Criar aplicativos de negócios com o Microsoft Silverlight

José Antônio da Cunha – outubro 2013

Resumo: Silverlight é rico de recursos diretamente aplicáveis para os desenvolvedores que criam aplicativos de negócios. A ferramenta foi aprimorada pelo Microsoft Visual Studio ® e Microsoft Expression Blend ®. Este artigo descreve uma arquitura para construir aplicações de negócios.

Introdução

Silverlight tem evoluído rapidamente desde seu lançamento original em 2007. A ênfase inicial para o Silverlight foi em fornecer ricas experiências de mídia, permitindo aos designers e desenvolvedores para adicionar ricos graus de interatividade, mídias e animação para seus sites. A partir da versão Silverlight 4, foram acrescentados, mais recursos, fazendo do Silverlight uma opção atraente de tecnologia para a construção de aplicações de negócios.

O que é aplicações de negócio?

Então o que queremos dizer com a "aplicação de negócio"? Um aplicativo de negócios tem uma finalidade distinta ou objetivo, e, geralmente, ajuda a empresa a economizar dinheiro ou tempo, ou talvez os ajuda a melhorar a satisfação do cliente. Os exemplos mais comuns podem incluir aplicações de CRM, e-commerce e aplicações de solicitação de compra, e sistemas de gestão de RH dos funcionários. Este estilo de aplicação tem características comuns.

Aplicações de negócios:

- **São dirigidos a dados**: Os dados são muito importantes, o aplicativo precisa ser capaz de exibir os dados de maneiras diferentes. O aplicativo também precisa ser capaz de tomar a entrada do usuário através de campos de formulário, validá-lo, e depois salvar e atualizá-lo adequadamente.
- **Exigir validação de dados robusto**. Para garantir a melhor experiência do usuário, a validação dos dados de entrada deve ocorrer no cliente com retorno imediato para o usuário. Para segurança adicional, a validação também devem ser fornecidos no servidor.

- **Necessidade de ser seguro**. Os dados apresentados pela aplicação normalmente precisa variar dependendo de quem está acessando o aplicativo. Autenticação é essencial. Além disso, a funcionalidade oferecida pela aplicação muitas vezes precisa variar dependendo da função do usuário.
- **Deve apresentar ao usuário uma experiência profissional, eficiente e agradável**. Uma relação profissional que se encaixa nos requisitos de branding das empresas e permite aos usuários fazer seus trabalhos de forma eficiente e eficaz é geralmente mais importante do que extravagantes efeitos visuais. Um aplicativo de negócios também precisa visualizar dados em muitas maneiras diferentes, utilizando uma variedade de tipos de controle diferentes, que vão desde caixas de texto simples, caixas de combinação e grades de dados a controles mais sofisticados que fornecem gráficos e relatórios.
- **Muitas vezes, precisam ter acesso aos recursos locais**. Muitos aplicativos de negócios também precisam ter acesso aos recursos locais, incluindo o sistema de arquivos local, outras aplicações client-side de produtividade empresarial, como o Microsoft Office Excel ® ou o Microsoft Office Outlook ®, impressoras, microfones, webcams, a área de transferência, e assim por diante.
- **Dependem da integração com serviços externos e internos**. Muitos aplicativos de negócios precisam acessar serviços e aplicações existentes para fornecer dados críticos essenciais para o funcionamento do aplicativo de negócios. Estes podem ser acessados localmente através da rede interna ou remotamente através da Internet. O acesso é normalmente através de chamadas REST ou SOAP.

Por que escolher o Silverlight para aplicativos de negócios?

Os usuários de aplicativos de negócios esperam interfaces ricas e funcionais, e com bom desempenho, tais como aqueles tipicamente fornecidas pelos aplicativos desktop os quais estão acostumados. Com as mais recente melhorias para o Silverlight, os recursos ricas para interface de usuário e as ferramentas associadas fornecido pelo Visual Studio e Expression Studio, agora você pode construir rapidamente este tipo de aplicação e usufruir dos benefícios de prestação Web e implantação.

Os seguintes recursos, em particular, fazem do Silverlight uma tecnologia viável e atraente para a construção de aplicações de negócios:

- **WCF RIA Services**. WCF Rich Internet Application (RIA) Serviços e Visual Studio 2010+ oferece uma solução elegante para lidar com a transmissão de dados entre as camadas de sua aplicação, validação de dados e controle de alterações. Ao fazer isso, eles fornecem um modelo unificado para o lado do cliente e do servidor para o desenvolvimento, tornando um trabalho tradicionalmente difícil, para o desenvolvedor, muito mais fácil.
- **Ricos controles de dados**. Silverlight oferece uma rica biblioteca de mais de 60 controles complementados por open source e pacotes de controle de fornecedores. As novas funcionalidades ricas de controles de vinculação de dados, como o DataGrid, ContentControl, DatePicker e gráficos fornecidos pelo Silverlight Toolkit, tornar muito mais fácil, exibir dados de uma maneira atraente. Novos controles como o controle **RichTextArea**, torna muito mais fácil capturar a entrada de texto formatado. Trabalhar com grandes quantidades de dados e

manipulação de dados e paginação também é muito mais fácil com o controle DataPager, que em grande parte automatiza esse trabalho.

- **Página de navegação**. Silverlight fornece uma página de quadro de navegação, tornando muito mais fácil para apoiar a navegação entre páginas. Você pode controlar como e quando a página de navegação deve interagir com o jornal histórico do navegador, e também fornece mapeamento URI, permitindo-lhe usar URIs significativo para navegar entre as diferentes páginas em seu aplicativo.
- **Out-of-browser suporte**. O recurso fora do navegador, permite que os usuários continuem trabalhando enquanto estiver desconectado da rede. Por exemplo, você pode escrever um aplicativo que sincroniza suas alterações de volta para o servidor na próxima vez que ligar. Out-of-browser aplicativos também podem funcionar como aplicativos confiáveis com mais privilégios, fazendo tarefas que requerem o acesso a recursos locais possível. Por exemplo, você pode acessar o sistema de arquivos local, interagircom aplicativos do Office, ou ter acesso ao teclado completo em modo de tela cheia.
- **Validação de código compartilhado**. O suporte ao recurso de validação significa que você pode escrever sua lógica de validação, uma vez, e ter o código reutilizado automaticamente no cliente e no servidor. Você também pode aplicar declarativamente suporte de validação básica, tais como comprimento e verificações de intervalo.
- **Acesso a recursos locais**. Silverlight permite, que as aplicações fora-de-browser Silverlight, possam acessar os recursos locais, tais como o sistema de arquivos e área de transferência, bem como outros aplicativos locais e dispositivos que fornecem uma interface COM. Utilizando o suporte COM permite que seus aplicativos de negócios do Silverlight, possam se integrar com aplicativos do Office como o Excel ou Outlook e realizar muitas outras tarefas antes não possíveis de aplicações web.
- **Suporte de impressão**. O suporte de impressão do Silverlight permite imprimir páginas inteiras e itens de dados específicos diretamente de seus aplicativos de negócios.
- **Apoio de comando**. Silverlight agora fornece suporte para os comandos da mesma forma que o WPF faz para aplicações desktop. Comandos fornecem uma maneira de operações especificas de data-bind em seu aplicativo para controles na interface do usuário do aplicativo. Esta abordagem de comando suporta padrões como o Model-View-ViewModel (MVVM) que promovem a separação de código, manutenção e testabilidade.
- **Arraste e solte**. Aplicações Silverlight agora permite que os usuários arraste os arquivos para suas aplicações Silverlight. Por exemplo, um usuário pode arrastar e soltar um documento do Excel em uma aplicação Silverlight e ter no aplicativo, automaticamente seus dados em um controle DataGrid.
- **Apoio ao botão direito do mouse**. Agora você pode criar seus próprios menus pop-up para melhorar ainda mais a experiência do usuário e, facilitar o acesso a contextos específicos características e funções.

Assim, enquanto muitas características essenciais para aplicações de negócios já estão disponíveis, como você deve estruturar seu aplicativo Silverlight para melhor aproveitá-las? Como você deve realizar acesso a dados? Como você deve passar

dados através da rede entre UI ricas e banco de dados? Estabelecer e compreender a arquitetura é essencial.

Considerações de arquitetura

Parte da arquitetura de aplicação Silverlight tem muito em comum com os tradicionais aplicações HTML / ASP.NET / PHP, com uma diferença importante. Quando você usa o Silverlight, a lógica de apresentação está localizado apenas no cliente no navegador do usuário.

Arquitetura de aplicações HTML / ASP.NET / PHP

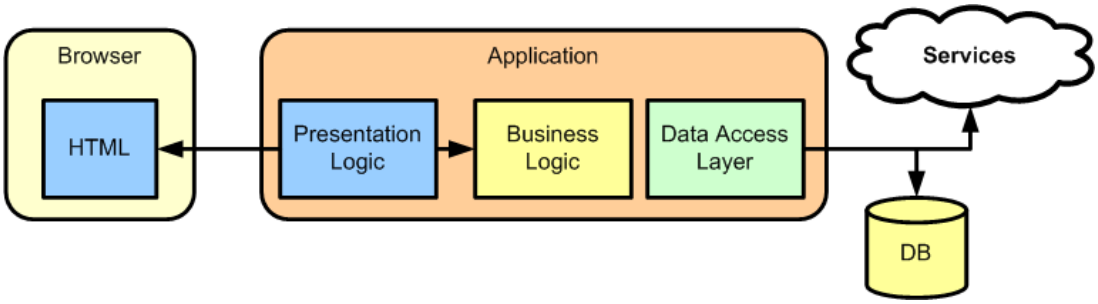

Figura 1: Arquitetura de aplicações HTML / ASP.NET / PH

Com este modelo muito comum, a aplicação é dividida em três camadas: negócios, apresentação e dados. No modelo ASP.NET / PHP convencional, a lógica de apresentação no servidor é responsável por gerar a necessária marcação HTML, que é enviado para o navegador do cliente para contruir a interface do usuário do aplicativo. Em muitos sistemas, a lógica de apresentação e de negócios, e muitas vezes a lógica de acesso a dados, é mais estreitamente associadas a Figura 1, que sugere, uma clara separação em níveis distintos, o que é desejável, e oferece muitos benefícios, incluindo flexibilidade, facilidade de manutenção e escalabilidade. Com a tecnologia AJAX, a riqueza das interfaces apresentadas por este tipo de aplicação tem aumentado significativamente, mas o Silverlight permite-lhe tirar a riqueza e funcionalidade a um outro nível.

Arquitetura Rich Internet Application

O Rich Internet Application modelo (RIA) oferecido pelo Silverlight muda um pouco, fornecendo um modelo híbrido que combina o poder do Silverlight para exibir ao usuário uma experiência muito rica (mais parecido com uma aplicação desktop tradicional) no cliente, com os três modelos convencionais Web-tier. A Figura 2 mostra a arquitectura de RIA.

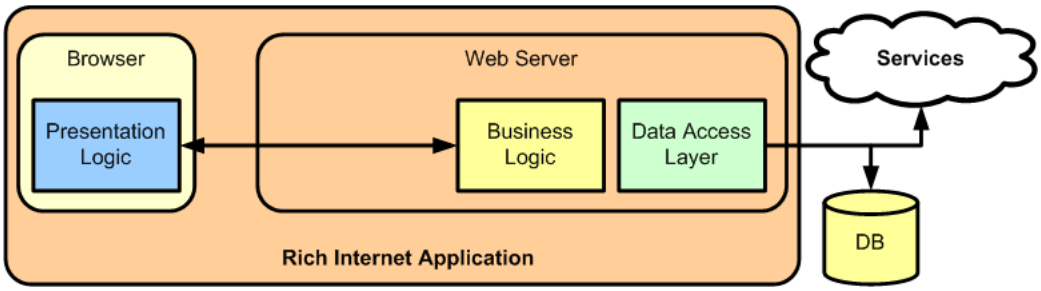

Figura 2: Silverlight / RIA Application Architecture

Observe como a camada de apresentação estar agora completamente do lado do cliente (em execução no Silverlight plug-in dentro do navegador do usuário). Neste modelo, o cliente é realmente uma extensão do servidor e você deve pensar no RIA como uma aplicação lógica única. Isto levanta uma questão fundamental: Como você deve lidar com a comunicação entre cliente e servidor, através do limite de confiança que é a rede?

WCF RIA Services

Com a introdução do WCF RIA Services (a seguir designado RIA Services) fornece uma solução para este problema, e ao fazê-lo resume muito da complexidade subjacente de se comunicar através de um limite de confiança do cliente para o servidor. O objectivo fundamental para serviços de RIA é o de proporcionar uma coerência , entre as extremidades de uma aplicação, que liga as exibições de apresentação de um lado com a base de dados do outro. Isto, juntamente com o fato de que o RIA service, abstrai o detalhe de ter que lidar com a comunicação assíncrona, simplifica a tarefa de desenvolvimento global.

Um serviço de domínio no servidor é exposto à apresentação do lado do cliente através da camada de classe de contexto de domínio (gerado automaticamente, a partir da classe de serviço de domínio, das ferramentas do Visual Studio, quando você construir seu projeto). A solução RIA de Serviços em camadas, no topo da arquitectura da aplicação RIA, é mostrado na Figura 3.

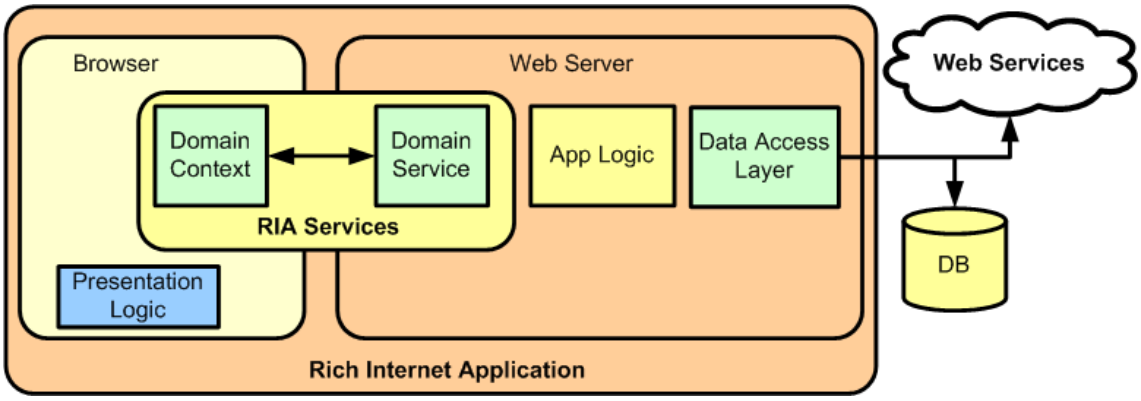

Figura 3: WCF RIA Services, the domain service and domain context classes

Padrões

Há uma série de padrões de projeto que você deve considerar na construção de seus aplicativos de negócios usando Silverlight, a fim de melhor estrutura e modularizar suas aplicações e torná-los inerentemente mais testável, extensível, escalável e sustentável. Enquanto eles não são obrigatórios para a construção de aplicações Silverlight, padrões como Model-View-ViewModel (MVVM) e injeção de dependência são altamente recomendados.

Model-View-ViewModel

O padrão MVVM é fundamentalmente sobre a separação de interesses dentro da sua base de código. Por exemplo, MVVM permite que você separe seus pontos de vista (a apresentação da interface do aplicativo para o usuário) de dados e lógica. Os projetistas podem projetar a View no Expression Blend, enquanto o ViewModel podem ser desenvolvidos no Visual Studio. Um dos principais benefícios de MVVM é que ele melhora a capacidade de teste de seu código, movendo a lógica da aplicação para fora da view e para fora do ViewModel. Assim, quanto mais você se move a partir da View para o ViewModel, mais fácil fica o teste. Isso também ajuda quando se trata de depurar seu código.

Suas Views, views models, e models são todas separadas e são fracamente acoplados. Eles se comunicam um com o outro, mas apenas através de interfaces definidas.

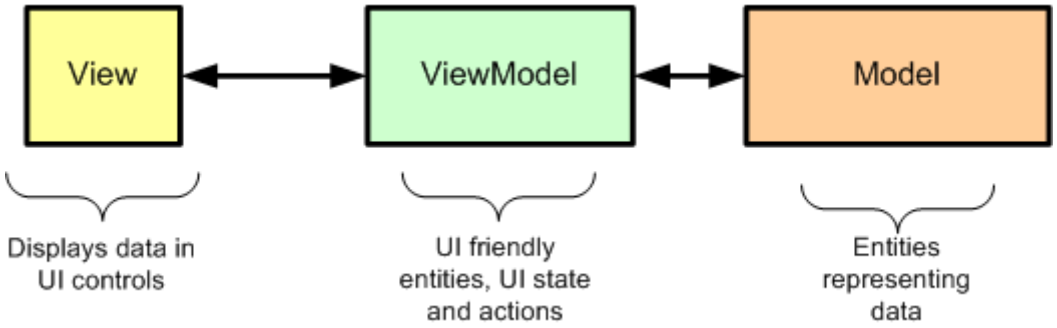

Figura 4: Model-View-ViewModel Pattern

Então, qual é a diferença entre a View, o ViewModel e o model?

- **View**. A View contém os controles e comportamentos que formam a interface do usuário. Ele também pode conter animações, aspectos de navegação, temas e outros recursos interativos utilizados para montar a interface visual. Toda a visão deve ser definida declarativamente (em XAML), sempre que possível, a fim de beneficiar (por exemplo) o Expression Blend e apoio designer do Visual Studio. A View também contém as ligações declarativa (novamente no XAML) que identificam os itens de dados que serão apresentados para o usuário. O ponto de ligações para os nomes das propriedades do item de dados, mas não têm consciência de onde essas propriedades são nem de onde eles vêm. Você deve manter o código behind da View o mínimo posível.

- **ViewModel**. O ViewModel representa os dados para a View, e lida com a comunicação entre visão e modelo através de ligações (**Biding**). Ele não contém elementos de interface do usuário, mas fornece os pontos de entrada para as hierarquias de dados. Também pode moldar e combinar os dados de maneiras diferentes, de modo que a View possa se comunicar com ele.
- **INotifyPropertyChanged** e **INotifyCollectionChanged** como formas de notificar a View quando algum aspecto dos dados mudou. O ViewModel também ouve mudanças a partir da tela através de suas ligações.
- **Model**. O modelo é simplesmente os dados, ignorando tudo o mais dentro de sua aplicação. Por exemplo, em um aplicativo de RH, seria conter suas entidades de empregados. O modelo não precisa saber de onde os dados vem. Poderia vir de um serviço WCF, o WCF RIA Services, um serviço RESTful (como o Twitter, um Feed RSS, ou Amazon), um banco de dados SQL Server e assim por diante. O modelo pode também conter validação.

Dependency Injection

Injeção de dependência é outro padrão central para a construção de aplicações modulares e dissociadas e, é usado dentro da abordagem MVVM para fornecer conexões de baixo acoplamento entre o Model, View e ViewModel. Sem injeção de dependência, essas conexões precisaria ser hard-coded. Injeção de dependência permite inserir ou "injetar" as classes a usar. Por exemplo, você pode usar diferentes classes quando o teste de unidade em vez de em tempo de execução.

A chave para a injeção de dependência é programar usando uma interface em vez de um tipo concreto ou classe. Isto permite-lhe construir implementações diferentes, tais como implementações de simulação para fins de teste. Considere uma interface usada para expor um modelo para um aplicativo, como no código a seguir.

```
public interface ITimeSheetSystem
{
  ObservableCollection<Employee> GetEmployees();
}
```

Ao programar contra o ITimeSheetSystem interface, diferentes implementações podem ser prestadas, incluindo uma para simular testes de unidade.

```
public class MockTimeSheetSystem : ITimeSheetSystem
{
  public ObservableCollection<Employee> GetEmployees()
  {
    return new ObservableCollection<Employee> { new Employee{ID=1,Name=”Sam
Smith”,
                                          Age=25, Department=”Sales”}};
  }
}
```

PRISM (Composite Application Guidance for WPF and Silverlight)

PRISM é um conjunto de ferramentas que você pode usar para criar aplicativos que são testáveis, modular e extensível. Ele permite que você construir interfaces de usuário

compostas para WPF e Silverlight aplicativos, e permite-lhe apoiar o desenvolvimento de aplicações em várias equipes. PRISM separa os elementos da interface do usuário específicas dos elementos de apresentação e lógica de negócios da aplicação, e mostra como você pode testá-los de forma independente. Ele também mostra como você pode usar o padrão MVVM para compartilhar informações não-UI código e componentes entre Silverlight e WPF aplicativos. PRISM inclui também outros padrões de apoio ao desenvolvimento de aplicações modulares, extensíveis.

PRISM fornece os seguintes componentes:

- **Shell.** Este contém todas as views que posteriormente serão carregados. É um recipiente para as regiões de interface do usuário diferentes que você precisa para exibir. Você pode pensar nisso como a sua tela principal para o usuário, onde todos os outros controles e elementos são apresentados.
- **Regions**. O shell fornece regiões, ou espaços reservados nomeados, em que você pode colocar as views. A interface do utilizador pode consistir em um ou muitos regiões. Pense em regiões como os lugares onde os elementos que o usuário irá interagir com são apresentados.
- **Moduls**. Módulos permitem que você construa sua aplicação como um conjunto discreto de blocos modulares independentes um do outro. Eles fornecem uma maneira de você particionar sua solução. Cada módulo é uma entidade independente, que não fazem referência direta à outros módulos. Se necessário, você pode então usar comandos ou eventos para se comunicar entre eles.
  Os módulos são realmente importantes para o desenvolvimento de aplicativos de negócios, porque muitas vezes há fases de um projeto onde diferentes indivíduos ou equipes estão trabalhando em partes da aplicação geral. Estes indivíduos ou equipes podem trabalhar nos módulos sem saber muito sobre o que os outros estão fazendo. PRISM ajuda a integrar estes módulos, mesmo que sejam libertados em tempos diferentes. Ele também permite que seu aplicativo para carregar módulos sob demanda.
- **Container Dependency Injection**. PRISM usa a Unidade container de injeção de dependência. O recipiente é o objeto que coordena a criação de objetos e os objetos que eles dependem. Injeção de dependência é um ingrediente fundamental para a construção de testáveis, estruturas modulares e dissociado e aplicações. Para mais informações sobre a Unidade, ver http://msdn.microsoft.com/en-us/library/dd203101.aspx.
- **Bootstrapper**. O bootstrapper inicia o aplicativo e carrega o shell (o recipiente principal interface do usuário). Ele também registra e carrega todos os módulos necessários e fornece uma maneira para que você execute tarefas de inicialização que você possa precisar para realizar.
- **Events Aggregation**. Fornece uma maneira menos rígida para publicar e subscrever eventos. Isso permite que diferentes componentes de uma aplicação para comunicar uns com os outros sem precisar de uma referência direta.

Para aplicações Silverlight negócios mais complexas, proporcionando uma série de funcionalidades e enfrentando o desafio de apresentar muitas views diferentes de dados para o usuário, usando PRISM oferece algumas vantagens significativas,

impondo disciplina e estrutura para o seu software e fornecendo um conjunto de ferramentas muito úteis e técnicas. PRISM fornece um número de itens distintos, mas você pode optar por usar cada uma delas de forma seletiva. Você pode usar apenas os pedaços de PRISM você quer (como comandante) sem o uso de todos eles.

Para mais informações sobre o PRISM, consulte http://msdn.microsoft.com/en-us/library/dd458809.aspx.

Conclusão

Aplicações de negócios têm um conjunto de requisitos funcionais e habilidades que têm, tradicionalmente, exigiu um cliente rico.

Silverlight juntamente com seus frameworks e ferramentas de suporte, oferece uma alternativa viável e que reúne o melhor dos modelos de desktop e aplicações Web. A ferramenta aprimorada pelo Visual Studio 2010+ e Expression Blend 3+ diminui muito a dor de cabeça de criar aplicativos de negócios e diminui em muito o tempo de desenvolvimento da aplicação. Isso ajuda você a construir aplicações de negócios a um custo menor. Silverlight é mais produtivo como uma plataforma para aplicações CRUD do que qualquer outro, porque você não precisa mais gastar tempo em um trabalho tedioso para empacotar dados entre objetos de negócios e controles.

Recursos adicionais

Para mais informações sobre como usar o Silverlight para a construção de linha de negócios de aplicativos, consulte:

Silverlight

- www.microsoft.com/silverlight/developer
- http://www.silverlight.net
- "Data Validation with Silverlight 3 and the DataForm" at http://msdn.microsoft.com/en-us/magazine/ee335695.aspx
- Silverlight TV http://channel9.msdn.com/shows/SilverlightTV/

Visual Studio

- www.microsoft.com/visualstudio

Expression Blend and Sketchflow

- www.microsoft.com/expression

RIA Services

- http://silverlight.net/getstarted/riaservices/

PRISM

- http://msdn.microsoft.com/en-us/library/dd458809.aspx
- http://www.codeplex.com/CompositeWPF

MEF

- http://www.codeplex.com/mef

- Building Composable Apps in .NET 4 with the Managed Extensibility Framework" at http://msdn.microsoft.com/en-us/magazine/ee291628.aspx

Printing

http://silverlight.net/learn/videos/all/printing-api-basics/