



AGENDA

- Controller Factories
- Controller Action Invokers
- Filters
- Model Binders
- Highway OnRamp MVC

TIM RAYBURN
PRINCIPAL CONSULTANT
MICROSOFT MVP
AUTHOR
AND FRAMEWORK HACKER

CONTROLLER FACTORIES
WHICH OBJECT DO I CREATE?



ICONTROLLERFACTORY



```

 4.0.1 - Windows - Help
File Edit View Insert Tools Options Help
WindowsControllerFactory.cs [1] WebApplication1
Assembly System.Web.Mvc.dll, v5.0.0.0
using System;
using System.Web.Routing;
using System.Web.SessionState;

namespace System.Web.Mvc
{
    public interface IControllerFactory
    {
        IController CreateController(RequestContext requestContext, string controllerName);
        SessionStateBehavior GetControllerSessionBehavior(RequestContext requestContext, string controllerName);
        void ReleaseController(IController controller);
    }
}

```

```
Assembly System.Web.Mvc.dll, v5.0.0.0

using System;
using System.Web.Routing;
using System.Web.SessionState;

namespace System.Web.Mvc
{
    public interface IControllerFactory
    {
        IController CreateController(RequestContext requestContext, string controllerName);
        SessionStateBehavior GetControllerSessionBehavior(RequestContext requestContext, string controllerName);
        void ReleaseController(IController controller);
    }
}
```

```
Assembly System.Web.Mvc.dll, v5.0.0.0

using Castle.MicroKernel;
using System.Collections.Generic;
using System.Diagnostics;

namespace Highway.DemoWebApp.App_Architecture.Services.Core
{
    // Developed by ericstefan koumar at http://docs.castleproject.org/Windsor.WindsorTutorialPartTwo.PluggingWindsorInAspNet.aspx
    public class WindsorControllerFactory : DefaultControllerFactory
    {
        private readonly IKernel kernel;

        public WindsorControllerFactory(IKernel kernel)
        {
            this.kernel = kernel;
        }

        public override void ReleaseController(IController controller)
        {
            kernel.ReleaseComponent(controller);
        }

        [DebuggerStepThrough]
        protected override IController GetControllerInstance(RequestContext requestContext, Type controllerType)
        {
            if (controllerType == null)
            {
                throw new ArgumentNullException("controllerType");
            }
            return (IController)kernel.Resolve(controllerType);
        }
    }
}
```

```
Assembly System.Web.Mvc.dll, v5.0.0.0

using System;
using System.Web.Routing;
using System.Web.SessionState;

namespace System.Web.Mvc
{
    public interface IControllerFactory
    {
        IController CreateController(RequestContext requestContext, string controllerName);
        SessionStateBehavior GetControllerSessionBehavior(RequestContext requestContext, string controllerName);
        void ReleaseController(IController controller);
    }
}
```

```
Assembly System.Web.Mvc.dll, v5.0.0.0

using Castle.MicroKernel;
using System.Collections.Generic;
using System.Diagnostics;

namespace Highway.DemoWebApp.App_Architecture.Services.Core
{
    // Developed by ericstefan koumar at http://docs.castleproject.org/Windsor.WindsorTutorialPartTwo.PluggingWindsorInAspNet.aspx
    public class WindsorControllerFactory : DefaultControllerFactory
    {
        private readonly IKernel kernel;

        public WindsorControllerFactory(IKernel kernel)
        {
            this.kernel = kernel;
        }

        public override void ReleaseController(IController controller)
        {
            kernel.ReleaseComponent(controller);
        }

        [DebuggerStepThrough]
        protected override IController GetControllerInstance(RequestContext requestContext, Type controllerType)
        {
            if (controllerType == null)
            {
                throw new ArgumentNullException("controllerType");
            }
            return (IController)kernel.Resolve(controllerType);
        }
    }
}
```

```
Assembly System.Web.Mvc.dll, v5.0.0.0

using System;
using System.Web.Routing;
using System.Web.SessionState;

namespace System.Web.Mvc
{
    public interface IControllerFactory
    {
        IController CreateController(RequestContext requestContext, string controllerName);
        SessionStateBehavior GetControllerSessionBehavior(RequestContext requestContext, string controllerName);
        void ReleaseController(IController controller);
    }
}
```

```
Assembly System.Web.Mvc.dll, v5.0.0.0

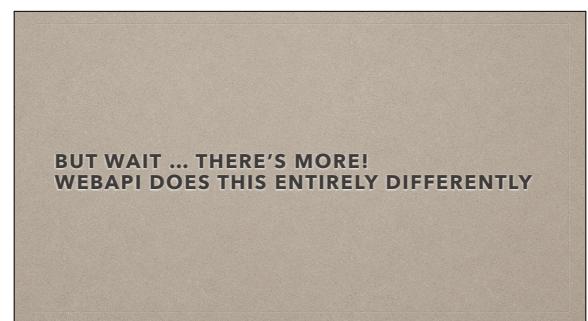
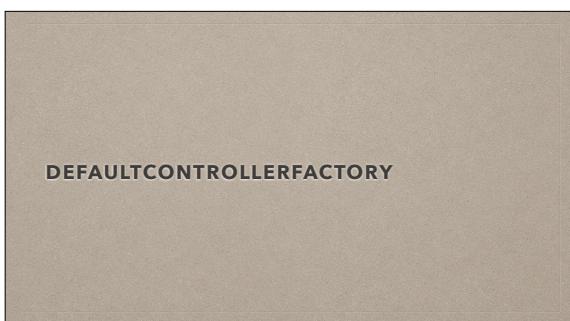
using Castle.MicroKernel;
using System.Collections.Generic;
using System.Diagnostics;

namespace Highway.DemoWebApp.App_Architecture.Services.Core
{
    // Developed by ericstefan koumar at http://docs.castleproject.org/Windsor.WindsorTutorialPartTwo.PluggingWindsorInAspNet.aspx
    public class WindsorControllerFactory : DefaultControllerFactory
    {
        private readonly IKernel kernel;

        public WindsorControllerFactory(IKernel kernel)
        {
            this.kernel = kernel;
        }

        public override void ReleaseController(IController controller)
        {
            kernel.ReleaseComponent(controller);
        }

        [DebuggerStepThrough]
        protected override IController GetControllerInstance(RequestContext requestContext, Type controllerType)
        {
            if (controllerType == null)
            {
                throw new ArgumentNullException("controllerType");
            }
            return (IController)kernel.Resolve(controllerType);
        }
    }
}
```



HTTPCONTROLLERACTIVATOR

```
public class WindsorHttpControllerActivator : IHttpControllerActivator
{
    private readonly IWindsorContainer container;

    public WindsorHttpControllerActivator(IWindsorContainer container)
    {
        this.container = container;
    }

    public IHttpController Create(
        HttpRequestMessage request,
        HttpControllerDescriptor controllerDescriptor,
        Type controllerType)
    {
        var controller =
            (IHttpController)this.container.Resolve(controllerType);

        request.RegisterForDispose(
            new Release(
                () => this.container.Release(controller)));

        return controller;
    }
}
```

A screenshot of the Visual Studio code editor showing the implementation of the `IHttpControllerActivator` interface. The code defines a class `WindsorHttpControllerActivator` that implements the `IHttpControllerActivator` interface. It uses the `IWindsorContainer` to resolve the controller type and registers it for disposal using a `Release` action.

```
using System;
using System.Net.Http;
using System.Web.Http.Controllers;
using System.Web.Http.Dispatcher;
public class WindsorHttpControllerActivator : IHttpControllerActivator
{
    public IHttpController Create(HttpRequestMessage request, HttpControllerDescriptor controllerDescriptor, Type controllerType)
    {
    }
}
```

```
public class WindsorHttpControllerActivator : IHttpControllerActivator
{
    private readonly IWindsorContainer container;

    public WindsorHttpControllerActivator(IWindsorContainer container)
    {
        this.container = container;
    }

    public IHttpController Create(
        HttpRequestMessage request,
        HttpControllerDescriptor controllerDescriptor,
        Type controllerType)
    {
        var controller =
            (IHttpController)this.container.Resolve(controllerType);

        request.RegisterForDispose(
            new Release(
                () => this.container.Release(controller)));

        return controller;
    }
}
```

A screenshot of the Visual Studio code editor showing the same implementation as above. A question mark icon is placed next to the `Release` lambda expression in the `RegisterForDispose` call, likely indicating a point of interest or a question about its functionality.

```
using System;
using System.Net.Http;
using System.Web.Http.Controllers;
using System.Web.Http.Dispatcher;
public class WindsorHttpControllerActivator : IHttpControllerActivator
{
    public IHttpController Create(HttpRequestMessage request, HttpControllerDescriptor controllerDescriptor, Type controllerType)
    {
    }
}
```

```
public class WindsorHttpControllerActivator : IHttpControllerActivator
{
    private readonly IWindsorContainer container;

    public WindsorHttpControllerActivator(IWindsorContainer container)
    {
        this.container = container;
    }

    public IHttpController Create(
        HttpRequestMessage request,
        HttpControllerDescriptor controllerDescriptor,
        Type controllerType)
    {
        var controller =
            (IHttpController)this.container.Resolve(controllerType);

        request.RegisterForDispose(
            new Release(
                () => this.container.Release(controller)));

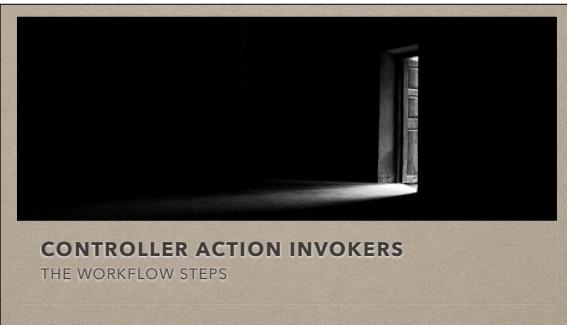
        return controller;
    }
}
```

A WINDSOR HTTPACTIVATOR IMPLEMENTATION

```
*private class Release : IDisposable
{
    private readonly Action release;

    public Release(Action release)
    {
        this.release = release;
    }

    public void Dispose()
    {
        this.release();
    }
}
```



```

+ModelBinderDictionary Binders { get; set; }

    *ActionResult CreateActionResult
    *ActionDescriptor FindAction
    *ControllerDescriptor GetControllerDescriptor
        FilterInfo GetFilters
            *object GetParameterValue
    IDictionary<string, object> GetParameterValues
        *bool InvokeAction
            *ActionResult InvokeActionMethod
    ActionExecutedContext InvokeActionMethodWithFilters
        *void InvokeActionResult
    ResultExecutedContext InvokeActionResultWithFilters
    AuthenticationContext InvokeAuthenticationFilters
    AuthenticationChallengeContext InvokeAuthenticationFiltersChallenge
        *AuthorizationContext InvokeAuthorizationFilters
            *ExceptionContext InvokeExceptionFilters

```

