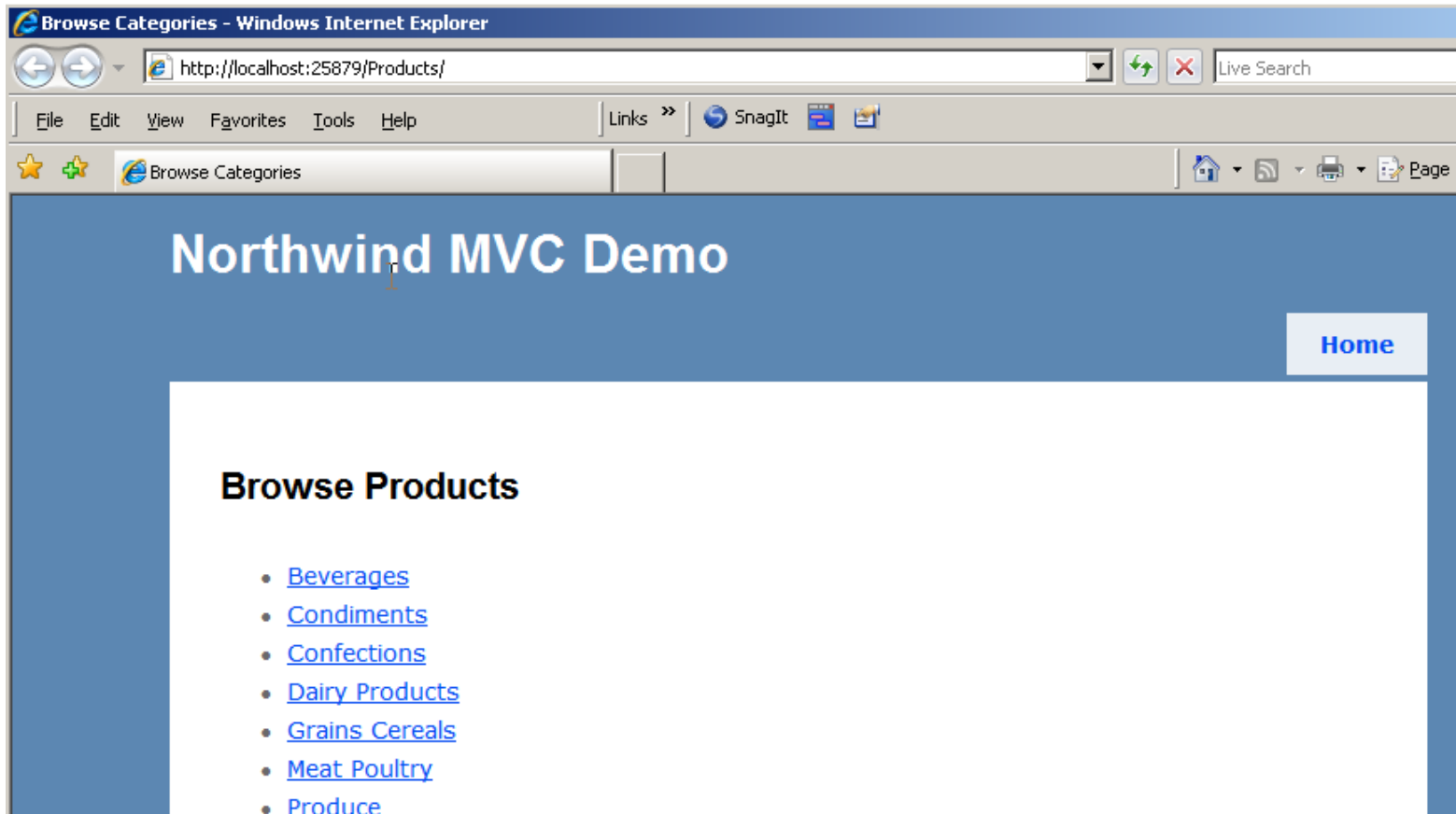


## ASP.NET MVC – XSS and AutoBind vulns in MVC Example

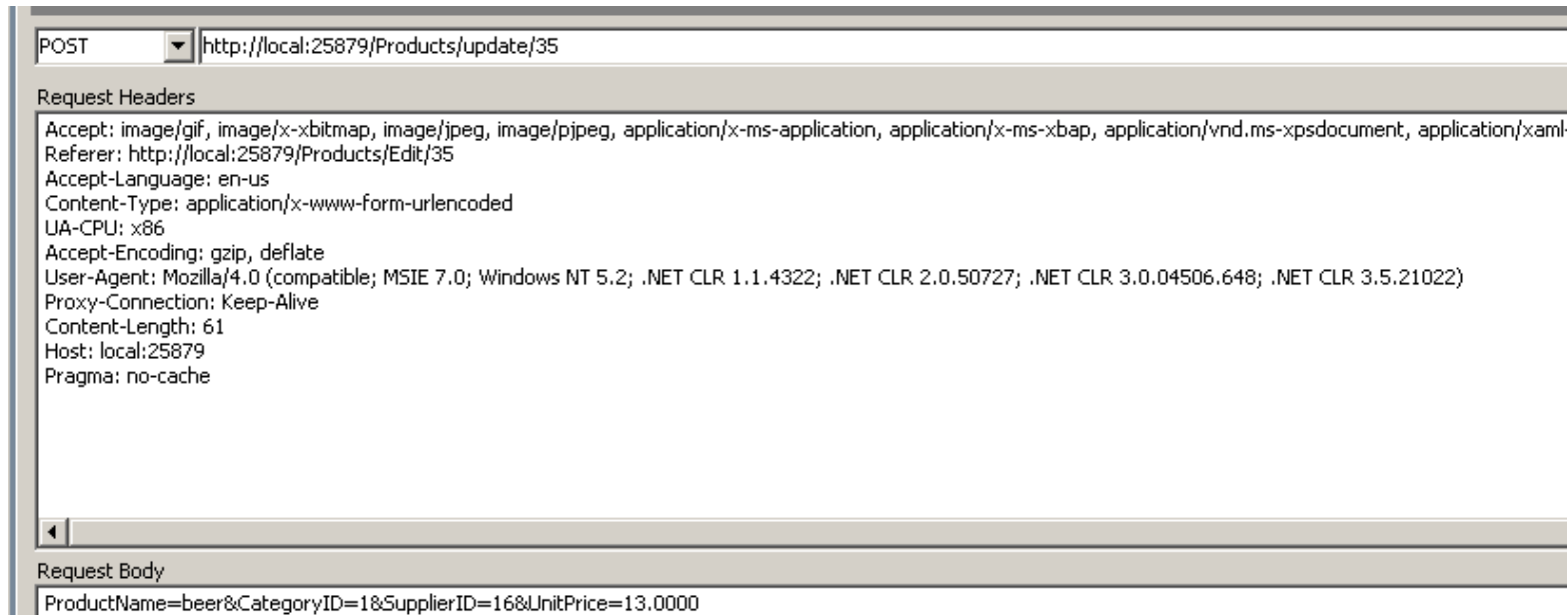
Downloaded <http://haacked.com/archive/2008/05/23/updated-northwind-demo.aspx> unzipped and successful run it ☺ (nice one click example)



### AUTO BIND Vulnerability

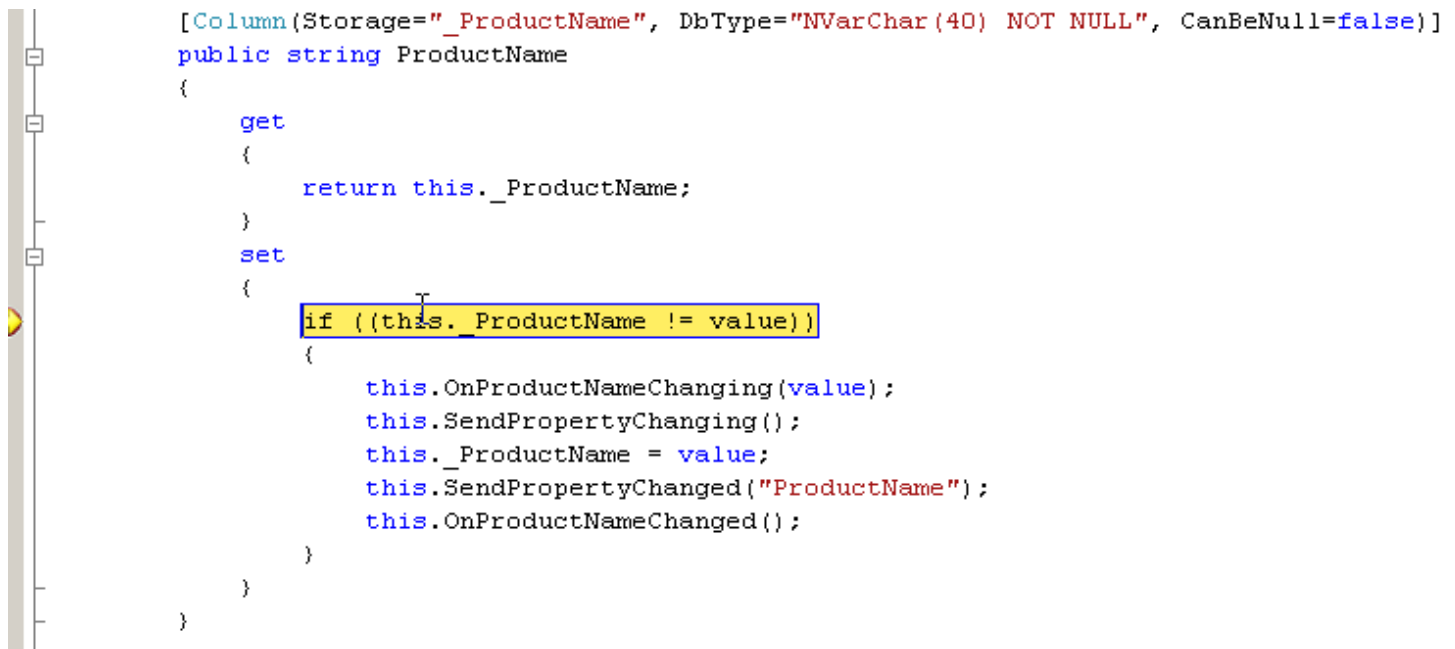
A screenshot of a web form for editing a product. The form has four input fields: "Name:", "Category:", "Supplier:", and "Unit Price:". The "Name:" field contains the text "Steeleye Stout<h1>asd</". The "Category:" field is a dropdown menu with "Beverages" selected. The "Supplier:" field is a dropdown menu with "Bigfoot Breweries" selected. The "Unit Price:" field contains the value "13.0000". Below the fields are two buttons: "Save" and "cancel".

The HTTP of Submit of the Edit looks like this:

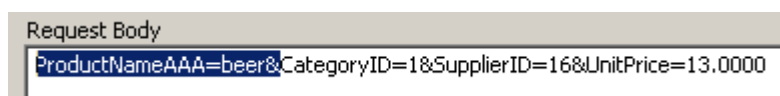


and these filed names map into variables names:

For example ProductName



Let's see what happens if we try:



we got a redirect

Name	Value
Product NameAAA	beer
CategoryID	1
SupplierID	16
UnitPrice	13.0000

Transformer Headers **TextView** ImageView HexView Auth

```
<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href="/Products/Edit/35">here</a>.</h2>
</body></html>
```

This product has the Product ID of 35

Products: Que...NORTHWND.MDF

Detail.aspx

Northwind.designer.cs

ProductsController.cs

MvcHandler.cs

Controller.cs

ControllerActionInvoker.cs

	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

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

Request Body  
 ProductName=beer&CategoryID=1&SupplierID=16&UnitPrice=13.0000&ProductID=100

Bingo, the setter was invoked ☺

Products: Que...NORTHWND.MDF Detail.aspx **Northwind.designer.cs** Product

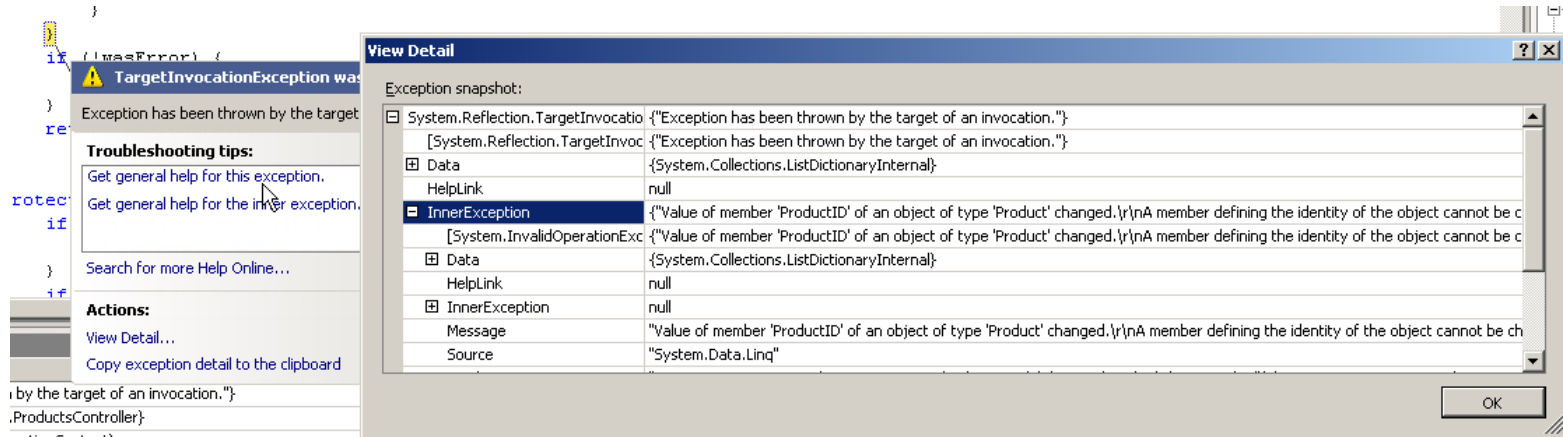
NorthwindDemo.Models.Product

```
[Column(Storage="_ProductID", AutoSync=AutoSync)
public int ProductID
{
    get
    {
        return this._ProductID;
    }
    set
    {
        if ((this._ProductID != value))
        {
            this.OnProductIDChanging(value);
            this.SendPropertyChanging();
            this._ProductID = value;
            this.SendPropertyChanged("ProductID");
            this.OnProductIDChanged();
        }
    }
}
```

Value Type

{NorthwindDemo.Models.Product}	Northwind
100	int

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



```
<html>
<head>
<title>Value of member 'ProductID' of an object of type 'Product' changed.<br>A member defining the identity of the object cannot be changed.<br>Consider adding a new object with new identity and deleting the existing one instead.</title>
```

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

Name: 
Category: 
Supplier: 
Unit Price:

Products

- ProductID
- ProductName
- SupplierID
- CategoryID
- QuantityPerUnit
- UnitPrice
- UnitsInStock
- UnitsOnOrder
- ReorderLevel
- Discontinued

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

Request Body

```
ProductName=beer&CategoryID=1&SupplierID=16&UnitPrice=13.0000&UnitsInStock=100
```

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:

```
Transformer Headers TextView ImageView HexView Auth
<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href="/Products/List/Beverages">here</a>.</h2>
</body></html>
```

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

Name:

Category:

Supplier:

Unit Price:

[cancel](#)

## Beverages



[Steeleye Stout](#)

**asd**

Price: \$13.00 ([Edit](#))



[Côte de Blaye](#)

Price: \$263.50 ([Edit](#))

Trying a script tag

**Save** [cancel](#)



And Bingo 😊

# Beverages



a



Let look at what happens in the code

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

```
List.aspx Products: Que...NORTHWND.MDF) Detail.aspx Northwind.designer.cs ProductsController.cs
Client Objects & Events
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/Site.Master" AutoE
<%@ Import Namespace="NorthwindModel" %>
<%@ Import Namespace="System.Collections.Generic" %>
<asp:Content ContentPlaceHolderID="MainContent" runat="server">

    <h2><%= ViewData.CategoryName %></h2>

    <ul>

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

            <li id="prod<%= product.ProductID %>">
                <%= product.ProductName %>

and the one on the Detail page like this
```

```
List.aspx Products: Que...NORTHWND.MDF) Detail.aspx Northwind.designer.cs
Client Objects & Events
<%@ Page Language="C#" MasterPageFile="~/Views/Shared/
<asp:Content ID="Content1" ContentPlaceHolderID="MainC

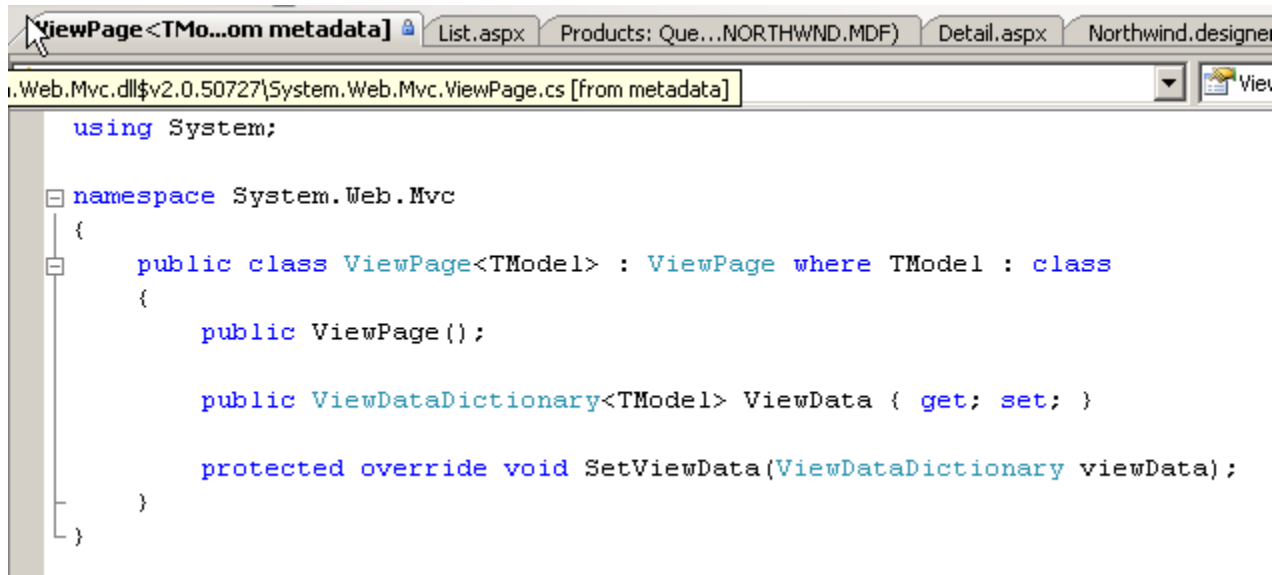
    <asp:Label ID="foo" runat="server" />

    <h2><%= ViewData.Model.ProductName %></h2>

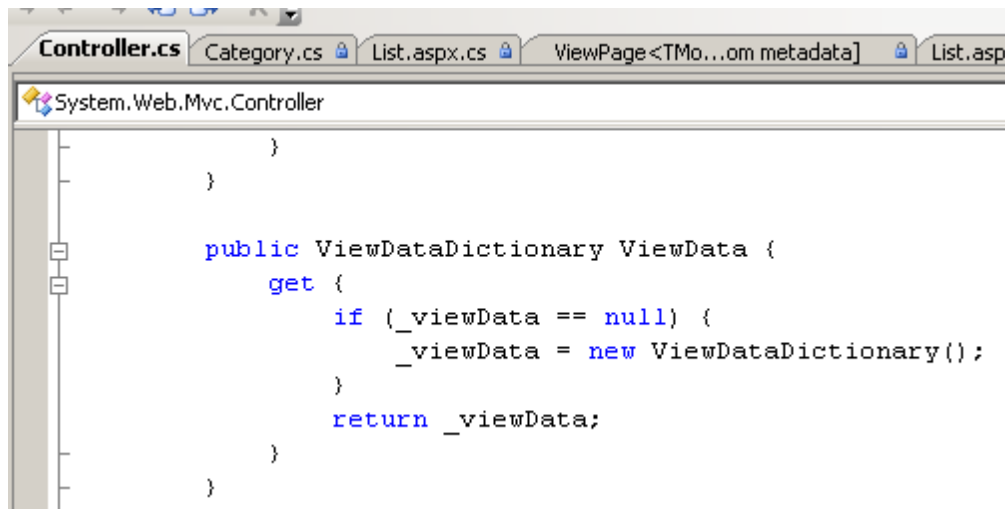
    <table class="productdetail">
```

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)

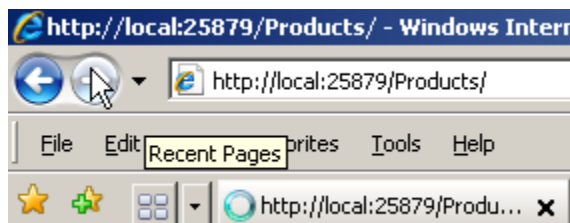
The ViewData object is defined inside the MVC dll:



created here:



## Following a Http Request



Opening this page



```

ProductsController.cs
Controller.cs Category.cs List.aspx.cs ViewPage<TModel, TView>.cs List.aspx

NorthwindDemo.Controllers.ProductsController

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

```

fires his controller:

```

Northwind.designer.cs
NorthwindRepository.cs ProductsController.cs Controller.cs Category.cs List.aspx.cs ViewPage<TModel, TView>.cs List.aspx Products: Queue<Product>.NORTHWND.MDF

NorthwindDemo.Models.Category

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

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

```

and

and

```

Northwind.designer.cs
NorthwindRepository.cs ProductsController.cs Controller.cs Category.cs List.aspx.cs ViewPage<TModel, TView>.cs List.aspx Products: Queue<Product>.NORTHWND.MDF

NorthwindDemo.Models.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<TModel, TView>.cs List.aspx Products: Queue<Product>.NORTHWND.MDF

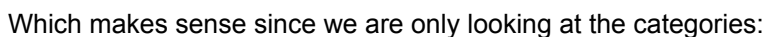
System.Web.Mvc.ControllerActionInvoker

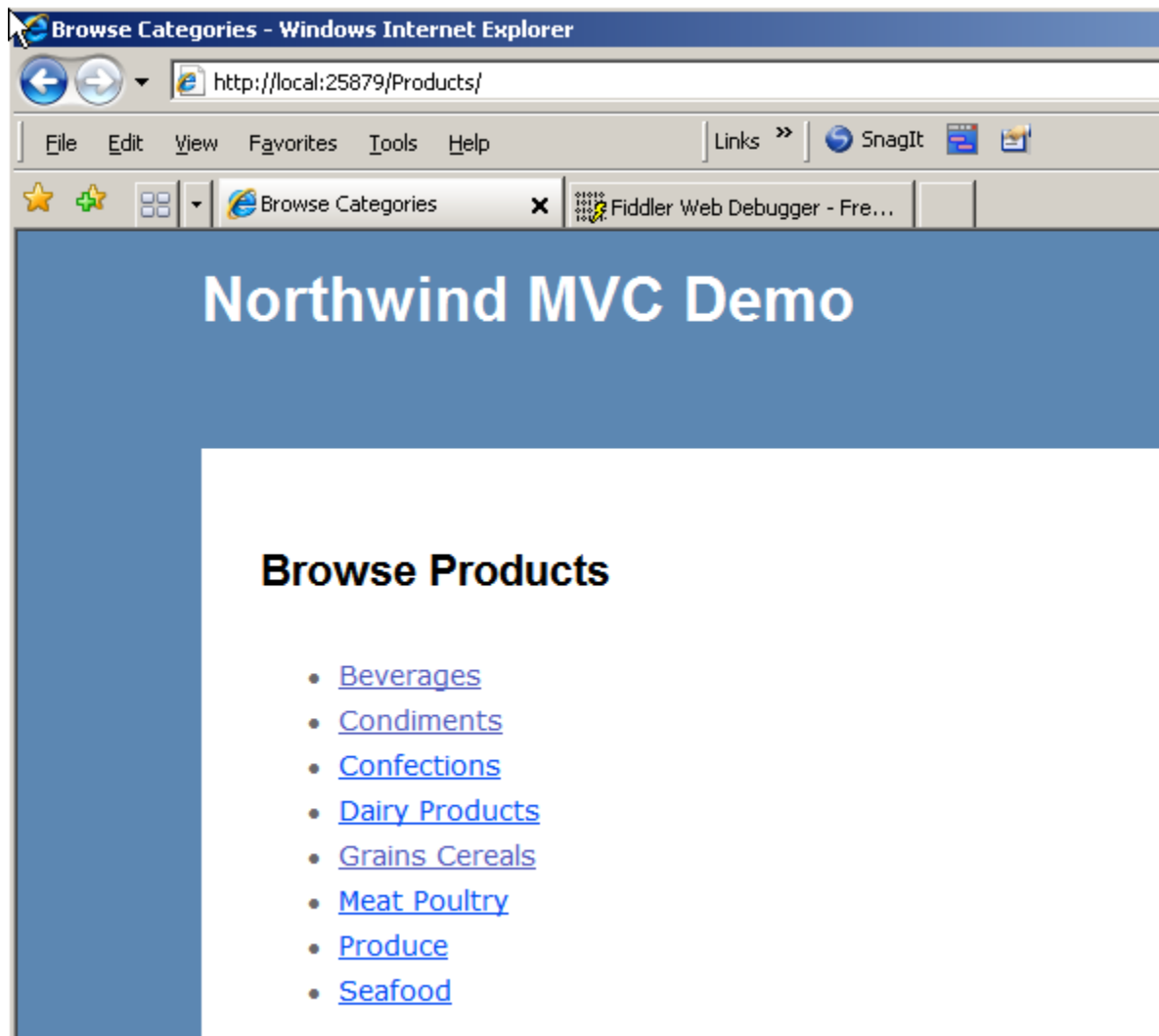
InvokeActionMethod(MethodInfo methodInfo, IDictionary<string, object> parameters, CultureInfo culture)

{
    CultureInfo.CurrentUICulture,
    MvcResources.ControllerActionInvoker_ParameterDictionary_Invalid,
    methodInfo.Name);
}

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

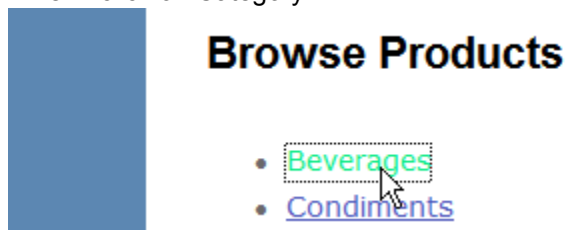
```





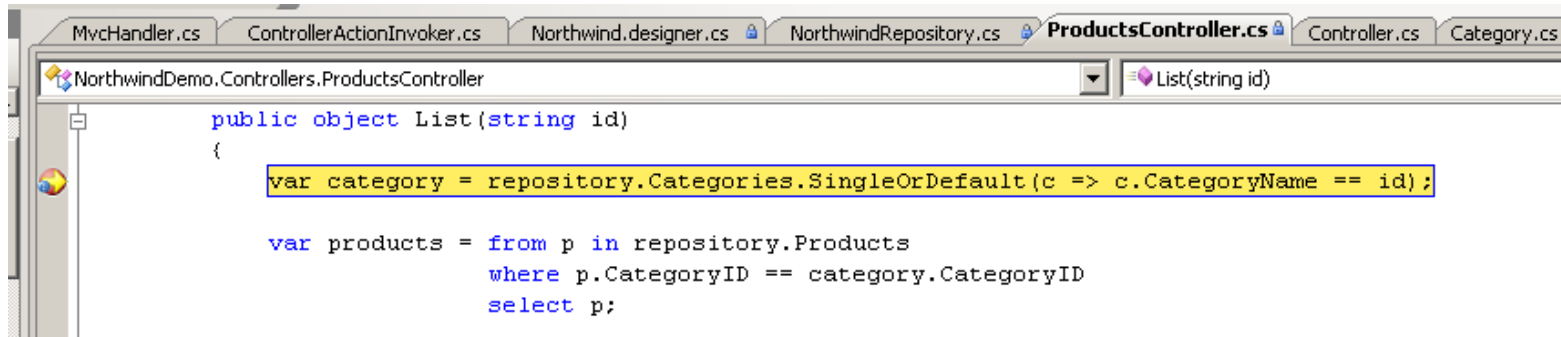
#### Next Example: List of Products in Category

When I click on Category:



Address: <http://local:25879/Products/List/Beverages>  
(URL)

This controller 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 beginning

Name	Value
ViewData	Count = 0

after

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

ViewData	Count = 2
[0]	{[Title, Hello World!]}
[1]	{[CategoryName, Beverages]}
Raw View	

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

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

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

model	Count = 8
[0]	{NorthwindDemo.Models.Product}
[1]	{NorthwindDemo.Models.Product}
[2]	{NorthwindDemo.Models.Product}
[3]	{NorthwindDemo.Models.Product}
[4]	{NorthwindDemo.Models.Product}
[5]	{NorthwindDemo.Models.Product}
[6]	{NorthwindDemo.Models.Product}

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, object> parameters)

    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>
            "
            <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 as 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)