Emanuela Amorim Jorge
Joice Sena de Camargos
Lívia Delgado A. Carneiro
Nícolas Catarina Parreiras

Model View Controller

Belo Horizonte

Julho de 2018

Emanuela Amorim Jorge
Joice Sena de Camargos
Lívia Delgado A. Carneiro
Nícolas Catarina Parreiras

Model View Controller

Pesquisa acadêmica apresentada ao curso de Técnico em Informática do Centro Federal de Educação Tecnológica de Minas Gerais como atividade avaliativa da disciplina Linguagem de Programação II.

Professor: Prof. Cristiano Amaral Maffort

Belo Horizonte
Julho de 2018

Resumo

Model View Controller (MVC) é um padrão de arquitetura que separa um aplicativo em três componentes lógicos principais: o modelo , a exibição e o controlador. Cada um desses componentes é criado para lidar com aspectos de desenvolvimento específicos de um aplicativo. O MVC é uma das estruturas de desenvolvimento da Web padrão da indústria mais usadas para criar projetos extensíveis.

Palavras-chave: Framework. Padrão de arquitetura. Web.

Abstract

Model View Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create extensible projects.

Keywords: Framework. Architectural pattern. Web.

Sumário

1 INTRODUÇÃO 9

2 O QUE É MVC 11
2.1 Modelo 11
2.2 Visão 11
2.3 Controlador 12

3 COMO SURGIU 13

4 COMO FUNCIONA 15
4.1 Aplicação WEB 15

5 SPRING FRAMEWORK 17
5.1 Módulos 17
5.1.1 Web 17
5.1.1.1 Recursos 17

6 VANTAGENS 19

7 DESVANTAGENS 21

8 CONCLUSÕES 23

1 Introdução

Atualmente, muitos softwares e frameworks estão utilizando do padrão MVC para o desenvolvimento de seus aplicativos/sites. O MVC (Model, View e Controller) é uma arquitetura permite dividir as funcionalidades de seu sistema/site em camadas, essa divisão é realizada para facilitar resolução de um problema maior.

Embora a plataforma Java forneça uma grande variedade de funcionalidades de desenvolvimento de aplicativos, ela não tem meios de organizar os blocos básicos de construção em um todo coerente, deixando essa tarefa para arquitetos e desenvolvedores.

O componente de Inversão de Controle do Spring Framework (IoC) fornece um meio formalizado de compor componentes diferentes em um aplicativo totalmente funcional pronto para uso. O Spring Framework codifica padrões de design formalizados como objetos de primeira classe que pode integrar aplicativos. Várias instituições usam o Spring Framework desta maneira para projetar aplicações robustas e de fácil manutenção.

O trabalho de pesquisa tratará primeiramente do conceito de MVC, especificando suas camadas. Em seguida como surgiu o padrão e como ele funciona, para depois se aprofundar no Spring Framework. Por fim é citados algumas das vantagens e desvantagens da utilização do MVC.

2 O que é MVC

MVC é um padrão de arquitetura de software que separa a informação da interface com a qual o usuário interage.

É uma forma de estruturar um projeto de forma que a interface de interação (view) esteja separada do controle da informação em si (models), separação essa que é intermediada por uma outra camada controladora (controllers).

2.1 Modelo

A camada model representa e manipula seguindo as regras do negócio. Essa Camada é responsável pela Lógica

O modelo encapsula o estado e comportamento da aplicação além de ser o único componente do MVC que faz interface da aplicação frente à fonte de dados, que normalmente é representada pelo banco de dados da aplicação.

Devido aos bancos serem relacionais, é necessário no modelo existir um mapeamento dos objetos do software orientados a objeto, para as tabelas do banco de dados. Essa técnica é o Mapeamento Objeto-Relacional (ORM – Object Relational Mapping). Além desta técnica, há o padrão DAO (Data Access Object – Objeto de acesso a dados) que permite melhor acesso a dados por parte dos objetos.

A lógica de negócio não fica apenas separada da apresentação, ela sequer sabe da existência da mesma. O model é a combinação dos dados e dos métodos que os manipulam. Com isso, o reuso provido pelo MVC se dá principalmente neste componente (modelo), sendo este o núcleo funcional do sistema.

2.2 Visão

Essa camada é responsável pela apresentação, é a interface de representação do modelo, ou seja, trata-se da fronteira entre usuário e o sistema em si. A view pode dar forma mais conveniente, exibir alguns atributos e ocultar outros, atuando como um filtro para os dados do modelo. É papel do controlador definir a view apropriada para a exibição da resposta obtida pela requisição feita.

No modelo tradicional do MVC, a view é composta por GUI's (Graphical Users Interface) enquanto que no MVC web a visualização das informações acontece através das páginas HTML (HyperText Markup Language – Linguagem de Marcação de Hipertexto) para informação estática e JSP, ASP, PHP, dentre outras, para informações dinâmicas.

Outra diferença entre o MVC original e o web é que as views no original recebiam notificações do model, feitas por sua vez, através da implementação de interfaces que definem objetos observadores e observados. No caso de aplicações web estas notificações são sempre mediadas pelo controlador, principalmente pela necessidade do reuso, é na visão que ocorrem todas as interações do usuário que serão tratadas pelo controlador para chamar os métodos apropriados no modelo.

Este componente pode ser considerado o mais flexível do MVC, podendo ser facilmente alterado ou substituído. Novas visões também podem ser facilmente implementadas, sem afetar em nada a estrutura do sistema.

2.3 Controlador

Essa camada é o controlador onde controla a comunicação entre o modelo e a visão.

Pode ser considerado como a fronteira entre os outros dois componentes do MVC e sua finalidade é controlar interações que ocorram a partir do usuário que trabalha sobre elementos na camada de visão e descobre o que essa entrada significará para o modelo.

As entradas que esses componentes recebem são, normalmente, eventos de mouse, entradas de teclado, entre outras. Estes eventos por sua vez serão traduzidos em requisições de serviços para outro componente que deverá tratar, existe um controlador para cada função da aplicação, mas algumas estratégias podem ser adotadas para que se desenvolva um controle central e se evite código duplicado.

Em aplicações Web uma solução comumente adotada para a criação deste controle central é a implementação do padrão Front-Controller. Um objeto é criado e serve como ponto de entrada principal para todas as requisições que vem da view, este objeto mapeia a requisição para o tratador adequado. O Struts é um exemplo de framework que faz o papel de Front-Controller em aplicações Java para web.

3 Como surgiu

O MVC surgiu como uma forma melhorada para construção de interfaces gráficas, mas tomou grande amplitude e passou a ser utilizado, de uma forma geral, na arquitetura de sistemas complexos.

O cientista da computação Trygve Mikkjel Heyerdahl Reenskaug foi o responsável pelo Model-View-Editor que foi o primeiro nome dado ao padrão arquitetural de projeto criado em meados da década de 70.

Transpondo os sistemas que existiam em sua época e focando um estudo nas GUI's (Graphical User Interface), Reenskaug criou a primeira implementação para o MVC que surgiu como uma forma otimizada de se construir tais interfaces gráficas com o usuário e, segundo o próprio Reenskaug, a concepção do nome foi uma das partes mais duras, pois era extremamente difícil encontrar um bom nome para a diferente arquitetura de componentes.

4 Como funciona

Um dos principais objetivos do padrão MVC é a organização do código de uma aplicação em camadas, realizando assim a separação física dos componentes do software. objetivando desta forma agrupar componentes por responsabilidades em comum.

A fundamentação da divisão das funcionalidades de um sistema em camadas surgiu como alternativa para solucionar alguns problemas existentes nas aplicações monolíticas, nas quais, dados e código eram armazenados em uma mesma máquina, na qual todas as funcionalidades eram definidas em um único módulo contendo uma grande quantidade de linhas de código e de difícil manutenção.

A necessidade de compartilhar a lógica de acesso aos dados entre vários usuários, impulsionou o desenvolvimento de aplicações em duas camadas. Nesta estrutura, a base de dados é armazenada em uma máquina específica (servidor) diferente das máquinas que executam as aplicações (clientes). Um problema desta abordagem é o gerenciamento de versões, pois para cada alteração de uma das regras implementadas obriga a atualização dos aplicativos em todas as máquinas clientes.

4.1 Aplicação WEB

Porém, a Internet trouxe para os usuários e desenvolvedores uma nova visão de aplicativos. Neste novo paradigma, a aplicação (vista na Internet como site) é disponibilizada por meio de um servidor web no qual o usuário apenas realiza requisições. As requisições são processadas no servidor web e as respostas são enviadas ao usuário.

Desta forma, surgiu a necessidade de responder, dinamicamente, às solicitações dos usuários, característica essa que contribuiu para os esforços dos desenvolvedores em separar a lógica de negócio da interface com o usuário, ressurgindo o modelo MVC de desenvolvimento.

Este modelo consiste em uma tríade de classes freqüentemente usadas em sistemas interativos para construção de interfaces com o usuário. A implementação deste modelo mantém o núcleo funcional do sistema independente da interface. Assim, as interfaces internas podem permanecer estáveis, mesmo quando a interface necessita ser alterada para se adaptar a novas plataformas e dispositivos de interação.

Com isso a apresentação, a lógica e o acesso ao banco de dados estão separados em camadas específicas, tornando os sistemas mais manuteníveis e garantindo a independência entre estas camadas. Desta forma, as camadas de negócio podem ser divididas em classes podendo ser agrupadas em pacotes ou componentes reduzindo as dependências entre as

mesmas. Esta divisão facilita a reutilização por diferentes partes do aplicativo e até por aplicativos diferentes. O modelo MVC se aplica como uma excelente arquitetura para o desenvolvimento de sistemas corporativos com base na web.

5 Spring Framework

O Spring Framework é uma plataforma Java que fornece suporte abrangente à infraestrutura para o desenvolvimento de aplicativos Java.

5.1 Módulos

O Spring Framework consiste em recursos organizados em cerca de 20 módulos. Esses módulos são agrupados em Core Container, Data Access/Integration, Web, AOP, Instrumentation e Test.

5.1.1 Web

A estrutura de MVC do Spring Web é projetada em torno de um DispatcherServletque envia solicitações para manipuladores, com mapeamentos de manipulador configuráveis, resolução de visualização, resolução de localidade e de tema, além de suporte para o upload de arquivos. O manipulador padrão é baseado nas anotações @Controllere @RequestMapping, oferecendo uma ampla variedade de métodos de manipulação flexíveis. Com a introdução do Spring 3.0, o @Controllermecanismo também permite criar sites e aplicativos RESTful, por meio da @PathVariableanotação e de outros recursos.

No Spring Web MVC, pode-se usar qualquer objeto como um comando ou objeto de backup de formulário, não precisa implementar uma interface ou classe base específica do framework. A vinculação de dados do Spring é altamente flexível, por exemplo, trata erros de tipo como erros de validação que podem ser avaliados pelo aplicativo, não como erros do sistema. Portanto, não precisa duplicar as propriedades de seus objetos de negócios como strings simples e sem tipos em seus objetos de formulário, simplesmente para lidar com envios inválidos ou para converter as Strings corretamente. Em vez disso, muitas vezes é preferível vincular-se diretamente aos seus objetos de negócios.

5.1.1.1 Recursos

- Clara separação de papéis: Cada função - controlador, validador, objeto de comando, objeto de formulário, objeto de modelo DispatcherServlet, mapeamento de manipulador, etc - pode ser atendido por um objeto especializado.

- Configuração direta de classes de framework e de aplicação como JavaBeans: Esse recurso de configuração inclui referências fáceis em contextos, como de controladores da Web a objetos de negócios e validadores.

- Código comercial reutilizável , sem necessidade de duplicação: Use objetos de negócios existentes como objetos de comando ou formulário em vez de espelhá-los para estender uma determinada classe base de estrutura.

- Ligação personalizável e validação: Tipo incompatíveis como erros de validação em nível de aplicativo que mantêm o valor ofensivo, a data e a vinculação de número localizadas e assim por diante, em vez de objetos de formulário somente de String com análise manual e conversão em objetos de negócios.

- Mapeamento de manipulador personalizável e resolução de visualização: O mapeamento de manipuladores e as estratégias de resolução de visão variam de uma simples configuração baseada em URL a estratégias de resolução sofisticadas e específicas. O Spring é mais flexível que os frameworks web MVC que exigem uma técnica específica.

- Transferência de modelo flexível: A transferência de modelos com um nome/valor Map oferece suporte à fácil integração com qualquer tecnologia de visualização.

6 Vantagens

As vantagens em se utilizar o MVC no desenvolvimento de uma aplicação estão diretamente relacionadas com a manutenibilidade que se consegue atingir quando se divide um sistema em módulos específicos, que tratam de responsabilidades diferentes e únicas. Ainda, dentro deste contexto, a reusabilidade é bem explorada, pois uma mesma funcionalidade que é desenvolvida para uma janela (ou página web), pode ser reutilizada em outras janelas que a requisitem.

7 Desvantagens

A maior desvantagem deste modelo é o auto custo no desenvolvimento de uma aplicação, pois, o processo tende a levar mais tempo, tendo em vista que todas as classes deverão ser organizadas em pacotes que representem a sua funcionalidade. Isso obriga aos desenvolvedores a seguir um padrão no desenvolvimento de qualquer parte da aplicação, aumentando desta forma o prazo de entrega do produto. Ainda, uma outra desvantagem é que este padrão exige a presença de um profissional especializado que domine os conceitos apresentados, aumentando os gastos com treinamentos e conscientização da efetiva adoção do padrão.

8 Conclusões

O MVC que separa a lógica de negócios no model, a apresentação na view e a interação entre eles no controller, também se apresenta como uma boa escolha para a construção de aplicações web interativas. Isto devido ao fato de neste tipo de aplicação haverem grandes quantidades de interações de diversos tipos de usuários e buscas e exibições de dados.

As camadas MVC é muito importante por agilizar o desenvolvimento e até a manutenibilidade, como pois é dividido em camadas fica mais fácil de fazer alterações nos código, isso foi uma evolução tecnológica pois com a agilidade é mais fácil de manipular o código, podendo modificar qualquer camada sem modificar as outras.

O MVC, portanto, demonstram que é um padrão arquitetônico que se adéqua de forma concisa a soluções na web. Isso, graças à separação de camadas conforme a sua funcionalidade.