

Como fazer Deploy no ASP.NET 4?

Renato Haddad

Microsoft Most Valuable Professional MVP, MCT, MCPD e MCTS

Abril 2010

Tecnologias

Visual Studio .NET 2010, ASP.NET 4

Sumário

Neste artigo vou mostrar como fazer o deploy de uma aplicação ASP.NET 4 com o VS 2010 e o MSBuild

Introdução

O deploy (instalação) no servidor de aplicações ASP.NET sempre foi uma atividade de risco em função de fazer upload de arquivos incorretos, ou ainda de versões que não estejam funcionando corretamente, ou seja, com erros de compilação. Muitos desenvolvedores ASP.NET criam Web Application ou Web Site, e a cada alteração nos códigos fontes, seja no aspx ou na programação (.cs ou .vb) é preciso compilar, testar e se tudo estiver correto, fazer o upload para o servidor.

Este é o processo normal e comum de uma aplicação. No caso de Web Application é possível apenas subir o arquivo alterado porque o próprio .NET fará a recompilação direto no servidor, mas o risco é grande, pois se o arquivo estiver com erro, o site não irá rodar. Por outro lado, se você fizer o deploy no servidor apenas com os arquivos necessários para rodar a aplicação, aí sim, tudo estará teoricamente correto. Neste caso é preciso fazer upload de todos os arquivos aspx e os códigos estarão numa dll a cada alteração realizada nos fontes. No entanto, dependendo do tamanho da aplicação e da velocidade de upload, nem sempre esta maneira é uma boa solução.

Estrutura do Web.Config no ASP.NET 4?

Quem nunca passou pela situação de ter que ficar inserindo diversos comentários no Web.Config para diferenciar os ambientes de desenvolvimento, produção e testes. Como exemplo, se pegarmos a string de conexão para os diversos cenários, com certeza você deve controlar toda a string de conexão colocando e retirando comentários no Web.Config. Isto sem falar em outras variáveis da aplicação que são possíveis de inserir no Web.Config como por exemplo, servidor de email, contas de email, autenticação e autorização, enfim, tudo o que uma aplicação ASP.NET tem de mais comum.

Agora, no VS 2010, o ASP.NET 4 permite criar diversos arquivos Web.Config com diferentes configurações, sendo que cada arquivo contém as respectivas variáveis para os diversos cenários. Na hora da execução, debug e deploy você pode selecionar o cenário que quiser, e dinamicamente o Web.Config será montado focado no cenário escolhido.

Vamos a um exemplo, abra o Visual Studio 2010, selecione New Project. Conforme a figura 1, em Templates, selecione Visual C# / Web e no tipo de aplicação use ASP.NET Web Application. O nome do projeto será ArtigoWebDeploy localizado em c:\ProjetosNET4. Por fim, clique no botão OK para criar o projeto.

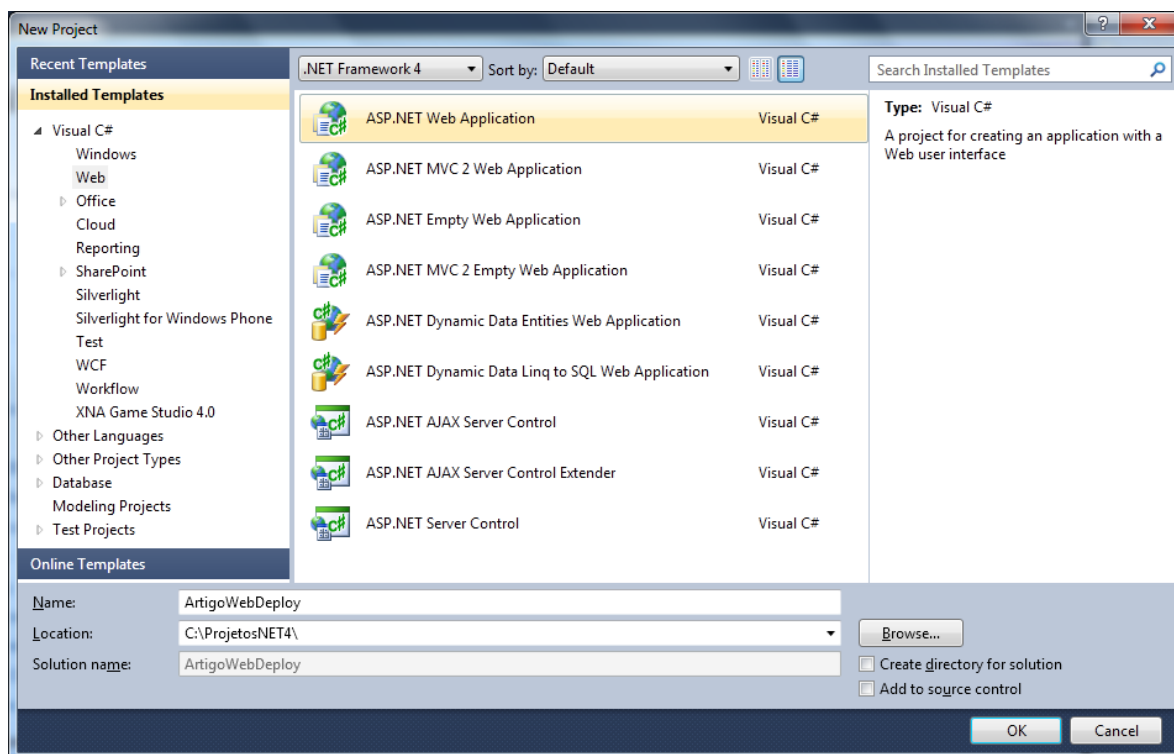


Figura 1 – Criação do projeto

O VS 2010 cria uma estrutura padrão de projetos ASP.NET com diversos formulários, arquivo de estilo CSS, a master page, etc. O que importa é que há o Web.Config. Como iremos precisar da string de conexão com o banco de dados para exibir os produtos num gridView, vamos usar o Entity Framework para isto. No Solution Explorer, dê um clique com o botão direito e adicione um novo item conforme a figura 2, onde em Templates seja Data / ADO.NET Entity Data model chamado Northwind.edmx.

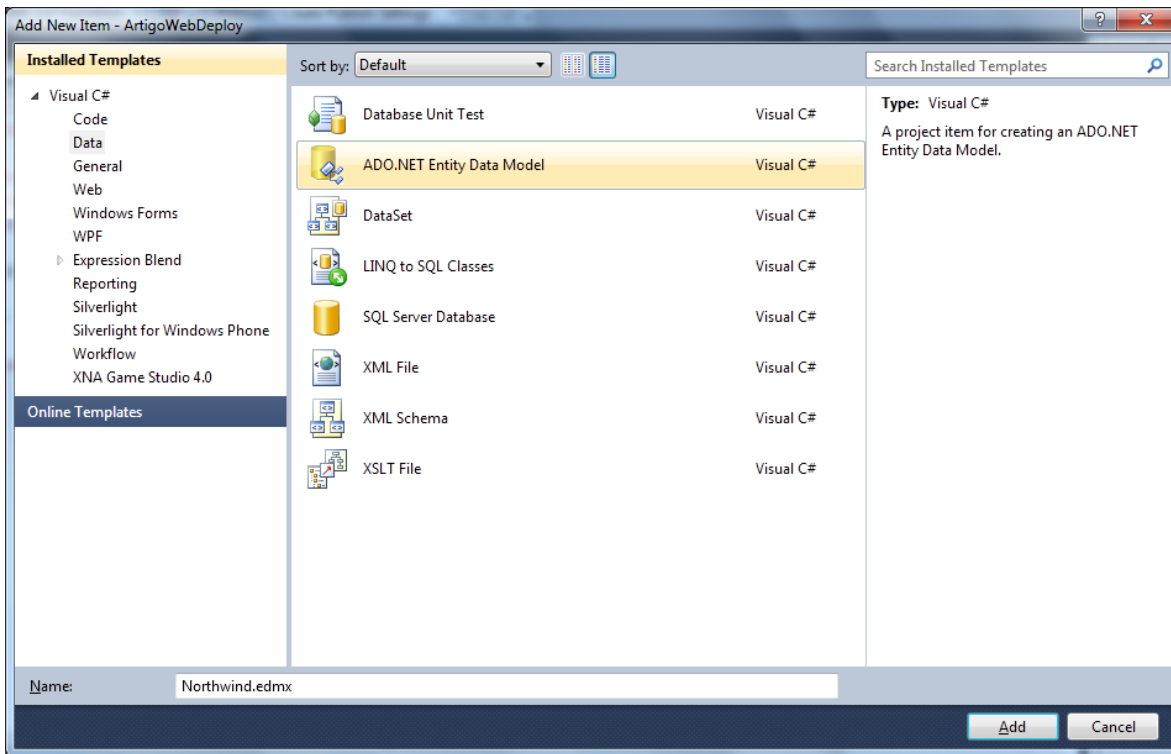


Figura 2 – Template para o Entity Framework

Em seguida, na janela aberta, selecione Generated From Database e clique no botão Next. Seguindo no assistente, aponte a conexão para o Northwind, que é o banco de dados de exemplo no SQL Server, portanto, se você não tiver baixado o banco diretamente no www.codeplex.com. Conforme a figura 3, o checkbox "Save entity connection settings in Web.Config as" está selecionado e informa que a string de conexão chamada NorthwindEntities será armazenada no arquivo Web.Config. É exatamente isto que queremos.

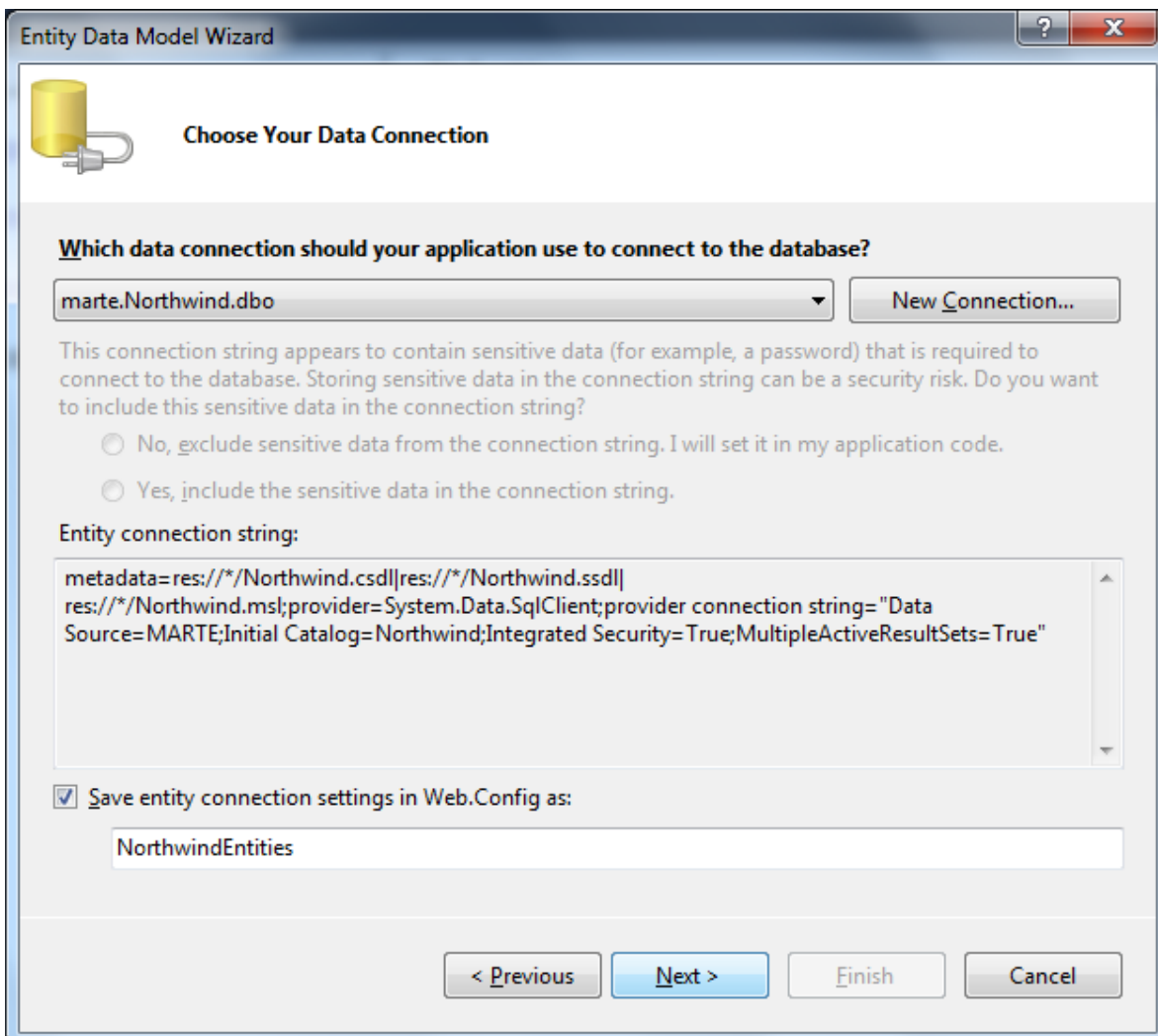


Figura 3 – Chave para o Web.Config

Clique no botão Next e no objeto Tables, selecione a tabela de Products. Clique em Finish para finalizar e conforme a figura 4, você verá o modelo de objeto relacional (ORM) criado contendo a entidade Products. Para efeito de testes é isto que precisamos. Pressione CTRL + S para salvar o ORM e feche-o.

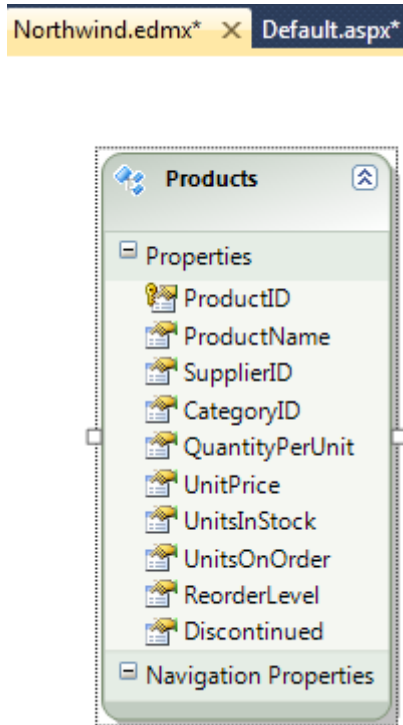


Figura 4 – ORM de Produtos

Abra a página default.aspx e altere o html para o código da listagem 1, o qual contém um gridView para exibir os dados do produto e um label chamado lblDeploy. Fique à vontade para formatar o gridView da maneira que quiser, o importante é que os produtos serão exibidos nele.

```
<asp:Content ID="BodyContent" runat="server"
ContentPlaceHolderID="MainContent">
    <h2>
        Web Deploy no ASP.NET -
        <asp:Label ID="lblDeploy" runat="server" />
    </h2>
    <br />
    <asp:GridView ID="GridView1" runat="server">
    </asp:GridView>
</asp:Content>
```

Listagem 1 – Estrutura do html

Agora vamos alterar o Web.Config para focar em 3 ambientes de deploy. Antes de mais nada, abra o Web.Config e note que a string de conexão com o banco de dados contém a seguinte declaração da listagem 2. Note que a chave chama-se NorthwindEntities e é justamente este conteúdo da chave que será trocado em tem pode execução de acordo com o tipo de deploy.

```
<add name="NorthwindEntities"
connectionString="metadata=res://*/Northwind.csdl|res://*/Northwind.ssdl|res://*/Northwind.msl;
provider=System.Data.SqlClient;
provider connection string="Data Source=Servidor;
Initial Catalog=Northwind;
Integrated Security=True;
MultipleActiveResultSets=True";"
providerName="System.Data.EntityClient" />
```

Listagem 2 – String de conexão

Vamos aproveitar que o Web.Config está aberto e insira uma nova chave para usarmos em qualquer lugar da aplicação. Veja na listagem 3 que é preciso criar a sessão appSettings, adicionar a chave (key) chamada tipo e o valor (value) que pode conter qualquer texto, neste caso é debug. Todas as variáveis da aplicação que você quiser utilizar na aplicação deverão ser incluídas neste local.

```
<appSettings>
  <add key="tipo" value="debug"/>
</appSettings>
```

Listagem 3 – Nova chave

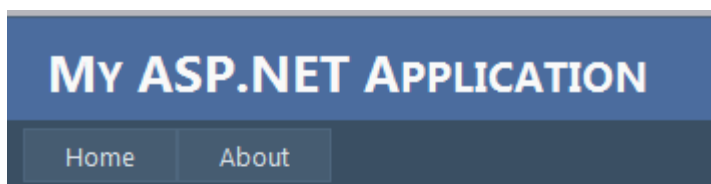
Pressione CTRL + S para salvar o Web.Config e volte para o default.aspx. Pressione a tecla F7 para exibir os códigos da página default.aspx.cs e no Page_Load da página digite o código da listagem 4 que irá inserir no lblDeploy o texto contido na chave tipo do appSettings, assim como preencher o gridView com as informações dos primeiros 15 produtos através do LINQ (Language Integrated Query).

```
protected void Page_Load(object sender, EventArgs e)
{
    // lê a chave tipo no appSettings do Web.Config
    lblDeploy.Text = ConfigurationManager.AppSettings["tipo"];

    // preenche o gridView
    using (NorthwindEntities ef = new NorthwindEntities())
    {
        var dados = (from p in ef.Products
                    select new
                    {
                        p.ProductName,
                        p.UnitPrice,
                        p.UnitsInStock }
                    ).Take(15);
        GridView1.DataSource = dados;
        GridView1.DataBind();
    }
}
```

Listagem 4 – Código do Page_Load

Salve o projeto, pressione a tecla F6 para fazer o Build. Se tudo compilou com sucesso, aparecerá a mensagem Build succeeded na barra de status. Agora execute (pressione F5) o formulário no navegador e veja o resultado, conforme a figura 5.



WEB DEPLOY NO ASP.NET - DEBUG

ProductName	UnitPrice	UnitsInStock
Chai	21,0000	25
Chang	26,0000	20
Aniseed Syrup 2	250,0000	17
Chef Anton's Cajun Seasoning	25,0000	50
Chef Anton's Gumbo Mix	21,0000	3
Grandma's Boysenberry Spread	25,0000	139

Figura 5 – Execução da aplicação

Até aqui está tudo perfeito e rodando. Note que o texto DEBUG veio exatamente da chave tipo que está no Web.Config. O que temos neste cenário é a realidade de todo desenvolvedor. Imagine que você precise alterar o conteúdo tanto da string de conexão quanto da chave tipo para executar num ambiente de produção. O que você faria? Provavelmente iria duplicar as linhas, comentar os conteúdos, e mover o Web.Config para o servidor de produção. Vou ainda mais longe para ampliar o conhecimento deste artigo. Que tal, incluir um ambiente de testes, ou seja, teremos uma outra string de conexão e outra chave tipo com outro conteúdo. Com certeza você triplicaria os dados no Web.Config e iria controlar quais dados deverão ser usados conforme o ambiente, e é claro, iria usar o recurso de comentar e descomentar.

O que o ASP.NET 4 oferece para Deploy?

Como temos três cenários com chaves distintas, o ASP.NET 4 resolve este problema e nos ajuda e muito com uma facilidade incrível. Primeiro, você precisa observar no Solution Explorer que existem 2 arquivos padrão Web.Config, sendo: Web.Debug.config e Web.Release.config. Para visualizá-los, basta expandir o nó do Web.config.

Agora precisamos criar os demais Web.config para os ambientes de produção e testes. No Solution Explorer, clique com o botão direito no nome da solução (neste caso, ArtigoWebDeploy) e selecione Configuration Manager. Será aberta a janela da figura 6.

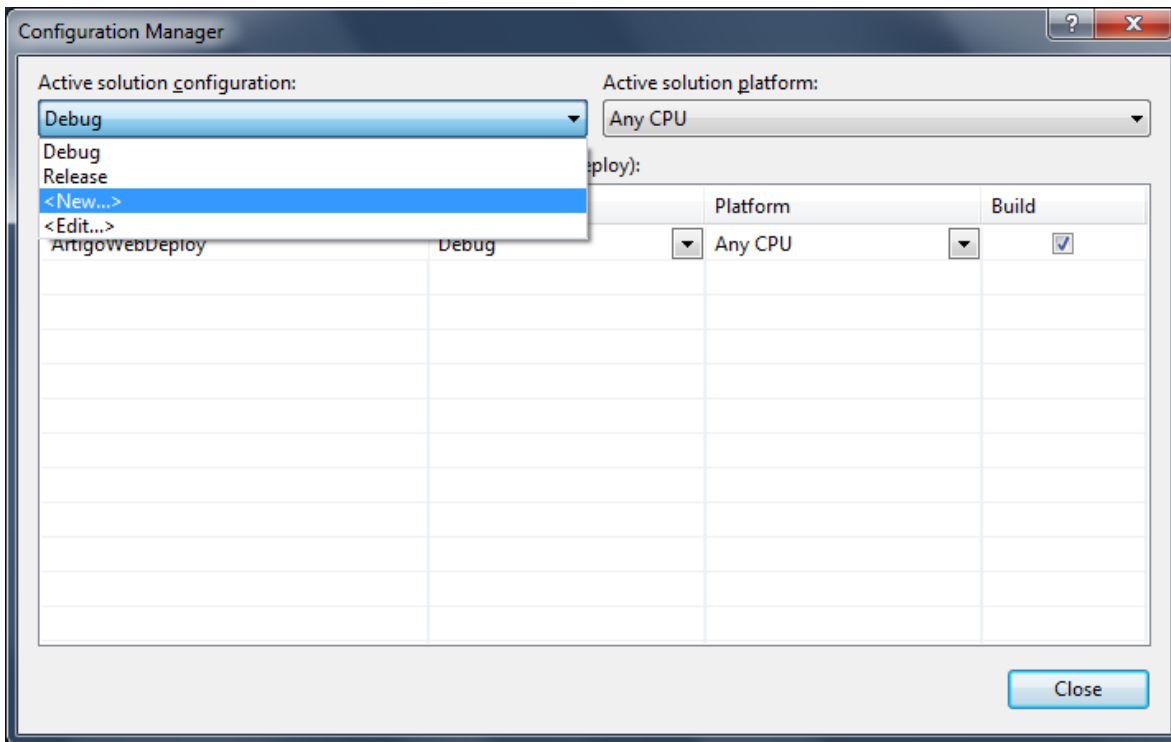


Figura 6 –Nova configuração do Web.config

Selecione a opção New, em Name digite producao, e no drop Copy settings from, selecione Debug. Isto significa que o debug servirá como base para o Web.config de produção. Ao final clique no botão OK e repite os passos para criar o Web.config de testes também baseado no debug.

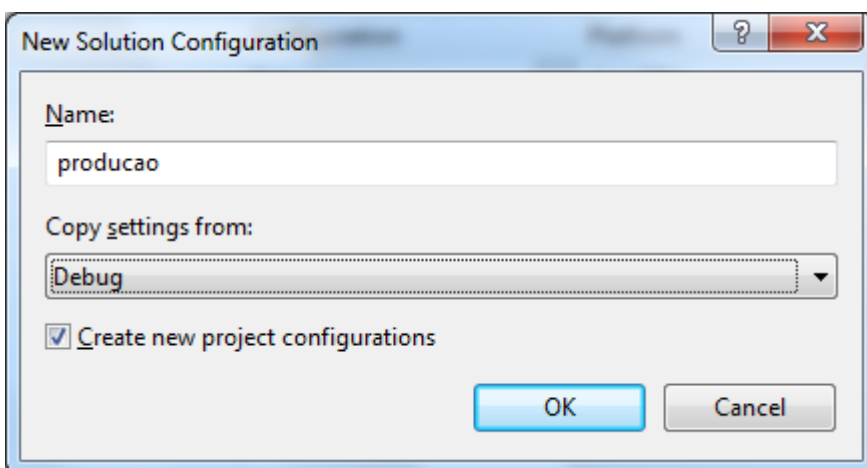


Figura 7 – Web.config de produção

Quando você fechar a janela da figura 7, os dois novos arquivos Web.config não aparecerão no Solution Explorer. Isto porque é preciso primeiro recompilar a solução (botão direito no Solution Explorer / Rebuild Solution), e em seguida, clique com o botão direito no Web.config e selecione Add Config Transforms. Ao final, você visualizará os quatro arquivos Web.config no Solution Explorer, conforme a figura 8.

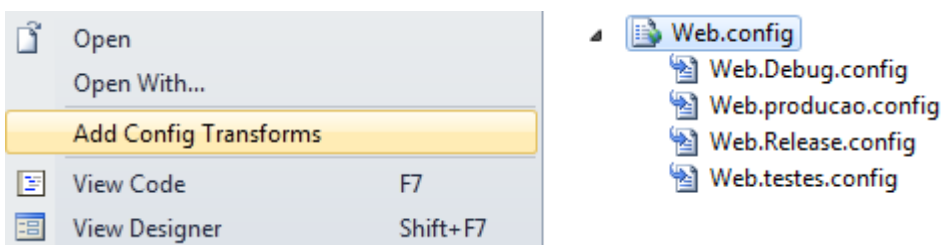


Figura 8 – Transform do Web.config

Então, chegou a hora de desmistificar o que rola nos bastidores. Uma aplicação ASP.NET 4 tem apenas 1 arquivo Web.config, no entanto, quando fizemos o Add Config Transforms foram gerados os 4 arquivos de configuração. Cada um destes arquivos contém customizações específicas para cada ambiente, o qual será gerado dinamicamente o Web.config final quando da compilação. Isto significa dizer que o Web.config é o arquivo base, e todas as customizações nos demais arquivos é o que serão substituídos dinamicamente.

Para exemplificar melhor, abra o arquivo Web.producao.config. Este arquivo contém uma estrutura XML com alguns comentários. Então, digite o conteúdo conforme a listagem 5, sendo que a string de conexão você deve descomentar, alterar o nome para NorthwindEntities, que é o mesmo nome da chave no Web.config, assim como o Data Source para o

ServidorProducao. Isto significa que você deverá alterar a string de conexão apontando para o servidor de produção, e o ServidorProducao é apenas um nome fictício que dei, no seu caso, digite o nome do servidor que contém o banco de dados. Note ainda que adicionei a sessão appSettings contendo um novo texto no value, sendo: servidor de producao. Aqui cabe um importante comentário a saber. O **xdt:Transform** irá substituir o valor de value exatamente na chave key existente no arquivo Web.config. O responsável em localizar a chave é o **xdt:Locator**. Isto vale para a string de conexão também. Ou seja, quando você fizer o deploy para publicar a aplicação, estes dados serão substituídos para gerar o Web.config final.

```
<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
  <connectionStrings>
    <add name="NorthwindEntities"
connectionString="metadata=res://*/Northwind.csdl|res://*/Northwind.ssdl|res://*/North
wind.msl;provider=System.Data.SqlClient;
provider connection string=&quot;Data Source=ServidorProducao;Initial
Catalog=Northwind;Integrated Security=True;MultipleActiveResultSets=True&quot;;"
      xdt:Transform="SetAttributes" xdt:Locator="Match (name) " />
    </connectionStrings>

  <appSettings>
    <add key="tipo" value="servidor de producao" xdt:Transform="SetAttributes (value) "
xdt:Locator="Match (key) " />
  </appSettings>

  <system.web>
    <compilation xdt:Transform="RemoveAttributes (debug) " />
  </system.web>
</configuration>
```

Listagem 5 – Customização do Web.config de produção

Repita o mesmo passo para o arquivo Web.testes.config, de acordo com a listagem 6

```
<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
  <connectionStrings>
    <add name="NorthwindEntities"
connectionString="metadata=res://*/Northwind.csdl|res://*/Northwind.ssdl|res://*/North
wind.msl;provider=System.Data.SqlClient;
provider connection string=&quot;Data Source=ServidorTestes;Initial
Catalog=Northwind;Integrated Security=True;MultipleActiveResultSets=True&quot;;"
      xdt:Transform="SetAttributes" xdt:Locator="Match (name) " />
    </connectionStrings>

  <appSettings>
    <add key="tipo" value="servidor de testes" xdt:Transform="SetAttributes (value) "
xdt:Locator="Match (key) " />
  </appSettings>
```

Listagem 6 – Customização do Web.config de testes

Desta forma você pode ter diversas configurações para todos os ambientes que desejar. O próximo passo é montar a estrutura para publicar a aplicação. Abra as propriedades do projeto e na guia Package/Publish Web você terá todas as opções necessárias para publicar o site. Aqui você deve observar que em Configuration contém todos os Web.config disponíveis, e como queremos publicar no servidor de testes, selecione Active (testes), conforme a figura 9.

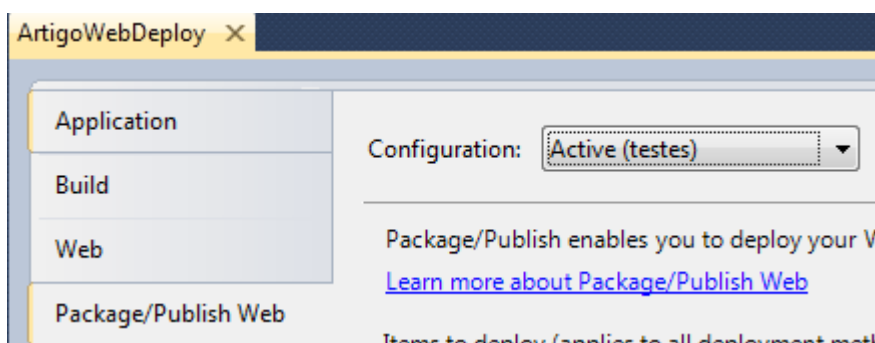


Figura 9 – Propriedades do Publish

Em seguida, na barra de ferramentas Web One Click Publish, clique no dropdown Create Publish Settings e selecione New para criar uma nova configuração., conforme a figura 10. Na janela aberta troque o nome da configuração para testes. Em Publish method, selecione File System porque iremos publicar localmente na máquina. Em Target Location digite c:\teste que é o local

onde todos os arquivos serão publicados. É interessante deixar a opção Delete all existing files prior to publish selecionada para que todos os arquivos sejam excluídos antes da publicação, assim evitamos de manter eventuais lixos no destino.

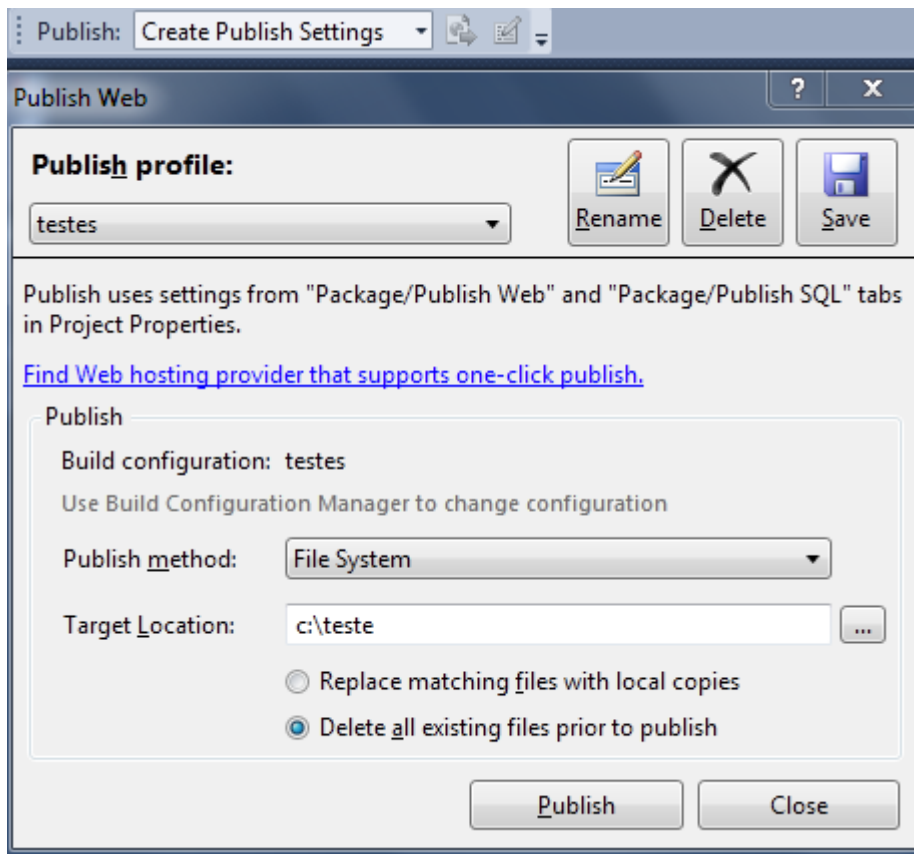


Figura 10 – Publish Web

Para publicar, basta clicar no botão Publish. Se você acompanhar o processo de publicação na janela Output, verá as seguintes linhas, sendo que ao final, consta como publicado com sucesso.

```
----- Build started: Project: ArtigoWebDeploy, Configuration: testes Any CPU -----
ArtigoWebDeploy -> C:\ProjetosNET4\ArtigoWebDeploy\bin\ArtigoWebDeploy.dll
----- Publish started: Project: ArtigoWebDeploy, Configuration: testes Any CPU -----
Connecting to c:\teste...
Transformed Web.config using Web.testes.config into obj\testes\TransformWebConfig\transformed\Web.config.
Copying all files to temporary location below for package/publish:
obj\testes\Package\PackageTmp.
Deleting existing files...
Publishing folder /...
Publishing folder Account...
Publishing folder bin...
Publishing folder Scripts...
Publishing folder Styles...
===== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped =====
===== Publish: 1 succeeded, 0 failed, 0 skipped =====
```

Através do windows explorer abra a pasta c:\teste e veja todo o site publicado. Dentro da pasta bin terá uma dll com todos os códigos escritos na aplicação. Mas, o que nos interessa é o arquivo Web.config, portanto, abra-o no VS 2010. Localize a string de conexão e o appSettings e certifique-se que os dados são os definidos no Web.testes.config. Desta forma, se você enviar esta pasta para o servidor ou via FTP, o web.config está pronto.

Para efeito de publicação direto no servidor, na janela de configuração do Publish Web, se o Publish method estiver como Web Deploy, você pode configurar o Service URL conforme a figura 11.

```
To publish locally (requires administrative rights), use:
localhost or <LocalComputerName>
To publish remotely through Remote Agent Service (typically used for intranets, requires administrative rights), use:
http://<RemoteComputerName>
To publish remotely through Windows Management Service, use the value specified by your hosting provider, such as:
<HostedRemoteServer> or https://<HostedRemoteServer>:8172/Msdeploy.axd
```

Figura 11 – Service URL

Na figura 12, veja um exemplo desta configuração que usei no Road Show que fizemos em 16 cidades para a publicação diretamente no servidor da orcsweb nos EUA usando o Web Deploy. O endereço para consulta é <http://173.46.159.126/default.aspx>.

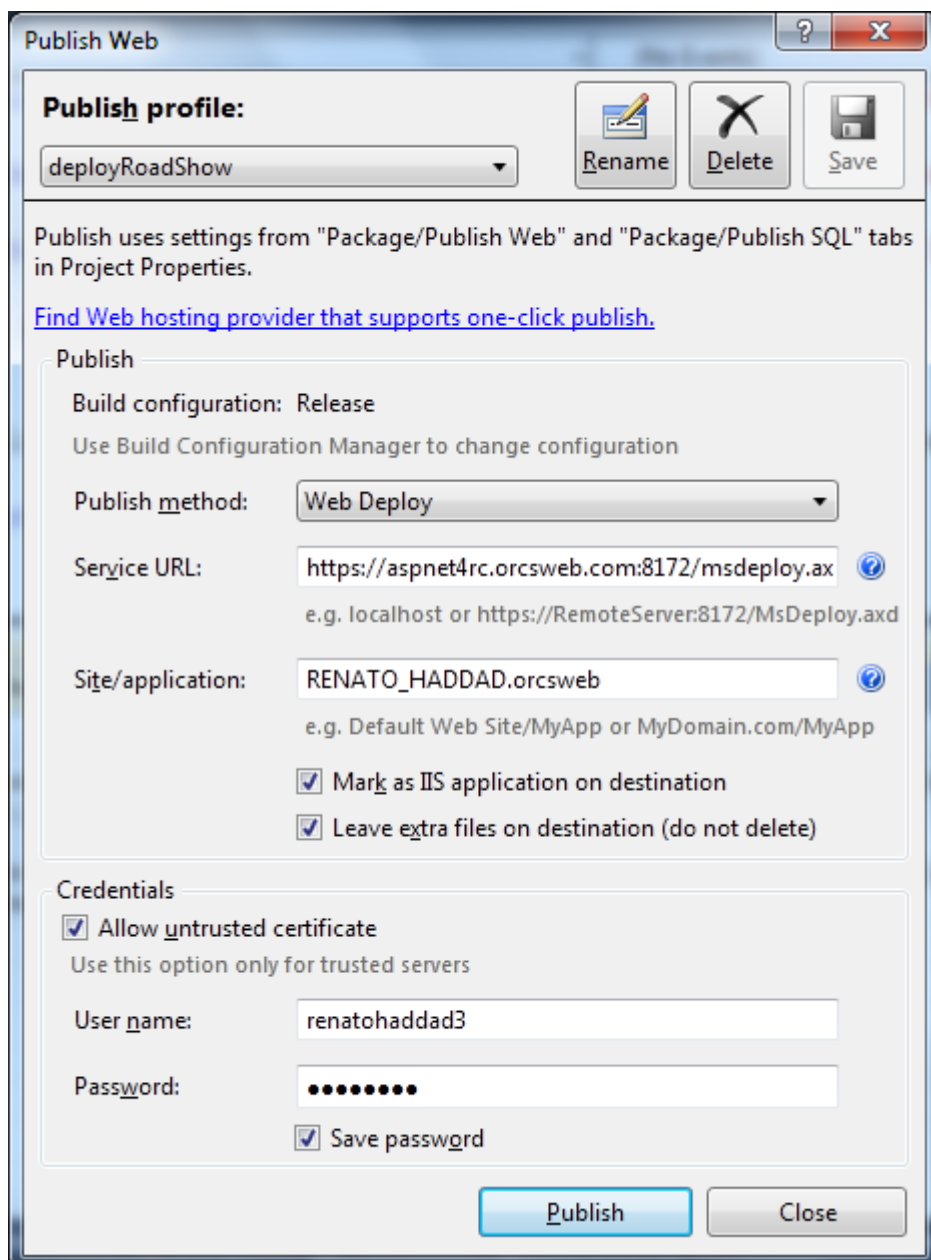


Figura 12 – Publicação direto no servidor

Na figura 13 você encontra outras interessantes configurações para o pacote Web Deployment Package Settings, sendo:

- Create deployment package as a zip file: é possível criar um pacote zipado contendo todos os arquivos da aplicação
- IIS Web site/application name to use on the destination server: nome do site ou da aplicação a ser usada no IIS

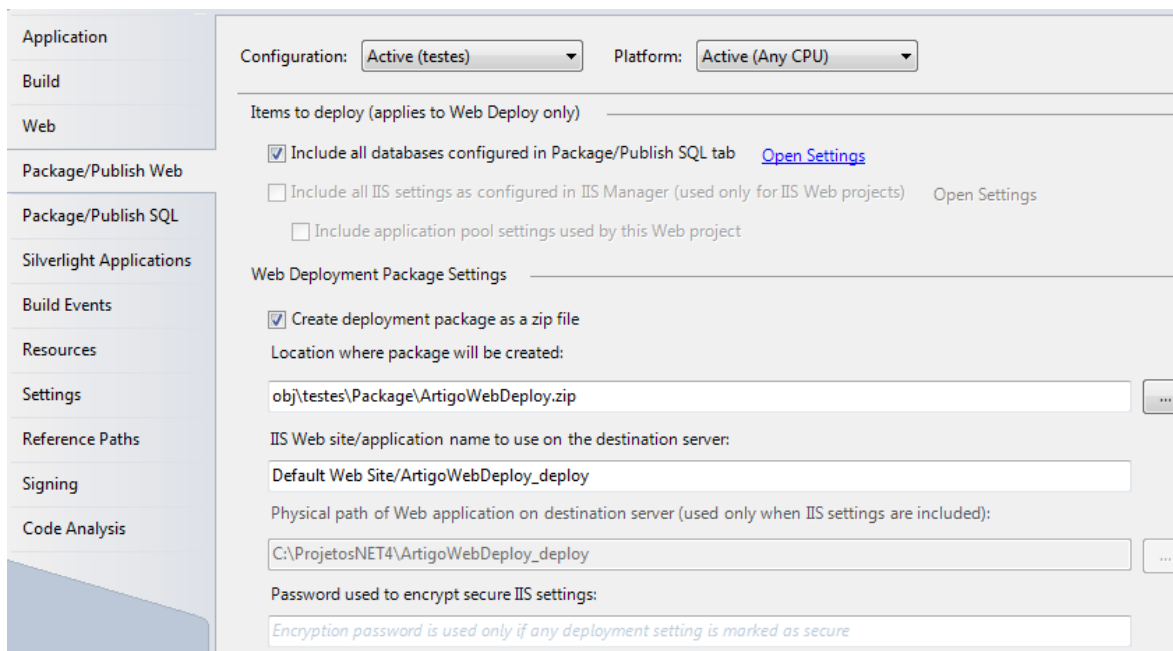


Figura 13 – Demais configurações

Enfim, como os recursos de deploy e publicação do ASP.NET 4 e o Web Deploy com o MS Deploy, tudo ficou mais fácil para qualquer time de desenvolvimento.

Conclusão

Estruturar uma boa forma de usar diversas configurações no Web.config para a aplicação é o que irá diferenciar a produtividade de publicar e testar os projetos de Web. Invista no deploy do ASP.NET 4 para criar todos os cenários possíveis para atender aos clientes o mais rápido possível.

Referências + Tópicos Relacionados

Help do Visual Studio .NET 2010

DVD de treinamentos de ASP.NET 4 disponível no www.renatohaddad.com

Sobre o Autor

Renato Haddad (rehaddad@msn.com – www.renatohaddad.com) é MVP, MCT, MCPD e MCTS, palestrante em eventos da Microsoft em diversos países, ministra treinamentos focados em produtividade com o VS.NET 2010, ASP.NET 4, Entity Framework, Reporting Services e Windows Mobile. Visite o blog <http://weblogs.asp.net/renatohaddad> e a loja de treinamentos www.renatohaddad.com/loja