Lição 7

Introdução a MVC e ao Framework Struts

Objetivos

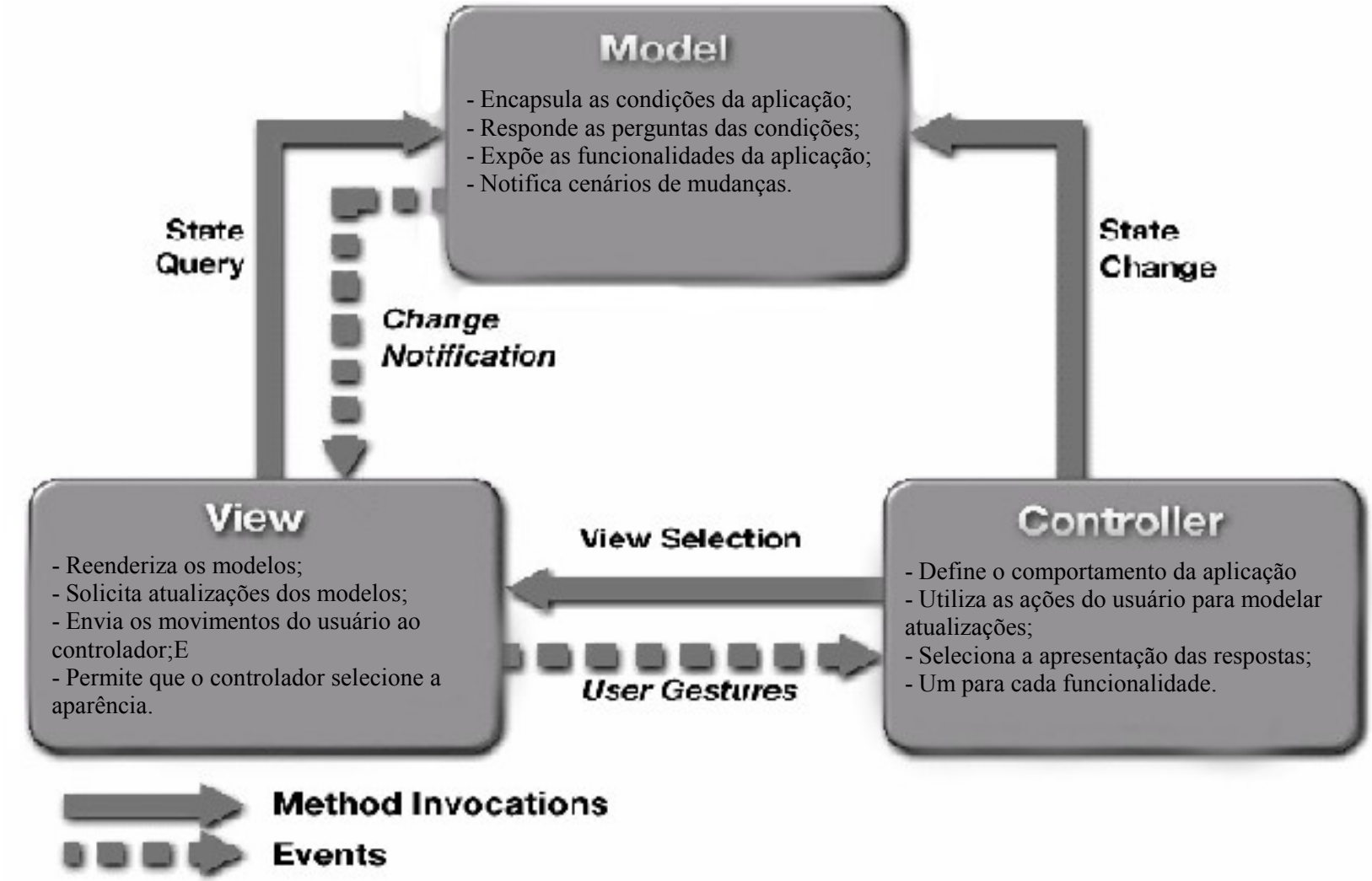
Ao final desta lição, o estudante será capaz de:

- Compreender o funcionamento da arquitetura MVC
- Utilizar o framework Struts no desenvolvimento de aplicações

Introdução à arquitetura MVC

- Padrão de arquitetura eficaz no desenvolvimento de projetos
- Código mais suscetível a mudanças é a interface do usuário
- Manter a lógica de negócios acoplada firmemente à interface
- Padrão MVC proporciona uma solução para estes problemas
- Divide a aplicação em *model*, *view* e *controller*
- Decompondo enquanto fornece uma série de recomendações sobre suas interações

Introdução à arquitetura MVC



Model

- Representa os dados usados por uma aplicação
- Define detalhes sobre recuperação, persistência e manipulação de dados
- Vantagens:
  - Fornece muitos benefícios durante a manutenção da aplicação desde que dados e operações não estejam espalhados pela da aplicação
  - Os componentes podem facilmente ser reutilizados em outras aplicações que necessitam de funcionalidades similares

View

- Detalhes da execução da interface com o usuário
- Interage com o usuário
- Vantagens:
  - Facilita a inclusão de um grupo separado do projeto na equipe do desenvolvimento
  - Torna possível fornecer múltiplas interfaces à aplicação

Controller

- Detalhes sobre a transição de programas *flow/screen*
- Responsável por capturar os eventos gerados pelo usuário na camada *view* e possibilitar a atualização dos componentes na camada *model*.
- Vantagens:
  - Componentes de *view* podem ser projetados de modo que não necessitam estar cientes de outros componentes
  - Atualizações dos componentes *model* são removidos da camada de apresentação

Introdução à arquitetura MVC

- Não devemos dizer que o padrão MVC apresenta todos os benefícios sem nenhum efeito colateral
- Dividir a aplicação em três componentes separados resulta em um aumento da complexidade
- Para aplicações pequenas que não se beneficiam do acoplamento fraco da camada *model*, este pode ser um obstáculo no uso deste padrão
- É melhor manter em mente que as aplicações freqüentemente começam pequenas e crescem para sistemas complexos, sendo assim, nessas ocasiões o acoplamento fraco sempre deve ser utilizado

Arquitetura *model 2*

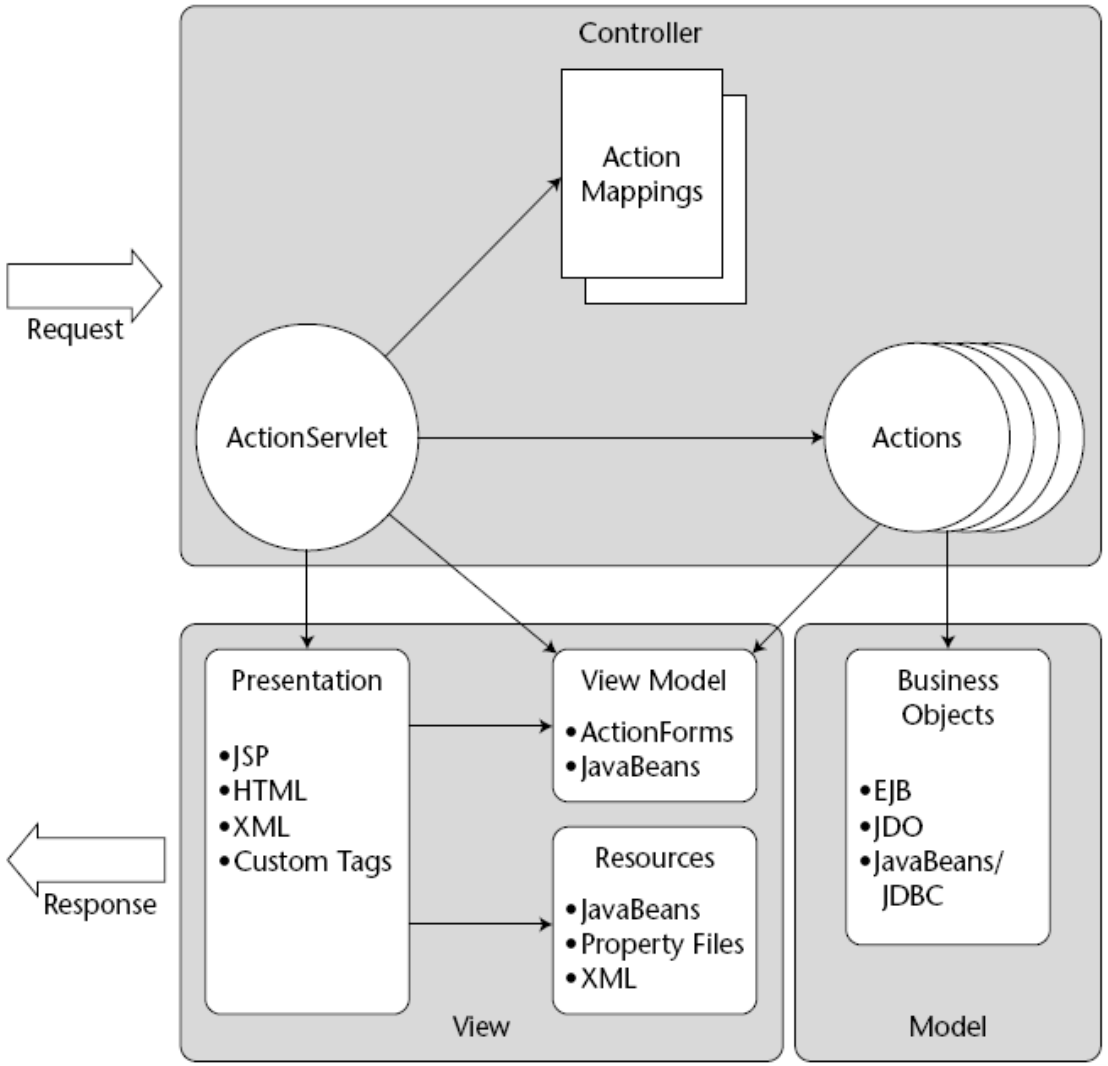
- MVC arquitetura ligeiramente modificada e adaptada para o uso de aplicações Web
- *Front Controller*, um controlador *servlet* que:
  - Fornece um único ponto do acesso para o restante da aplicação
  - Responsável pela gerência central do fluxo da aplicação
  - Proporcionar serviços igualmente como a manipulação segura e a gerência de usuários
  - Usa tipicamente configurações de *XML* para determinar o fluxo da aplicação e processar comandos
  - Geralmente emprega componentes auxiliares que servem como objetos *Command*

Arquitetura *model 2*

- Executar essa arquitetura pode ser facilitada com o uso de estruturas *third-party* existentes
- Estas estruturas fornecem muitos detalhes do encapsulamento
- Fornece uma funcionalidade adicional

Struts

- Uma estrutura de código aberto fornecida e controlada pela Apache Software Foundation



MVC e Struts

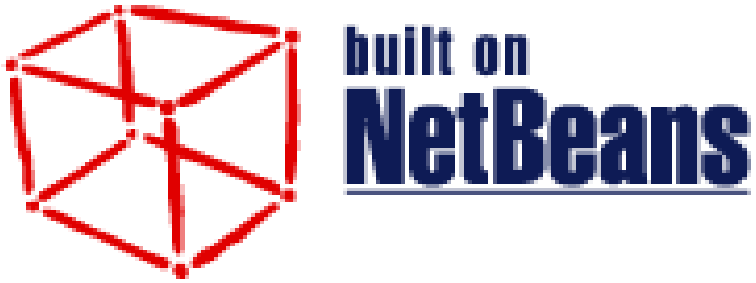
- Objetos fornecidos por Struts para as camadas:
  - Model
  - View
    - *struts-html*
  - Controller
    - *ActionServlet*
    - *Action*
    - *ActionForm*
    - *struts-config.xml*

ActionServlet

- No centro da implementação do controlador a estrutura *Struts* é *ActionServlet*
- *Serves* são um *Front Controller servlet* e fornecem um único ponto do acesso ao descanso da aplicação WEB
- Contém a lógica por trás da manipulação do pedido do cliente
- Sabe todos os detalhes lendo de um arquivo de configuração XML, chamado *struts-config.xml*
  - Qual *Action* a se chamar para segurar o pedido
  - Qual componente de Vista deve ser chamado em seguida
- Fornecido de modo operacional pela estrutura *Struts*; necessita somente ser configurado corretamente

web.xml

- Passaremos agora para o NetBeans

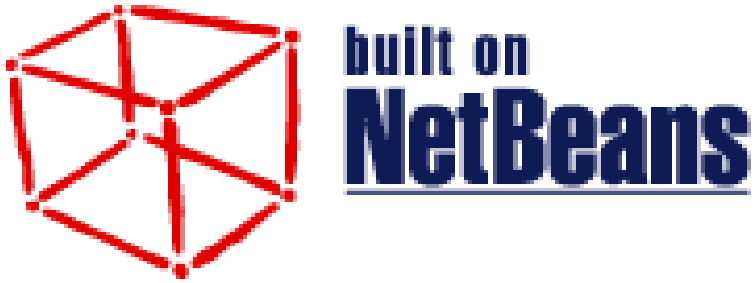


Action

- Todos os objetos *Action* definem um método chamado *execute()*
- Este método é chamado pelo *ActionServlet* para assegurar os pedidos do cliente
- *Struts* fornece somente colaboradores com a classe base *Action*
- Inclui objetos *Action* e os alimentadores de pedido
- Desenvolvedores devem criar uma aplicação da sub-classe desta super-classe e fornecer uma implementação para executar o método

Action

- Passaremos agora para o NetBeans



Action

- Atividades que uma *Action* deve executar são:
  - Recuperar a informação que o usuário forneceu associada ao *ActionForm bean*
  - Traduzir os dados do formulário em parâmetros requeridos pelo objeto de negócios, que implementa as funcionalidades
  - Recuperar o resultado da operação do objeto de negócios e de lá determine a próxima tela que seguinte o usuário deverá receber
  - Armazenar os dados dos resultados da operação de negócios na sessão ou objeto solicitado para uso no restante da aplicação

Action

- A estrutura somente instanciará uma única cópia dos objetos *Action*
- Irá usá-los para facilitar todas as solicitações
- Isto significa que devemos:
  - Codificar a *Action* para ser uma *thread* segura
  - Usar sempre variáveis locais e não variáveis globais

Action

- Informar a *ActionServlet* que componentes da tela serão responsáveis por retornar instancias de objetos *ActionForward*
- Têm acesso a objetos *ActionForward* com o uso do objeto *ActionMapping*
- Estes percursos são lidos pelo arquivo de configuração *ActionServlet*
- Instruir a *ActionServlet* à transmitir o controle para um mapeamento lógico chamado *success*
- Possui a seguinte instrução:

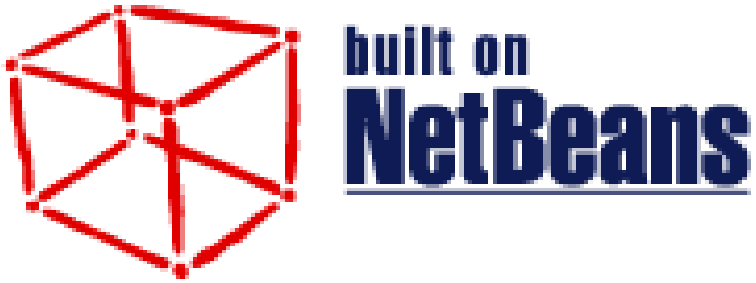
```
return mapping.findForward("success");
```

ActionForm

- Instâncias desta classe são usados para facilitar a recuperação dos dados dos formulários
- Cada instância representa um formulário ou uma série destes
  - Definem as propriedades que correspondem aos elementos do formulário(s) que representam e indicam que usam publicamente *getters* e *setters* acessíveis
  - *Actions* que necessitam dos dados dos formulários acionam métodos *getter* de instâncias *ActionForm*
- *Struts* fornecem a definição da super classe
- Desenvolvedores têm a responsabilidade de criar suas próprias implementações

ActionForm

- Passaremos agora para o NetBeans



ActionForm

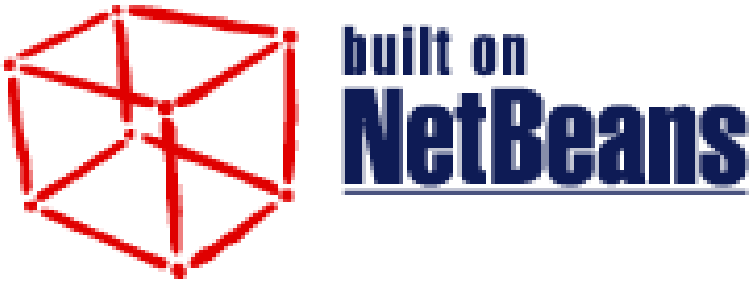
- Definir propriedades para cada elemento que será representado no formulário
- São usados para transferir dados entre a *view* e a *controller* e em ambas não é indicado para comportar estas regras
- Opcionalmente inclua um método de validação para confirmar a consistência dos dados antes que a *controller* envie-os à *Action*

struts-config.xml

- Arquivo de configuração para os componentes dos *Struts*
- Definir que ação é chamada por qual solicitante
- Formar os componentes para serem usados em cada ação
- Descrever um mapa de nomes lógicos para os caminhos reais

struts-config.xml

- Passaremos agora para o NetBeans



Controller

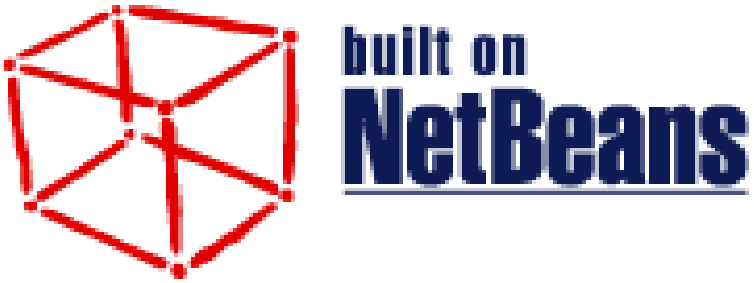
- Para uma única instalação:
  - Configurar o *ActionServlet* em seu ambiente
- Para cada gerente de formulários, que será adicionado à aplicação:
  - Criar um objeto *ActionForm*
  - Criar um objeto *Action*
  - Criar a configuração entre um objeto *ActionForm* na *struts-config.xml*
  - Criar a configuração entre um objeto *Action* na *struts-config.xml*
  - Configurar o envio por um objeto *Action*

View

- *Struts* pode empregar toda a tecnologia da camada de apresentação
- Na realidade é empregada na maioria de casos *JSPs* e/ou *HTML*
- *Struts* fornece para esta camada um conjunto de bibliotecas *tag* para permitir o uso de características de *Struts* para o preenchimento e validação automática do formulário

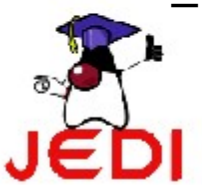
struts-html

- Passaremos agora para o NetBeans



View

- Para uma única instalação:
  - Configure as bibliotecas de *tags* para serem utilizadas em seu ambiente de desenvolvimento de aplicações
  - Coloque os arquivos *JAR* que contenham a implementação das bibliotecas de *tags* no diretório *WEB-INF/lib*
- Para cada formulário a ser criado:
  - Adicionar uma diretriz orientadora
  - Utilizar a *tag* <html:form>
  - Empregue os *Tags* incluídos na biblioteca de *Tags* S*truts-HTML* que têm a mesma funcionalidade (<html:text>, etc);
  - Os campos de entrada do formulário definido estejam presentes como propriedades do objeto *ActionForm*
  - Não se esqueça de fechar a *tag* <html:form>



Sumário

- Introdução a arquitetura MVC
  - Model, View e Controller
- Arquitetura model 2
- Struts
  - ActionServlet
  - Action
  - ActionForm
  - Arquivo struts-config.xml

Parceiros

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