

Módulo II

Arquitetura em Camadas

Prof. Ismael H F Santos

Ementa

- Arquitetura de camadas de Software
 - Arquiteturas em Camadas
 - Padrões para Arquiteturas em camadas
- Persistência de Objetos
 - Persistência de objetos
 - Estratégias de Persistência

FPSW-Java

Arquitetura
Em Camadas



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

3

Introdução

- Em aplicações OO de médio e grande porte, diversos aspectos devem ser considerados:
 - Apresentação
 - Lógica da aplicação
 - Lógica do negócio
 - **Persistência de Objetos**
 - Camada de Utilitários:
 - Controle de Exceções, Logging, comunicação, etc.
- **Persistência** : capacidade de uma aplicação manter suas informações entre sessões de uso.
 - Uma proporção significativa do esforço de desenvolvimento recai sobre a solução que o desenvolvedor deve dar a este problema.

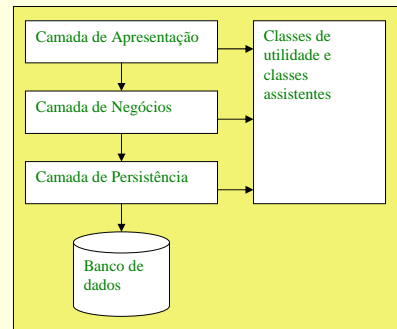
April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

4

Arquitetura em camadas

- Arquitetura em camadas visa a criação de aplicativos modulares, de forma que a camada mais alta se comunica com a camada mais baixa e assim por diante, fazendo com que uma camada seja dependente apenas da camada imediatamente abaixo.



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

5

Arquitetura em camadas

- **Camada de Apresentação:** Lógica de interface do usuário (GUI). O código responsável pela apresentação e controle da página e tela de navegação forma a camada de apresentação;
- **Camada de Negócios:** Código referente a implementação de regras de negócio ou requisitos do sistema;
- **Camada de persistência:** Responsável por armazenamento e recuperação dos dados quando solicitado. Objetivo é o de garantir uma independência da fonte de dados (arquivos, bancos de dados, etc) e ao mesmo tempo manter as informações entre diferentes sessões de uso.

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

6

Arquitetura em camadas

- **Banco de dados:** O BD existe fora da aplicação Java, é a atual representação persistente do estado do sistema.
- **Assistentes e Classes de utilidade:** São classes necessária para o funcionamento ou mesmo o complemento de uma aplicação ou parte dela, como por exemplo o Exception para tratamento de erros.

April 05

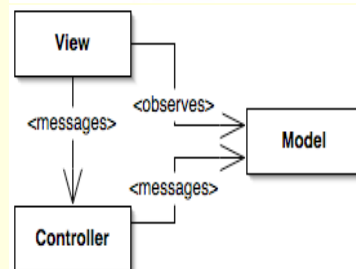
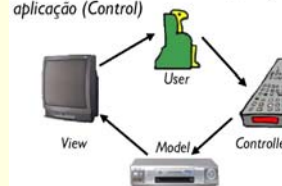
Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

7

Model-View-Controller (MVC) Design Pattern

- O padrão de arquitetura **MVC** (model-view-controller) surgiu nos anos 80 com a linguagem SmallTalk.
 - Criado por Trygve Reenskaug no fim dos anos 70
 - Usado no desenvolvimento de aplicações desktop por facilitar o desenvolvimento em camadas de aplicações que usam a orientação a objetos

Técnica para separar dados ou lógica de negócios (Model) da interface do usuário (View) e do fluxo da aplicação (Control)



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

8

Model-View-Controller (MVC) Design Pattern

■ O que é o MVC

- padrão projeto para o desenvolvimento de aplicações,
- A implementação de aplicações usando este padrão são feitas com recurso a frameworks, apesar de não ser obrigatória a utilização de uma para seguir o padrão.

■ Objetivo do MVC

- Isolar mudanças na GUI, evitando que estas mudanças acarretem em mudanças na Camada de Negicos da Aplcação (Application's Domain Logic)

■ Vantagens

- Facilita a manutenção
 - *Changes to business logic are less likely to break the presentation logic & vice-versa*
- Facilita o desenvolvimento por times multi-disciplinares:
 - *desenvolvedores – creating robust business code*
 - *designers – building usable and engaging UIs*

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

9

Model-View-Controller (MVC) Design Pattern

■ Camadas e respectivas funções

■ **Model:**

- Define as regras de acesso e manipulação dos dados
- Armazenados em bases de dados ou ficheiros, mas nada indica que sirva só para alojamento persistente dos dados.
- Pode ser usado para dados em memória volátil, p.e.: memória RAM, apesar não se verificar tal utilização com muita frequência. Todas as regras relacionadas com tratamento, obtenção e validação dos dados devem ser implementados nesta camada.

■ **View:**

- Responsável por gerar a forma como a resposta será apresentada, página web, formulário, relatório, etc...

■ **Controller:**

- Responsável por responder aos pedidos por parte do utilizador. Sempre que um utilizador faz um pedido ao servidor esta camada é a primeira a ser executada.

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

10

Model-View-Controller (MVC) Design Pattern

MVC

- Clearly separates business, navigation and presentation logic. It's a proven mechanism for building a thin, clean web-tier

Model

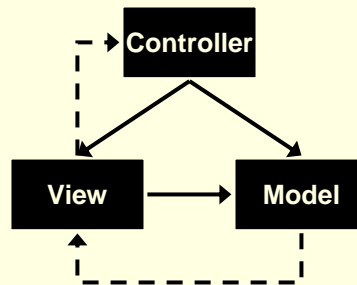
- The domain-specific representation of the information on which the application operates.

View

- Renders the model into a form suitable for interaction, typically a user interface element.

Controller

- Processes and responds to events, typically user actions, and may invoke changes on the model.



Note: the solid lines indicate a direct association, and the dashed line indicate an indirect association

<http://en.wikipedia.org/wiki/Model-view-controller>

April 05

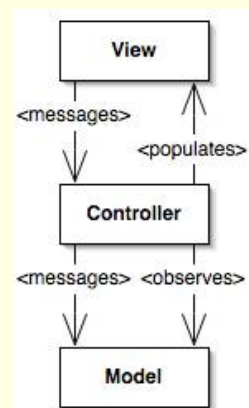
Prof. Ismael H. F. Santos - ismael@tegraf.puc-rio.br

11

MVC Next – Steve Jobs

- A next (Steve Jobs), resolveu modificar esse modelo oferecendo uma alternativa para sua linguagem de programação objective-c.

- Delega a responsabilidade de observar o modelo para a camada de **Controller** que, por sua vez, envia para a camada de visão as alterações ao invés da camada de **View** obter esses dados do **Model**.



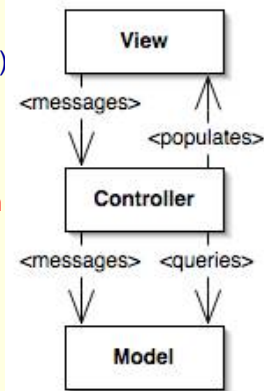
April 05

Prof. Ismael H. F. Santos - ismael@tegraf.puc-rio.br

12

MVC Model 2

- Com o crescimento das aplicações web baseadas no protocolo **HTTP** que é sem estado, não temos mais uma sessão permanentemente aberta entre o cliente e o servidor. Além disso o **HTTP** não prevê uma forma de “**enviar**” (push) informações do servidor para o cliente.
- Isto impede o trabalho do **Controller** que não pode mais enviar informações para a **View** sem ser solicitado. Para contornar o problema a **Sun** criou o **MVC Model 2**, baseado no padrão **FrontController**.
- Agora a camada **Controller** submete ações tentando acompanhar o processo de **request-response** do protocolo **HTTP** ao invés de observar a camada **Model**, criando um fluxo linear para a arquitetura das aplicações.



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

13

FPSW-Java

*Padrões para
Arquiteturas
em Camadas*

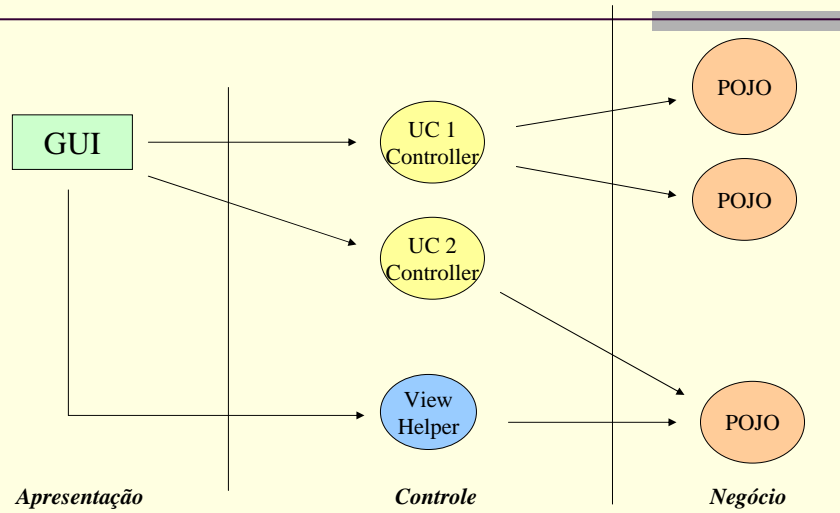


April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

14

Modelo de Camadas – Apps Desktop

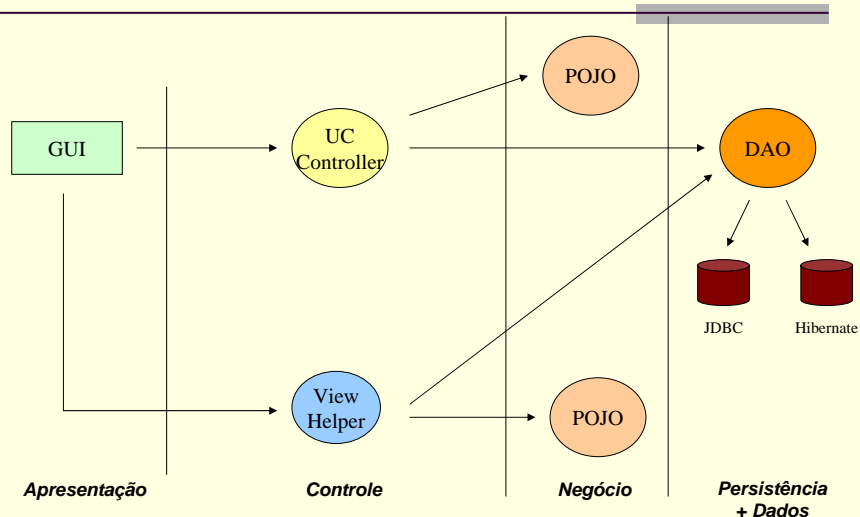


April 05

Prof. Ismael H. F. Santos - ismael@tegraf.puc-rio.br

15

Modelo de Camadas – Apps Desktop



April 05

Prof. Ismael H. F. Santos - ismael@tegraf.puc-rio.br

16

WebApp

Modelos
MVC - Web

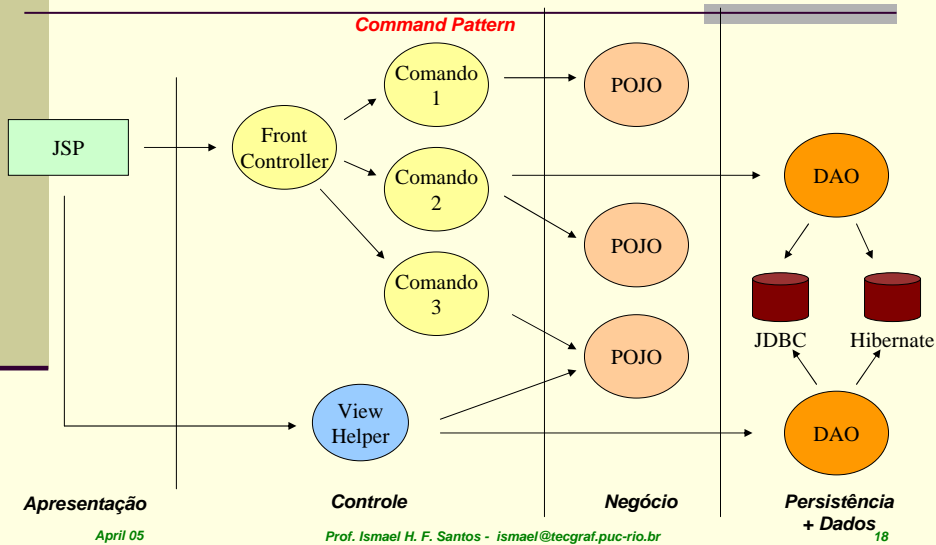


April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

17

Modelo de Camadas – Apps Web



Implementação do MVC para Web

- **Design centrado em páginas**
 - Aplicação JSP consiste de seqüência de páginas (com ou sem beans de dados) que contém código ou links para chamar outras páginas
- **Design centrado em servlet (FrontController* ou MVC)**
 - Aplicação JSP consiste de páginas, beans e servlets que controlam todo o fluxo de informações e navegação
 - Este modelo favorece uma melhor organização em camadas da aplicação, facilitando a manutenção e promovendo o reuso de componentes.
 - Um único servlet pode servir de fachada
 - Permite ampla utilização de J2EE design patterns

* FrontController é um J2EE design pattern. Vários outros design patterns serão identificados durante esta seção. Para mais informações, veja Sun Blueprints I71

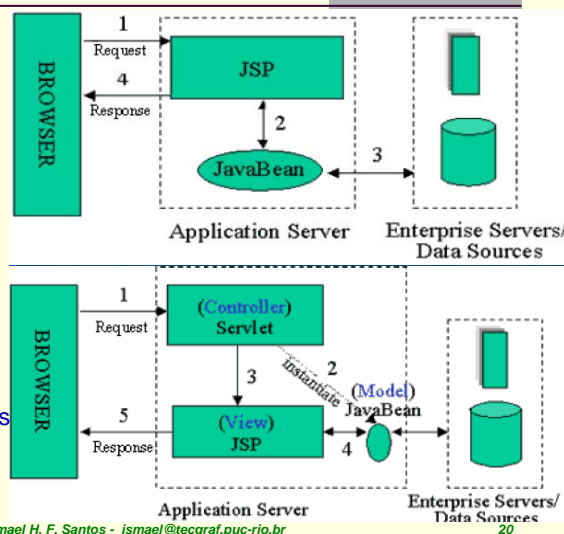
April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

19

Especificações do J2EE - Arquiteturas de aplicação Web

- **Model 1**
 - Recomendado para projetos pequenos.
 - E/S: Java Server Pages
 - Lógica de negócio: Java Beans e EJBs
- **Model 2**
 - Recomendada para projetos médios e grandes.
 - Variação do padrão MVC
 - Controller: Servlets
 - Model: JavaBeans e EJBs
 - View: Java Server Pages



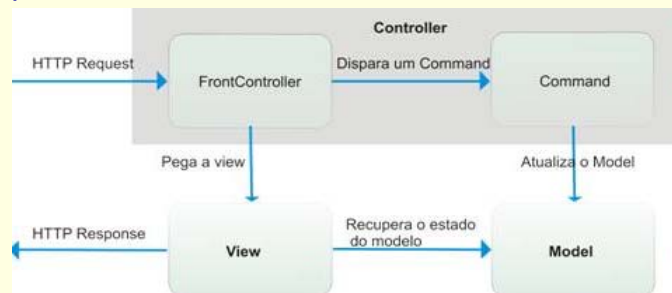
April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

20

Padrão Front Controller

- Padrão que consolida todas as requisições web em um único objeto manipulador, despachando o tratamento adequado dessas requisições conforme o comportamento esperado.



- A seguir apresentamos mais detalhes do padrão

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

21

Implementação do MVC para Web

- *Design centrado em páginas*
 - *Aplicação JSP consiste de seqüência de páginas (com ou sem beans de dados) que contém código ou links para chamar outras páginas*
- *Design centrado em servlet (FrontController* ou MVC)*
 - *Aplicação JSP consiste de páginas, beans e servlets que controlam todo o fluxo de informações e navegação*
 - *Este modelo favorece uma melhor organização em camadas da aplicação, facilitando a manutenção e promovendo o reuso de componentes.*
 - *Um único servlet pode servir de fachada*
 - *Permite ampla utilização de J2EE design patterns*

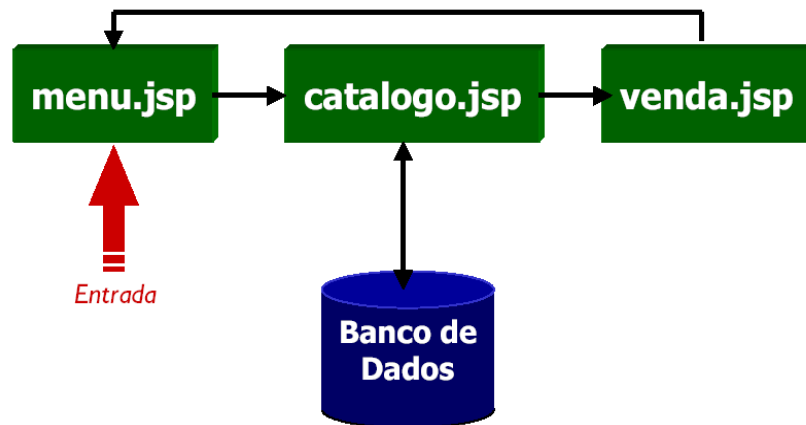
* FrontController é um J2EE design pattern. Vários outros design patterns serão identificados durante esta seção. Para mais informações, veja Sun Blueprints [7]

April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

22

JSP Model I - Centrado em páginas

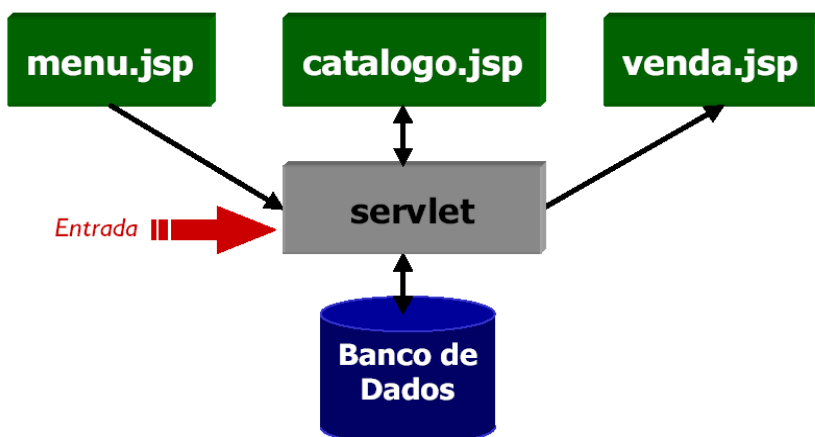


April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

23

JSP Model II - Centrado em servlet



April 05

Prof. Ismael H. F. Santos - ismael@tecgraf.puc-rio.br

24