UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
ESPECIALIZAÇÃO EM INFORMATICA:
ÊNFASE: ENGENHARIA DE SOFTWARE

Projetando a camada de apresentação no .NET Framework

por

Danilo Lopes Frota

Monografia de Final de Curso

Professor Roberto Bigonha
Orientador

Belo Horizonte, 18 de outubro de 2013

Resumo

Sumário

1. Introdução

Com a popularização da Internet como meio de comunicação ficou evidente a possibilidade do desenvolvimento de aplicações tipicamente projetadas para Desktops que pudessem utilizar a Web para obter os mesmos resultados. Uma das grandes preocupações para tornar este tipo de aplicação uma realidade é a camada de apresentação que geralmente é responsável pelo sucesso ou fracasso de uma solução com estas características.

A camada de apresentação representa a interface do usuário, sendo responsável pela interação com a aplicação. Para permitir que uma aplicação possa utilizar diferentes tipos de interface o núcleo do sistema (acesso a dados e negócios) é implementado em camadas separadas e utilizado através de eventos disparados pela camada de apresentação.

A dificuldade de projetar esta camada em um ambiente Web utilizando tecnologias tradicionais (ASP, PHP, PERL, etc...) é que cada uma delas pode apresentar resultados diferentes na visualização da interface dependendo do navegador utilizado, sendo necessário muitas vezes implementar uma interface diferenciada para cada Browser que utilizar a aplicação. Outro inconveniente é que a programação lógica de um formulário e seu layout não podem ser separados, impedindo a produtividade de especialistas em programação e Design. Além disso, a camada de apresentação baseada em Browser possui muitas limitações gerando dificuldade de aprendizado, baixa produtividade e conseqüentemente insatisfação do usuário.

Uma possível solução para a implementação da camada de apresentação é o desenvolvimento de formulários baseados em Windows que acessam recursos diretamente da Web. Este tipo de projeto solucionaria as limitações da camada baseada em Browser mas tornaria o processo de atualização dos formulários pouco confiável e complexo.

Diante dos problemas apresentados iremos demonstrar duas técnicas que podem ser utilizadas para solucionar os principais problemas da construção e implementação da camada de apresentação em um ambiente Web. Para tal, utilizaremos a tecnologia do .NET Framework como plataforma de desenvolvimento. Esta tecnologia que será detalhada no capítulo seguinte possibilita a construção de uma camada de apresentação Web mais rica, independente do navegador, produtiva e bastante similar à programação tradicional, além de fornecer mecanismos de distribuição automatizados para uma camada baseada em Windows através de Web. Utilizando esta plataforma como base para a implementação estaremos avaliando fatores como performance, interface gráfica, distribuição, segurança, portabilidade e custos para o desenvolvimento de uma aplicação proposta. Nos capítulos seguintes descreveremos a plataforma .NET e as soluções para a camada de apresentação baseadas nesta plataforma, bem como os ganhos e restrições das soluções apresentadas..

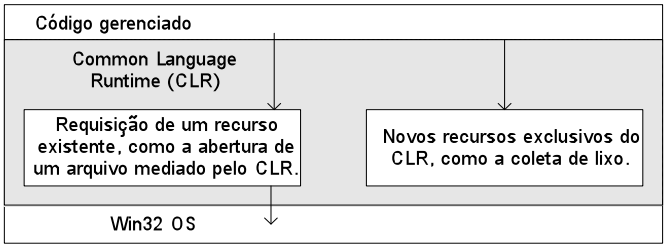
2. Tecnologia

2.1. .NET Framework

Segundo a Microsoft, .NET Framework é um ambiente para construir, ampliar e operar XML [1]WebServices e outras aplicações. Ele possui três componentes principais: a Common Language Runtime, a Biblioteca de Classes e o ASP.NET

2.1.1. Common Language Runtime

A Common Language Runtime – CLR (Linguagem Comum de Tempo de Execução) é um ambiente que propicia um conjunto de serviços mais rico que o sistema operacional padrão Win32. Ela é responsável por manter a execução de todos os aplicativos desenvolvidos com a Biblioteca de Classes do .NET Framework, provendo serviços como gerenciamento de memória, threads [2]e remoting[3]. Assim, todo o código executado pelo CLR é denominado código gerenciado e o código que não é executado pelo CLR é denominado código-não-gerenciado.



Execução gerenciada em CLR

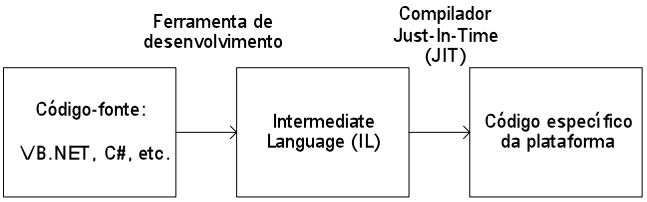
Toda a ferramenta de desenvolvimento que suporte a CLR compila o código fonte em um código intermediário chamado Intermediate Language (IL). Este código não é específico de nenhum sistema operacional ou linguagem .NET, pois como toda linguagem que suporta a CLR gera o mesmo IL não existirá diferenças de implementação quando forem executados pela Common Language Runtime.

Para executar um aplicativo existe um compilador chamado Just-In-Time (JIT) que compila o IL em um código nativo específico para o sistema operacional e arquitetura alvo.

[1] XML é um anagrama cujas letras vêm de eXtensible Markup Language, o formato universal dos documentos e dados disponíveis na Internet. XML é um protocolo de padrão industrial administrado pelo W3C

[2] Thread: Entidade básica na qual o sistema operacional aloca um determinado tempo da CPU

[3] .Net Remoting: fornecem mecanismos que permitem objetos se interagirem através de domínios de aplicações diferentes.



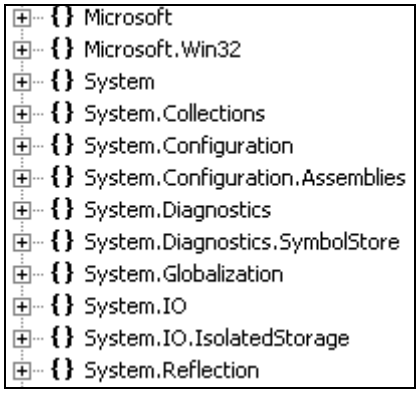
Código fonte compilado para IL

No que se refere à segurança os componentes gerenciados podem fornecer vários níveis de confiança dependendo de uma variedade de fatores inclusive sua origem (Internet, rede local ou o próprio computador onde está sendo executado). Isso significa que o componente gerenciado poderá ter ou não acesso a arquivos, operações de registro e outras funções permitindo executar código oriundo da Web com maior segurança.

2.1.2. Biblioteca de Classes

A biblioteca de classes é uma coleção hierárquica orientada a objetos integrada ao CLR para desenvolver desde aplicações de linha de comando ou de interfaces gráficas baseadas em Windows ou Browser. Estas classes estão divididas em pacotes denominados espaços de nomes (Namespaces) que são utilizadas dependendo do resultado que desejamos alcançar. Por exemplo, um Namespace contêm um bloco de construção para aplicativos Windows, outro para networking e outro para desenvolvimento Web. O conceito de espaço de nomes evita que nomes de funções e objetos usados em um aplicativo interfiram com nomes escritos por outros desenvolvedores.

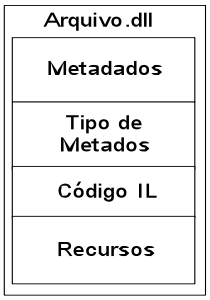
A biblioteca de classes também define alguns tipos básicos denominados Common Type System – CTS (Sistema de Tipos Comuns). Tipo é a representação de dados, no nosso caso a especificação dos mais fundamentais, como um inteiro que é representado com 32 bits. A utilização do CTS é fundamental para facilitar a interoperabilidade e robustez entre as linguagens utilizadas no .NET Framework.



Exemplo da Biblioteca de Classes.NET Framework

2.1.3. Montagens

Ao compilar um aplicativo, o código IL é armazenado em uma montagem (Assembly). Uma montagem é definida como sendo uma coleção lógica de arquivos executáveis (exe) e bibliotecas (.dll) contendo recursos e códigos de um aplicativo. Cada montagem contém um manifesto que é uma descrição de metadados dos códigos e recursos contidos dentro da montagem. Como todos os arquivos necessários para a execução da aplicação estão dentro do Assembly, a distribuição se torna muito simples porque nenhuma informação extra se faz necessária nos sistemas alvo (presumindo que o CLR esteja instalado)



Exemplo de uma montagem

A montagem também é utilizada para especificação de espaço de nomes ,versão, e segurança.

O .NET Framework permite compartilhar montagens utilizando o Global Assembly Cache (GAC), um diretório que armazena todas as montagens compartilhadas. Entretanto, ao utilizarmos compartilhamento de arquivos nos deparamos com o problema das colisões de nomes. Para garantir a utilização de nomes únicos, o manifesto da montagem utiliza criptografia de chave pública em conjunto com o nome do arquivo para formar um *nome forte* e evitar as colisões.

O manifesto de uma montagem possui também a informação da *versão de compatibilidade* que é utilizada pelo CLR para o controlar as versões requisitadas pelo cliente. Quando um cliente executa uma aplicação, o CLR procura a versão necessária para a qual o cliente foi projetado. Caso a versão exata não seja encontrada, o arquivo não será carregado, evitando o conflito de versões.

Os recursos de segurança operam por montagem permitindo diferentes níveis de privilégios para cada Assembly. O CLR permite que o administrador especifique os privilégios que cada montagem de código gerenciado possui de acordo com o grau de confiança das mesmas. Isso é feito através da configuração de um arquivo baseado em XML que especifica níveis de permissão

para empresa, máquina ou usuário. Assim, a configuração de segurança de uma aplicação .NET torna-se bastante flexível.

2.1.4. ASP.NET

Inicialmente, a Web utilizava páginas estáticas para visualização de informações evoluindo para páginas geradas dinamicamente de acordo com a interação do usuário. Estas páginas podiam ser geradas por um ambiente em tempo de execução Web chamado de Active Server Pages (ASP) como parte do Internet Information Server (IIS) que disponibilizava páginas requisitadas pelo usuário. Entretanto, este ambiente possuía vários problemas para a construção de um aplicativo de n-camadas, sendo considerado a parte fraca da cadeia de desenvolvimento. Como as páginas são criadas dinamicamente quando um cliente faz uma requisição, o script do lado do servidor leva um tempo relativamente longo para processar muitas solicitações por segundo, pois o código é interpretado. Para gerar uma página dinâmica o servidor deverá executar uma lógica de programação que contêm instruções detalhadas do HTML que deverá ser gerado, sendo que muitas vezes, estas instruções deverão ser escritas para diferentes tipos de navegadores que fazes a requisição.

Para solucionar estes e outros problemas, o .NET Framework oferece a partir do ASP.NET a criação de páginas compiladas que tenham acesso direto aos recursos do Framework. Isso significa que o ASP.NET poderá utilizar qualquer linguagem que suporte o CLR para programação lógica dos formulários.

2.1.4.1. Aplicação Servidora no .NET Framework

O modelo baseado em ASP.NET permite a separação do layout e a programação através de formulários chamados Web Forms (formulários da Web) para exibir a interface do usuário. Os WebForms são formulários Windows para Word Wide Web que introduzem um novo modelo de programação, pois são responsáveis pelo desenho da sua própria interface (gerando HTML) e pelo disparo de eventos tanto do lado cliente como do lado do servidor, ou seja, o ASP.NET traduzirá o Web Form em uma versão HTML que independe do navegador e de qualquer plataforma. Além disso, estes controles permitem a utilização de um rico conjunto de eventos e propriedades de forma semelhante à programação em ambiente Windows .

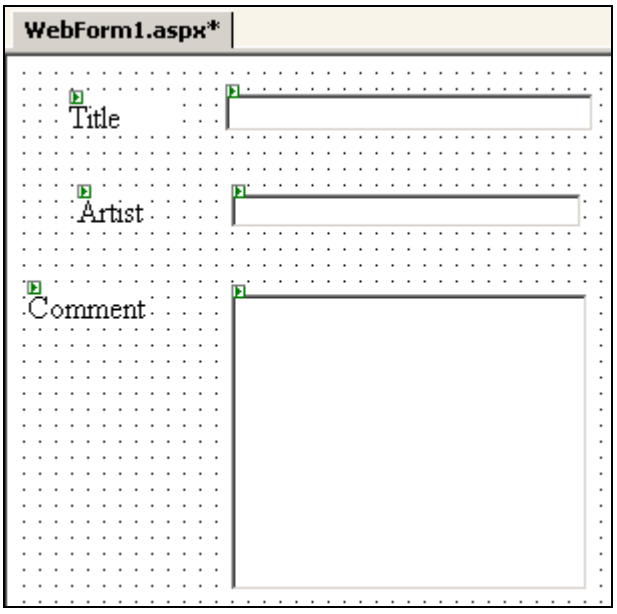


Ilustração 1 : Exemplo de um WebForm

2.1.5. Aplicação Cliente no .NET Framework

Aplicações cliente estão tradicionalmente ligadas à programação baseada em Windows (processadores de texto, planilhas eletrônicas, aplicações de negócios, etc). Estas aplicações geralmente utilizam janelas, cardápios, botões e outros elementos de interface gráfica para usuários e disponibilizam recursos locais como acesso ao sistema de arquivos e periféricos.

A evolução do desenvolvimento de aplicações baseadas em cliente no .NET Framework é um conjunto de classes que possibilita o desenvolvimento rápido de formulários Windows. O conjunto destas classes é chamado de Windows Form. O Windows Form foi criado para construção de aplicações baseadas em Windows reduzindo significativamente o custo de distribuição e desenvolvimento por tratar-se de uma ferramenta rápida e de fácil operação que utiliza um modelo de programação robusto e consistente com modelos baseados na WIN32 API. Assim, o Windows Form fornece classes .NET contendo componentes pré-fabricados de interface de usuário para várias características comuns para a maioria dos desenvolvedores.

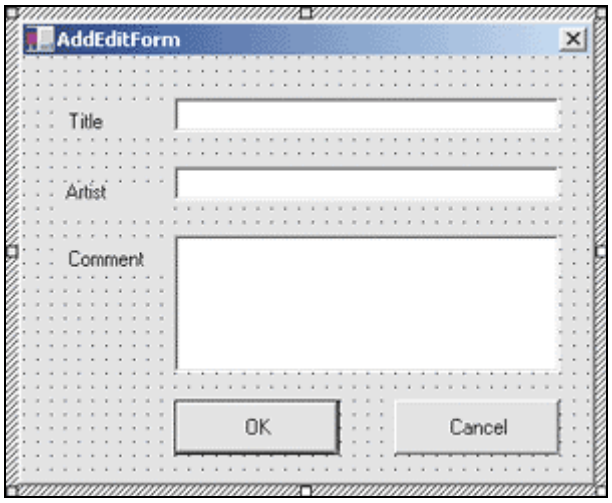


Ilustração 2 : Exemplo de um Windows Form

2.1.6. WebServices

O lado do servidor também pode oferecer através do ASP.NET serviços conhecidos como WebServices. Rod Howard classifica WebServices como uma aplicação acessível via protocolos Web padronizados. Um destes protocolos é o SOAP (Simple Object Access Protocol) baseado em XML para codificar e transmitir dados de uma aplicação.
O cliente de um WebSevice pode ser um aplicativo Windows criado com Windows Forms ou um aplicativo ASP.NET que utiliza Web Forms. Qualquer plataforma que utilize o protocolo SOAP poderá utilizar os serviços providos pelos WebServices.

3. Ferramentas

3.1. Visual Studio .NET

No nosso exemplo, utilizaremos o Visual Studio.NET como ferramenta de desenvolvimento e o C# como linguagem de programação. O VS .NET não é essencial para o desenvolvimento de aplicativos, já que poderíamos utilizar um editor de texto básico e compilar o código utilizando um compilador do .NET framework através de uma linha de comando. Contudo, podemos utilizar alguns recursos do ambiente como:

- Recursos visuais de projeto para aplicativos Windows Form e Web Form
- Editor de texto adequado à linguagem utilizada
- Depuração avançada
- Navegação pelos elementos do projeto, como código, imagens, sons, etc.

3.2. Aplicação proposta

Para demonstrar e comparar as soluções, a nossa aplicação exemplo utilizará o modelo de 3 camadas (acesso a dados, negócios e apresentação) com uma camada auxiliar que chamaremos de Web Services.
A camada de apresentação será dividida em duas. Uma camada de apresentação baseada em Web (Web Form) e outra camada baseada em Windows (Windows Form). Ambas as camadas de apresentação irão consumir recursos somente da camada Web Services que por sua vez irá prover serviços das camadas de negócios e dados. Os detalhes da aplicação serão descritos no Capitulo 4.

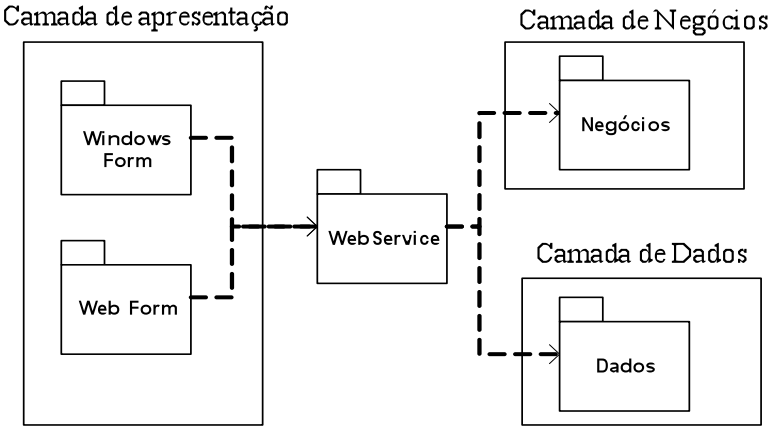


Ilustração 3: arquitetura proposta

4. Técnicas propostas

4.1. Aplicações baseadas em Browser versus aplicações baseadas em Desktop

Segundo Chris Sell, aplicações baseadas em Browser são pobres e limitadas se comparadas com aplicações Windows. Uma aplicação baseada em Windows é mais rápida, rica em relação à interface e possui uma curva de aprendizado mais eficiente que sua versão Web. Evidentemente que aplicações baseadas em Web possuem a vantagem de serem executadas por diferentes Browsers e plataformas. Entretanto, considerando as vantagens de utilização da camada de apresentação baseada em Windows (utilizando Windows Form) mostraremos a seguir uma solução baseada neste tipo de camada e outra baseada em Browser.

4.2. Conhecendo as técnicas

4.2.1. Smart Clients

4.2.1.1. Definição

A combinação das vantagens das aplicações baseados em Windows Form (rica interface gráfica e gerenciamento de estados simplificado) com as vantagens da aplicação baseadas em Web (distribuição pela Internet, acesso ao servidor para dados e serviços) é denominada pela comunidade .NET como Smart Clients. Smart Clients é uma solução caracterizada pela sua relação com Windows Form na implementação do lado cliente e com Web Services para a implementação do lado servidor. Como os Web Services não irão gerenciar estados e interface não é necessária nenhuma camada híbrida para gerar a interface gráfica que uma aplicação Web típica necessita.

4.2.1.2. Estados da Interface

De acordo com Karli Watson, estado é um termo utilizado para descrever as informações permanentes que devem ser mantidas além de um único uso ou visita a um Web Site.
Como o protocolo HTTP é stateless (sem estado) os aplicativos Web não possuem meios de saber qual usuário está visitando o site ou se o mesmo está retornando. Para capturar este tipo de informação geralmente são utilizados cookies que é uma tecnologia usada para armazenar informações na máquina cliente.

No caso do Smart Client, uma vez que a camada de apresentação baseada em Windows Form foi instalada na máquina cliente os estados das interfaces são gerenciados automaticamente como parte da implementação da sua funcionalidade, ao contrário do que acontece utilizando-se um Browser, onde o gerenciamento dos estados é o centro das preocupações dos desenvolvedores. No Smart Cliente, não existe round-trips para construção da interface porque toda lógica e dados necessários já estão no cliente. Assim, como não existe o uso de time-out de sessão, o cliente sabe exatamente quando está desconectado permitindo que a aplicação possa ser executada off-line.

4.2.1.3. Distribuindo a aplicação a partir de um Servidor Web

Com a utilização do .NET Framework, os problemas de conflitos de versão de bibliotecas (.dll) foram solucionados, pois as aplicações executadas neste ambiente são isoladas uma das outras. Com este problema solucionado uma aplicação baseada em .NET framework permite que os administradores de sistema

distribuam e atualizem a camada de apresentação da mesma forma como são feitas as atualizações na Web. Esta tecnologia é chamada de No-Touch Deployment.

4.2.1.4. No-Touch Deployment

Este processo espelha a distribuição de aplicações baseados em Browser. Utilizando métodos de biblioteca de Classes do .NET Framework, uma aplicação pode ser escrita para que os assemblies necessários possam ser baixados diretamente de um servidor Web remoto.

A primeira vez que em determinado assembly é referenciado ele é descarregado no *download assembly cache* do computador cliente e carregado para ser utilizado. Posteriormente, se o assembly requisitado já está no computador cliente e a versão da camada de apresentação do servidor Web não se alterou, a cópia local será carregada. Se a versão do servidor Web for atualizada, a nova versão será baixada e executada. Assim, as atualizações são automaticamente propagadas para o cliente evitando tráfico de dados desnecessário na rede.

4.2.1.5. Segurança

Para assegurar que o código que está sendo executado na máquina cliente não efetue acessos não autorizados (já que o Smart Client é capaz de utilizar vários recursos da máquina cliente) é possível definir diferentes níveis de confiança para a aplicação, dependendo da origem e de outras características como assinatura digital do autor.
No ambiente de execução Runtime do .NET Framework a .CLR executa checagens de segurança de baixo nível denominada *acesso seguro de código*. Esta tecnologia garante que o código executará apenas operações permitidas pelas políticas de segurança definidas. Assim se uma permissão do Assembly for negada o mesmo não será carregado.

4.2.2. Thin Client

4.2.2.1. Definição

Thin Cliente é uma característica de uma aplicação cuja parte cliente não executa nenhum tipo de processamento significativo. No nosso caso, a máquina cliente (Thin Client) deve possui apenas um Browser para visualização da camada de apresentação. Esta é enviada pelo servidor Web como páginas HTML, sendo que o processamento referente ao funcionamento da aplicação também é feito pelo servidor Web. Em nosso contexto, o termo Thin Cliente estará sempre sendo associado à utilização de Web Forms para visualização da camada de apresentação baseada em Browser.

4.2.2.2. Estados

No ASP tradicional existem algumas limitações em relação ao funcionamento de uma sessão. Estas limitações que foram solucionadas no ASP.NET incluem.

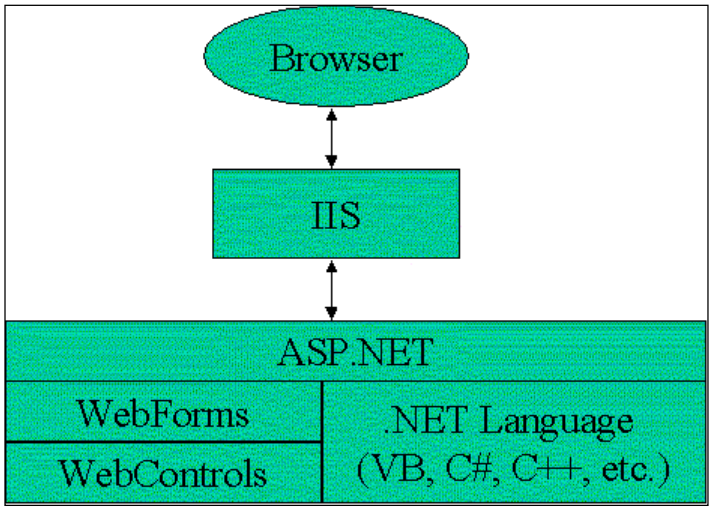
- Processamento Independente: Ao contrario do ASP tradicional, se o processo que hospeda o ASP falhar ou for reiniciado a sessão permanecerá disponível, pois o gerenciamento da sessão pode ser executado em um processo separado.
- Suporte a configurações de Web Farms[4]: Em uma Web Farm utilizando ASP tradicional o usuário se move pelos servidores enquanto a sessão é especifica por maquina (cada servidor ASP processa sua própria sessão). Com o ASP.NET estaremos utilizando um modelo de sessão out-of-process[5] permitindo que todos os servidores compartilhem a mesma sessão.
- Independência de Cookies. Clientes que não permitem a utilização de Cookies não podem utilizar as vantagens dos estados de sessão. O ASP.NET permite um gerenciamento de estados da sessão independente de cookies e similar a programação tradicional.

4.2.2.3. Distribuindo uma aplicação Thin Client

Ao contrário do Smart Client, aplicações Thin Client podem ser executadas por qualquer máquina ou dispositivo que possua um browser http para permitir que a camada de apresentação possa ser utilizada na forma de páginas

[4] Web Farms são um conjunto se servidores utilizados para suportar uma ou mais aplicações Web.
[5] A sessão é executada em um espaço de processo separado do cliente.

HTML que serão “baixadas” sob demanda. Logicamente o servidor da camada de apresentação deverá possuir o IIS instalado juntamente com o ASP.NET



Arquitetura do ASP.NET

4.2.2.4. Mapeamento dos objetos na camada de apresentação

A grande maioria dos objetos utilizados pelo Windows Form possui uma versão similar na forma de WebForm Controls. Estes objetos possuem também os eventos e propriedades mais utilizadas em uma camada de apresentação baseada em Windows Form. Desta forma, o ASP.NET contêm controles prontos que são utilizados em páginas HTML da mesma forma que os controles do Windows Form são utilizados em aplicativos.

Ao criamos um formulário WebForm (extensão .aspx) criaremos automaticamente uma classe que será utilizada para programação lógica do formulário (cs, no caso do C#).

Esta classe será referenciada no código html pela diretiva codebehind (código por trás) e terá o mesmo nome do arquivo .aspx. Desta forma, teremos o layout HTML e a programação lógica do formulário separado em arquivos diferentes. Os controles gráficos podem ser inseridos no WebForm de forma semelhante os controles utilizados no Windows Form.

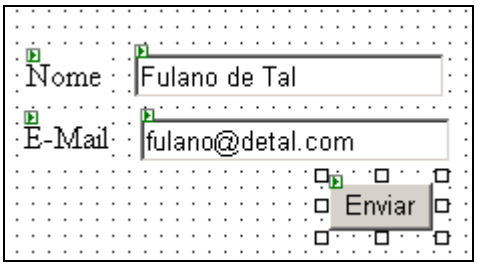


Ilustração 4: Exemplo do WebForm Exemplo.aspx

```
<%@ Page language="c#" Codebehind="Exemplo.aspx.cs"
AutoEventWireup="false" Inherits="WebUI.Exemplo" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
        <body MS_POSITIONING="GridLayout">
                <form id="Exemplo" method="post" runat="server">
                        <asp:TextBox id="txtNome" style="Z-INDEX: 100; LEFT:
75px; POSITION: absolute; TOP: 37px" runat="server">Fulano de Tal</asp:TextBox>
                        <asp:Label id="lblEmail" style="Z-INDEX: 105; LEFT: 20px;
POSITION: absolute; TOP: 70px" runat="server">E-Mail</asp:Label>
                        <asp:TextBox id="txtEmail" style="Z-INDEX: 104; LEFT:
78px; POSITION: absolute; TOP: 69px" runat="server">fulano@detal.com</asp:TextBox>
                        <asp:Button id="CmdEnviar" style="Z-INDEX: 103; LEFT:
172px; POSITION: absolute; TOP: 102px" runat="server" Text="Enviar"></asp:Button>
                        <asp:Label id="lblNome" style="Z-INDEX: 102; LEFT: 21px;
POSITION: absolute; TOP: 39px" runat="server">Nome</asp:Label>
                </form>
        </body>
</HTML>
```

Exemplo do código HTML presente no arquivo Exemplo.aspx

O arquivo Exemplo.cs (referenciado no código HTML) conterá toda à parte de programação lógica do formulário permitindo que programadores e Web Designers trabalhem de forma independente com conseqüente aumento da produtividade da equipe de desenvolvimento.

```
namespace WebUI
{
    /**/
    public class Exemplo : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.TextBox txtNome;
        protected System.Web.UI.WebControls.Label lblNome;
        protected System.Web.UI.WebControls.Button CmdEnviar;
        protected System.Web.UI.WebControls.TextBox txtEmail;
        protected System.Web.UI.WebControls.Label lblEmail;

        private void Page_Load(object sender, System.EventArgs e)
        {
            // Put user code to initialize the page here
        }

        Web Form Designer generated code
    }
}
```

Ilustração 5: Conteúdo do arquivo Exemplo.cs

Após a publicação do projeto, quando a página aspx é requisitada pelo cliente, o IIS passa a requisição para o processador de execução. O processador de execução compila a página aspx na primeira vez que é encontrada, lê a classe compilada e cria o código atrás do objeto que por sua vez contêm os manipuladores de evento. O código por trás cria os controles e os instrui a desenhar o HTML que é enviado ao IIS e repassado ao cliente.

5. Aplicação prática das técnicas propostas

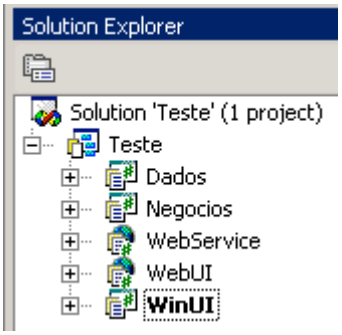
5.1. Aplicação proposta

Para nosso estudo definimos uma pequena aplicação para que possamos descrever soluções para a implementação da camada de apresentação utilizando Smarts Clients (Windows Form) e Thin Clients (WebForms).

A nossa aplicação exemplo será um sistema de financeiro de contas a pagar. Para fins de estudo iremos detalhar apenas o funcionamento de alguns formulários da camada de apresentação supondo que as outras camadas estejam implementadas.

Criaremos a nossa aplicação exemplo em C# utilizando o VS.NET como ferramenta de desenvolvimento.
Para criar a aplicação seguiremos os seguintes passos:

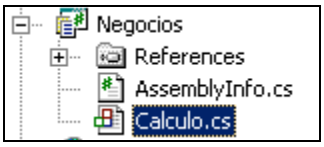
1- Criação de um novo Solution denominado Teste.sln.
2- Criação de um projeto denominado Negócios que irá conter os objetos de negócio.
3- Criação de um projeto denominado Dados que conterá as informações relativas ao acesso ao banco de dados.
4- Criação de um projeto WebServices que será responsável pela publicação dos serviços da camada de negócios e de dados através de Web.
5- Criação de um projeto (Windows Application) denominado WinUI que conterá a camada de apresentação Windows (Smart Client).
6- Criação de um projeto (WebForm) denominado WebUI que conterá a camada de apresentação Web (Thin Client).



Detalhando a camada WebService

Em nosso aplicativo criaremos alguns métodos públicos na camada de negócios e de dados. Estes métodos serão acessados pela camada de WebService que por sua vez proverá estes serviços para as camadas de apresentação.

Para exemplificar este processo criaremos um método publico chamado *calculaMulta* dentro de um componente chamado *Calculo* na camada de negócios. Este método retornará a multa calculada de acordo com o percentual estipulado.

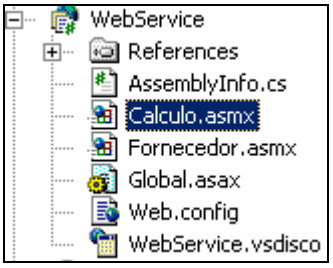



```
public decimal calculaMulta(decimal valor , decimal percentual)
{
        return (valor * percentual);
}
```

Na camada de dados criaremos um método publico chamado *inserir* dentro do componente *Fornecedor.* Este método irá inserir um novo cliente no banco de dados.


```
public bool inserir(string nome,string tipo,string endereco,string cidade,
                    string estado, string telefone)
{
        ... ... ...
}
```

Dentro da camada de WebService criaremos um serviço chamado Calculo.asmx para expor os métodos do componente Calculo da camada de negócios e um serviço chamado Fornecedor.asmx para expor os métodos do componente Fornecedor da camada de dados.



Para termos acesso os componentes das outras comandas adicionaremos uma referencia ao projeto *Dados* e outra ao projeto *Negócios*.

Para expor os serviços do WebService criamos métodos com mesmo nome e assinatura dos procedimentos públicos das camadas referenciadas, instanciamos os objetos e passamos os parâmetros com a mesma assinatura para o objeto criado. Para que a camada de apresentação possa utilizar os métodos do WebService devemos inserir a diretiva [Web Method] antes do procedimento. Assim, a camada de apresentação (Smart Client) ou (Thin Client) terão o mesmo procedimento de chamada para os serviços oferecidos pelo WebService.

```
[WebMethod]
public decimal calculaMulta(decimal valor,decimal percentual)
{
        Negocios.Calculo oCalculo = new Negocios.Calculo();
        return oCalculo.calculaMulta(valor,percentual);
}
```

5.2. Implementando Smart Clients

Na camada de apresentação do projeto WinUI utilizaremos o Windows Forms para implementação da solução Smart Client. Para que o projeto WinUI utilize os serviços do WebServices devemos primeiramente criar uma referencia Web para os serviços. No nosso exemplo criaremos uma referencia chamada wsDados (para o serviço de dados) e wsNegocios (para o serviço de negócios)

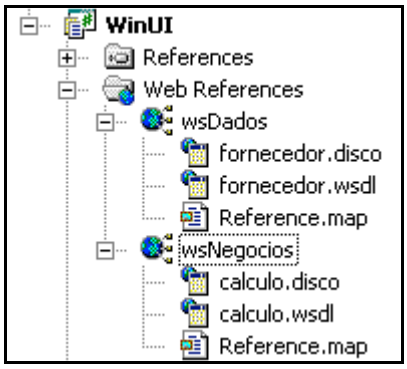
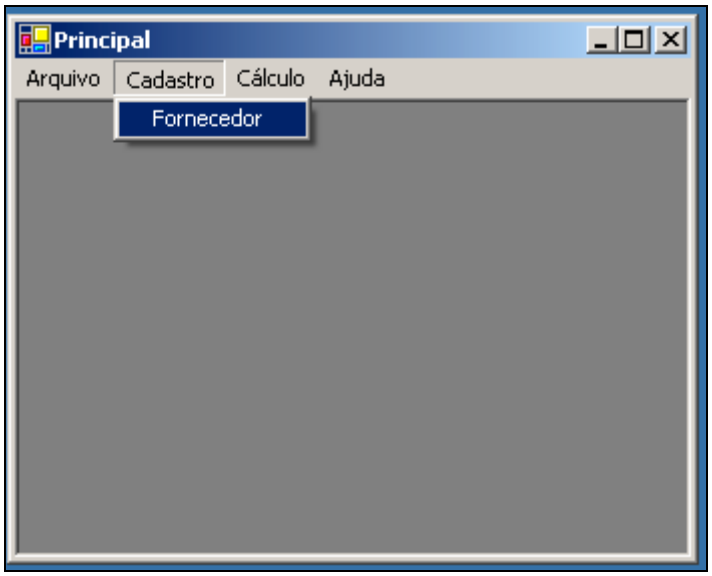


Ilustração 6: Referencias para o WebService

Para implementação do Smart Client seguiremos os passos abaixo:

- Criar um formulário principal do tipo Windows Form para o projeto. Este formulário será composto por um menu que chamará os outros formulários utilizados no projeto.



- Criar um formulário (Windows Form) para o cadastro de fornecedores.

Ilustração 7

- Inserir nos eventos apropriados as chamadas para os WebServices correspondentes. No caso da gravação de um novo fornecedor, por exemplo, instanciamos o serviço Fornecedor e chamamos o método *inserir*.

```
private void cmdGravar_Click(object sender, System.EventArgs e)
{
        ...
        wsDados.Fornecedor oFornecedor = new wsDados.Fornecedor();
        seInseriu = oFornecedor.inserir ( nome, tipo, endereco, cidade,
                                          estado, telefone);
        ...
}
```

- Criar um formulário (Windows Form) para o cálculo de multa

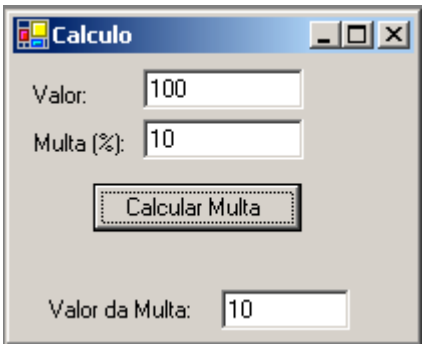


Ilustração 8: Formulário de Multa

- Inserir nos eventos apropriados as chamadas para os WebServices correspondentes. No caso do calculo de multa instanciamos o serviço *Calculo* e chamamos a função *calculaMulta*

```
private void CmdCalcular_Click(object sender, System.EventArgs e)
{
    wsNegocios.Calculo oCalculo = new wsNegocios.Calculo();
                    txtResultado.Text = oCalculo.calculaMulta(
                    Convert.ToDecimal(txtValor.Text),
                     Convert.ToDecimal(txtPercentualMulta.Text)).ToString();
}
```

- Compilar o projeto depois de inserido todo o código necessário para a implementação.
- Acessar as propriedades da pasta WinUI\Bin\Debug pelo Windows Explorer. Na pasta “Compartilhamento da Web” definir o nome do compartilhamento como “Teste”.
- Executar a seguinte url no Windows Explorer: http://localhost/Teste/WinUI.exe

Com este procedimento o arquivo WinUI.exe (arquivo gerado após compilar o projeto - contem a camada de apresentação) será “baixado” para o Global Assembly Cache (GAC) e apresentado ao usuário como um formulário Windows Form com todas as características de um formulário Windows. O acesso ao WebServices somente será feito quando chamarmos algum procedimento que irá disparar um acesso ao serviço (no nosso exemplo, um clique no botão Inserir no formulário de fornecedores). O controle do estado da interface será automaticamente gerenciado pelo Windows Form, minimizando a utilização da rede na execução de tarefas como atualização da interface do usuário.
A utilização dos WebServices para prover os serviços necessários para a camada de apresentação garante a segurança do código (já que o mesmo se encontra no servidor web), sendo que a atualização destes serviços independem da camada de apresentação, ou seja, os formulários que já se encontram no CAG serão utilizados exceto se eles também forem atualizados.
O tamanho do executável que o GAC deverá “baixar” pode ser grande (entre 1 a 5 MB) dependendo do tamanho da aplicação e código existente dentro dos formulários. Uma alternativa para evitar que toda a camada de apresentação precise ser baixada primeiramente antes da sua execução e criar uma solução separada para o Smart Client inserindo cada formulário (ou um conjunto deles) em um projeto separado.

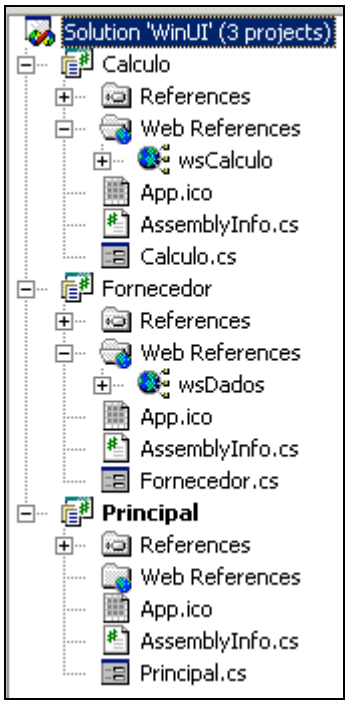


Ilustração 9: Solução separada para Smart Client

O projeto principal e compilado como WindowsApplication e os projetos como Class Library. O resultado da compilação é um executável WinUI.exe que possui apenas o formulário principal e uma biblioteca .dll para cada formulário independente. Com isso o poderemos enviar para o GAC arquivos menores que podem ser atualizados independentemente uns dos outros minimizando ainda mais a utilização da rede para distribuição dos arquivos. Assim, o GAC passará a receber arquivos cujo tamanho possuía ordem de grandeza em MB para algumas dezenas de KB.

5.3. Implementando Thin Client

Na camada de apresentação WebUI utilizaremos o ASP.NET juntamente com os WebForms e WebControls para implementação da solução Thin Client. Para que o projeto WebUI utilize os serviços do WebServices criaremos uma referencia Web para os serviços (semelhante ao que foi feito na solução anterior).

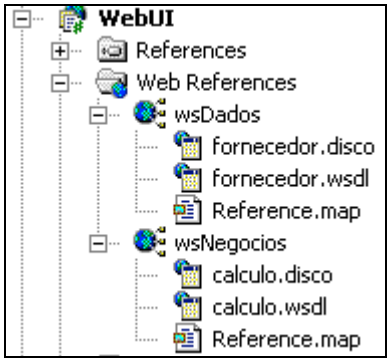


Ilustração 10: Referencias do Thin Client ao WebServices

Para implementação do Thin Client seguiremos os passos abaixo:

- Criar um formulário principal (WebForm) para o projeto. Este formulário será composto por hiperlinks que chamarão os outros formulários da aplicação.
  Obs.: Existem vários métodos para implementar um menu em um Browser inclusive WebControls de terceiros. Nossa aplicação irá utilizar o método mais simples, um HyperLink WebControl para chamada de cada um dos formularios.

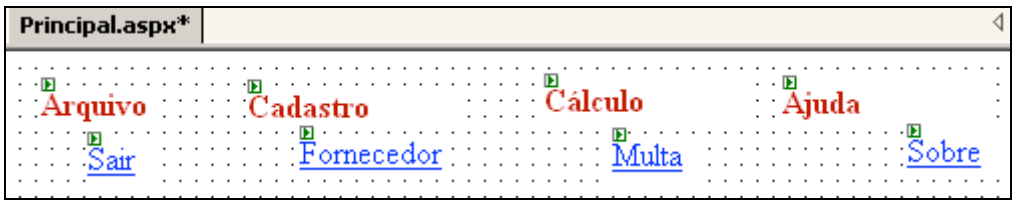


Ilustração 11: Exemplo de um formulário principal utilizando WebForms

- Criação de um formulário (WebForm) para o cadastro de fornecedores. Este cadastro terá os mesmos campos e controles presentes no projeto baseado em Windows Form. O formulário (Fornecedor.aspx) assim que criado terá uma classe de mesmo nome com extensão .cs (para C#) onde iremos inserir o código nos respectivos eventos para o funcionamento do mesmo.

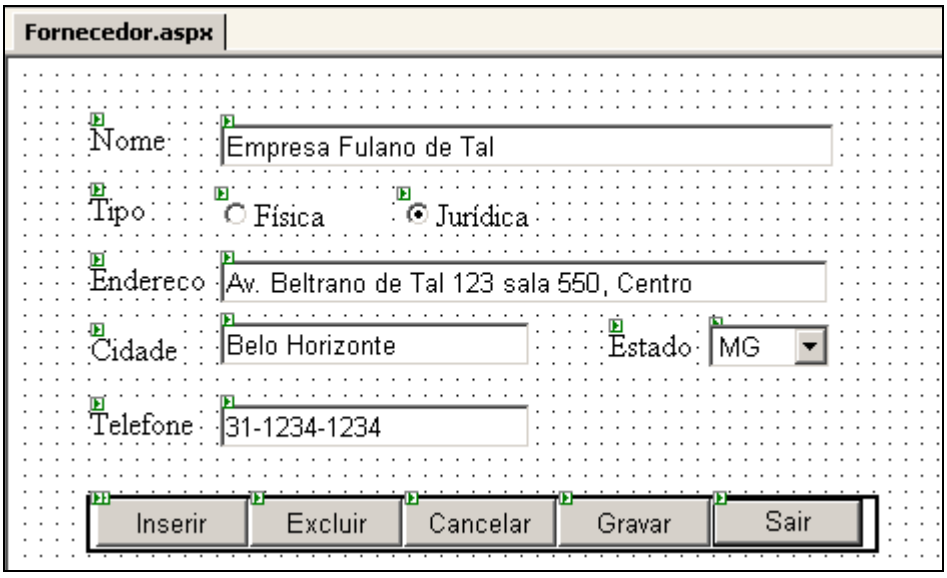


Ilustração 12: Exemplo de um WebForm de Cadastro de Fornecedores

- Inserir nos eventos apropriados as chamadas para os WebServices correspondentes. Os objetos gráficos utilizados no WebForm (chamados de WebContros) possuem propriedades e eventos semelhantes os mesmos objetos utilizados do Windows Forms. No nosso exemplo, iremos inserir exatamente o mesmo código para chamada do método inserir que foi utilizado na solução Smart Client.

```
private void cmdGravar_Click(object sender, System.EventArgs e)
{
        ...
        wsDados.Fornecedor oFornecedor = new wsDados.Fornecedor();
        seInseriu = oFornecedor.inserir ( nome, tipo, endereco, cidade,
                                          estado, telefone);
}
```

- Criação de um Web Form para o cálculo de multa

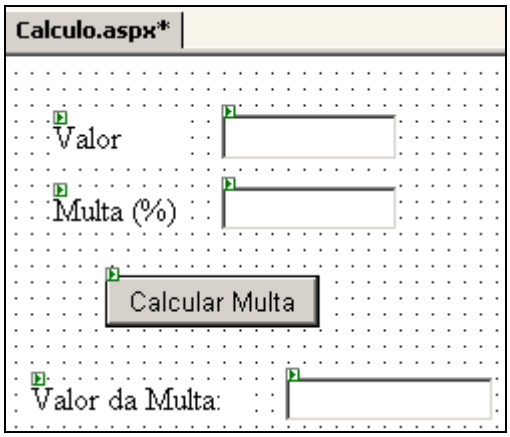


Ilustração 13:WebForm para Calculo de Multa

- Inserir nos eventos apropriados as chamadas para os WebServices correspondentes. No caso do calculo de multa iremos inserir o mesmo código da versão Windows Form no evento Click do botão Calcular Multa.

```
private void CmdCalcular_Click(object sender, System.EventArgs e)
{
    wsNegocios.Calculo oCalculo = new wsNegocios.Calculo();
                    txtResultado.Text = oCalculo.calculaMulta(
                    Convert.ToDecimal(txtValor.Text),
                     Convert.ToDecimal(txtPercentualMulta.Text)).ToString();
}
```

- Definir o formulário Principal.aspx como sendo a página inicial e compilar o projeto depois de inserido todo o código necessário para a implementação.
- Executar a seguinte url no Windows Explorer: http://localhost/Teste/Teste_WebUI/Principal.aspx

Com este procedimento o ASP.NET instalado no servidor irá verificar qual Browser (e versão) efetuou uma requisição para a pagina e irá gerar o HTML especifico para o navegador de forma automática. A parte lógica dos formulários (código dos eventos, etc) serão acessados via biblioteca .dll (no nosso exemplo: WebUI.dll que foi criado depois de compilarmos o projeto) . Como a parte lógica do formulário esta toda encapsulada em um arquivo compilado o tempo de resposta de uma aplicação ASP.NET e muito menor que o tempo de resposta de uma aplicação ASP tradicional onde o código é interpretado.
O controle de gerenciamento de estados apesar de ser mais sofisticado no ASP.NET ainda exige round-trips para seu funcionamento. As chamadas feitas aos WebServices são idênticas às realizadas pelo Windows Form e os controles básicos são praticamente idênticos na usabilidade e no controle de eventos.

6. Avaliação e comparação dos resultados

6.1. Interface Gráfica

Smarts Clients baseados no Windows Form possuem uma interface rica para interação com o usuário além de permitir a utilização de gráficos sofisticados. Já o Thin Client baseado no Web Form possui interface gráfica limitada e dependente do Browser (apesar de gerar HTML específico para um deles os recursos gráficos disponíveis variam entre os navegadores), reduzindo a usabilidade da aplicação.

6.2. Performance

Como o Smart Client pode ser executado totalmente na máquina Cliente ele terá um rápido tempo de resposta para aplicações que exigem um alto grau de interatividade o que pode ser comprovado na nossa aplicação exemplo. Já o Thin Client precisa efetuar requisições para o servidor na execução da maioria das suas operações afetando significativamente o tempo de resposta.

6.3. Capacidade de execução off-line

Aplicativo baseado em Browser é inteiramente dependente da conexão a com Internet. Se esta conexão não apresentar uma velocidade adequada ou se tornar indisponível a aplicação não poderá ser utilizada. Já aplicações baseadas em Smart Clients podem prover mecanismos para trabalho off-line uma vez que os assemblies da camada de apresentação foram baixados localmente.

6.4. Uso eficiente de recursos das máquinas clientes

Smart Clients podem ter acesso completo aos recursos da máquina local ao contrário das aplicações Thin Client que possuem acesso bastante limitado os recursos locais.

6.5. Distribuição

Smart Clients através do No-Touch Deployment são baixados quando necessário, instalados e executados diretamente da máquina cliente sem a necessidade de registrar componentes para o seu funcionamento na forma de Windows Form. É possível também fragmentar esta camada de apresentação em vários arquivos e maximizar a velocidade de execução inicial.

Já o Thin Client deve executar o .Net Framework no servidor para que o mesmo envie a camada de apresentação na forma de páginas HTML para o cliente.

6.6. Plataforma

Smart Clients necessitam que o .Net Framework esteja sendo executado na máquina cliente ao contrário do Thin Client que precisa do Framework apenas no servidor. Para utilizar a camada de apresentação no Thin Client, o cliente precisa apenas de um browser HTTP independente de plataforma.

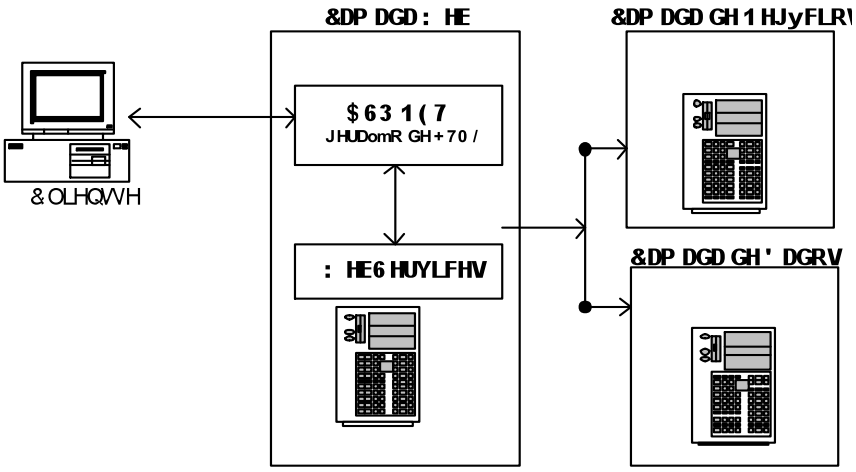
6.7. Modelo de Programação

Apesar de possuírem modelos de programação diferentes: Smart Clients (baseados em Windows) e Thin Client (baseado em Browser), a plataforma .NET torna a programação de ambos os modelos bastante parecida. Como a mesma linguagem de programação pode ser utilizada em ambos os modelos, o tempo para desenvolvimento de um modelo baseado em um já existente torna-se relativamente rápido.

6.8. Custos

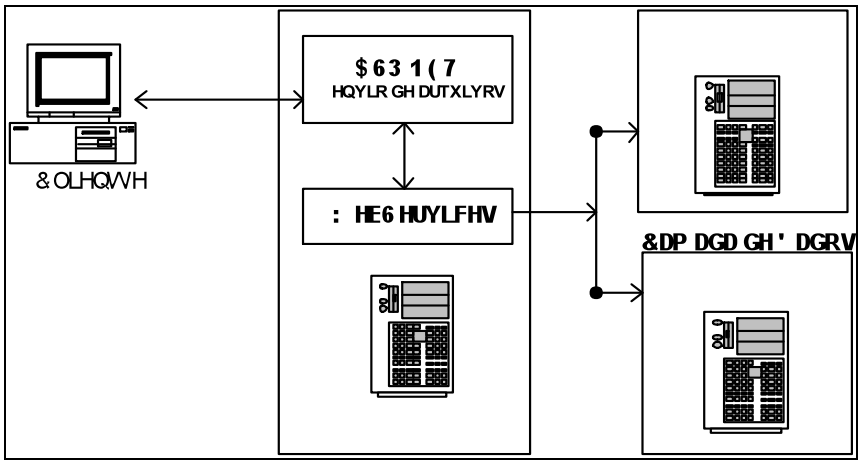
Para o Thin Client gerar a camada de apresentação é necessário que o ASP.NET reconheça o Browser que enviou uma requisição, gere o HTML específico para este navegador e o envie para o cliente. Para acessar os serviços de negócios e dados (no nosso exemplo) a camada de apresentação já deverá estar funcional para interagir com o WebService.

Se considerarmos a utilização de um servidor específico para cada camada,teremos: um servidor para hospedar a camada de apresentação (ASP.NET) juntamente com o WebService, outro servidor para hospedar a camada de negócios e um último para hospedar a camada de dados.



Para fins de comparação iremos supor que nossa aplicação utilize 60% do processamento do Servidor Web para gerar HTML e 40% para utilização da camada de WebServices.

Se utilizarmos Smart Clients com a mesma arquitetura proposta teremos no servidor Web os arquivos referentes à camada de apresentação que serão baixados. Se estes arquivos estiverem fragmentados (um arquivo por formulário) teremos um tamanho compatível com os mesmos formulários na versão Web (no nosso exemplo, o formulário de fornecedores foi apenas 2% maior que a versão HTML gerada). Considerando o pior caso para a utilização do Thin Client (a aplicação está sendo baixada pela primeira vez pelos usuários) teremos um processamento muito parecido com a solução Thin Client, pois os arquivos da camada de apresentação serão enviados para a máquina cliente. Entretanto, depois que todos arquivos estiverem no GAC, o processamento do servidor Web será praticamente para o acesso ao WebService diminuindo drasticamente o número de requisições. Se levarmos em conta os custos deste servidor para atender um grande número de requisições, a solução Smart Client sem dúvida oferece uma relação cont mais atraente pelo fato de não precisar utilizar os recursos da camada Web para gerar a camada de apresentação. Dependendo da relação entre a quantidade de requisições Web / WebServices o custo total da solução poderá variar significativamente entre as soluções propostas.


&OLHQWH
$63 1(7
HQYLR GH DUTXLYRV
: HE6 HUYLFHV
&DP DGD GH ' DGRV

7. Conclusão

As recomendações para utilização das soluções propostas variam de acordo com o fim da aplicação a ser utilizada.

Aplicações Thin Cliente baseadas em WebForms deverão ser utilizadas sempre que for necessário atingir um público que não necessite de muitos recursos para visualização da camada de apresentação (neste caso, apenas um navegador), que necessite de interação com diversas plataformas e que a limitação de recursos do sistema do usuário não seja um empecilho para a aplicação. Páginas de comércio eletrônico e aplicações que necessariamente utilizam os recursos limitados do navegador para garantir a segurança de uma aplicação são exemplos que devem adotar o Thin Client como solução para desenvolvimento e distribuição.

Já as aplicações Smart Client baseadas em Windows Form deverão ser utilizadas quando a usabilidade da aplicação exigir interfaces ricas e gráficos sofisticados ou quando a aplicação possuir como requisito a capacidade de utilizar recursos locais do cliente bem como um tempo de resposta relativamente pequeno a uma requisição. Este tipo de camada é recomendado sempre que possível para aplicações que possam contar com máquinas clientes utilizando o .NET Framework. Entre as aplicações recomendadas para este tipo de camada podemos citar: jogos, aplicações gráficas e de entrada de dados, sistemas de ponto de vendas, etc. Apesar do custo inicial de distribuição para a instalação da plataforma .NET nos clientes, o custo da execução da aplicação no servidor diminui, visto que o número de requisições por segundo ao WebService aumentará de acordo com a relação entre a quantidade de requisições Web / WebServices, ou seja, quanto mais interativa for a aplicação o custo do servidor da camada de apresentação será proporcionalmente mais baixo que sua versão Thin Client.

8. Bibliografia

Watson, Karli , “Beginning C# Programando”, Markron Books, São Paulo, 2002

Plant, David S. , “Iniciando Microsoft .NET”, Markron Books, São Paulo, 2002

Sells , Chiris, “State Sanity Using Smart Clients”, MSDN Library, Outubro 2002

Howard, Rob, “Web Services with ASP.NET”, MSDN Library, Outubro 2002

Howard, Rob , “ASP.NET Session State”, http://www.msdn.microsoft.com.br

Sheriff , Paul D. “Introduction to ASP.NET and Web Forms”, MSDN Library, Outubro 2002

.NET Framework Developer’s Guide , “Introduction to Web Forms”, http://www.msdn.microsoft.com.br

Smart Cliente Application Model and the .NET Framework, http://www.msdn.microsoft.com.br

No-Touch Deployment in the .NET Framework, MSDN Library, Outubro 2002

Security and Versioning Models in the Windows Forms Engine Help You Create and Deploy Smart Clients, MSDN Library, Outubro 2002

Issues in .NET Application Architecture, http://www.msdn.microsoft.com.br

.Net Framework Help, Visual Studio.NET