
SGB
Sistema de
Gestão Bibliográfica

SGB_TUTORIAL_MVC

Versão 0.1

Histórico de Revisões

Nome	Alterações	Data	Versão
Henrique Resende Hirako	Criação do Documento	02/11/2012	0.1

Sumário

1 Introdução

Uma aplicação geralmente é formada por um conjunto de códigos de acesso a dados, de regras de negócio e de apresentação ou interface. Quando esses códigos estão misturados vários problemas podem surgir. A interdependência entre esses componentes pode provocar dificuldade de manutenção, impossibilidade de reuso, repetição de código no caso de inclusão de novas apresentações e mesmo o código de acesso a dados pode sofrer os mesmos problemas. O padrão Model-View-Controller (MVC) – em português Modelo-Visão-Controlador – soluciona essas dificuldades na medida em que propõe desacoplar o acesso aos dados e as regras de negócio da maneira como o usuário pode visualizá-los.

2 Definição

Model-view-controller (**MVC**) é um um padrão de arquitetura 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. Com o aumento da complexidade das aplicações desenvolvidas torna-se fundamental a separação entre os dados (Model) e o layout (View). Desta forma, alterações feitas no layout não afectam a manipulação de dados, e estes poderão ser reorganizados sem alterar o layout.

2.1 Modelo

Modelo (Model) – responsável por manter o estado da aplicação. Ele é mais que uma classe para armazenar os dados, nele devem estar contidas todas as **regras de negócio que se aplicam a esses dados.** O modelo também é responsável por comunicar-se com o banco de dados, se for necessário.

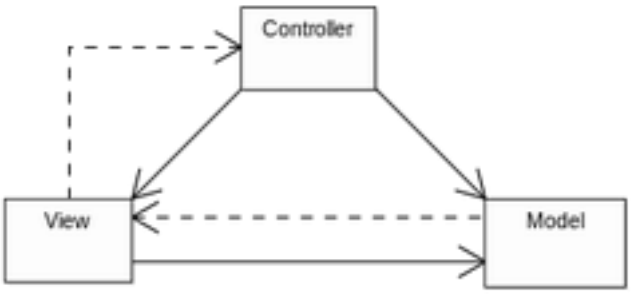
2.2 Visão

Visão (View) – especifica exatamente como o modelo deve ser apresentado. É a interface do usuário. A visão é dinâmica, se adequando a qualquer modificação no modelo.

2.3 Controlador

Controlador (Controller) – o controlador traduz as interações do usuário com a visão, mapeando-as para tarefas que o modelo irá realizar.

2.4 Diagrama



Um diagrama simples exemplificando a relação entre Model, View e Controller. Obs.: as linhas sólidas indicam associação direta e as tracejadas indicam associação indireta.

3 Explicação da definição

O model-view-controller resolve este problema através da separação das tarefas de acesso aos dados e lógica de negócio, lógica de apresentação e de interacção com o utilizador, introduzindo um componente entre os dois: o Controller. MVC é usado em

padrões de projeto de software, mas MVC abrange mais da arquitetura de uma aplicação do que é típico para um padrão de projeto.

Descrição do Padrão

MVC diz como os componentes da aplicação interagem.

Note: A partir do momento em que dividimos os nossos componentes em Camadas podemos aplicar o MVC nestas. Geralmente isto é feito definindo a Camada de Negócios como o Model, a Apresentaçãocomo a View. O componente Controller exige um pouco mais de controle. Logo, cuidado para não confundir MVC com separação de camadas. Camadas dizem como agrupar os componentes. O MVC diz como os componentes da aplicação interagem.

O MVC baseia-se em 2 princípios fortes. - O Controller Despacha as Solicitações ao Model - A View observa o Model.

3.1 Model

A representação "domínio" específica da informação em que a aplicação opera. Por exemplo, aluno, professor e turma fazem parte do domínio de um sistema acadêmico. É comum haver confusão pensando que Model é um outro nome para a camada de domínio. Lógica de domínio adiciona sentido a dados crus (por exemplo, calcular se hoje é aniversário do usuário, ou calcular o total de impostos e fretes sobre um determinado carrinho de compras).

Muitas aplicações usam um mecanismo de armazenamento persistente (como banco de dados) para armazenar dados. MVC não cita especificamente a camada para acesso aos dados, porque subentende-se que estes métodos estariam encapsulados pelo Model.

3.2 View

"Renderiza" o model em uma forma específica para a interação, geralmente uma interface de usuário.

3.3 Controller

Processa e responde a eventos, geralmente ações do usuário, e pode invocar alterações no Model. É lá que é feita a validação dos dados e também é onde os valores postos pelos usuários são filtrados.

MVC é muito visto também em aplicações para Web, onde a View é geralmente a página HTML, e o código que gera os dados dinâmicos para dentro do HTML é o Controller. E, por fim, o Model é representado pelo conteúdo de fato, geralmente armazenado em bancos de dados ou arquivos XML.Ainda que existam diferentes formas de MVC, o controle de fluxo geralmente funciona como segue:

1. O usuário interage com a interface de alguma forma (por exemplo, o usuário aperta um botão)
2. O Controller manipula o evento da interface do usuário através de uma rotina pré-escrita.
3. O Controller acessa o Model, possivelmente atualizando-o de uma maneira apropriada, baseado na interação do usuário (por exemplo, atualizando os dados de cadastro do usuário).
4. Algumas implementações de View utilizam o Model para gerar uma interface apropriada (por exemplo, mostrando na tela os dados que foram alterados juntamente com uma confirmação). O View obtém seus próprios dados do Model. O Model não toma conhecimento direto da View.

5. A interface do usuário espera por próximas interações, que iniciarão o ciclo novamente