O MVC é um conceito muito importante no mundo da programação. Esta é uma sigla que significa Model-View-Controller, ou Modelo-Visualização-Controlador, em português. Ele é usado para organizar o código de muitos tipos de software, facilitando o desenvolvimento e tornando o programa mais fácil de ser mantido e atualizado. Vamos entender melhor como isso tudo funciona.

**Por que o Modelo MVC é importante?**

Imagine que você está construindo uma casa. Se você simplesmente começar a construir sem um plano, pode acabar com cômodos estranhos ou estruturas mal posicionadas. Mas, se você definir bem os projetos antes de começar, sua casa será melhor organizada e será mais fácil de viver. O mesmo se aplica ao código de um programa, onde a casa é equivalente ao seu aplicativo e o plano é o modelo MVC.

Ao seguir o modelo MVC, dividimos o código do nosso aplicativo em três partes:

1 - O Modelo, que é onde todos os dados e as regras de negócio são processados.

2 - A Visualização, que é a interface de usuário, onde você vê os resultados das operações.

3 - E o Controlador, que é como o cérebro que opera as outras duas partes, recebendo entradas do usuário e enviando comandos para o Modelo e a Visualização.

**Como o Modelo, a Visualização, e o Controlador interagem?**

Vamos utilizar um simples exemplo de um site de cadastro de livros para explicar como o MVC funciona.

No caso do Modelo, pense nele como o repositório de todos os livros que estão cadastrados no site. Ele guarda todas as informações, como título, autor e gênero do livro. Ele também tem todas as regras, como verificar se o livro já está cadastrado ou não.

A Visualização é a página que você vê em seu navegador. Ela apresenta um formulário para adicionar um novo livro e uma lista de todos os livros já cadastrados.

E finalmente, temos o Controlador. Digamos que você quer adicionar um novo livro. Você digita as informações no formulário e clica em 'Enviar'. Aqui é onde o Controlador entra em ação. Ele “pega” a informação que você digitou, verifica com o Modelo se o livro já existe (se for a regra) e, se tudo estiver certo, pede para o Modelo adicionar o novo livro. Em seguida, instrui a Visualização para mostrar a nova lista de livros com o que você acabou de adicionar.

Vê como tudo trabalha junto? Cada parte faz um trabalho específico, mas todas se comunicam para manter as coisas funcionando perfeitamente.

**Como o Modelo MVC ajuda na organização do código e na manutenção do aplicativo?**

Quando o código é bem organizado, é muito mais fácil identificar problemas, fazer atualizações e adicionar novas funcionalidades. Por exemplo, se um dia você decidir que quer mudar a cor de fundo do seu site, você só precisa ir na *Visualização* e fazer a mudança. Se decidir mudar as regras para adicionar um novo livro, vai direto para o *Modelo*.

Essa organização também facilita muito a vida de qualquer outra pessoa que possa trabalhar no seu código no futuro. E se você decidir oferecer seu site em diferentes linguagens, por exemplo, você pode simplesmente adicionar uma nova *Visualização*, sem precisar mexer no *Modelo* ou no *Controlador*.

Então, o Modelo MVC é um grande aliado na hora de desenvolver um aplicativo. Além de facilitar o entendimento e a manutenção do código, ajuda a prepará-lo para as mudanças e melhorias que sempre surgem ao longo do tempo.

**MVC em Java**

Em Java, o padrão MVC era comumente aplicado utilizando as tecnologias JSP ou Thymeleaf. Com o JSP (Java Server Pages), é possível misturar códigos Java e HTML para criar páginas web dinâmicas. Já no Thymeleaf, os códigos não se misturam, mas temos páginas HTML dentro da pasta resources do projeto. Essas páginas se comunicam com o código Java e fazem com que possamos visualizar as informações no navegador.

Essas tecnologias faziam com que a View do MVC estivesse dentro da aplicação Java. Porém, atualmente o mais comum no mercado é ter uma separação entre front-end e back-end. No back-end, temos a Model e o Controller. O back-end fornece informações que serão utilizadas pelo front-end, que será responsável pela parte da View.

Agora que você sabe a teoria, é hora de colocar a mão na massa e começar a praticar. E lembre-se: cada novo conceito aprendido é um passo à frente em sua jornada como desenvolvedor. Boa sorte!

## Pergunta sobre o MVC

Suponha que você seja a pessoa desenvolvedora responsável pelo ADOPET, um site de adoção de animais. Seu chefe recentemente lhe pediu para transformar a aplicação linha de comando do site em uma API, para melhorar a interação dos usuários com o sistema. Então, você cria uma primeira rota HTTP para a aplicação.

A fim de aplicar seus conhecimentos recém-adquiridos, você precisa escolher o código correto para criar essa primeira rota, que será mapeada como /animais. Qual das alternativas abaixo representa corretamente o código em Spring para iniciar este trabalho?

@GetMapping("/animais")

**public** **String** **getAnimais**(){

**return** "Todos os animais";

}

Este código em Spring Boot é uma maneira correta de criar uma primeira rota HTTP para visualizar todos os animais.