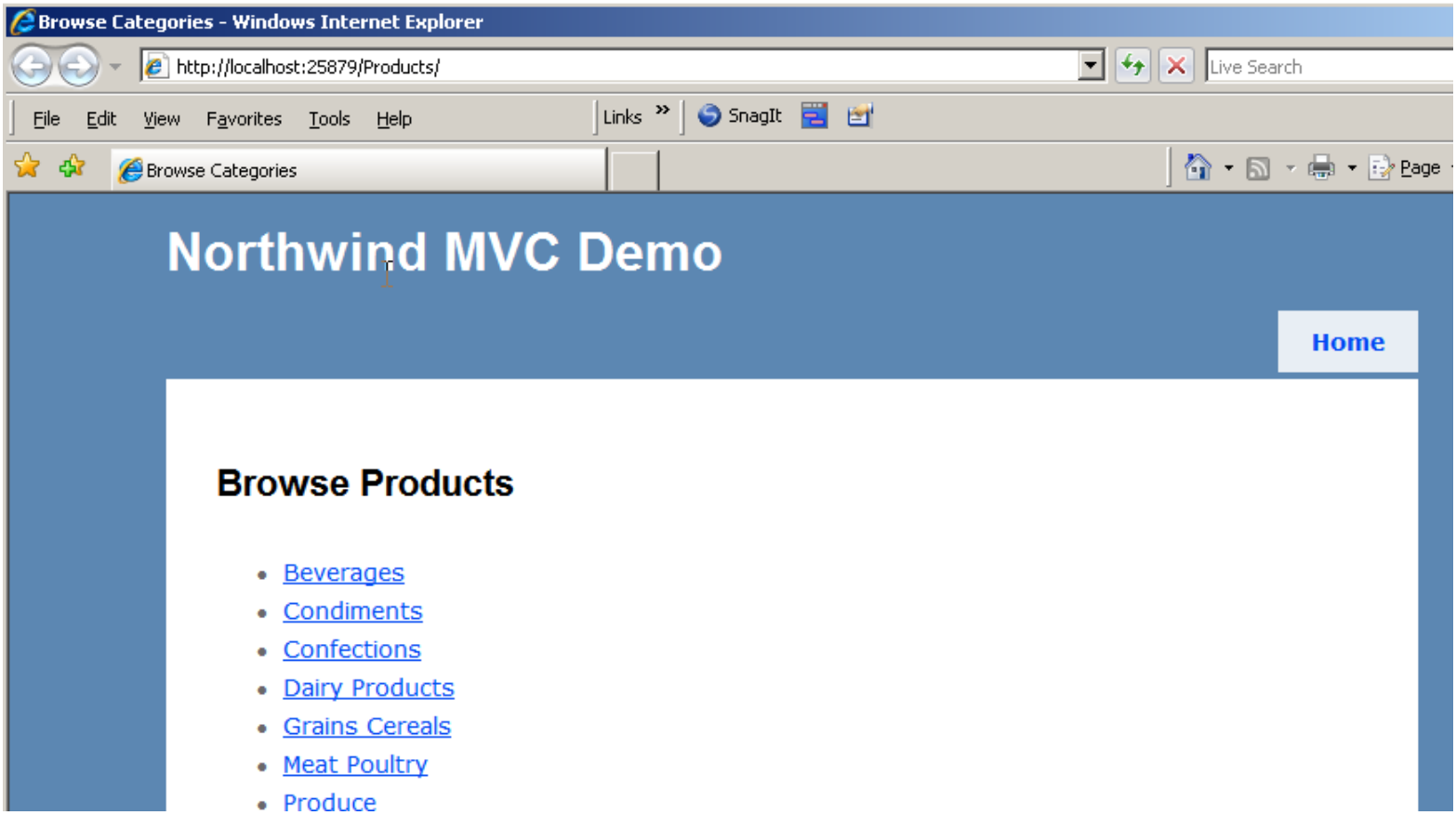
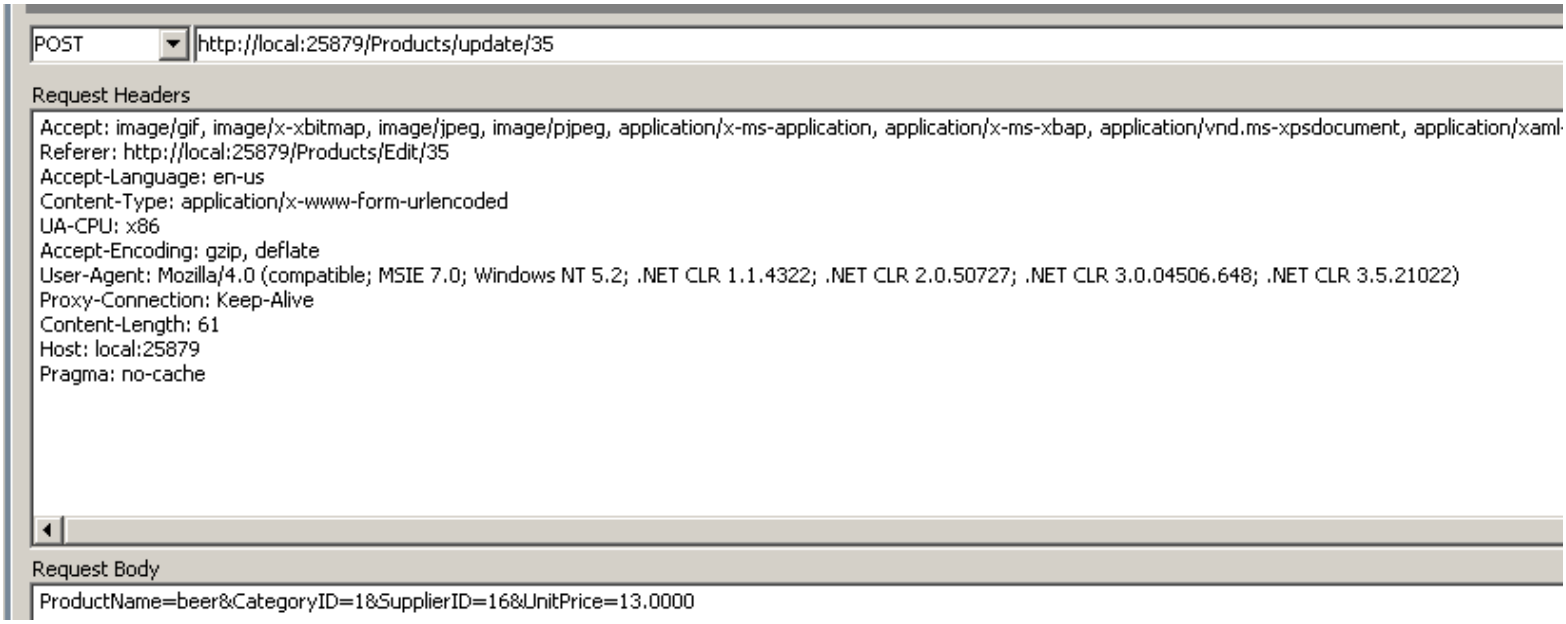
ASP.NET MVC – XSS and AutoBind vulns in MVC Example

Downloaded http://haacked.com/archive/2008/05/23/updated-northwind-demo.aspx unziped and successful run it ☺ (nice one click example)



AUTO BIND Vulnerability

The HTTP of Submit of the Edit looks like this:

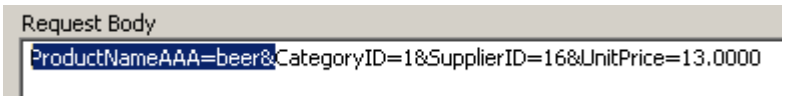


and these filed names map into variables names:

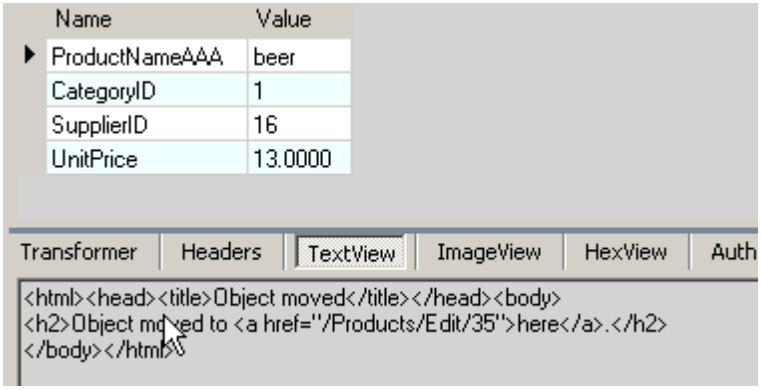
For example ProductName

```
        [Column(Storage="_ProductName", DbType="NVarChar(40) NOT NULL", CanBeNull=false)]
        public string ProductName
        {
            get
            {
                return this._ProductName;
            }
            set
            {
                if ((this._ProductName != value))
                {
                    this.OnProductNameChanging(value);
                    this.SendPropertyChanging();
                    this._ProductName = value;
                    this.SendPropertyChanged("ProductName");
                    this.OnProductNameChanged();
                }
            }
        }
```

Let's see what happens if we try:

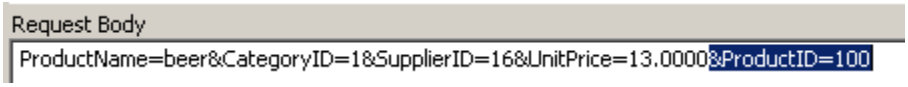


we got a redirect

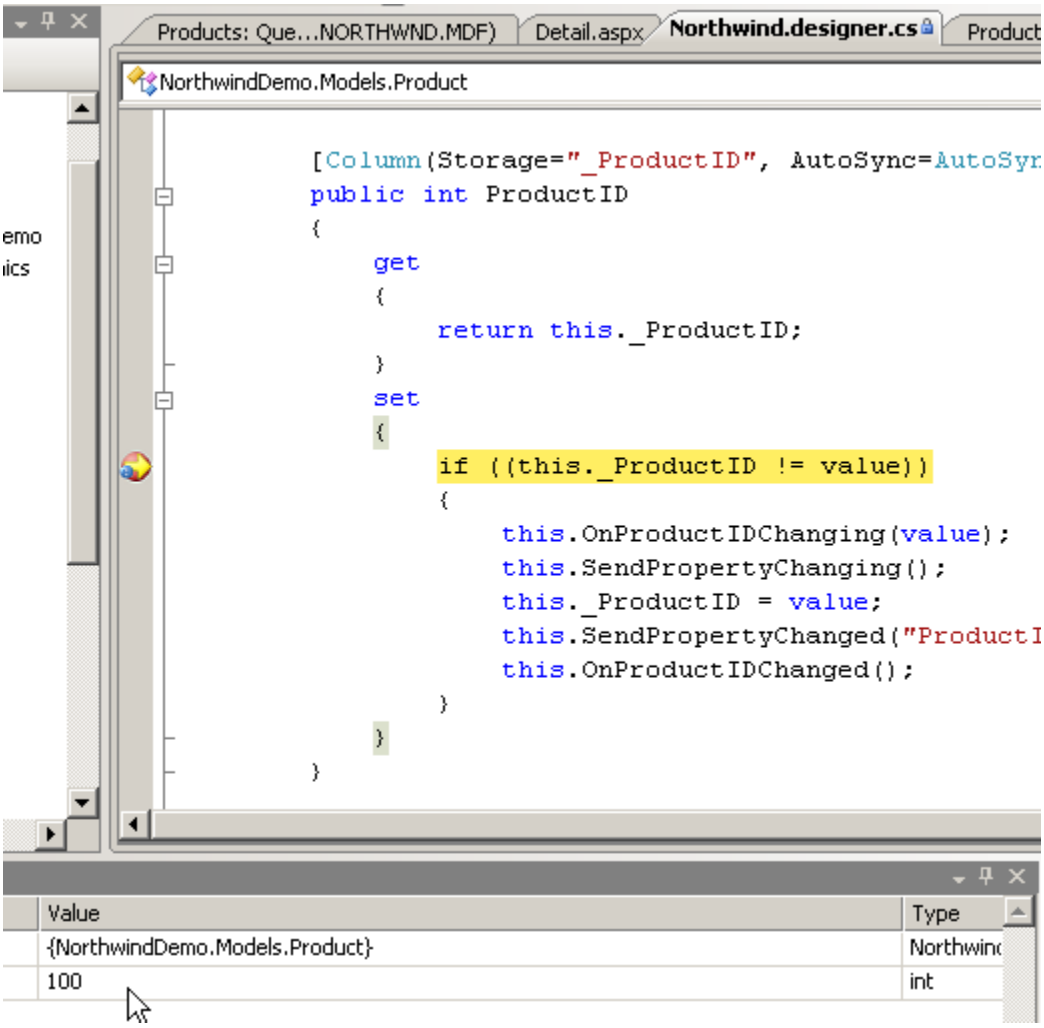


This product has the Product ID of 35

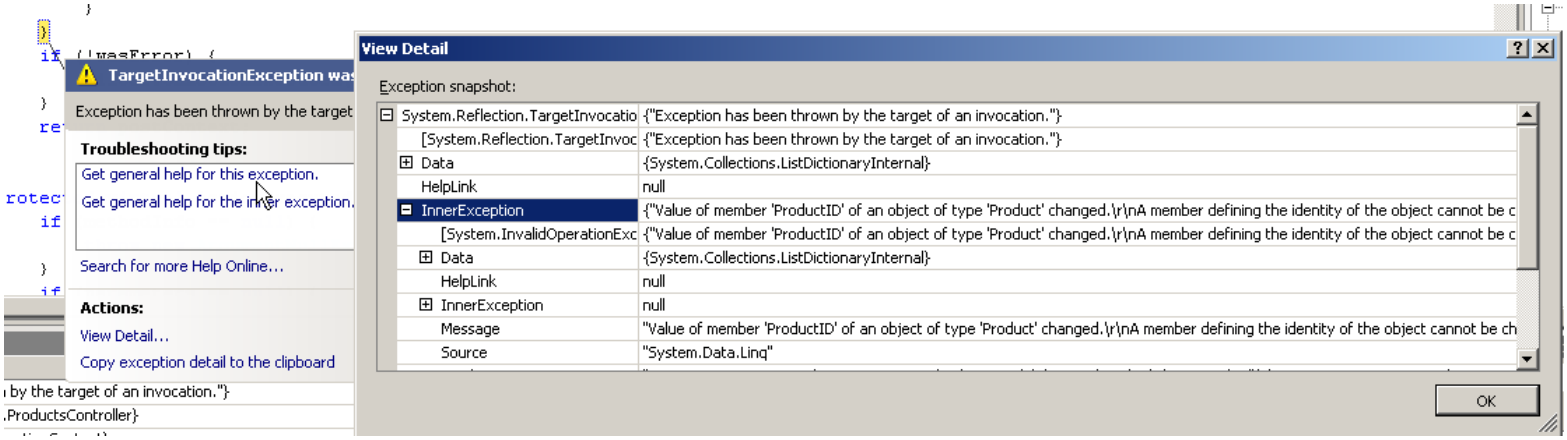
What happens if we add a new field called ProductID and set it to 100

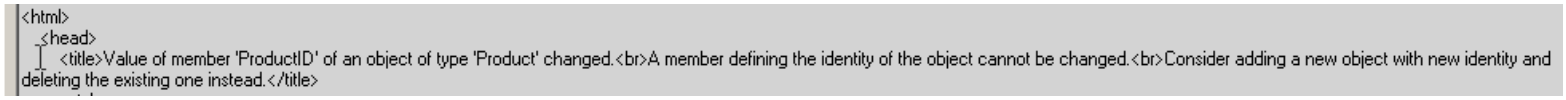


Bingo, the setter was invoked ☺

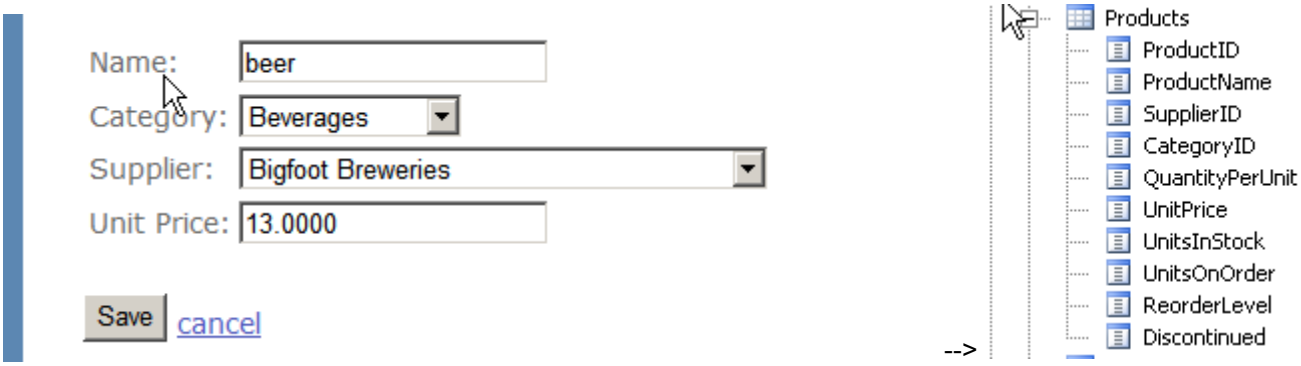


And it sort of worked, because the error is related to the fact that we can't change the value of an identity field:



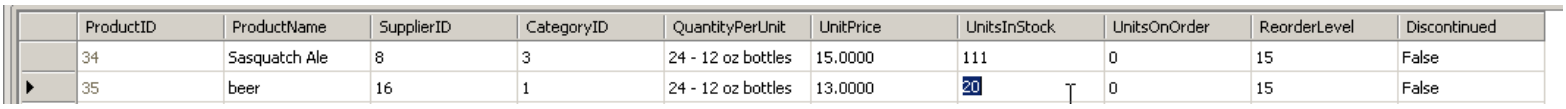


Just to complete the PoC, let's change something else that is not exposed by the GUI:

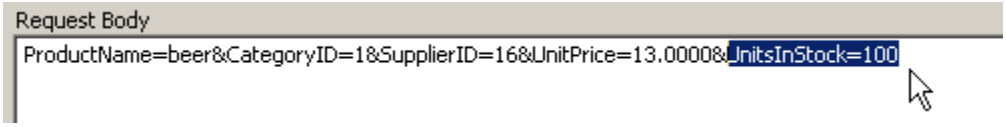


Let's try UnitsInStock

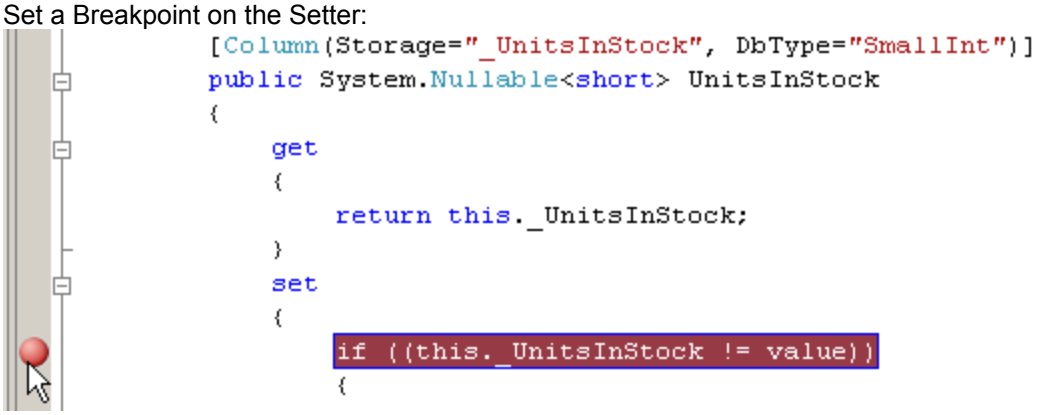
Currently on 20



ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
34	Sasquatch Ale	8	3	24 - 12 oz bottles	15.0000	111	0	15	False
35	beer	16	1	24 - 12 oz bottles	13.0000	20	0	15	False



Set a Breakpoint on the Setter:

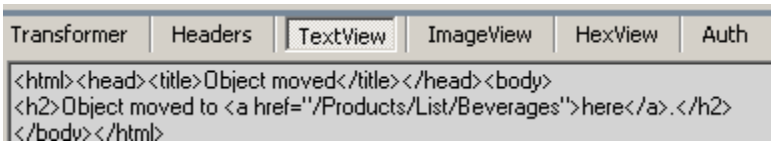


```
[Column(Storage="_UnitsInStock", DbType="SmallInt")]
public System.Nullable<short> UnitsInStock
{
    get
    {
        return this._UnitsInStock;
    }
    set
    {
        if ((this._UnitsInStock != value))
        {
```

submit the request and we have an Hit:

```
[Column(Storage="_UnitsInStock", DbType="SmallInt")]
public System.Nullable<short> UnitsInStock
{
    get
    {
        return this._UnitsInStock;
    }
    set
    {
        if ((this._UnitsInStock != value))
        {
```

The request completed successfully:



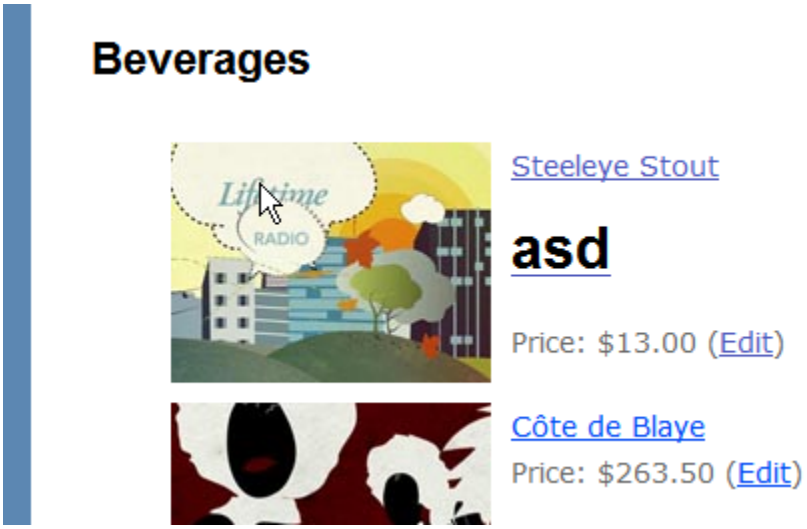
And the value was updated

▶	35	beer	16	1	24 - 12 oz bottles	13.0000	100	0	15	False
	36	Inlagd Sill	17	8	24 - 250 g jars	19.0000	112	0	20	False

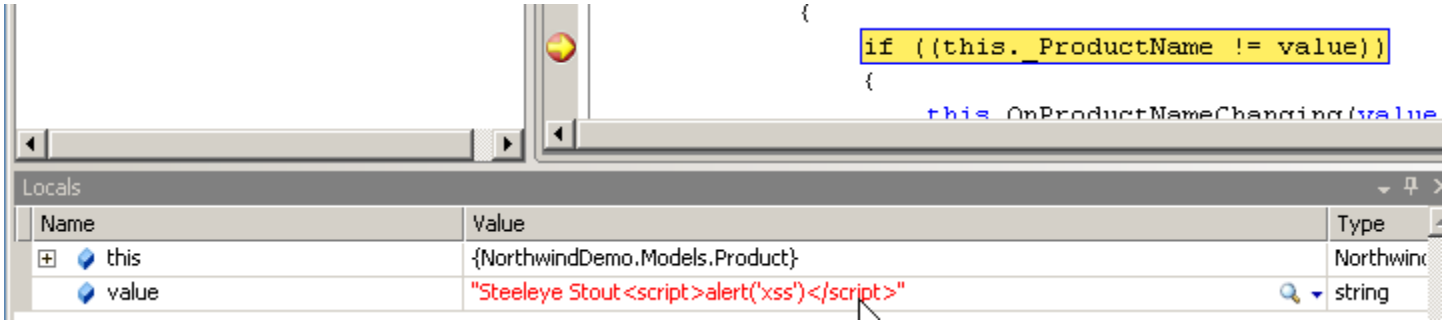
So this confirms that the ASP.NET MVC is vulnerable to the same vulnerability has the Spring Framework :) :) :)

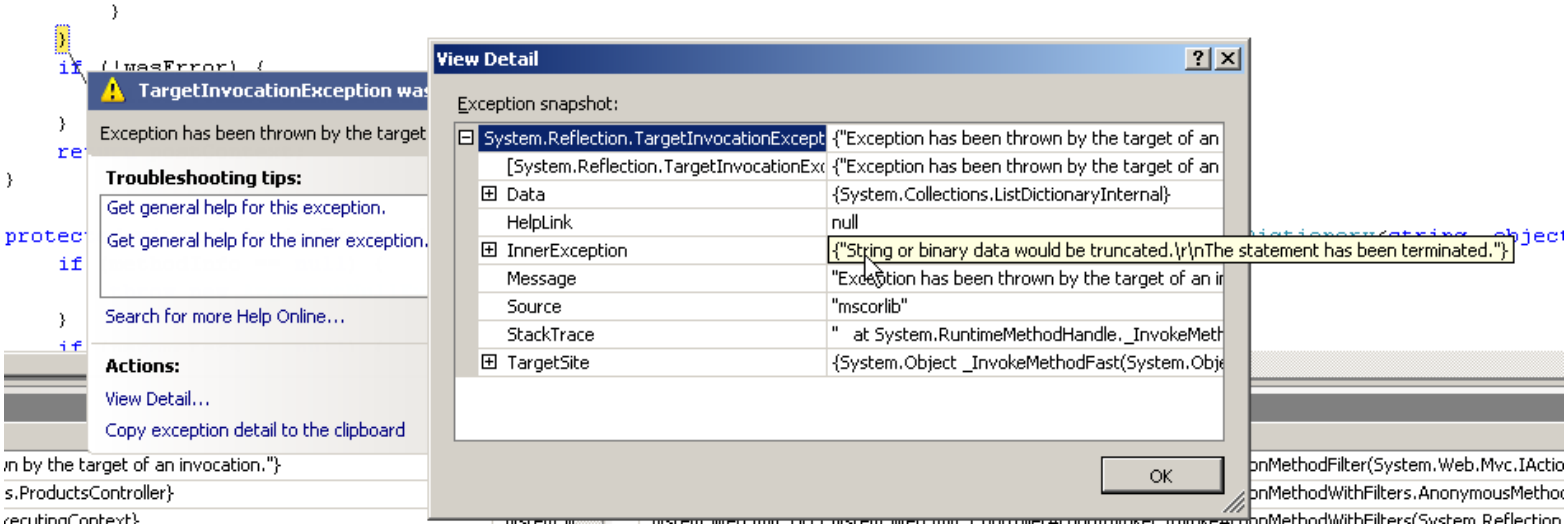
XSS by Design

Another Vulnerability is that MVC Apps are vulnerable to XSS by design.
Inserting payload on Name



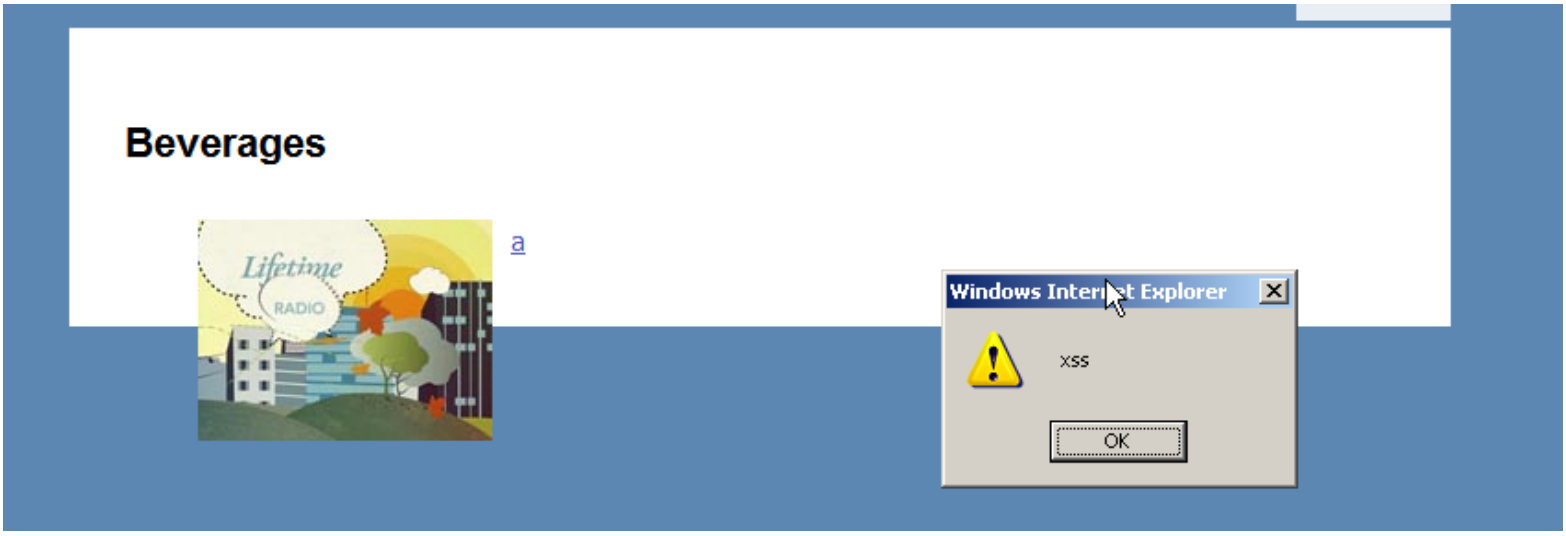
Trying a script tag





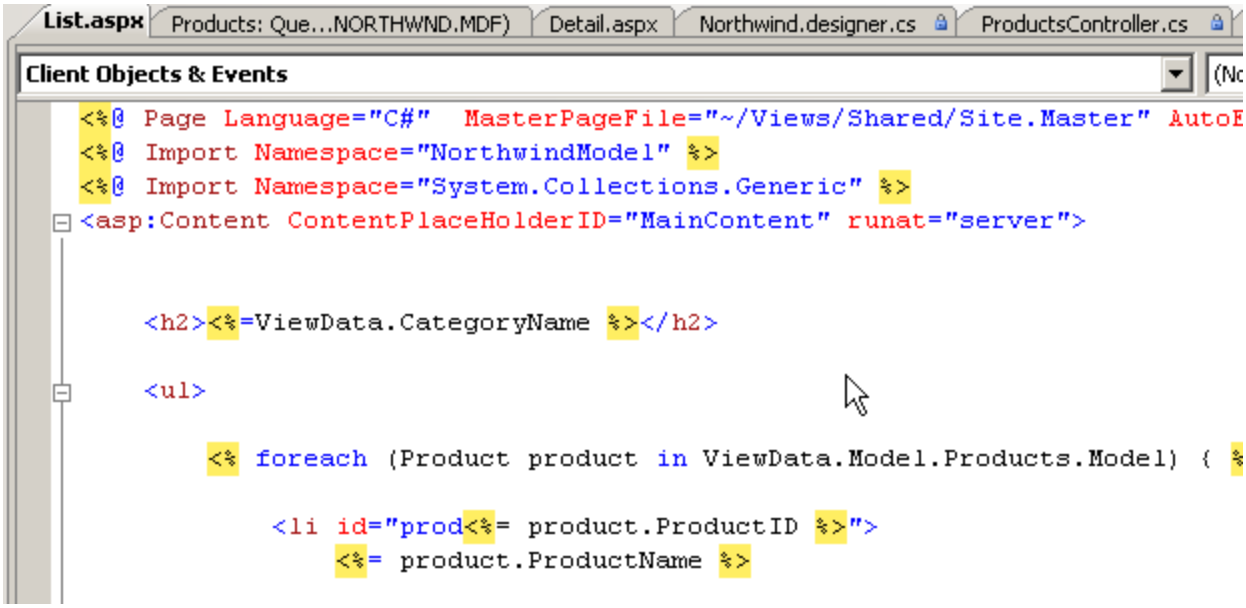
payload was too big, let’s try something small

And Bingo ☺

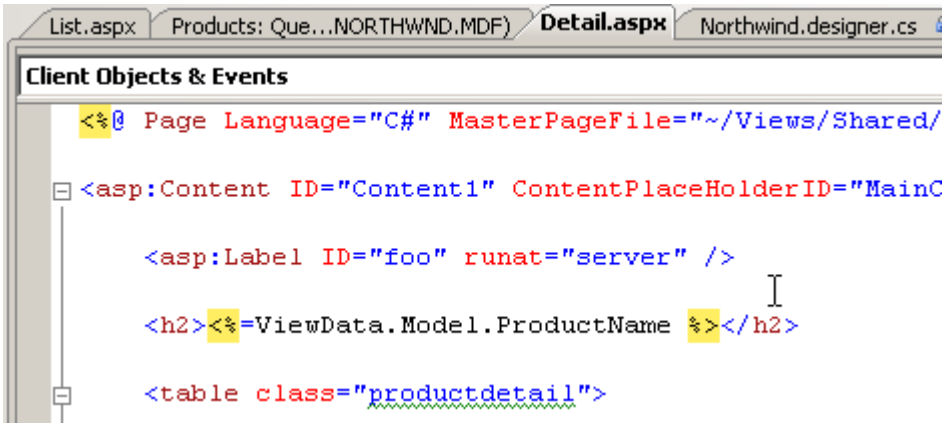


Let look at what happens in the code

The code on the List.aspx page looks like this:

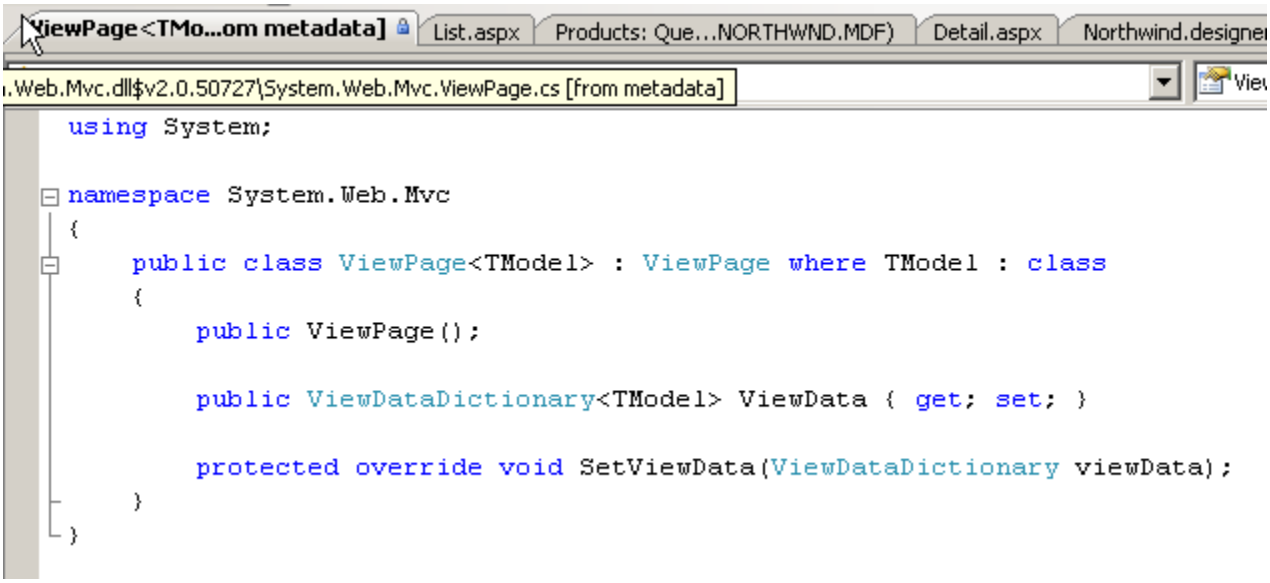


and the one on the Detail page like this

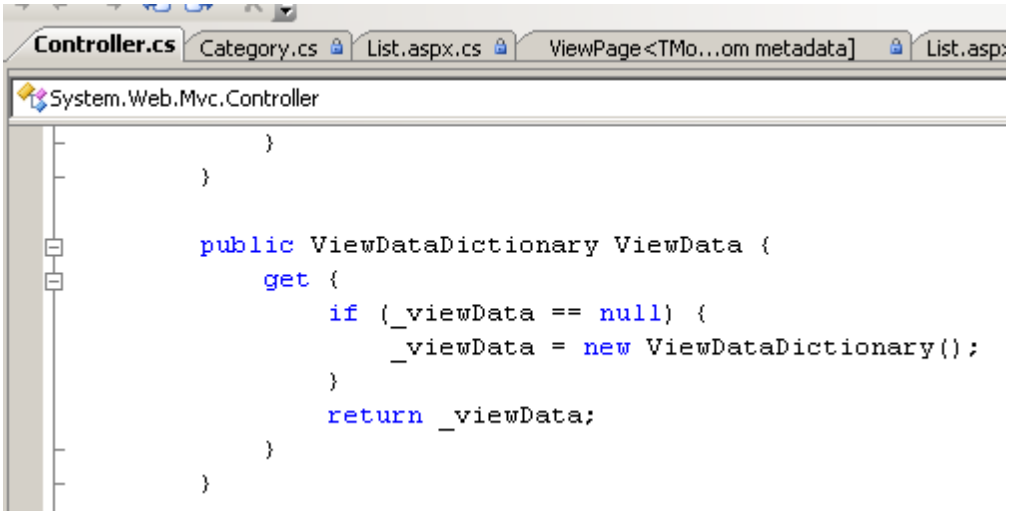


Basically what is happening is that the data placed in the ViewData is not encoded (as it goes in , or as it is used on the aspx page)

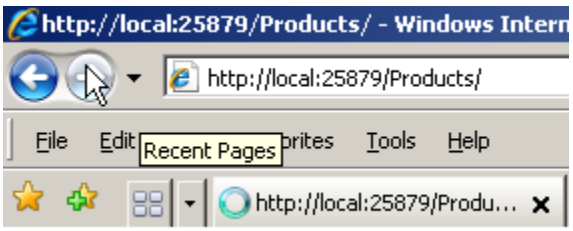
The ViewData object is defined inside the MVC dll:



created here:



Following a Http Request

Opening this page 

fires his controller:

ProductsController.cs | Controller.cs | Category.cs | List.aspx.cs | ViewPage<TMo...om metadata] | List..

NorthwindDemo.Controllers.ProductsController

```
public object Categories()
{
    return View("Categories", repository.Categories.ToList());
}
```

and

```
public virtual IQueryable<Category> Categories
{
    get
    {
        return this.dataContext.Categories;
    }
}
```

-

and

```
public System.Data.Linq.Table<Category> Categories
{
    get
    {
        return this.GetTable<Category>();
    }
}
```

Northwind.designer.cs | NorthwindRepository.cs | ProductsController.cs | Controller.cs | Category.cs | List.aspx.cs | ViewPage<TMo...om metadata] | List.aspx | Products: Que...NORTHWND.MDF)

NorthwindDemo.Models.Category — Category()

```
private EntitySet<Product> _Products;

Extensibility Method Definitions

public Category()
{
    this._Products = new EntitySet<Product>(new Action<Product>(this.attach_Products), new Action<Product>(this.detach_Products));
    OnCreated();
}
```

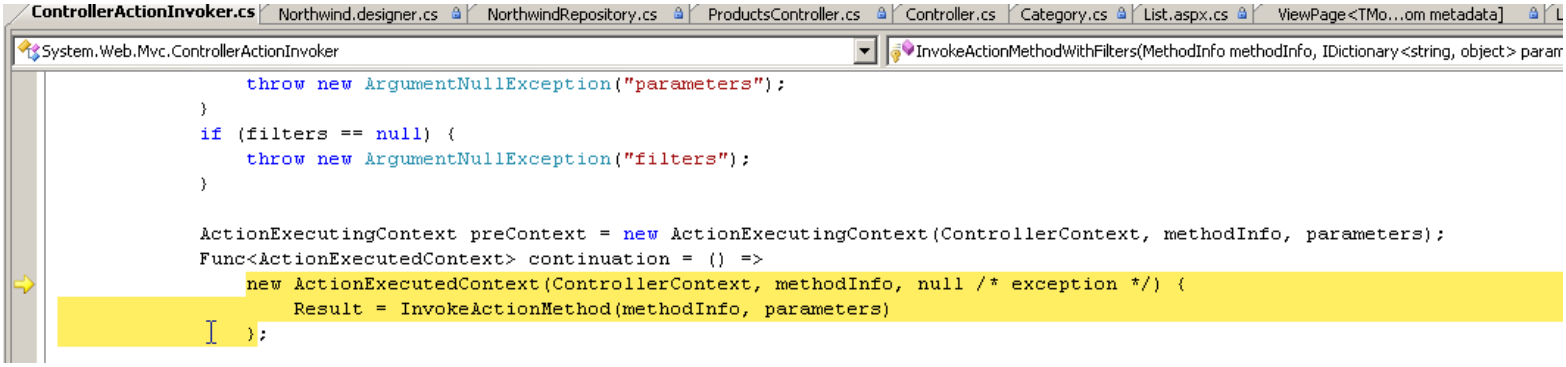
This loops for a while (it is executing the `repository.Categories.ToList());`)

goes back up to :

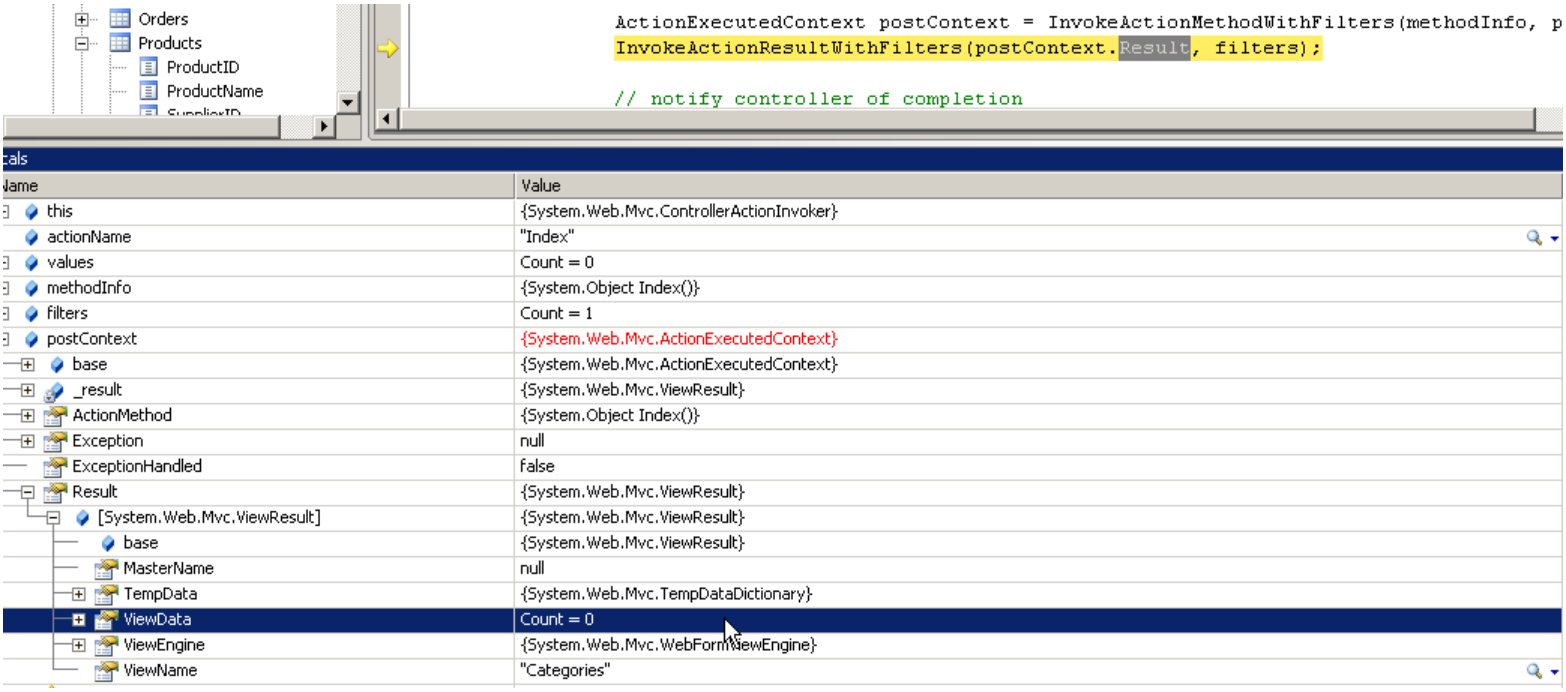
ControllerActionInvoker.cs | Northwind.designer.cs | NorthwindRepository.cs | ProductsController.cs | Controller.cs | Category.cs | List.aspx.cs | ViewPage<TMo...om me

System.Web.Mvc.ControllerActionInvoker — InvokeActionMethod(MethodInfo methodInfo, IDictionary<string, object> p

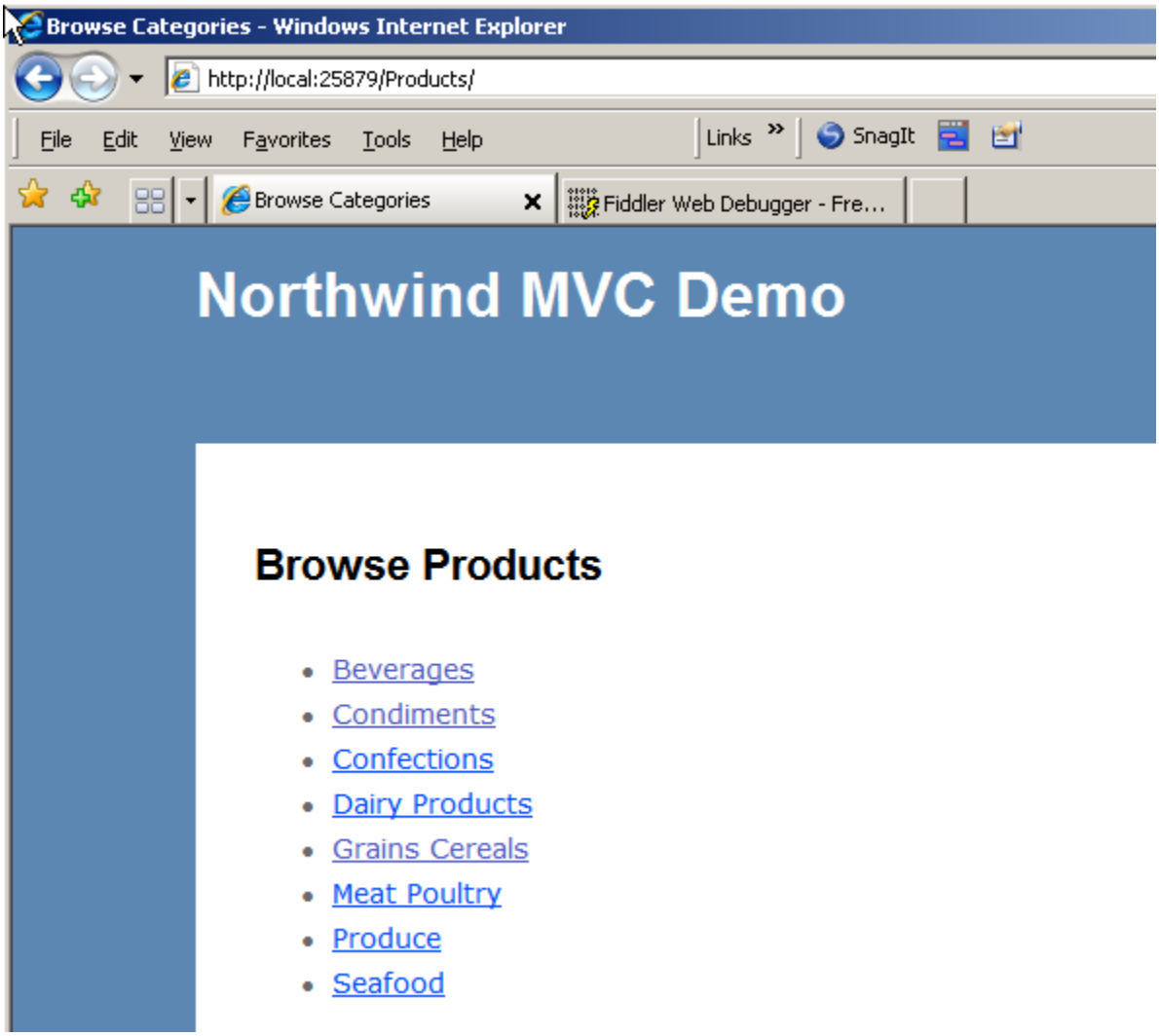
```
                CultureInfo.CurrentUICulture,
                MvcResources.ControllerActionInvoker_ParameterDictionaryIsInvalid,
                methodInfo.Name));
        }

        IController controller = ControllerContext.Controller;
        object[] parametersArray = (from parameterInfo
                                    in parameterInfos
                                    select ExtractParameterFromDictionary(parameterInfo, parameters)).ToArray();
        object returnValue = methodInfo.Invoke(controller, parametersArray);
```



In this example the ViewData Dictionary object is not populated:

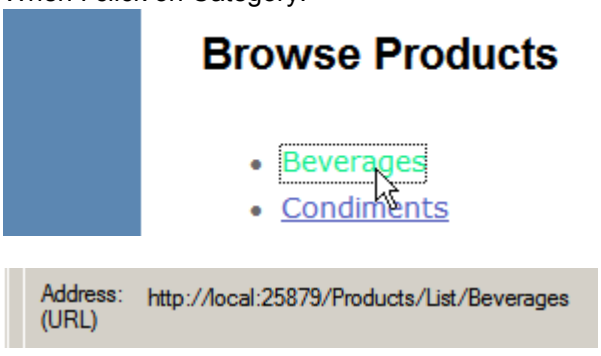


Which makes sense since we are only looking at the categories:

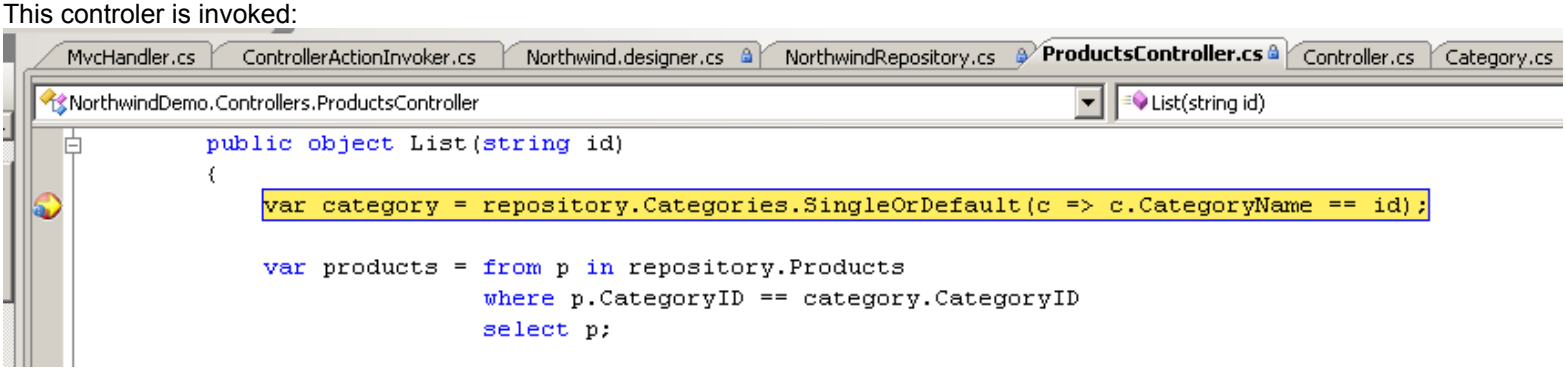


Next Example: List of Products in Category

When I click on Category:



This controler is invoked:



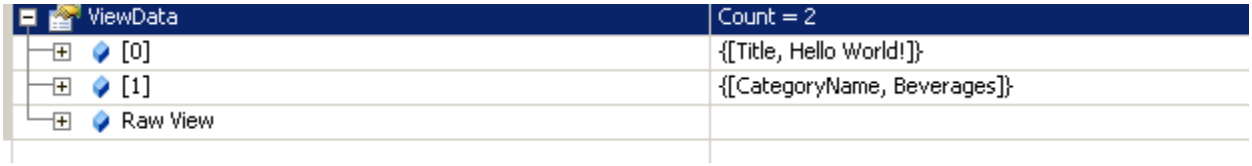
who gets the data from the database:

```
var category = repository.Categories.SingleOrDefault(c => c.CategoryName == id);

var products = from p in repository.Products
               where p.CategoryID == category.CategoryID
               select p;
```

and starts to populate the ViewData Dictionary, which is empty in the beggining

after

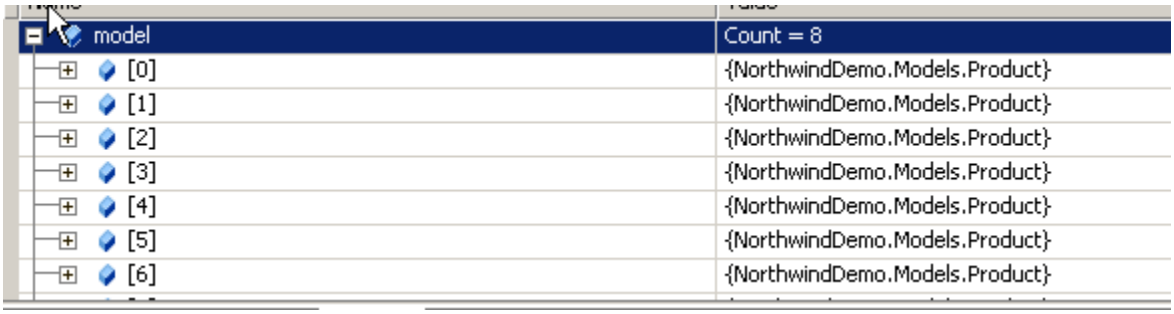
```
ViewData["Title"] = "Hello World!";
ViewData["CategoryName"] = id;
```



and this will create a view with the a list of Products

```
return View("ListingByCategory", products.ToList());
```

```
        protected internal ViewResult View(string viewName, object model) {
            return View(viewName, null /* masterName */, model);
        }
```



Return back to:

MvcHandler.cs | ControllerActionInvoker.cs | Northwind.designer.cs | NorthwindRepository.cs | ProductsController.cs | Controller.cs | Category.cs | List.aspx.cs

System.Web.Mvc.ControllerActionInvoker — InvokeActionMethod(MethodInfo methodInfo, IDictionary<string, obje

```
            IController controller = ControllerContext.Controller;
            object[] parametersArray = (from parameterInfo
                                        in parameterInfos
                                        select ExtractParameterFromDictionary(parameterInfo, parameters)).ToArray().
            object returnValue = methodInfo.Invoke(controller, parametersArray);

            if (returnValue == null) {
                return new EmptyResult();
            }

            ActionResult actionResult = (returnValue as ActionResult) ??
                new ContentResult { Content = Convert.ToString(returnValue, CultureInfo.InvariantCulture) };
            return actionResult;
        }
```

The data show is:

Which was created by this View:

```
<h2><%= ViewData["CategoryName"] %></h2>

<ul class="productlist">

    <% foreach (var product in ViewData.Model) { %>

        <li>
            <img src="/Content/Images/<%=product.ProductID%>.jpg" alt="<%= Html.AttributeEncode(product.ProductName) %>"
            <div>
                <a href="/Products/Detail/<%=product.ProductID %>"> <%=product.ProductName %> </a>
                <br />
                Price: <%=String.Format("{0:C2}", product.UnitPrice)%>
                  <span class="editlink"">
                     (<%=Html.ActionLink("Edit", "Edit", new { ID=product.ProductID })%>)
                </span>
            </div>

        </li>

    <% } %>
```

In this case the ViewData.Model has the product list

And has we saw before we can put XSS payloads on (for example) the ProductName field which is shown unencoded

There are also issues on who can call what (authorization), see these posts:
- http://weblogs.asp.net/fredriknormen/archive/2007/11/25/asp-net-mvc-framework-security.aspx
- http://geekswithblogs.net/AzamSharp/archive/2008/02/24/119946.aspx

Some notes:

- Where is the ViewState? It looks like they dropped and in this case it would prevent this from being exploited
- What about PageValidation? What is that also disabled? (that allows XSS payloads on the way in)