Dependency Injection in MVC 6 Controller

Author: David Huang

Date: 25, Feb., 2015

1, Introduction

We know we can use Defaultcontrollerfactory and IControllerfctory in MVC5 to generate a custom controller for dependency injection. In MVC 6, IActionResultHelper once was used to achieve this. However, this interface now is obsoleted last month. I need to find out the way to achieve the same goal in VS 2015 CTP15. Without inheriting from base Controller class in MVC 6, controller class still can work as a middleware to return new content result or new json result to client. MVC 6 brings base Controller class back to allow View to be return as actionresult as it work in MVC 5. So the data in action result can be displayed in Razor view. However, if we inject dependencies into controller default constructor, razor will return an error to tell us the service is not activated before we can do something to fix this problem. After struggling, I finally can find out the way to fix this problem as this documentation shown here.

2, Get Started

Create a new controller called POCOController in my existing MVC 6 project as below

```
public class POCOController :Controller
{
     public IActionResult Users()
    {
        return View();
    }
    public IActionResult Index()
    {
        return null;

    }
```

MVC 6 controller still can inherit its methods and properties from Controller base class and can return View as MVC 5 did. I created a Users() action here and expected the user data in this action can be pushed into users.cshtml razor view. The data in this action should be injected from data service provider. Therefore, We can add a new folder called Services into project and add a new class called UserProvider.cs that implement its interface called IUserProvider.cs in the same folder as codes shown below

```
public interface IUserProvider
{
 object GetUser(string Id);

}
```

```
public class UserProvider : IUserProvider
    {

        List<User> _user = new List<User>()
        {
         new User {   Name = "David",
            Address = "23 Water Road St Leonards" },
         new User {   Name = "Jeff",
            Address = "23 Water Road St Leonards" },

        }; //todo get from dbcontext

        public object GetUser(string name)
        {
            return _user.Find(x => x.Name == name);
        }


    }
```

Now we have a data service provider we can consume it in controller. If we do not use DI, we can simply initiate a new UserProvider object in POCOController class. However, based on SOLID design pattern, We need to consider DI consistently. DI is the first citizen in Asp.Net VNext in VS2015. Therefore, if we want to consume data inside controller, we need to inject those data in. Another error will come if we simply inject this data provider interface into controller. The error will tell us we cannot activate this data service in controller. Why we can have such error occurred. This is because DefaultControllerFactory in MVC 6 is different from the one in MVC 5. MVC 6 uses a container IServiceProvider here to receive the data service dependency. So before we can infuse data service with controller, we need to inject this IServiceProvider into controller so DefaultControllerFactory can activate it before it can receive data service dependency injection. Now we can update this POCOController class as below

```
public class POCOController :Controller
{
 protected readonly IServiceProvider _serviceProvider;
 public POCOController(IServiceProvider serviceProvider)
 {
     _serviceProvider = serviceProvider;
 }
  public IActionResult users()
 {
      return View();
 }
 public IActionResult Index()
 {
     return null;

 }

}
```

This is the template we can use for data service injection in controller. We simply inject the interface of service provider template in MVC 6 here. Now the next step we need to do is to implement data service here.

ServiceProvider class in MVC 6 uses GetService() method to emit the method from data service provider. It only can take the type as parameter. AS we know, based on SOLID design pattern, it is not a good practice to initiate a new type inside controller. We need to inject this type here. How can we do this?

We use startup.cs root class that acts as the same action as the bootstrap.cs class in WPF PRISM design pattern. MVC6 introduces Fung DI container inside ConfigureServices() method in startup class. So we now can inject our UserProvider class in startup.cs file as below

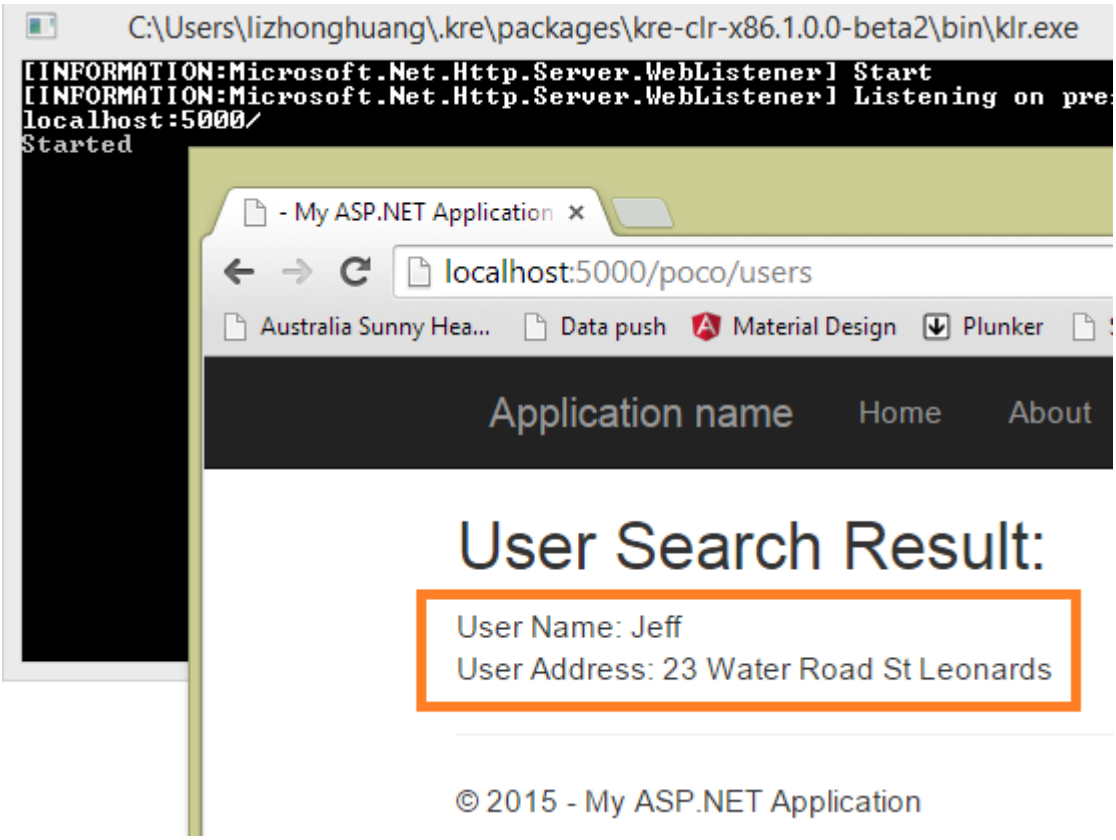
```
public void ConfigureServices(IServiceCollection services)
 {
       ……..
     services.AddMvc();
     services.AddSingleton<UserProvider>();

 }
```

Now whole application can see this UserProvider class. We now go back to POCOController class to update the code as below

```
  public class POCOController :Controller
{
 protected readonly IServiceProvider _serviceProvider;
 public POCOController(IServiceProvider serviceProvider)
 {
     _serviceProvider = serviceProvider;
 }
  public IActionResult users()
 {
     var userService = _serviceProvider.GetService(typeof(UserProvider));
     ViewData.Model = ((UserProvider)userService).GetUser("Jeff");

      return View();
 }
 public IActionResult Index()
 {
     return null;

 }

}
```

ServiceProvider GetService() emits object from UserProvider type. We now can simply call the method in this data service provider object to fetch data out service. Thoss data then can be displayed in Razor view as example below

```
<h2>User Search Result: </h2>

<div>User Name: @Model.Name</div>

<div>User Address: @Model.Address</div>
```



3, SUMMARY

So now we can inject serviceprovider template into controller to make default controller as a custom controller that allows us to inject data from outside. Serviceprovider emits the service object that can be consumed in controller. This is really a nice architecture in MVC we expected for a long time.