Composite Applications with **Prism** in **WPF** - Part 1

Snesh Prajapati, 22 Nov 2015

In this series of articles, we will learn about **Prism** Framework by creating a demo Composite **WPF** application. We will learn how to use various components of **Prism** to build a Composite application by mocking practical organization level scenario.

Download Source Code - Till Part 1 - 1.5 MB

Introduction

In this series of articles, we will learn about **Prism** Framework by creating a demo Composite **WPF** application. We will learn how to use various components of **Prism** to build a Composite application by mocking practical organization level scenario. To make it easy to read and understand this article series is consist of three parts. Outline of each part is mentioned below. In this article we will cover first part of the series.

Prerequisites: Before going through the series of this article, I assume you would be having basic understanding of **WPF**, specially concepts given in below points, if not please have a look at the links provided here:

- Model-View-ViewModel (MVVM)
- ICommand Interface
- DelegateCommand
- Dependency Injection with Unity

Outlines

Part 1

- Overview of Final Demo
- Architecture of Final Demo

- What is **Prism**
- What is Shell
- Creating Shell and Defining Regions

Part 2

- Creating Common Infrastructure
- Creating Modules

    1. Navigation Module
    2. Employee Module
    3. Software Module
    4. Hardware Module
    5. Request Module

- Loading Modules
- Issues in Demo Application

Part 3

- What is Event Aggregator
- Use of Event Aggregator
- What is IActiveAware Interface
- Use of IActiveAware

Overview of Final Demo

In this article series, we will be creating a demo application called "ITHelpDeskDemoApp". We are mocking an application which would be used to raise request for IT (Information Technology) related needs for employees in an organization. So the purpose of this demo application would be to provide ITHelpDesk service to the employees within the organization. By using this demo application employee can raise request to get softwares and hardwares. By providing employee id, one can get employee detail and the request associated with that employee.

We will start implementation of demo application, from first part of this article series and then keep enhancing this application till third part of this article. "ITHelpDeskDemoApp" application will have five modules called Navigation, Employee, Software, Hardware and Request module. Brief description of each module is given below:

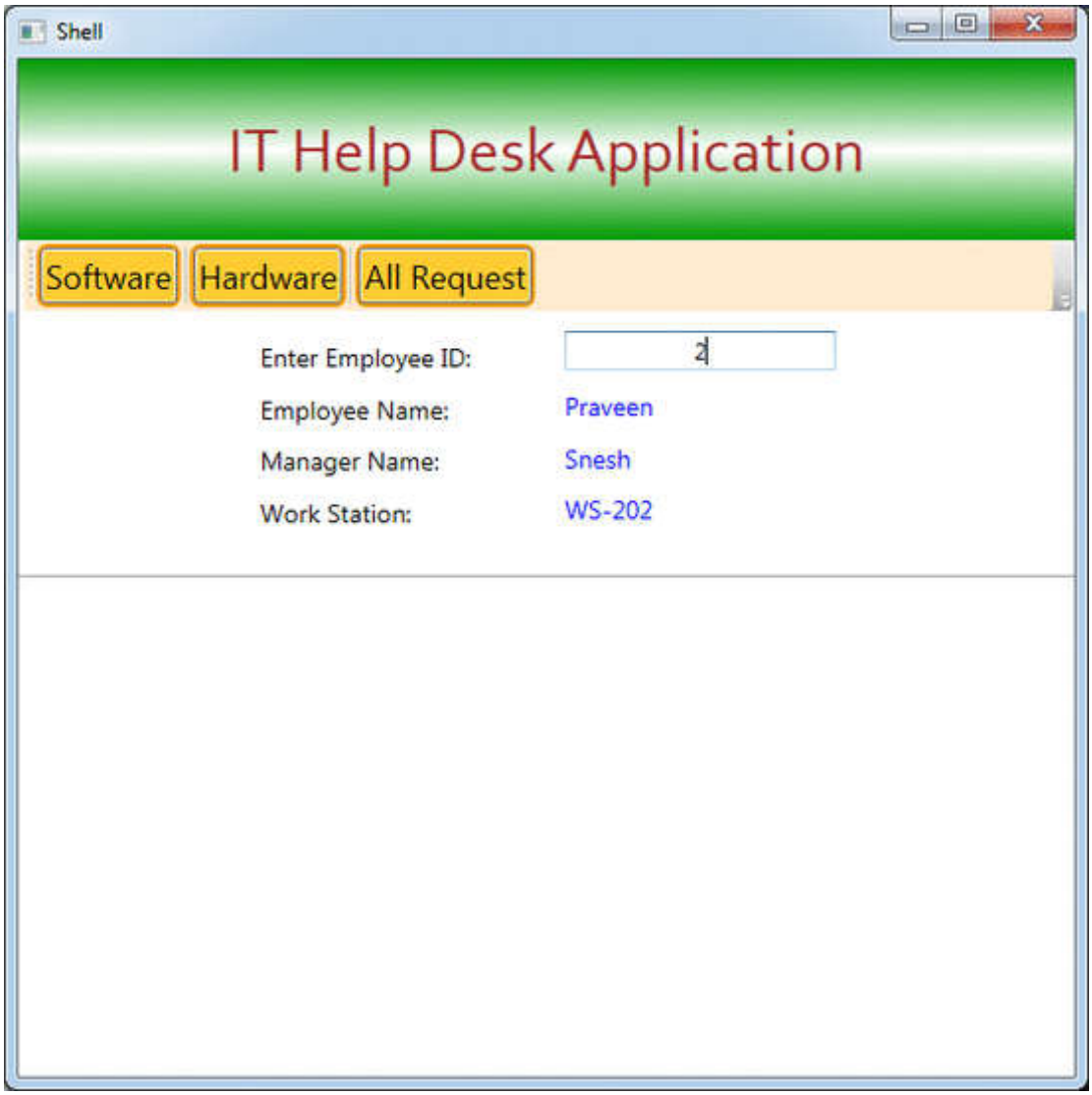
Navigation Module: Navigation Module creates Navigation/Menu Bar which contains three buttons called as Software, Hardware and "All Request". Navigation Module is loaded with the Shell/Statup window.

Employee Module: Employee Module displays information of employee when the employee enters his/her Employee ID. This Module is loaded with the Shell/Startup window.

Software Module: Software Module is used to raise a request for a particular type of software. This module is loaded **on demand** while user clicks on Software button in Navigation bar.

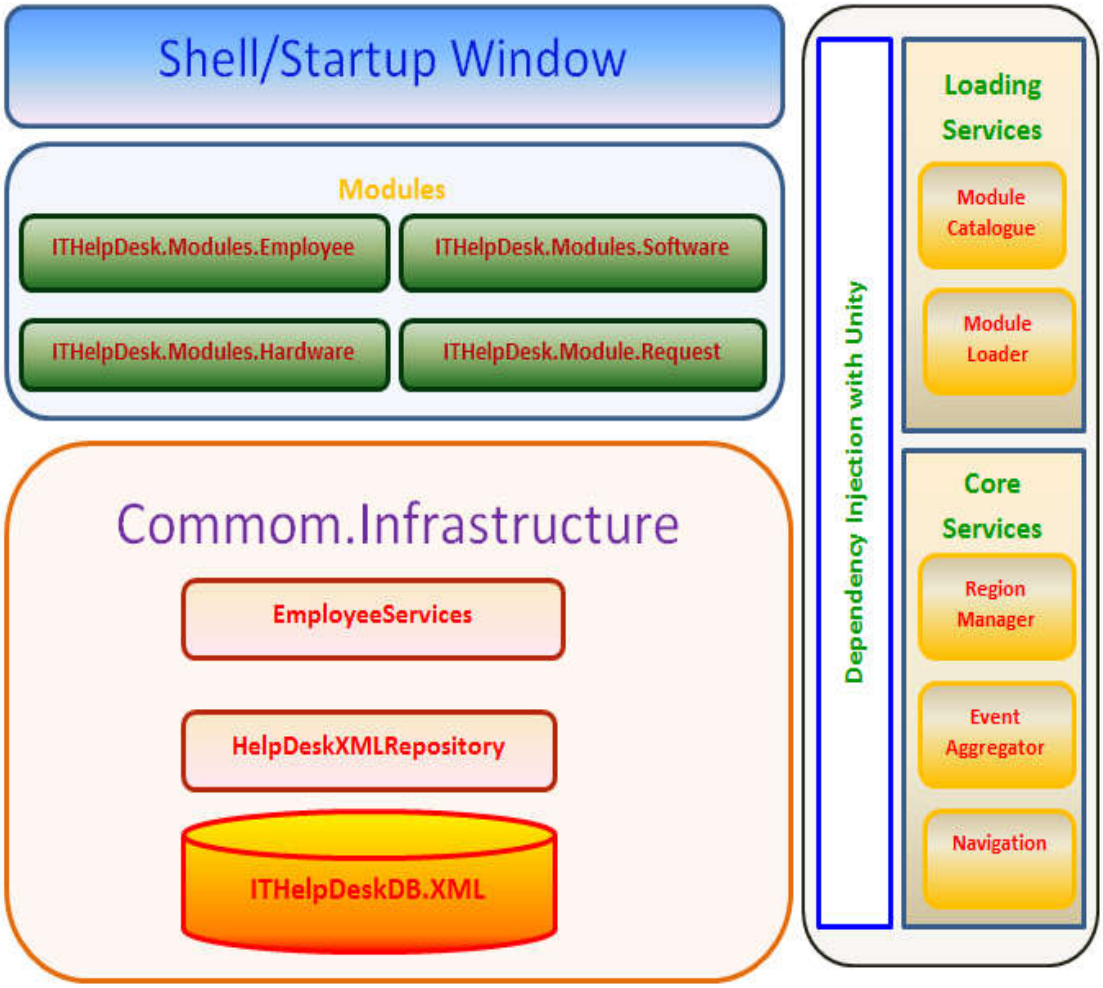
Hardware Module: Hardware Module is used to raise a request for a particular type of hardware. This module is loaded **on demand** while user clicks on Hardware button in Navigation bar.

All Request Module: All Request Module is used to see the detail of request raised by Employee/User. This module is loaded **on demand** while user clicks on "All Request" button Navigation bar.



Architecture of Final Demo

Below image represents the architecture of our demo project:



As we can see the architecture of demo is a typical **Prism** app. At top level we have Shell and Modules, which is our UI part and those consist of Views. We are having related ViewModels in those projects too.

At bottom we have infrastructure module which have EmployeeService. It is supposed to provide all employees related services to the app. Often ViewModels call it to get employee related data and save employee requests. The benefit of having a service is that it abstracts the mechanism of information sources or its manipulation. So if datasource changes (it may be a RDBMS or XML files or web service) or we need to use multiple sources, only service code will change. For the simplicity we are using XML file as a database here and we have a HelpDeskXMLRepository to interact with XML file.

At right side of the architecture diagram, we have a vertical bar which represents Cross Cutting Concerns for entire application. We are using Unity as a dependency injection framework. With the help of unity, we are using many services provided by **Prism** such as Module Loader, Event Aggregator etc. We will explain about those in detail as we will use those while creating the demo.

What is **Prism**

Prism is a framework for building **"Composite Applications"** using **WPF**/Silverlight. **Prism** framework provides a set of libraries which helps us in building modular enterprise applications. By using those libraries, we can categories application into multiple modules, load them as per need and orchestrate them very easily. For more learning about **Prism**, please have a look at **Prism** Basic Introduction and **Prism** Page on MSDN, and few more things about **Prism**.

What is Shell

Prism framework provides many features, Shell is one of them. Shell is a standard naming for a xaml file which is used as Startup window but we can give any name to Shell.xaml file. In this file we write code which creates common layout and provide a consistent look and feel to the

application. In Shell, we define the regions where module can be loaded initially at application startup or on demand.

It is similar to MasterPage in ASP.NET (or Layout page in ASP.NET MVC) which has logo/branding features, navigation bars, header, footer etc.

With **Prism** based application, as a first step, Shell need to be created. For that there are some standard steps need to be taken. Let us follow those steps given in next section.

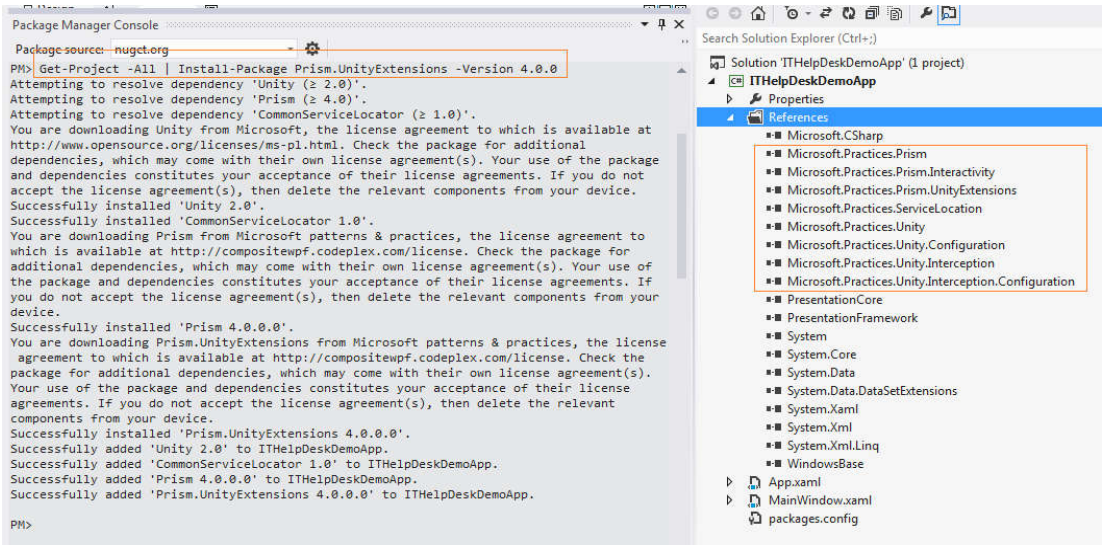
Creating Shell and Defining Regions

At first step of **Prism** based application, we need to create Shell/Startup window then define regions in the Shell. That's what we are going to achieve with this section.

Let us create a **WPF** application project called ITHelpDeskDemoApp by selecting **WPF** application template in Visual Studio.
Open "Package Manager Console" window by navigating Tools -> Library Package Manage -> Package Manager Console in Visual Studio.
Write following command in "Package Manager Console" window and press enter. It will download and add all required libraries to all projects within the solution as shown in below screenshot.

Hide Copy Code

```
Get-Project -All | Install-Package Prism.UnityExtensions -Version 4.0.0
```



Adding a region in Shell
Remove MainWindow.xaml file from the project. Add a new Window by selecting Window (**WPF**) called as Shell.xaml.

Open Shell.xaml file and add following namespace to the root element of the XAML file, as shown below. This namespace provides some classes which are required while defining the regions in Sell.

Hide Copy Code

```
xmlns:prism=http://www.codeplex.com/CompositeWPF
```

Write following code in Shell.xaml file to create common layout for the application and defines regions for modules.

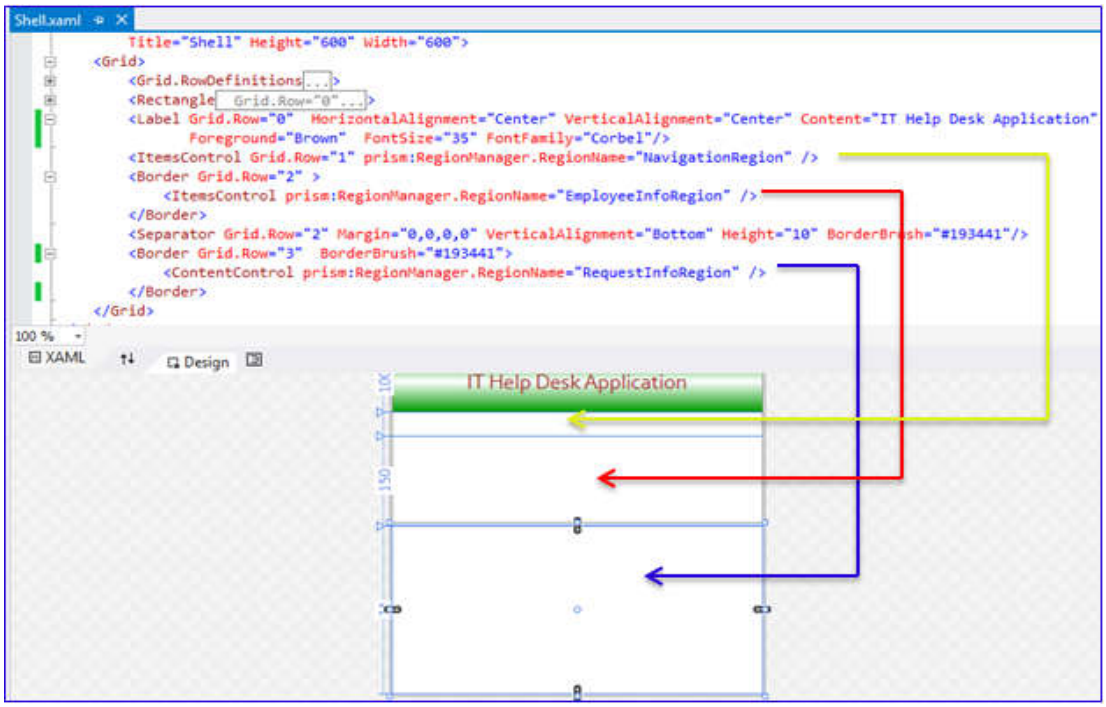
Hide Copy Code

```
<Grid>
     <Grid.RowDefinitions>
         <RowDefinition Height="100" />
         <RowDefinition Height="40"/>
         <RowDefinition Height="150" />
         <RowDefinition />
     </Grid.RowDefinitions>
     <Rectangle  Grid.Row="0" >
         <Rectangle.Fill>
```

```
            < inearGradient r sh  tartPoint="0 0" EndPoint="0 0.5"  preadMethod="Reflect">
                <Gradient top  olor=" 00  00"  ffset="0"/>
                <Gradient top  olor=" hite"  ffset="1" />
            </ inearGradient r sh>
        </Rectangle.Fill>
    </Rectangle>
    < a el Grid.Row="0"  Hori ontalAlignment=" enter" VerticalAlignment=" enter"  ontent="I  Help Desk
Application"  Foregro nd=" rown"  Font i e=" 5" FontFamily=" or el"/>
    <Items ontrol Grid.Row="1" prism:RegionManager.Region ame=" a igationRegion" />
    < order Grid.Row=" " >
        < ontent ontrol prism:RegionManager.Region ame="EmployeeInfoRegion" />
    </ order>
    < eparator Grid.Row=" " Margin="0 0 0 0" VerticalAlignment=" ottom" Height="10"  order r sh=" 1  441"/>
    < order Grid.Row=" "   order r sh=" 1  441"   >
        < ontent ontrol prism:RegionManager.Region ame="Re  estInfoRegion" />
    </ order>
</Grid>
```

In above xaml code, **ItemsControl** is used to define two regions one for NavigationModule called as "NavigationRegion" and second for EmployeeModule called as "EmployeeInfoRegion". A **ContentControl** is used to define region for Software/Hardware/"All Request" modules called as "RequestInfoRegion". "**prism**:" is alias to bring namespace for RegionManager class. In both controls (ItemsControl/ContentControl), **RegionManager.RegionName** attribute is used which is an attached property of RegionManager class. This property is responsible to create region for module with the specified name. **RegionManager** class is responsible for creating and maintaining group of regions for modules.

Pictorial description of Shell is shown below.



Add ITHelpDeskBootstrapper.cs file to the root directory of the project and write following lines of code in this file.

Hide Copy Code

```
        sing Microsoft.Practices.Prism.UnityExtensions
        sing Microsoft.Practices.Unity
        sing  ystem. indows
       namespace I HelpDeskDemoApp

           class I HelpDesk ootstrapper : Unity ootstrapper

               protected o erride Dependency  ject  reate hell

                   ret rn  ontainer.Resol e<shell>

               protected o erride  oid Initiali e hell
```

```
                    ase.Initiali e hell
                App.  rrent.Main indow =   indow this. hell
                App.  rrent.Main indow. how



</shell>
```

UnityBootstrapper

In **Prism** framework, **Bootstrapper** is an abstract class. It provides methods required for application initialization stage. But we do not use it directly as we need a Unity (dependency injection framework) too, so we will use **UnityBootstrapper** class. **UnityBootstrapper** class is an abstract class which inherits from Bootstrapper class. This way if we used UnityBootstrapper, automatically we can use all methods given by UnityBootstrapper as well as Bootstrapper class.

In above lines of code, we are inheriting ITHelpDeskBootstrapper class from UnityBootstrapper. **CreateShell** is an abstract method of Bootstrapper class which we must override. This method is used to create Shell/Startup window of the application. **InitializeShell** is a virtual method of Bootstrapper class which we need to override because it is responsible to show Shell/Statup window. Now question comes if these both methods are from Bootstrapper class then why we are inheriting ITHelpDeskBootstrapper class from UnityBootstrapper, why not from Bootstrapper directly? This is because later in this application we will need to use ConfigureContainer method which is given by UnityBootstrapper. UnityBootstrapper provide more methods to register dependencies and other functionality required to implement in **Prism** application.

Setting up Startup Window

In **WPF**, StartupUri is used to launch MainWindow.xaml when application starts. Remove the reference of MainWindow.xaml from App.xaml file which is specified as StartupUri (Uniform Resource Identifier). In this demo, we are using **Prism**, now it is responsibility of Bootstrapper class to launch Shell/Startup window.

Open App.xaml.cs file and inside the constructor of App class, create an instance of ITHelpDeskBootstrapper and call Run method.

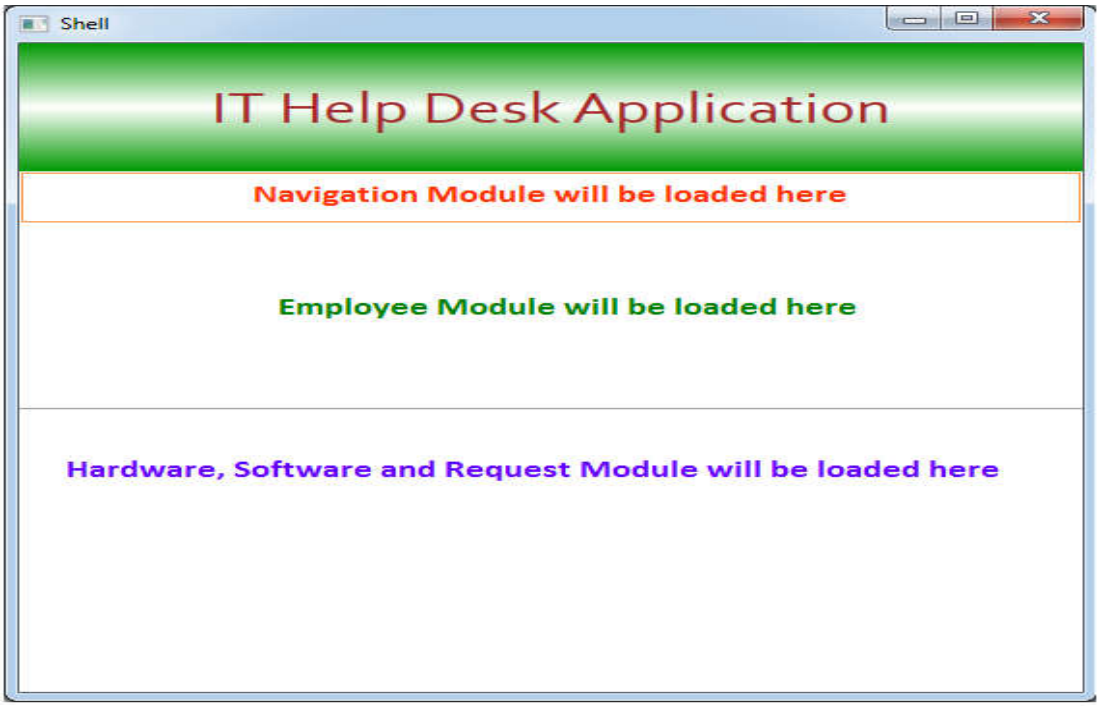
Hide Copy Code

```
p  lic partial class App : Application

    p  lic App

        I HelpDesk ootstrapper  ootstrapper = new I HelpDesk ootstrapper
         ootstrapper.R n
```

Run the application, hope you will be able load Shell.xaml file as Startup window.



Conclusion

In this first part of the article series, we learned about how to create a Shell with regions. In next part of this article series, we will continue with same demo and enhance it further. Your comments/suggestions and queries are most welcome. Thanks.

License

Share

TWITTER FACEBOOK

About the Author

Snesh Prajapati
Software Developer
India

I am a Software Developer working on Microsoft technologies. My interest is exploring and sharing the awesomeness of emerging technologies.

You may also be interested in...

Flip jQuery Plugin for Previewing Web Content

AdES: An Implementation of CAdES for Windows in C++

Comments and Discussions

You must Sign In to use this message board.

e r h mme ts

Spacing e ed Layout pe Per page pd te

First Prev Next

Exceptional explanation

Member 12288978 **14-Sep-16 1:35**

I am new to composite application framework and Prism, but your straightforward explanation on these often confusing terms (MVVM, Prism, and App flow) and architectural illustration of Prism is of exceptional quality . Thank you so much, now I can do more with this knowledge I have.

Sign In · View Thread

Re: Exceptional explanation

Snesh Prajapati 14-Sep-16 4:50

Thanks for encouraging note

Sign In · View Thread

Please add part 2

Member 12128238 **25-Nov-15 0:31**

Request to please add part and part 3

Sign In · View Thread

Re: Please add part 2

Snesh Prajapati 25-Nov-15 2:27

Sure, I will do in few days. Thanks.

Sign In · View Thread

Re: Please add part 2

Snesh Prajapati 28-Nov-15 0:01

I would like to inform you that Part 2 is available now. Thanks.

Sign In · View Thread

Re: Please add part 2

@nkit Bajpai 10-Apr-16 17:31

Hi,

URL of part 2 please ?

Thanks,
Ankit

Sign In · View Thread

Re: Please add part 2

Snesh Prajapati 10-Apr-16 18:53

Hi Ankit,

It is Composite Applications with Prism in WPF - Part 2[^]

Best Regards,
Snesh

modified 11-Apr-16 2:11am.

Sign In · View Thread

Re: Please add part 2

@nkit Bajpai 14-Apr-17 23:39

Snesh,

I am following your article.
Could you please share the link to part 3. It will be a good idea to share those links when are posting the article. This will help us a lot.

Thanks,
Ankit Bajpai

Sign In · View Thread

Re: Please add part 2

Snesh Prajapati 18-Apr-17 4:37

It's there in Outline section: "Part 2" and "Part 3" Text are having hyperlinks.

Anyway, following is the link for Part 3:

Composite Applications with Prism in WPF - Part 3

Thanks.

Sign In · View Thread

Error 3: The name "RegionManager" does not exist in the namespace "http://www.codeplex.com/CompositeWPF" Mr.PoorEnglish 22-Nov-15 11:46

Hello!

I downloaded your code-sample, and found all required references and files as you described it.
But the Shell.xaml - File does not compile on my system.
VisualStudio 2013 Community

can you help me?

Regards

Sign In · View Thread

Re: Error 3: The name "RegionManager" does not exist in the namespace "http://www.codeplex.com/CompositeWPF" Snesh Prajapati 22-Nov-15 14:53

Sure.

"RegionManager" must work if you have added following namespace in XAML file:

Hide Copy Code

```
xmlns:prism="http://www.codeplex.com/ omposite PF"
```

And "Microsoft.Practices.Prism.dll is referenced.

Can you please download it once again and sometime problems may occur if you download code from internet so first unblock before unzipping:
http://www.thewindowsclub.com/unblock-file-windows-8[^]

There is following resource mentioning similar issue and he resolved another way:
https://social.msdn.microsoft.com/Forums/expression/en-US/da89a78e-8d50-4470-aaa1-356c834b04fd/prism-interoperation?forum=blend[^] (On this page scroll at bottom to get how he fixed that issue.)

Otherwise create a new WPF app and install Prism 4 by Nuget and then try to create Shell as we created in attached demo.

Please try those. It is interesting to me as I have tested with Visual Studio 2012, 2013 and 2015 too on different machines and for me downloaded code is working fine.

Please share how the issue is fixed in your case. Thanks.

modified 22- o -1 21:2 pm.

Sign In · View Thread

Last Visit: 31-Dec-99 18:00 Last Update: 14-Sep-18 5:51 Refresh 1

General News Suggestion Question Bug Answer Joke Praise Rant Admin