Programação Java para Iniciantes

Aula 9 – Parte 1

Prof. Rogério Napoleão Jr.

Padrões de Projetos

- Nas últimas décadas, ocorreu na indústria de engenharia de software um enorme progresso no campo dos padrões de projeto.

- Utilizar padrões de projeto reduz substancialmente a complexidade do processo de design.

Padrões de Projetos

- Benefícios:
  - Ajuda a construir um software confiável com arquiteturas testadas e confiança acumulada pelas empresas.
  - Promove a reutilização de projetos em futuros sistemas.
  - Ajuda a identificar equívocos comuns e armadilhas que ocorrem na construção de sistemas.

Padrões de Projetos

- Benefícios:
  - Ajuda a projetar sistemas independente da linguagem.
  - Estabelece um vocabulário comum de projeto entre os desenvolvedores.
  - Encurta a fase de projeto no processo de desenvolvimento de um software.

Padrões de Projetos

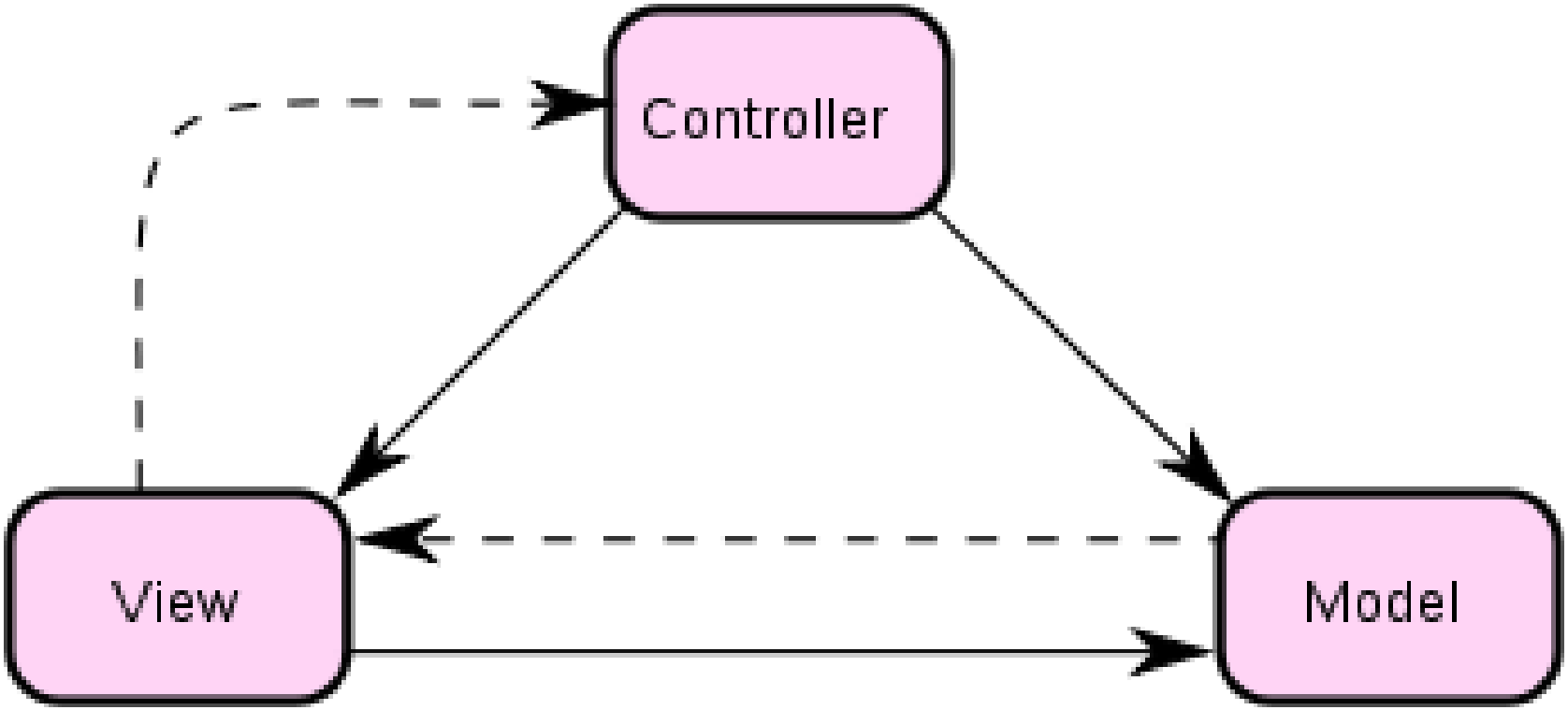
- Padrões arquiteturais
- Padrões de alto nível (arquitetura de sistemas)
  - **Model-View-Controller**
  - **Layers**

MVC

- **Model-view-controller** (**MVC**) é um modelo de desenvolvimento de Software, atualmente considerado um “Design Pattern" 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



MVC

- Porque surgiu a necessidade de utilizar MVC?
  - As telas (JSP's por exemplo) continham as regras e controles do sistema, de forma independente.
  - Qualquer mudança podia resultar em um conjunto de mudanças em várias páginas com consequências não previsíveis.
  - A complexidade cresce rapidamente e o que a princípio parecia simples, acaba se tornando algo complexo.
  - Não possuia independência da View.

MVC – Model

- Classes de domínio da aplicação.
- Responsáveis por executar a **regra de negócio**, ou seja, o real trabalho em si.
- Modela o problema do negócio.
- Independente da camada Controller e View.

MVC – Controller

- Envia mensagens ao Model dizendo o que fazer.
- Interface entre o Model e a View.
- O Controller e o Model devem sempre ser separados (o que fazer Vs. como fazer)

MVC – View

- O usuário precisa ver o que a aplicação está fazendo.
- Interface de saída.
- Exibem aspectos do Model.
- Model deve ser independente da View, porém deve possuir métodos de acesso.

MVC

- Sempre visar a independência de código.
- O Model nunca deve estar contido no Controller e na View.
- A View deve representar o estado do Model.
- O Controller deve conversar com o Model e View e não manipulá-los.