Utilizando Servlets e JSP em conjunto

Instituto de Computação

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br

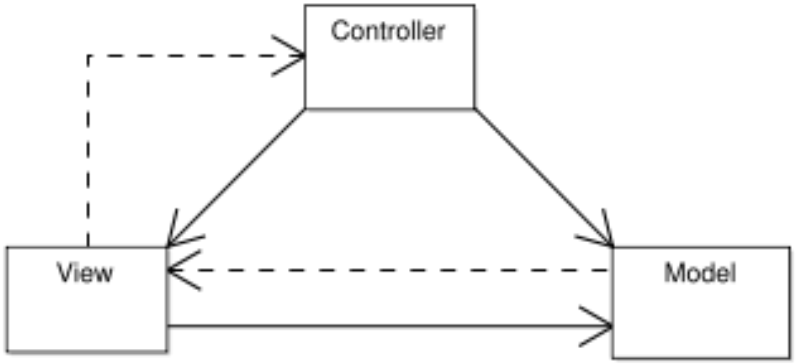
Estilo MVC

- Estilos arquiteturais servem para dar uma diretriz de como a arquitetura do sistema deve ser construída
  - Engenharia de software, diferentemente das demais engenharias, não está sujeita a “leis da natureza”
  - Estilos arquiteturais são “regras artificiais” que devem ser seguidas pelos desenvolvedores
- Um dos principais estilos arquiteturais é o MVC
  - MVC = Model-View-Controller
  - Filosofia básica: os elementos do sistema devem ser separados em três tipos: modelo, visão e controle

Estilo MVC

- Modelo (*Model*):
  - São os elementos derivados do processo de análise
  - Representam os principais conceitos do domínio
  - São usualmente persistidos em banco de dados
- Visão (*View*):
  - São os elementos criados durante o projeto para fazer interface com o usuário
  - Normalmente manipulam elementos de modelo
- Controle (*Controller*):
  - São os elementos que fazem a orquestração
  - Executam os casos de uso

Estilo MVC



- Regras:
  - Entidades de modelo não conhece ninguém (a não ser outras entidades de modelo)
  - Entidades de visão conhece somente entidades de modelo (e outras entidades de visão)
  - Entidades de controle conhece tanto entidades de modelo quanto entidades de visão (e outras entidades de controle)

Estilo MVC

- Vantagens:
  - Separação de responsabilidades clara
  - Alto grau de substituição da forma de interface com o usuário
  - Alto grau de reutilização das entidades do modelo
  - Possibilidade de múltiplas interfaces com o usuário, trabalhando de forma simultânea (modo texto, janelas, web, celular, etc.)

Servlet x JSP (revisitado!)

- Servlet:
  - Java é a linguagem principal
  - Indicado para implementar regras de negócio
  - Fácil acesso a banco de dados
- JSP:
  - HTML é a linguagem principal
  - Indicado para interface com o usuário
- Classes Java
  - Representam as entidades do domínio
  - Também conhecidas como JavaBeans, VO (Value Objects) ou POJO (Plain Old Java Objects)
  - Ponte entre Servlet e JSP

Servlets e JSP em conjunto

- Model: JavaBeans
- View: JSP
- Controller: Servlet

Camada Model: JavaBeans

- Derivados dos diagramas de classe da aplicação
- Representam as entidades do domínio
- Exemplo para domínio bancário
  - Usuario
  - Conta
  - Transacao
  - Deposito
  - Saque
  - Investimento
  - Etc.

Camada Controller: Servlet

- Representam os casos de uso do sistema
- Esses Servlets devem
  1. Acessar os dados preenchidos dos formulários
  2. Processar esses dados, acessando o BD se necessário
  3. Criar JavaBeans com o resultado do processamento
  4. Armazena os JavaBeans na requisição, para que o JSP tenha acesso em seguida
  5. Encaminhar o fluxo de execução para o JSP

Camada Controller: Servlet

- Armazenamento e leitura do *JavaBean* no Request
  - No Servlet:

```
Conta conta = new Conta(...);
request.setAttribute("conta", conta);
```

  - No JSP:

```
<jsp:useBean id="conta" type="br.uff.ic.Conta"
             scope="request" />
```

Camada Controller: Servlet

- Encaminhando para o JSP
  - Uso de RequestDispatcher
  - Exemplo

```
RequestDispatcher dispatcher =
  request.getRequestDispatcher("resultado.jsp");
dispatcher.forward(request, response);
```

Camada View: JSP

- Finalmente, projete os JSP para apresentar os resultados processados pelos Servlets
- Eles devem conter toda a interface com o usuário
- Esses JSP devem:
  1. Acessar os dados disponibilizados pelos Servlets na requisição
  2. Apresentar esses resultados em HTML