

Mar 06

O porque da camada de negócios no meu MVC

Postado em : Sexta-feira, 6 de Março de 2009 | Autor: Paulo Teixeira

Ontem estive conversando com um amigo desenvolvedor assim como eu.

Ele e eu conversávamos sobre frameworks que devemos adotar e tudo mais... Nisso saiu a questão de padrões de desenvolvimento utilizado. E ele me perguntou como eu padronizava minhas aplicações.

Eu dei uma breve explicação e mostrei umas classes para ele e meu padrão **MVC**.

Para que entendam a questão segue o formato.

- **business**
- **models**
- **controllers**
- **views**

Daí veio a questão, "Só não entendi o por que do business?". Bom achei interessante a questão que ele levantou. Dei alguns exemplos e um deles gostaria de compartilhar com todos.

Na estrutura adotada eu organizo as informações da seguinte maneira em meu desenvolvimento.

- business - (camada de negócios), nesta camada eu armazeno meus acessos a banco, como uma classe **DAO**, tenho métodos padrões que fazem acesso por PK, acesso geral, Save e Delete denominados da seguinte forma:

- **Request()**
- **RequestAll()**
- **Save()**
- **Delete()**

Tenho algumas propriedades que me servem como auxílio para as consultas mais complexas.

- **BQL** (Business Query Language)
- **Expression**
- **OrderBy**

Detalhando as propriedades acima:

- BQL me serve como uma propriedade onde eu posso escrever uma consulta SQL nos padrões da minha camada de negócios a ser utilizada no lugar da consulta do método Request(). Quando eu seto uma consulta na propriedade BQL eu informo automaticamente para o método Request() que o mesmo usará o conteúdo de BQL como consulta.

- Expression permite que eu submeta apenas uma expressão nas minhas consultas. Ex:

```
<cfset objUsuarios = application.objeto.Users />  
<cfset objUsuarios.setExpression("perfil_usuario",objUsuarios.Equal,"1") />  
<cfset objUsuarios.Request() />
```

Desta forma, a minha consulta inserirá o parâmetro no WHERE da minha query: perfil_usuario = '1'.

- OrderBy permite assim como a propriedade Expression, inserir da mesma forma uma ordenação. Ex:

```
<cfset objUsuarios = application.objeto.Users />  
<cfset objUsuarios.setExpression("perfil_usuario",objUsuarios.Equal,"1") />  
<cfset objUsuarios.setOrderBy("nome_usuario",objUsuarios.OrderDesc) />  
<cfset objUsuarios.Request() />
```

Como podem entender, o código acima, será adicionado o parâmetro no WHERE e será ordenado decrescente mente pelo nome do usuário.

Bom, agora que conhecem as propriedades, que são setadas por meio de setters de acesso públicos e são recuperadas somente de dentro das suas próprias classes, vamos continuar com uma dica básica sobre a utilização de Business como camada de negócios no MVC.

Eu tenho uma funcionalidade no ColdFusion que já me resolveu vários problemas de desempenho e também já deixou alguns amigos impressionados. Trata-se do Query of Query como todos conhecem ou ao menos já ouviram falar.

Usando a camada de negócios eu tenho a possibilidade de usar a camada models para recuperar as queries feitas em business e através das queries usando Query of Query trabalhar com cache de dados.

Portanto as minhas consultas na camada de negócios (business) são todas cacheadas, e na camada de modelo eu recupero todas e executo as filtragens usando "QOQ", dessa forma, eu ganho em desempenho, pois minha aplicação não vai mais ao banco até o cache chegar em seu tempo limite ou alguma mudança ocorrer.

Isso também é muito legal para usar com Flex, pois as aplicações ficam muito rápidas, já que só acessam o banco no início da requisição ou quando modificamos alguma coisa.

Então gente, segue a dica espero que sirva para alguns, não deixei códigos nem os componentes e etc... dei somente a idéia para que vocês tenham liberdade para criar do jeito que quiser e modificar tudo que precisar.

As camadas **controllers** e **views** fazem o que no **MVC** são de suas responsabilidades mesmo.

Abraços a todos.