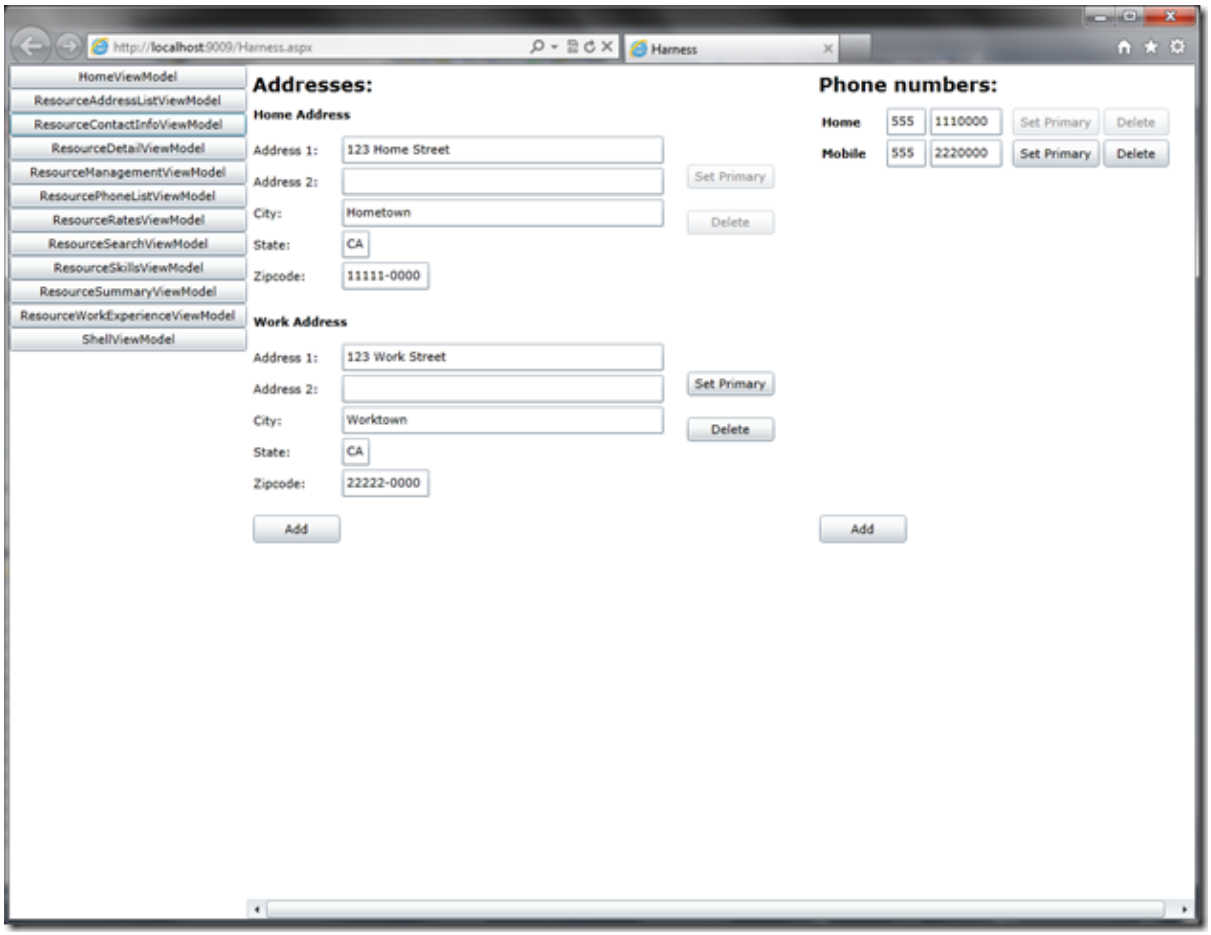
Contents

The screen harness is a simple, yet powerful productivity tool. Often developers waste valuable time by working directly in the main application. Every time a developer needs to try out a code modification, they have to build, then run the application, probably login, navigate to the screen they are working on and then see if the features work correctly. If they don't, they go back, make further code modifications and then build, run, login, navigate again. This is a costly cycle if done over and over again.

The screen harness provides a way to directly launch a ViewModel/View with one click. Very little setup is required to get going with the harness and be productive right out of the gate. How does it work? The screen harness automatically discovers the ViewModels and renders a button along the left side. When clicking on one of the buttons, the corresponding ViewModel is launched on the right side.



Bootstrapping

Bootstrapping the harness is very easy. Simply specify the HarnessViewModel as the root type in your bootstrapper. It's often convenient to have separate projects for the harness and the actual application and share the code between the two projects. This way by switching the Start Project/Start Page one can easily go back and forward between the harness and the application.

```
using Cocktail;
using DomainModel;
public class AppBootstrapper : CocktailMefBootstrapper<HarnessViewModel>
{
}
```

ViewModel discovery

ViewModel discovery in the screen harness is based on a marker interface. In order for a ViewModel to be discovered by the harness, the ViewModel must implement the IDiscoverableViewModel marker interface.

```
using Cocktail;
public class CustomerListViewModel : Screen, IDiscoverableViewModel
{
}
```

ViewModel setup

Sometimes it is necessary to perform some setup logic before the ViewModel can be launched. For example to setup certain dependencies without which the ViewModel wouldn't run properly. To perform this kind of setup logic, the ViewModel can implement the IHarnessAware interface. The interface's Setup() method will automatically be called before the ViewModel is activated.

```
using Cocktail;
public class CustomerListViewModel : Screen, IDiscoverableViewModel, IHarnessAware
{
   public void Setup()
    {
      // Perform necessary setup logic
   }
}
```

Video