Pacote de desenvolvimento ASP.NET MVC

Guilherme Cardozo Pinto **Ijacson Nogueira Dionizio** **Leandro Duarte Valente Nunes**

cardozogp@gmail.com **ijacson_dionizio@hotmail.com** **son.leandro@yahoo.com.br**

RESUMO

Com o aumento da complexidade das aplicações, tornou-se extremamente necessária a separação das diversas responsabilidades do sistema. Dividir as aplicações web em módulos denominados camadas mostrou-se um bom método para tal. A cada uma dessas camadas foi designada uma tarefa específica dentro do sistema, cujas responsabilidades serão apresentadas e abordadas neste artigo através do enfoque a tecnologia da Microsoft chamada ASP.NET MVC, bem como as melhorias da versão 2.0.

Palavras-Chave: ASP.NET MVC 1 e 2 ; WebForms; Camadas; Reutilização.

1. INTRODUÇÃO

O modelo MVC(Model-View-Controller) surgiu como uma nova forma de dividir as responsabilidades envolvidas na manutenção e apresentação das informações de uma aplicação web, apesar de ser fortemente utilizado também em ambientes desktop. A arquitetura foi desenvolvida para mapear as rotinas de entrada, processamento, saída de informação, bem como a persistência dos dados. Aplicações desenvolvidas sobre essa perspectiva são denominadas aplicações multicamadas. Um esboço desse paradigma está ilustrado na figura abaixo.

Figura 1- Comunicação no MVC

Model ou Modelo é a camada responsável pelas regras de negócio e persistência dos dados. Um componente View ou Visualização encaminha para um Controller ou Controle as requisições externas de um usuário da página web. Esse Controlador se comunica com o Mo-

delo, requisitando ou alterando os dados de interesse. Um Controller, é quem define como a aplicação web vai ser executada, passando as requisições ou chamadas do usuário para o modelo de dados. Baseado na ação do usuário da aplicação, no resultado do processamento e estado do modelo de dados, o controlador escolhe a View que será exibida no final do ciclo. Deve-se ressaltar que a View pode ser tanto um navegador web como um ambiente desktop, ou mesmo um telefone celular ou um Palm. Nesse artigo as Views serão abordadas em ambiente web tradicional. Normalmente, há um controlador que corresponde a cada funcionalidade da aplicação, isso nos remete a um dos princípios da Engenharia de Software, que define obrigatoriamente uma classe controladora para cada caso de uso do sistema.

2. PACOTE DE DESENVOLVIMENTO ASP.NET MVC

O Pacote de Desenvolvimento Asp.NET MVC é uma criação da Microsoft para implementar o modelo Model-View-Controller às páginas web. Para utilizar o Pacote de Desenvolvimento é preciso instalar a .NET Framework de Desenvolvimento 3.5 SP1 e o Visual Studio 2008. A Estrutura padrão de um projeto Asp.NET MVC está exemplificada na figura abaixo:

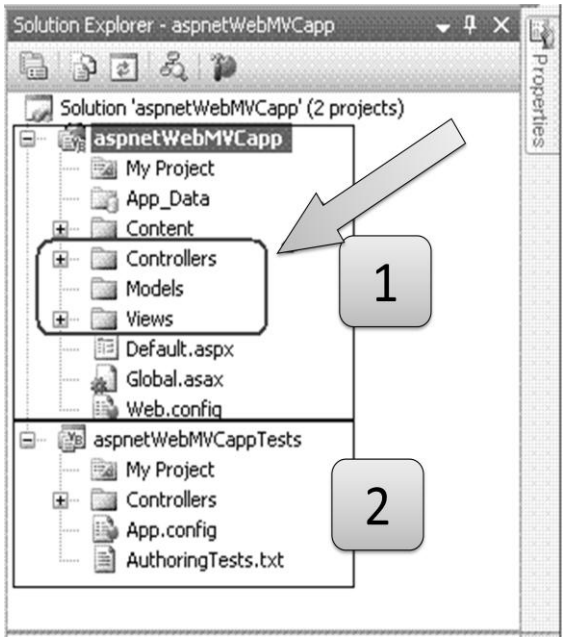

Figura 2 - Extrutura de um projeto ASP.NET MVC 1.0

Toda aplicação que utiliza o mencionado Pacote de Desenvolvimento contém 2 projetos principais. Um é o projeto web, onde será desenvolvida a aplicação e o outro projeto é responsável pela realização de testes unitários. A pasta Controllers é o local onde serão criadas as classes controladoras do sistema. Dentro da pasta Models estarão as classes responsáveis pela persistência e manutenção dos dados. E, por fim, a pasta Views conterá as visões da aplicação web.

Na grande maioria dos sistemas desenvolvidos para rodar em ambiente web, as URL's representam páginas web que estão armazenadas no servidor. Esse é o caso do Desenvolvimento Asp.NET Web Forms, a mais convencional forma de Desenvolvimento Asp.NET até o momento. Por exemplo, uma URL descrita da seguinte forma: \Produtos.aspx possui um template correspondente em disco, que representa uma página HTML. Quando é feita uma requisição http e a mesma chega ao servidor web, o Pacote de Desenvolvimento executa a rotina especificada pelo modelo e realiza o processamento da requisição. Já o Pacote de Desenvol-

vimento Asp.NET MVC mapeia a URL de uma forma em diferente do convencional. Ao invés de mapear a URL para um arquivo template no disco, o Pacote de Desenvolvimento mapeia a URL diretamente para as classes controladoras(Controllers). Essas classes controladoras recebem a requisição do usuário, quando necessário acessam o modelo de dados e chamam um componente View que gerará o HTML para o usuário. Isso é exemplificado na figura abaixo:

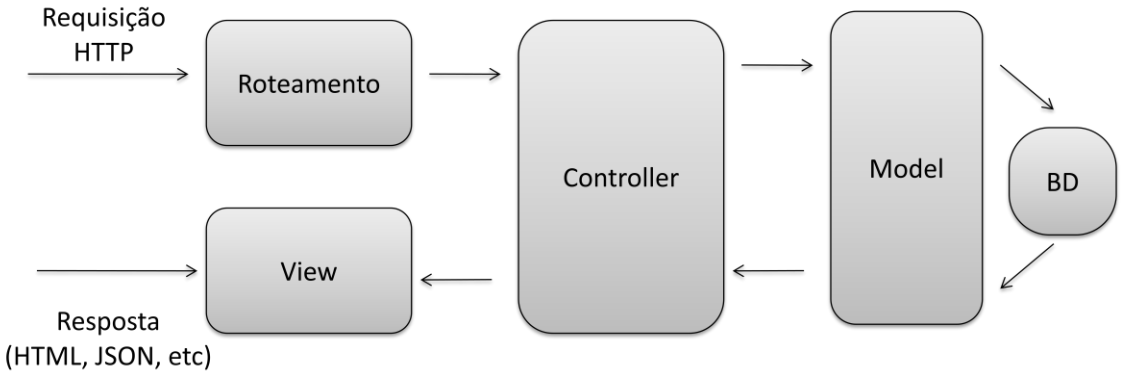

Figura 3 - Roteamento no ASP.NET MVC

O modelo padrão para fazer o roteamento de uma requisição HTTP é o modelo UrlPathController, ou seja, uma url iniciando com "/Produtos" será direcionada ou mapeada para uma classe controladora chamada "ProdutosController". Segundo o site www.macoratti.net:

> "O ASP .NET MVC mapeia as URLs diretamente para classes que são conhecidas como Controllers e a regra convencional de mapeamento de URL é que ela irá procurar a classe Controller equivalente ao nome informado na URL."

Uma das formas de utilizar o Pacote de Desenvolvimento MVC é fazer uso dos métodos Action. Esses métodos estão dentro das classes controladoras e são eles que respondem a cada requisição do usuário. Cada método Action responde a apenas uma requisição. E uma requisição pode ser processada por apenas um método Action. Quando um usuário faz uma requisição através de uma url no browser, o Pacote de Desenvolvimento MVC utiliza regras de roteamento descritas no arquivo Global.asax e determina o caminho do controller, que, por sua vez, determina qual método Action é o mais apropriado para responder a determinada requisição HTTP.

Por convenção, a url de uma requisição HTTP é considerada um caminho no qual estão contidos o nome da classe controladora e o nome do método Action, seguidos ou não de parâmetros.

Ex: Se um usuário fizer uma requisição HTTP utilizando a url http://aedb.com/Vendas/Produtos/Categorias, chamamos de sub-caminho o trecho: /Produtos/Categorias.

Neste exemplo, de acordo com as regras de roteamento do Pacote de Desenvolvimento Asp.NET MVC, Produtos é o nome da classe controladora e Categorias é o nome do método Action. Seguindo esse raciocínio, a regra de roteamento invoca o método Categorias, que está contido na classe Produtos, com o intuito de processar a requisição.

Em outro caso, se a url terminar como: /Produtos/Detalhes/5, a regra de roteamento passará, então, a tratar Detalhes como sendo método Action, e o número 5 será um parâmetro passado para o método Detalhes na requisição HTTP.

Outro ponto importante é como passar parâmetros para um método Action das classes controladoras. Uma das formas usuais é utilizar a API Request, nativa do Pacote de Desenvolvimento. Imagine a seguinte url: /Produtos/Detalhes?id=4. Poderemos passar o parâmetro utilizando a API da seguinte forma:

```
Public void Detalhes()
{
        int id = Convert.ToInt32(Request(“id”))
}
```

Outra forma, talvez mais simples, seria passar o valor do parâmetro da url de entrada como já sendo um argumento para o método Action. Seria da seguinte forma:

```
Public void Detalhes(int id)
{
}
```

A figura abaixo exemplifica o ciclo básico de uma aplicação Asp.Net MVC:

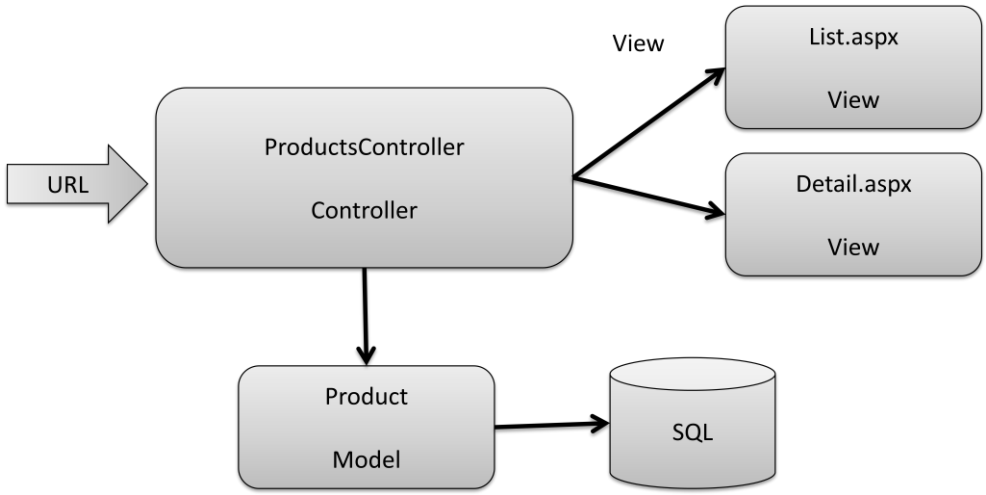

Figura 4 - Ciclo básico de vida

Os métodos Action das classes controladoras que tratam as requisições de entrada, utilizam os parâmetros contidos na url, executam a lógica adequada ao domínio, modificam o estado do modelo de dados, ou apenas retornam informações sobre o mesmo, e em seguida selecionam uma View para exibir a resposta de saída no navegador web.

3. CARACTERÍSTICAS DA ARQUITETURA MVC

O ASP.NET MVC fornece recursos tais como:

É possível testar uma unidade da aplicação sem ter que executar os controladores em um processo ASP.NET, o que torna o teste rápido e flexível. Você pode usar qualquer unidade de teste compatível com o .NET Framework.

Os componentes do Pacote de Desenvolvimento ASP.NET MVC são projetados de modo que possam ser facilmente substituídos ou personalizados. Você pode criar sua própria política de roteamento de URL, parâmetros, método de ação, e outros componentes.

O ASP.NET MVC também suporta a utilização de Dependency Injection (DI) e inversão de controle (COI). O DI permite injetar objetos em uma classe, sem ter que instanciar os objetos da classe em si. O COI estabelece que se um objeto requer outro, caso precisem estes podem ser acessados como sem fossem uma fonte externa ou um arquivo de configuração, facilitando o teste.

Possui um poderoso componente de mapeamento de URL que permite construir aplicações que têm URLs compreensíveis e pesquisáveis. A URLs não tem que incluir as extensões de nome de arquivo e são projetados para suportar os padrões de nomenclatura que funcionam com motores de otimização de pesquisa (SEO).

Suporte para usar a marcação existente na página ASP.NET (.aspx arquivos), controle de usuário (.ascx) e na página principal (arquivos.master). Você pode usar ASP.NET com recursos existentes na ASP.NET Pacote de Desenvolvimento MVC, como páginas mestras aninhadas, utilizando a expressão (<% =%>), controles de servidor, localização e assim por diante.

O ASP.NET MVC permite utilizar recursos como formas de autenticação, autenticação do Windows, autorização de URL, composição e funções de saída e cache de dados, gerenciamento de sessões, estado, perfil, sistema de configuração, e arquitetura do provedor.

4. ASP.NET WEB FORMS X ASP.NET MVC

O atual modelo de programação para internet em ASP.NET utiliza o Pacote de Desenvolvimento Web Forms para sua programação. Comparando-se os Web Forms com aplicativos desktop, o Windows está para a Internet assim como uma janela do Windows está para um web form.

Pode-se dizer que foi a primeiro Pacote de Desenvolvimento de Desenvolvimento web do Asp.NET. Foi criado em 2001/2002 e rapidamente tomou parte do mercado, deixando para trás definitivamente o Asp 3.

As principais vantagens do Asp.NET Web Forms são:

- Rápido, realmente RAD
- Designer Visual
- Controle de riscos
- Gerenciamento de estado abstraído
- Fácil de trabalhar
- Migração de Windows Form para Web Forms é muito simples
- Javascript “escondido” do programador
- Fácil interação com o SharePoint

Contudo ele também contém algumas desvantagens, como por exemplo:

- Difícil controle sobre HTML gerado
- Inicialmente tinha problemas com padrões
- Dificuldade de interação com Pacote de Desenvolvimentos javascript
- Arquitetura não estimulava busca de padrões arquiteturais, apesar de suportar
- Quase impossível testar GUI

Comparando as tecnologias, teoricamente Asp.NET MVC e Asp.NET Web Forms são funcionalmente equivalentes, no sentido de que uma equipe qualificada pode com êxito usar

ambos para construir uma solução Web, pois ambas atendem a todos os requisitos necessários para programação web.

O Pacote de Desenvolvimento Asp.NET MVC não está no mercado para substituir o Pacote de Desenvolvimento Web Forms e sim para fornecer mais opções de Desenvolvimento aos desenvolvedores web atuais. Entre as principais diferenças entre um e outro podemos citar:

- Web Forms é difícil de testar, apesar de não ser impossível, apenas trabalhoso, já o Asp.NET MVC possui toda uma estrutura voltada para seus testes automáticos.
- Asp.NET MVC exige mais conhecimentos de HTML, Javascrip, CSS e etc, sendo que no Web Forms você não precisa de tantos conhecimentos nesses requisitos pois são quase todos "escondidos" do programador.
- Arquitetura MVC é construída em cima da arquitetura MVC que faz uma separação da camada lógica da camada de negócio e apresentação e apesar de que em Web Forms você também poder fazer isso, não será como um padrão definido tal qual no MVC.
- Você pode aprender a desenvolver Asp.NET Web Forms facilmente e se tornar bom nisso, já em Asp.NET MVC você precisa ter um conhecimento maior em relação a sua estrutura e a programação avançada em HTML, Javascrip entre outros requisitos.

Portanto o sucesso no Desenvolvimento de aplicações web com o Pacote de Desenvolvimentos MVC ou Web Forms dependerá mais da aplicação em si do que a escolha pela diferença das vantagens e desvantagens desses dois Pacotes de Desenvolvimento.

> No final, o que importa é escolher entre um carro e uma moto ao fazer uma viagem. Cada viagem exige uma escolha, e tendo ambos os veículos disponíveis, estes devem ser vistos como oportunidades e não como maldições.
>
> http://msdn.microsoft.com/en-us/magazine/dd942833.aspx

5. REALIZANDO TESTES NO PACOTE DE DESENVOLVIMENTO ASP.NET MVC

Um desenvolvedor em início de carreira pode se perguntar: "Porque realizar teste em código que está funcionando". Talvez essa pergunta faça sentido para softwares mais simples ou que sofram poucas atualizações/manutenções. Quando o cenário é alterado e softwares de grande porte e que são atualizados constantemente entram em voga, a pergunta fica difícil de responder por que ela suscita outras como: "Como se prevenir de criar novos erros quando você está corrigindo um determinado trecho código?"

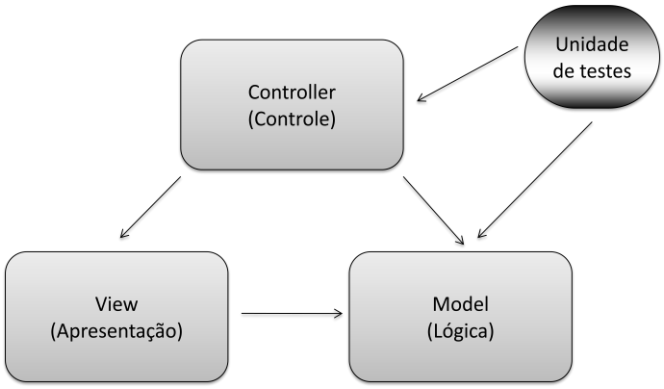

Figura 5 - Comunicação das camadas em um teste Unitário

Isso mesmo, criar novos erros. Ao desenvolver uma aplicação, um analista/programador está focado em um determinado cenário, e o código em questão funciona naquele cenário. Futuramente, outro analista, com outro cenário em mãos, irá corrigir um erro para o cenário no qual está focado. Se ele não testar o seu código para o antigo cenário, existe uma boa chance de ter criado um novo problema. Se a escala for aumentada para uma na qual

uma aplicação possui vários analistas resolvendo vários problemas em várias partes do software, o quão grande é a chance de novos erros serem gerados? Enorme.

Para evitar e agilizar os testes em softwares o Asp.NET MVC disponibiliza uma unidade de teste para garantir a qualidade no Desenvolvimento e manutenção da aplicação que será criada.

Para criar uma unidade de teste, deve-se primeiro fazer referência ao projeto principal conforme a figura:

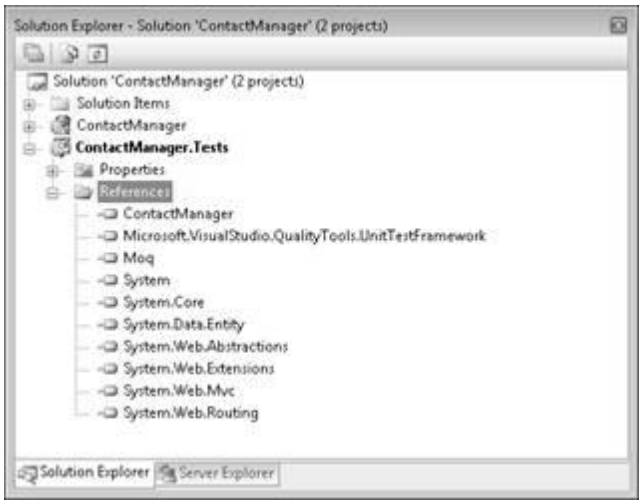

Figura 6 - Projeto para teste no Solution Explorer

Um teste basicamente verifica se um trecho de código está fazendo o que deveria fazer. Ou seja, no método abaixo é verificado se o objeto "contato" é criado com sucesso.

```
public void SucessoCriarContato()
{
        var contacto = Contato.CriarContato(-1, "José", "Silva", "555-5555",
"steve@somewhere.com");
        var resultado = _servico.CriarContato(contato);
        Assert.IsTrue(resultado);
}
```

Para executar um ou mais testes, basta ir ao menu **Test, Run, All Tests in Solution**. Se for mostrada uma janela semelhante a abaixo todos os testes foram bem sucedidos. Caso tenha havido algum problema o teste ficará marcado em vermelho para que o problema seja verificado pelo desenvolvedor e corrigido.

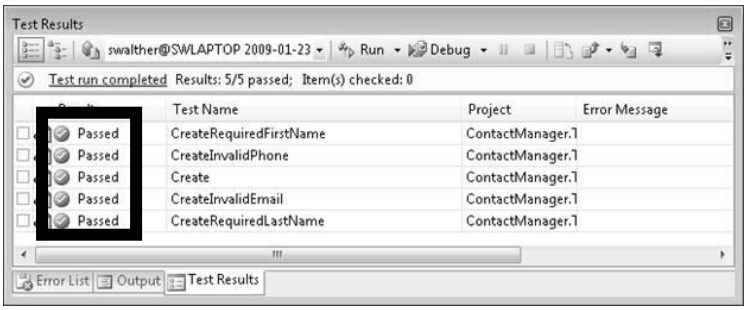

Figura 7 - Resultado dos testes realizados

Conforme já mencionado, em um software de grande porte, existem inúmeros métodos e funcionalidades. Os testes são formas rápidas e seguras de garantir o máximo possível de qualidade com tempo de Desenvolvimento relativamente curto.

6. ASP.NET MVC 2

O ASP.NET MVC 2 é uma melhoria do antigo ASP.NET MVC , sendo totalmente compatível com o mesmo. Essa nova versão já vem instalada por padrão no Visual Studio 2010. Quem usa versões anteriores da ferramenta de desenvolvimento precisa baixar o ASP.NET MVC 2 , e instalá-lo manualmente.

Segundo seus criadores a nova versão possui maior produtividade, segurança, performace e extensibilidade, além de proporcionar mais "Felicidade" ao desenvolvedor por permitir manter grandes projetos organizados através das áreas (que serão abordadas mais a frente), por possuir o básico necessário para início de um novo projeto, pelo upgrade simples da versão 1 para a 2, por ser um projeto Open Source, por já possuir integrado ao Pacote o JQuery e JQuery Validation e por fim a maior atenção ao feedback daqueles que usam a ferramenta.

7. O QUE HÁ DE NOVO?

No antigo ASP.NET MVC, já era possível perceber uma estrutura bem complexa, na organização dos diversos diretórios onde se localizavam todas as camadas, arquivos de banco de dados, arquivos de estilos, etc. Com o advento dessa nova tecnologia, fomos surpreendidos com um ambiente mais organizado do que antes. A estrutura dos diretórios tornou-se mais complexa, facilitando a organização. A figura abaixo mostra a janela do Solution Explorer, de uma aplicação desenvolvida no Visual Studio, usando o ASP.NET MVC 2.

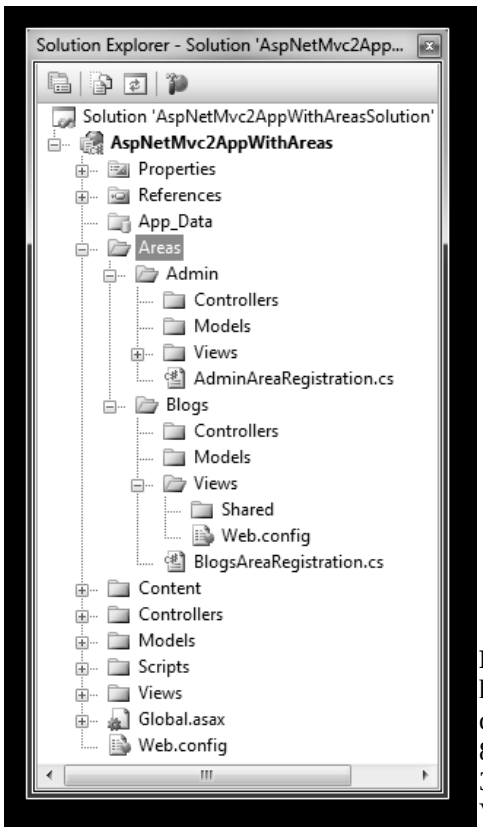

Figura 8 - http://download.microsoft.com/download/F/1/6/F16F9AF9-8EF4-4845-BC97-39791D5699C/WhatIsNewInMVC_2.pdf

8. NOVAS CARACTERÍSTICAS

Segundo o site oficial da Microsoft sobre o ASP.NET MCV houveram as seguintes inovações:

AREAS

As áreas são pequenas seções em que pode ser dividida uma grande aplicação web. Cada seção, chamada de Area, em inglês, é responsável por agrupar classes da camada controladora e/ou da camada de visão, que possuem características que permitam um agrupamento entre si. Na figura acima, nota-se duas seções, Admin, e Blogs, ambas responsáveis por uma parte da aplicação, com seus respectivos modelos, visões e controladores.

8.1 AUXILIADORES DE LAYOUT (HELPERS)

Através desta característica, é possível associar tipos de dados a componentes HTML na página. Explicando melhor, imagine um campo que contenha um dado do tipo *DateTime*. O ASP.NET MVC tem a capacidade de transformar esse campo em um calendário, em se tratando de HTML, é o componente *DatePicker*.

8.2 PROVEDORES DE VALIDAÇÃO DE MODELOS

As classes provedoras de validação de modelos são abstrações que representam provedores de lógica de validação para modelos. O ASP.NET MVC já possui seus próprios validadores baseados em atributos inclusos na classe *System.ComponentModel.DataAnnotations*. A partir do MV2 o desenvolvedor poderá criar seus próprios provedores de validação de modelos totalmente customizados.

8.3 VALIDAÇÃO NO LADO DO CLIENTE

O ASP.NET MVC 2 traz os validadores, que permitem desenvolver aplicações robustas e providas de mecanismos para tratar possíveis erros durante a interação com o usuário. Dentre os quais podemos citar:

- *Required* – esse validador obriga a digitação ou preenchimento de um campo.
- *RegularExpression* – esse validador permite avaliar um campo que contem uma expressão regular, ou seja, um conteúdo que deve obedecer a um padrão. Por exemplo, um campo para digitação de e-mail. Há regras bem definidas para esse tipo de informação. Assim, o referido validador avalia o conteúdo do campo.
- *StringLength* – esse validador serve para arbitrar um número máximo de caracteres que um campo poderá receber. Quando utilizamos o validador da seguinte forma: *StringLength(10)*, estamos dizendo que o respectivo campo poderá receber uma informação com, no máximo, 10 caracteres, incluindo espaços em branco.

8.4 SUPORTE PARA CONTROLLERS ASSÍNCRONAS

No servidor web, o ambiente de desenvolvimento mantém threads (partes de um programa) responsáveis por responder às requisições web. Quando uma requisição é feita, esse thread fica bloqueado para tal requisição, não podendo atender a outra requisição antes de acabar de processar a primeira. Tratando-se de um ambiente de alta disponibilidade, isso

seria um problema, já que o servidor poderia ficar sobrecarregado. Graças ao ASP.NET MVC 2, todo esse processo já pode ser realizado de forma assíncrona. O tempo para processar uma requisição é invariável, ou seja, independe se o processo é síncrono ou assíncrono. A diferença é que o servidor não fica mais bloqueado para responder a outras requisições enquanto completa a primeira. Isso aumenta a disponibilidade do servidor web.

8.5 SUPORTE PARA VALORES PADRÃO DE ATRIBUTOS EM PARÂMETROS DE MÉTODOS DE AÇÃO

A através da classe *System.ComponentModel.DefaultValue.Attribute,* pode-se suprir valores padrão como argumentos em parâmetros de métodos de ação, ou seja, é possível criar um método cuja assinatura de parâmetros já possui um valor padrão caso um parâmetro obrigatório não seja passado na chamada do método de ação.

8.6 MODELO DE META-DADOS E PROVEDOR DE MODELO DE META-DADOS

Através do *namespace System.ComponentModel.DataAnnotation* e da classe *ViewDataDictionary* é possível criar modelos e provedores de modelos de outros tipos de fontes de dados e até mesmo de arquivos XML.

Através da extração do modelo de Meta-dados pela classe *ModelMetadataProvider* é possível obter a estrutura do objeto *ModelMetadata* e dessa forma permitir que as classes *Helpers* gerenciem o Layout das View conforme os modelos obtidos.

8.7 SOBRECARGA DE MÉTODOS HTTP VERB'S

Para fazer isso, já que os browser's suportam apenas GET e POST HTTP Verb's, durante o POST o ASP.NET MVC 2 faz a leitura de um parâmetro oculto para fazer a emulação desses novos métodos. Os novos métodos são:

- HttpPostAttribute
- HttpPutAttribute
- HttpGetAttribute
- HttpDeleteAttribute
- AcceptVerbsAttributes

O parâmetro oculto foi nomeado como *X-HTTP-Method-Override* e só através dele a emulação dos novos verbos é possível. Além disso, para utilizar os novos métodos é preciso que a requisição real seja um POST.

8.8 EXIBIÇÃO DE ERROS DA MODEL NA VIEW

Agora a classe *Helper Html.ValidationSummary* possui um novo método que permite mostrar apenas erros da Model, ou seja agora, além dos erros em cada campo também poderão ser exibidos, antes destes, erros referentes à Model.

9. CONCLUSÃO

A tendência atual é de que as tecnologias de desenvolvimento web fiquem mais avançadas e propiciem mais possibilidades para o desenvolvedor criar. Embora o Pacote de Desenvolvimento Asp.NET MVC tenha deixado bem clara a separação entre as responsabilidades de cada camada no sistema, é possível desenvolver utilizando a metodologia de desenvol-

vimento multicamadas também em um ambiente Web Forms. A escolha entre um Pacote de Desenvolvimento ou outro pode depender do bom senso e habilidade técnica da equipe de Desenvolvimento. Um Pacote de Desenvolvimento não veio para substituir o outro. Muito pelo contrário. A tecnologia nasceu para criar mais possibilidades e, assim, enriquecer o mundo do Desenvolvimento Asp.NET.

E como se pode notar esse esforço para enriquecimento tem dado frutos já que pacote MVC 2 apresentou melhorias em segurança, performace e extensibilidade conforme o feedback da comunidade de desenvolvedores que utilizam a tecnologia.

REFERÊNCIAS

HANSELMAN, S; CONERY, R; HAACK, P. **Professional** Asp.Net Mvc 1.0. 1 ed. John Wiley Consumer, 2009.

ASP.NET - MVC - Model-View-Controller – Introdução. Disponível em < http://www.macoratti.net/08/06/asp_mvc1.htm> Acesso: Abril de 2010.

Página oficial da Microsoft sobre o ASP.NET MVC. Tutorial disponível em <http://www.asp.net/learn/mvc/tutorial-26-cs.aspx> Acesso: Abril de 2010.

Página oficial da Microsoft tratando das atualizações do ASP.NET MVC <http://www.asp.net/mvc/whatisaspmvc> Acesso: Julho de 2010