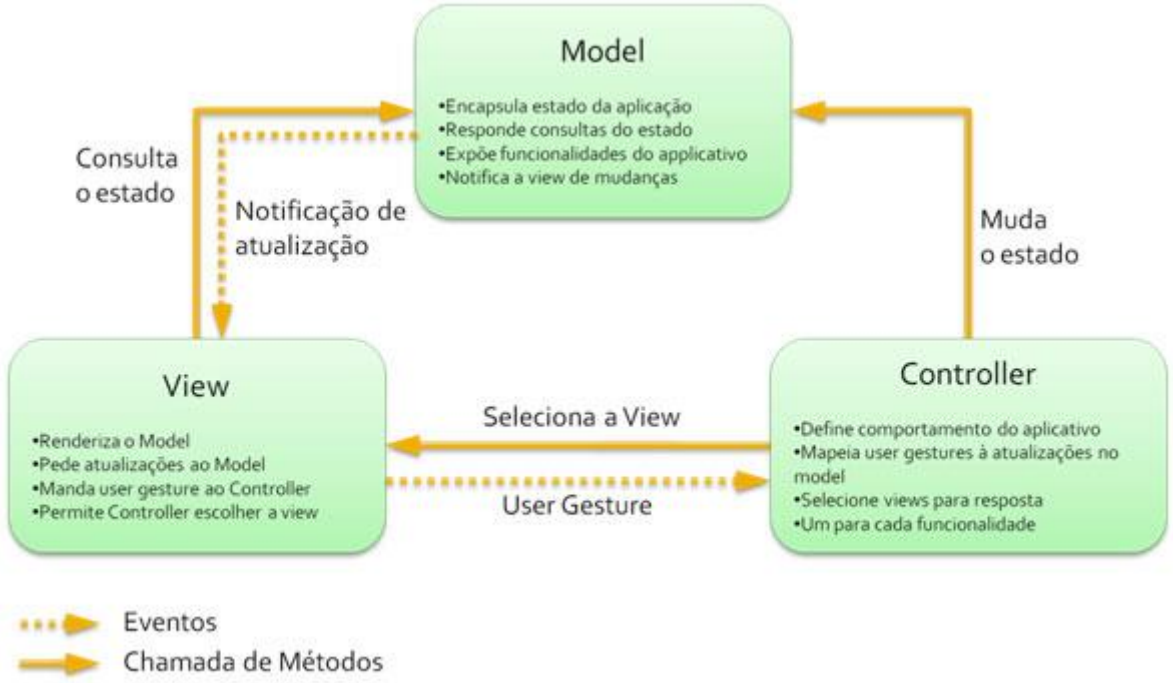
AULA 06 – M.V.C.

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 Apresentação como 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.

A figura abaixo descreve a funcionalidade e comunicação entre os componentes.



- ✓ **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.
- ✓ **View:** "Renderiza" o model em uma forma específica para a interação, geralmente uma interface de usuário.

- ✓ **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:

O usuário interage com a interface de alguma forma (por exemplo, o usuário aperta um botão).

O Controller manipula o evento da interface do usuário através de uma rotina pré-escrita.

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).

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.

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

9.1 Exemplo da organização de pacotes no NetBeans