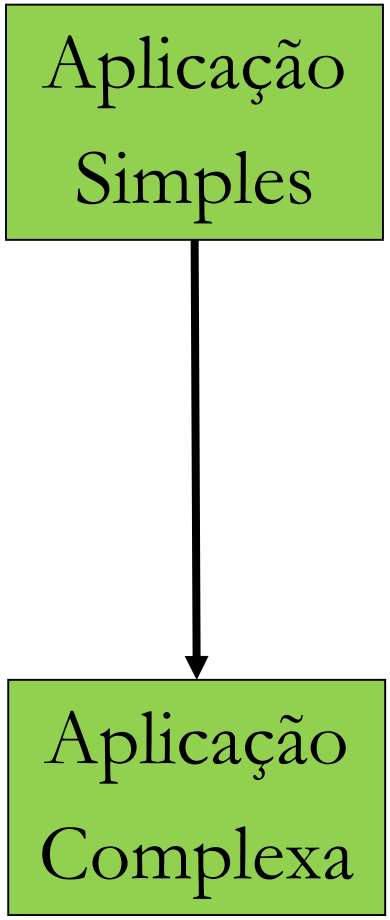
Programação IV
Aula 06 – MVC e DAO

Prof. Alex Marin

alex.marin@iffarroupilha.edu.br

Motivação



- Elementos de scripting que invocam diretamente o código de um servlet
- Beans
- **Combinação de Servlet/JSP (MVC)**
- **MVC com JSP + EL**
- MVC com beans, EL e um framework (JSF por exemplo)

MVC – Introdução

- O **MVC (*Model-View-Controller*)** é um modelo de desenvolvimento de Software, atualmente considerado uma "arquitetura padrão" utilizada na Engenharia de Software.

- O modelo isola a "lógica" (a lógica da aplicação) da interface do usuário (inserir e exibir dados), permitindo desenvolver, editar e testar separadamente cada parte.

MVC – Introdução

- Em outras palavras:
  - MVC é um conceito de desenvolvimento e design que tenta separar uma aplicação em três partes distintas.
  - Uma parte, a **Model**, está relacionada a lógica do negócio (processamento das informações);
  - Outra parte, a **View**, está relacionada a exibir os dados ou informações dessa aplicação;
  - E a terceira parte, a **Controller**, coordena as duas anteriores, exibindo a interface correta ou executando algum trabalho que a aplicação precisa completar.

MVC – Camadas

- Model
  - É o objeto que representa a verdadeira lógica do programa. Maneja os dados e controla suas transformações.
  - Não possui conhecimento específico dos controladores (controller) e das apresentações (view), nem sequer contém referência a eles.
  - Portanto, o Model são as classes que trabalham no armazenamento e busca de dados.

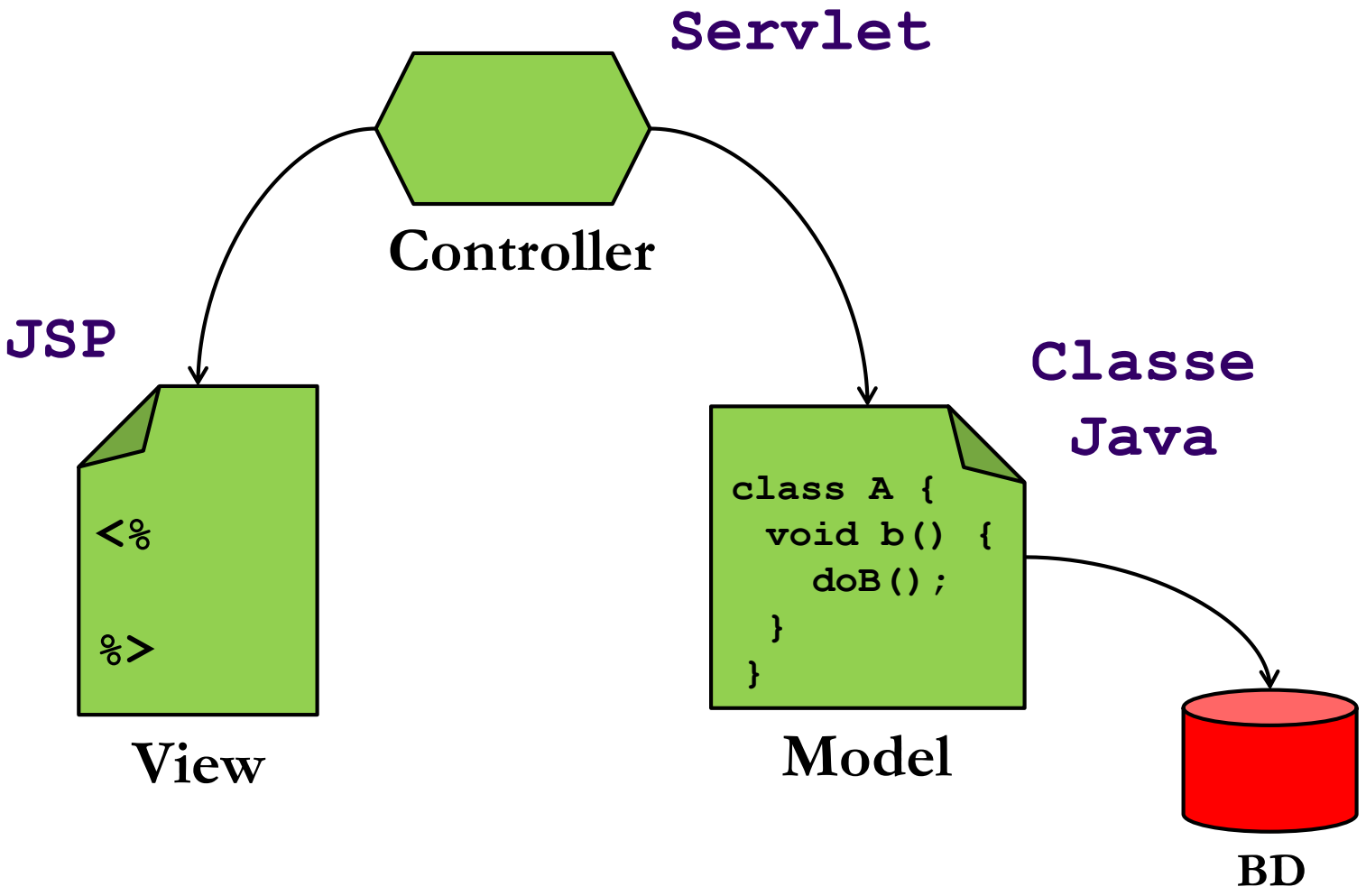
MVC – Camadas

- View
  - É o componente responsável pela apresentação visual dos dados processados pelo Model ao usuário.
  - Diferentes visões podem existir para um mesmo modelo, para diferentes propósitos.
  - Também é a parte que recebe os dados de entrada do usuário.

MVC – Camadas

- Controller
  - É o objeto que responde as ordens executadas pelo usuário, atuando sobre os dados apresentados pelo Modelo, decidindo como o Modelo deverá ser alterado e qual Apresentação (Visão) deverá ser exibida.
  - O controlador (*controller*) recebe a entrada de dados e inicia a resposta ao usuário ao invocar objetos do modelo, e por fim uma visão baseada na entrada. Ele também é responsável pela validação e filtragem da entrada de dados.

MVC no mundo Servlet & JSP



Um primeiro exemplo

- Formulário da página inicial

  `Confira: RefriIndex.jsp`

- Componente Model

  `Confira: RefriEspecialista.java`

- Componente Controller

  `Confira: RefriSabor.java`

- Componente View

  `Confira: ResultadoRefri.jsp`

Mais um exemplo

- Formulário da página inicial

  `Confira: Login.jsp`

- Componente Model

  `Confira: UsuarioBean.java`

- Componente Controller

  `Confira: Logar.java`

- Componente View

  `Confira: Logado.jsp`

O padrão DAO

- **DAO** (acrônimo de ***Data Access Object***), é um padrão para persistência de dados que permite separar regras de negócio das regras de acesso a banco de dados.

- Em uma aplicação que utilize a arquitetura MVC, todas as funcionalidades de bancos de dados, tais como obter as conexões, mapear objetos Java para tipos de dados SQL ou executar comandos SQL, devem ser feitas por classes de DAO.

O padrão DAO

- **DAO: o intermediário entre os mundos**
  - Arquivos texto são um conjunto de caracteres. Bancos de dados relacionais são um conjunto de tabelas, colunas, e linhas. Aplicações Java são um conjunto de objetos.

  - O DAO deve funcionar como um tradutor dos mundos. Suponha um banco relacional. O DAO deve saber buscar os dados do banco e converter em objetos para ser usado pela aplicação. Semelhantemente, deve saber como pegar os objetos, converter em instruções SQL e mandar para o banco de dados. É assim que um DAO trabalha.

O padrão DAO

- **Abstração**
  - Devido à sua qualidade de tradutor, o DAO abstrai a origem e o modo de obtenção / gravação dos dados, de modo que o restante do sistema manipula os dados de forma transparente, sem se preocupar com o que acontece por trás dos panos. Isso ajuda muito em processos de migrações de fonte de dados e testes unitários.

O padrão DAO

- **Unificação do acesso a dados**
  - É muito comum em códigos de programadores iniciantes vermos a base de dados sendo acessada em diversos pontos da aplicação, de maneira extremamente explícita e repetitiva.
  - Isso é um crime contra um bom design OO. Além de não abstrair a fonte dos dados, transforma o código em um espaguete difícil de manter.

O padrão DAO

- **Unificação do acesso a dados**
  - O DAO também nos ajuda a resolver este problema, provendo pontos unificados de acesso a dados.
  - Desse modo, a lógica de interação com a base de dados ficam em lugares específicos e especializados nisso, além de eliminar códigos redundantes, facilitando a manutenção e futuras migrações.

O padrão DAO

- **DAO na prática**
  - Geralmente, temos um DAO para cada objeto do domínio do sistema (Produto, Cliente, Compra, etc).
  - Cada DAO deve possuir uma interface, que especifica os métodos de manipulação de dados. Nossos códigos trabalharão apenas com as interfaces dos DAOs, desconhecendo a implementação utilizada. Isso é uma boa prática, não somente em termos de persistência, mas em vários outros pontos de uma aplicação.

O padrão DAO

- **DAO na prática**
  - O primeiro passo é definir classes POJO (Plain Old Java Object), as quais são semelhantes as tabelas do banco de dados, assim podemos manipular de forma igual para igual ao que está na base de dados.

Confira: **ClientesBean.java**

O padrão DAO

- **DAO na prática**
  - Para a conexão temos uma classe que faz a função de gerenciador (fábrica) de conexões.

Confira: **ConnectionFactory.java**

O padrão DAO

- **DAO na prática**
  - Para acessar os dados usamos o padrão DAO, assim encapsulamos todo o trabalho com o banco, e nossas classes que querer usar e manipular os dados simplesmente devem conhecer nossa classe DAO, sem se preocupar com abrir conexão, fechar e inserir comandos.

Confira: **ClienteDAO.java**

O padrão DAO

- **DAO na prática**
  - Para utilizar o DAO, seguindo o padrão MVC, é necessário implementar o **Controller** (Servlet) que receberá informações de uma **View** (página JSP ou HTML) e fará o acesso aos métodos do **Model** DAO.

Confira: **ClienteServlet.java**