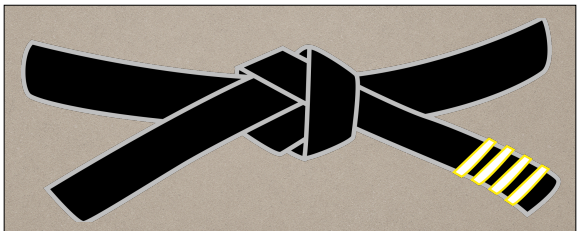


BLACK BELT MVC

EXTENDING THE FRAMEWORK

AGENDA

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

TIM RAYBURN

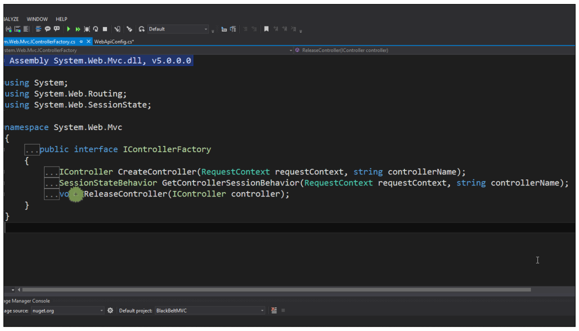
PRINCIPAL CONSULTANT
MICROSOFT MVP
AUTHOR
AND FRAMEWORK HACKER

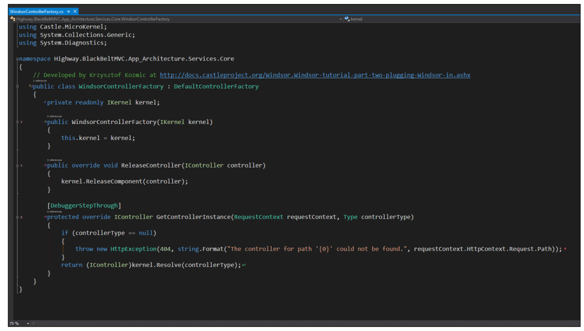
CONTROLLER FACTORIES

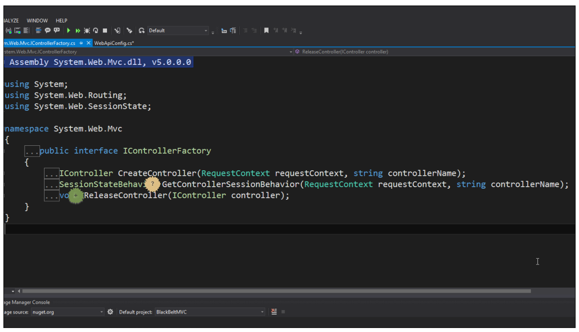
WHICH OBJECT DO I CREATE?

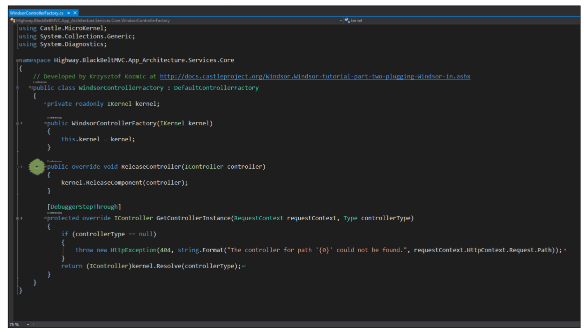
ICONTROLLERFACTORY

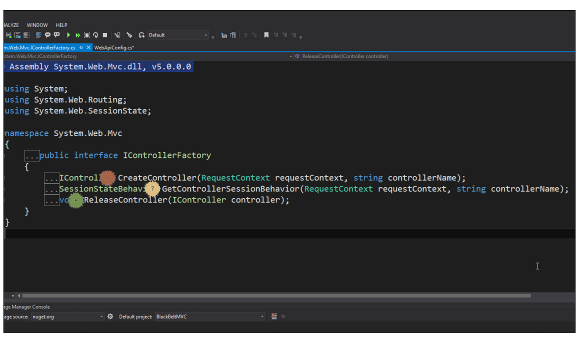
```
Assembly System.Web.Mvc.dll, v5.0.0.0

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);
    }
}
```

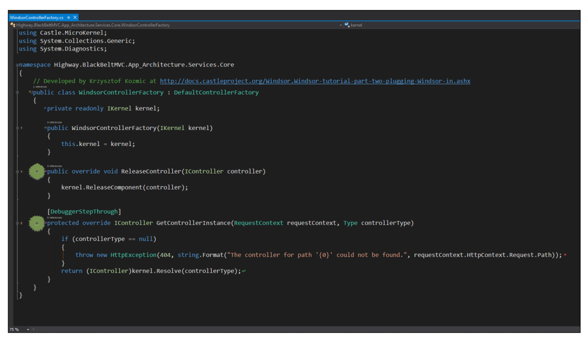










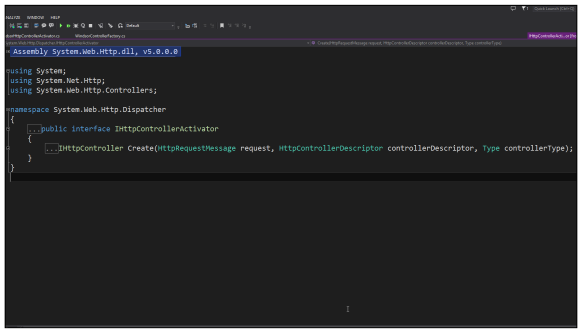


DEFAULTCONTROLLERFACTORY

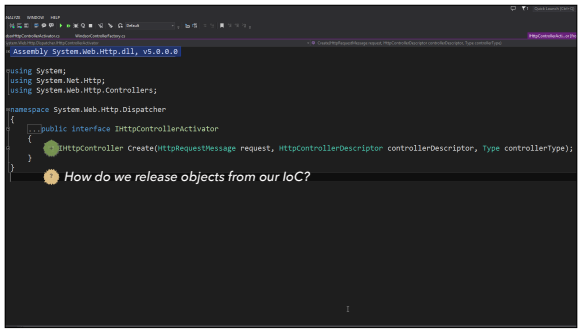
BUT WAIT … THERE'S MORE!
WEBAPI DOES THIS ENTIRELY DIFFERENTLY

HTTPCONTROLLERACTIVATOR

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

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




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

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




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

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

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

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

CONTROLLER ACTION INVOKERS

THE WORKFLOW STEPS

MODEL BINDERS

```
        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
```

HIGHWAY ONRAMP MVC

BECAUSE PLUMBING SHOULDN'T BE THIS HARD

FILTERS FOR CROSS-CUTTING CONCERNS

TYPES OF FILTERS

- IAuthenticationFilter
  - OnAuthentication
  - OnAuthenticationChallenge
- IAuthorizationFilter
  - OnAuthorization
- IActionFilter
  - OnActionExecuting
  - OnActionExecuted
- IResultFilter
  - OnResultExecuting
  - OnResultExecuted
- IExceptionFilter
  - OnException
- IOverrideFilter
  - FiltersToOverride