02

O padrão MVC

Códigos procedurais, que fazem muita coisa, são problemáticos. 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).

Separação de responsabilidades

Muitos sistemas tornam-se difíceis de manter porque existem trechos de código que tem muita responsabilidade, ou seja, fazem muitas coisas.

Um bom exemplo desse problema são as páginas JSPs que contém, além de código HTML, código Java e código SQL. Tudo isso é dificil 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.

Model-View-Controller

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.

Modelo: 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.

View: 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.

Controller: 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.

Frameworks MVC

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

Entender MVC é fundamental para desenvolver na Web, independente da tecnologia ou framework escolhido.