02

O padrão MVC

Muitos sistemas tornam-se difíceis de manter porque existem trechos de código que tem muita responsabilidade, ou seja, fazem muita coisas. Aplicações desktop são um exemplo empírico desse problema; muitos códigos legados em VB ou Delphi são difíceis de manter porque, um mesmo trecho de código é responsável por capturar dados do usuário (através de caixas de texto, botões, etc), tomar ações a partir desses dados (por exemplo, alterar uma tabela no banco de dados), e por fim alterar a interface do usuário para mostrar que o programa reagiu da maneira esperada (exibindo uma mensagem de sucesso).

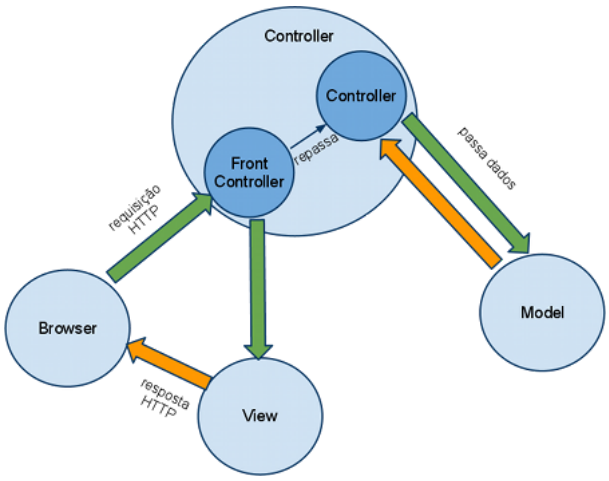
O problema da falta de divisão de responsabilidades não ocorre apenas no desktop. Na web, temos páginas ASP que contém, além de código HTML (para a apresentação do resultado), código VBScript com a lógica de negócio e código SQL acessando o banco de dados. Tudo isso é difícil de manter. O código fica espalhado e muitas vezes repetido.

Uma possível solução para o problema é tentar separar as diversas tarefas citadas acima, da seguinte forma:

Quando um usuário interage com o sistema, ele executa uma ação e obtém visualmente o resultado da mesma. Isto é, primeiro uma regra de negócios é executada (adicionar algo, atualizar algo, buscar algo), e depois informações são mostradas na tela.

Isso forma três camadas: A camada chamada de `Model` é responsável somente pelas regras de negócio; a camada de `View` é responsável unicamente por exibir os dados para o usuário final; por fim, a camada `Controller` é responsável por receber as ações feitas pelos usuários na `View` e convertê-las em chamadas de negócio dentro do Model. O padrão conhecido por MVC (Model-View-Controller) sugere que o programador faça exatamente essa separação no código.

Veja o desenho abaixo:



- Todo sistema possui regras de negócios, que podem ser simples ou complexas. Todas elas devem estar separadas em classes que tem essa regras como única responsabilidade. Ou seja, toda e qualquer ação de regra de negócio deve ser realizada por exclusivamente esse conjunto de classes.
- Interfaces de usuário também devem ser isoladas. Códigos de interface tendem a ser grandes e a sofrerem mudanças constantes. Por esse motivo, toda parte responsável pela exibição das informações para o usuário devem estar isoladas em outro ponto do sistema.
- Para conectar as ações que o usuário realiza na interface e fazer com que essas ações resultem em execuções de regra de negócio, é necessário que uma outra camada faça essa tarefa. Ou seja, essa camada recebe informações da camada de visualização e as transforma em invocações de regras de negócio.

O padrão MVC tem se mostrado uma maneira educada de separar essas camadas, e por isso sua utilização é crescente no mercado. Muitos frameworks, inclusive de outras linguagens de programação, como é o caso do `Ruby on Rails` (do mundo Ruby) e `VRaptor` (do mundo Java), adotam o MVC como solução para separar responsabilidades.

O ASP.NET MVC, como o próprio nome diz, segue o padrão MVC. Veja que os nomes adotados pelo MVC são idênticos aos do padrão MVC. Suas regras de negócio devem ficar em classes C# convencionais dentro da pasta `/Models`. O código responsável apenas pela interface do usuário são salvas na pasta `/Views`. Por fim, as classes controladoras, salvas no diretório `/Controllers`, conectam a view com os modelos.

Perceba a separação! Não há mistura de código! Isso faz do MVC um dos padrões mais populares para desenvolvimento de aplicações!

Leia um pouco mais sobre o MVC no site do Martin Fowler no site (http://martinfowler.com/eaaCatalog/modelViewController.html)http://martinfowler.com/eaaCatalog/modelViewController.html (http://martinfowler.com/eaaCatalog/modelViewController.html).