeado como *performLogin*
- Encontrado no bean referenciado com o nome *loginPage* ao botão
- O método ligado atua como a *action handler* quando o botão for pressionado

Navegação de Páginas

- Determina próxima tela mostrada para o usuário na submissão de um formulário
- Utiliza o valor de retorno do método que serve como o *action handler* para o botão de *submit*
- O valor do String é examinado nas regras de navegação definida dentro do arquivo de configuração

Navegação de Páginas

- Passaremos agora para o NetBeans



Sumário

- *JavaServer Faces*
- *JSF* e *Struts*
- *Controller*
- *FacesServlet*
- Método *Application*
- Configuração do JSF
- *Value Binding*
- Navegação de Páginas

Parceiros

- **Os seguintes parceiros tornaram JEDI™ possível em Língua Portuguesa:**