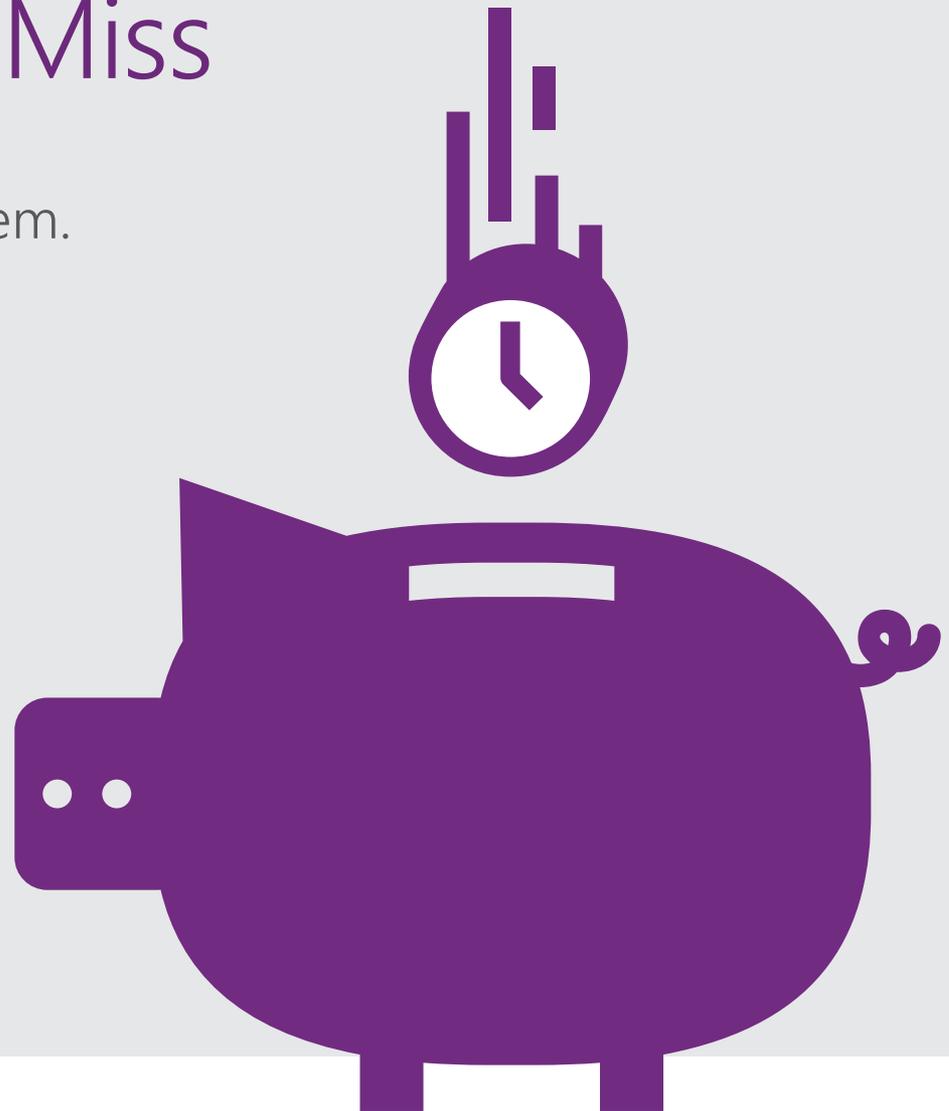


This may be out of date. [Read what's new here!](#)

# Time-Saving VS 2012 and ASP.NET 4.5 Features You Shouldn't Miss

See the highlights and learn  
how to make the most of them.



In this E-book  
you will learn how:

### Intellisense

help you discover  
JS and CSS features

### WAI Aria

makes the web  
usable by all

### HTML5 and CSS3

are available out of  
the box

### WebAPI

exposes data via  
HTTP service

### Strongly Typed Data Binding

ensures runtime  
confidence

### Request Validation

prevents yellow  
screens of death

### Page Inspector

puts an end to layout headaches

## Introduction:

Each Visual Studio release is like Christmas for developers around the globe. Not only does Microsoft evolve our beloved IDE, but they also pack all kinds of new features into ASP.NET. Visual Studio 2012 is no exception! It's packed full of awesome features and the new capabilities in ASP.NET allow developers to get their job done with less code!

Learning about all the new features, and more importantly, learning to use these features, can be a daunting task for any developer.

In this guide we will take a look at some of the highlights of the new IDE and platform. See how to use these features, and learn how they make your life as a developer easier! It is our hope that this short guide will make it possible for you to [download Visual Studio 2012](#), and leverage the new features in the IDE and ASP.NET 4.5.

**Jumpstart your Visual Studio 2012  
and ASP.NET 4.5 learning now!**

## About the Authors:



Joshua  
Holt

works as a Developer Support Specialist at Telerik focusing on ASP.NET, but tends to dabble in all of the products. Part of his role at Telerik is to work closely with the team, and help developers leverage the Telerik product line up. In his free time he likes to tinker with spatial data visualization projects, and chase clouds. Prior to working at Telerik, Joshua worked as a consultant for several Fortune 500 companies in the oil and gas industry.

[@jholt456](#)



Carl  
Berghem

is an Enterprise Solutions Consultant at Telerik specializing in the ASP.NET AJAX and ASP.NET MVC products. He has always been interested in web development and has played around with various web technologies since he was a child. In his free time Carl enjoys soccer, running and playing his guitar.

[@carlberghem](#)

## Contents:

### HOT NEW DESIGNER FEATURES 4

1. Intellisense Make JS and CSS Features Discoverable 4
2. WAI Aria Makes the Web Usable by All 8
3. HTML5 & CSS3 Support for Modern Web Applications 10

### NEW CODE FEATURES FOR BETTER WEB APPLICATIONS 11

1. Better Web APIs with ASPNET WebAPI 11
2. Strongly Typed Data Binding for Runtime Confidence 14
3. Lazy Request Validation Prevents YoD 16

### CATCHING BUGS BEFORE THEY CATCH YOU 18



## HOT NEW DESIGNER FEATURES

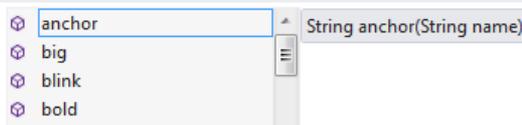
### 1 INTELLISENSE MAKE JS AND CSS FEATURES DISCOVERABLE

#### • JavaScript Intellisense

In previous versions of Visual Studio developers were left fending for themselves when it came to JavaScript development, with no aid from intellisense. While plugins and extra installations could fit this in to Visual Studio 2010, it is now offered out of the box with VS 2012. This means that functions related to the types of a variable will be displayed in intellisense, as opposed to either nothing or a list of all available JavaScript functions.

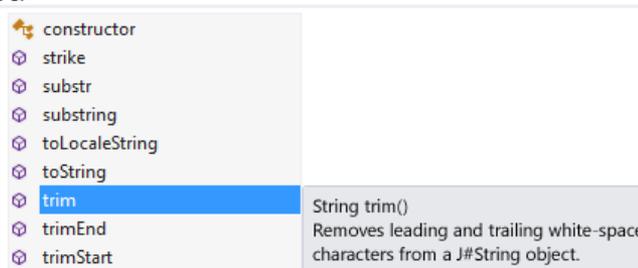
Here a string variable is defined within the document's ready function, and as can be seen only functions related to string manipulation appear within intellisense. Additionally, detailed information about functions will be displayed. Take trim() for example:

```
<script type="text/javascript">
    $(document).ready(function () {
        var myStringVariable = "This is a sample string";
        myStringVariable.
    });
</script>
```



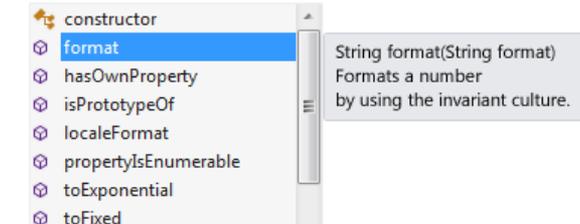
Now more information on what the trim() function does is provided to the developer.

```
<script type="text/javascript">
    $(document).ready(function () {
        var myStringVariable = "This is a sample string";
        myStringVariable.tr
    });
</script>
```



The same is true for numeric variables:

```
<script type="text/javascript">
    $(document).ready(function () {
        var myStringVariable = "This is a sample string";
        var myNumericVariable = 0;
        myNumericVariable.
    });
</script>
```



## • jQuery Intellisense and Documentation

VS 2012 goes a step beyond plain JavaScript intellisense and also provides the same functionality to one of the top JavaScript libraries: jQuery. A quick example of how deep this goes can be seen by attempting to bind to the document's ready event.

First typing out '\$(' provides detailed insight as to what can be accomplished by utilizing the jQuery selector.

```
<script type="text/javascript">
```

```
$(
```

```
</script> jQuery $(String selector, jQuery context)
```

1: \$(expression, context) – This function accepts a string containing a CSS selector which is then used to match a set of elements.  
2: \$(html) – Create DOM elements on-the-fly from the provided String of raw HTML.  
3: \$(elements) – Wrap jQuery functionality around a single or multiple DOM Element(s).  
4: \$(callback) – A shorthand for \$(document).ready().  
5: \$() – As of jQuery 1.4, if you pass no argument is to the jQuery() method, an empty jQuery set will be returned.

**selector:** 1: expression – An expression to search with.  
2: html – A string of HTML to create on the fly.  
3: elements – DOM element(s) to be encapsulated by a jQuery object.  
4: callback – The function to execute when the DOM is ready.

Since this is the document's ready event, the document variable should be passed. Moving on to the .ready() portion VS 2012 is eager to help:

```
<script type="text/javascript">
```

```
$(document).r
```

```
</script>
```

propertyIsEnumerable  
push  
pushStack  
queue  
ready  
remove  
removeAttr  
removeClass  
removeData

ready(Function fn)  
Binds a function to be executed whenever the DOM is ready to be traversed and manipulated.

And finally when the parenthesis is open to see what the ready function takes as a parameter the following appears:

```
<script type="text/javascript">
```

```
$(document).ready(
```

```
</script>
```

ready(Function fn)

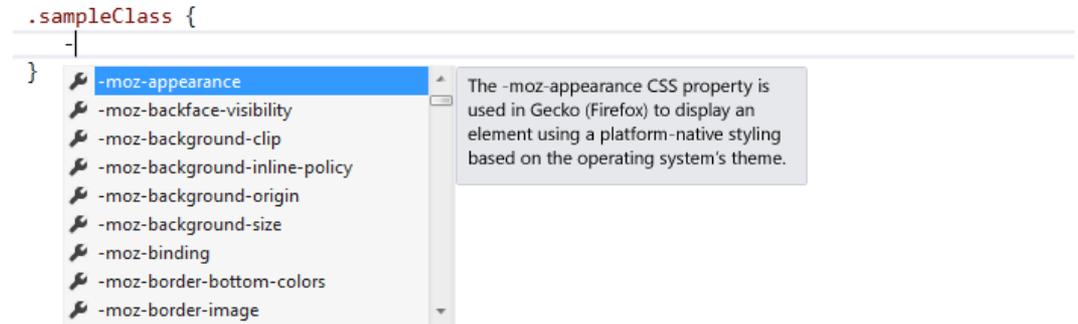
Binds a function to be executed whenever the DOM is ready to be traversed and manipulated.  
**fn:** The function to be executed when the DOM is ready.

As the images above depict, Visual Studio 2012 has stepped up its efforts when it comes to client-side development and the usage of the jQuery library.

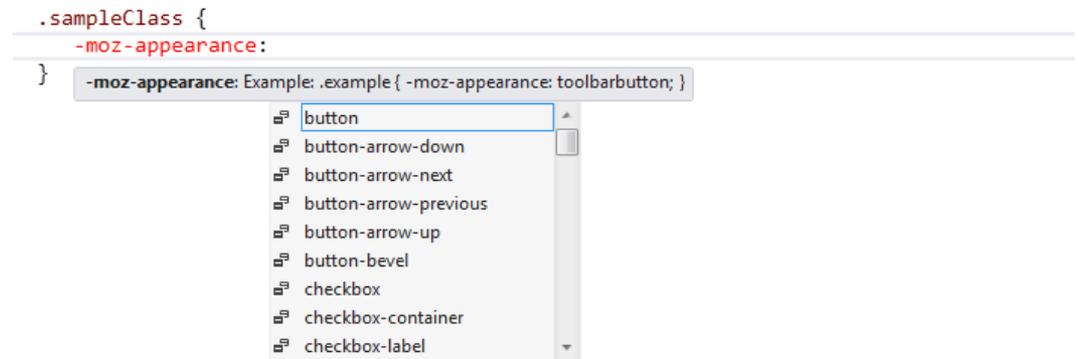
## • CSS Vendor-specific Intellisense

While intellisense for CSS was provided in VS 2010 and below, VS 2012 goes above and beyond by providing developers with the full list of vendor-specific options within a CSS file.

Simply typing “-” yields the following:



It also provides examples of the proper parameters that the CSS rule needs:



This allows developers easy access to all the vendor-specific CSS rules that are required to create cross-browser applications.

## • CSS Color Selection

Along the same lines, Visual Studio 2012 provides an integrated color-picker when a CSS rule related to a color is defined.

With using background-color, for example, a small bar containing a varying degree of colors appears to allow for quick selection.



However, if that is not enough a click of the arrow on the right will display a more in-depth color chooser which will satisfy any designer.



## The Bottom Line

Visual Studio 2012's new JavaScript and CSS features make it easy for developers and designers alike to build awesome web applications. For an in-depth look at all of the new features mentioned above, take a look at this [blog post](#)

## 2 WAI ARIA MAKES THE WEB USABLE BY ALL

### • What is WAI-ARIA?

WAI-ARIA is a set of specifications on how to increase the accessibility of modern day web applications that deal with “dynamic content and advanced user interface controls developed with AJAX, HTML, JavaScript, and related technologies” (quoting the official [WAI-ARIA page](#)). This comes from the [Web Accessibility Initiative \(WAI\)](#) of the [W3C](#).

### • Visual Studio 2012 and WAI-ARIA

Visual Studio 2012 studio does a couple of things to help developers with ensuring that their applications are WAI-ARIA compliant, which mainly stem from the intellisense feature. With this developers can easily define the WAI-ARIA roles and states for all of their HTML elements. In previous versions of Visual Studio only a Visual Studio Extension (additional install) would provide the same functionality.

What it boils down to is that ARIA is a set of additional semantic tags and attributes that describe and identify features for user interaction, how they relate to each other, and their current state.

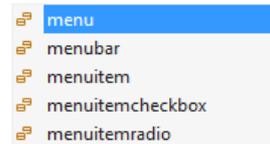
Often times this boils down to their roles, states, and functionality. Think of controls such as menus, trees, and grids.

All of these are pretty sophisticated controls that have a lot of dynamic-ness surrounding them.

### 1. Roles

Roles allow developers to specify what the purpose of each element is. These are then utilized by screen readers to deduce what each element is supposed to be. Without this role attribute the screen reader will have to attempt to parse the content and more or less guess what each element does. The image below takes a simple nav element, defines it as being a menu and then designates the child list elements to be menu items:

```
<nav role="menu">
  <ul>
    <li role="menuitem">Menu Item 1</li>
    <li role="menu"></li>
  </ul>
</nav>
```

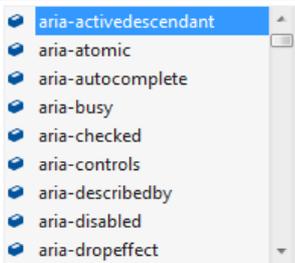


Notice how Visual Studio provides intellisense to aid in the definition of these elements. Developers can now quickly bring up the role property with intellisense and then utilize the same functionality to define the role of each of their elements – it cannot get easier!

## 2. States and aria-\*

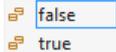
States in WAI-ARIA are meant to define the current state of the element. For example, when creating a treeview, there has to be a way to inform a screen reader whether or not a node is expanded or collapsed, which is exactly where these states come in to play. These are easily placed in an element by using the attributes that contain an aria-prefix.

```
<form id="form1" runat="server">
  <label id="firstLabel">First Name</label>
  <input id="firstName" type="text" aria- />
</form>
```



The image above shows Visual Studio 2012 in action and a large list of available options after aria- has been typed out. This new version of Visual Studio even provides the appropriate values these attributes accept:

```
<form id="form1" runat="server">
  <label id="firstLabel">First Name</label>
  <input id="firstName" type="text" aria-required="" />
</form>
```



## The Bottom Line

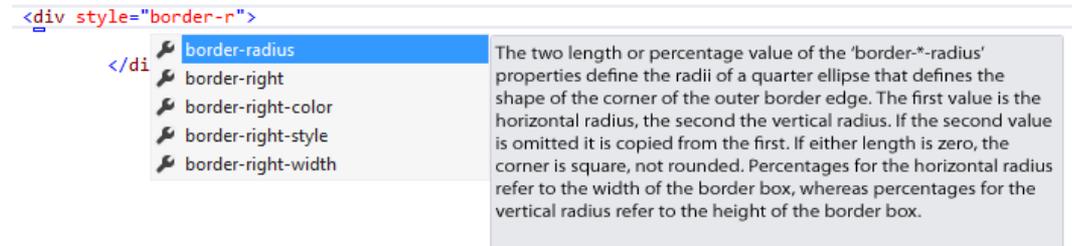
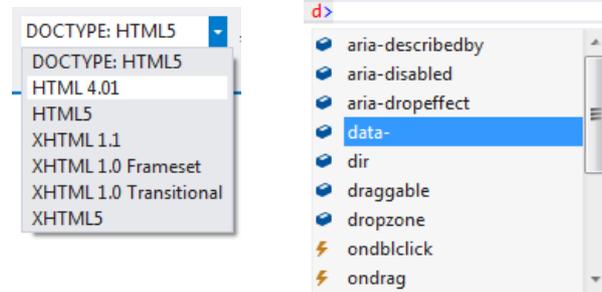
Accessibility is an often overlooked feature in modern web development. However, the new WAI Aria features in Visual Studio 2012 make it easy for developers to create accessible web applications. For an in-depth look at accessibility, and a complete example of Visual Studio 2012's capabilities take a look at this [blog post](#) 



### 3 HTML5 & CSS3 SUPPORT FOR MODERN WEB APPLICATIONS

In previous versions of Visual Studio, HTML5 and CSS3 validation were only available as a separate install. Many developers were unaware this add-on pack was even available. VS 2012 introduces built in support for CSS3 and HTML5.

These features can be accessed using intellisense, which helps take the guess work out for newcomers.



## The Bottom Line

All in all, VS 2012 makes it easier than ever to build web applications using these two emerging technologies!



## NEW CODE FEATURES FOR BETTER WEB APPLICATIONS

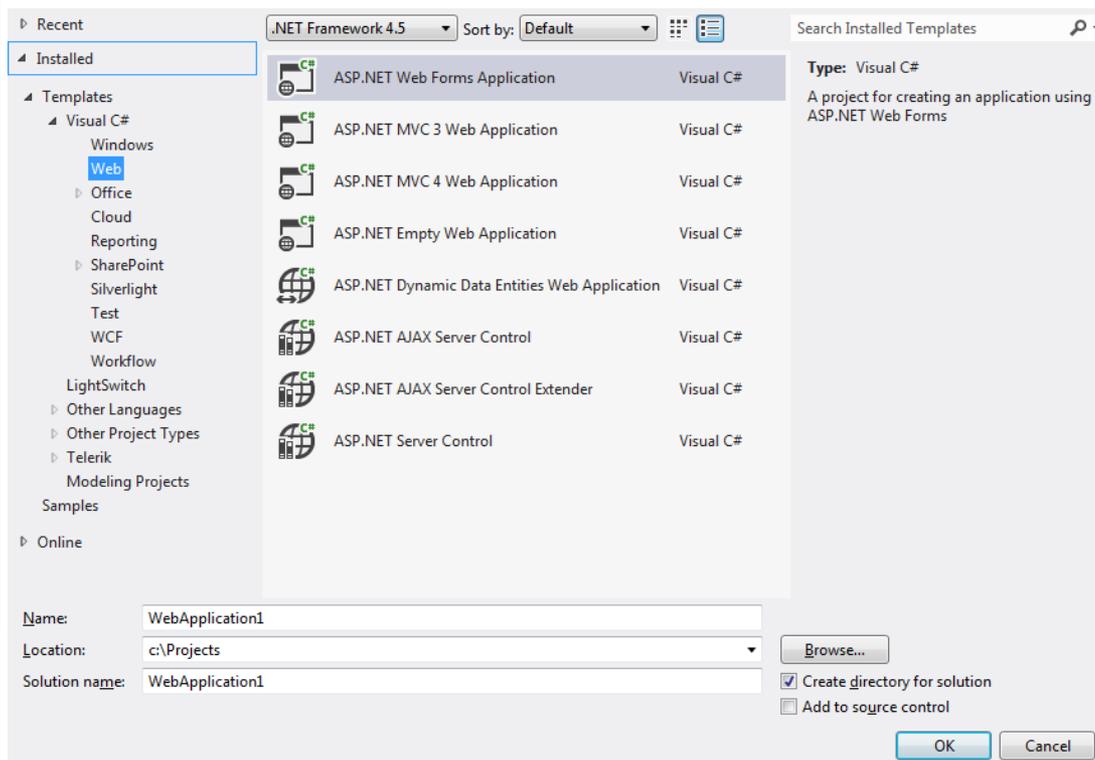
### 1 BETTER WEB APIS WITH ASP.NET WEBAPI

With the introduction of MVC 4 Microsoft has created a fantastic new way for developers to expose data via HTTP service. WebAPI is extremely easy to configure when compared to WCF services, and can send content in a multitude of formats. On top of that developers can easily extend it at almost any level. While WebAPI shipped with MVC 4, it can easily be used in all versions of ASP.NET!

#### • Getting Started with WebAPI

First create a new project.

For this example a standard ASP.NET AJAX project will work perfectly.



With the project created, it's time to add a few references. In the solution explorer, right click on the References folder and select Add Reference.

Add the following references:

**System.Web.Http.dll**  
**System.Web.Http.Common.dll**  
**System.Web.Http.WebHost.dll**

WebAPI is based on routes, which work like MVC routes. To make the service endpoints accessible, routes need to be configured in the application. To configure routes, open the Global.asax, and enter the following route in the Application\_Start method:

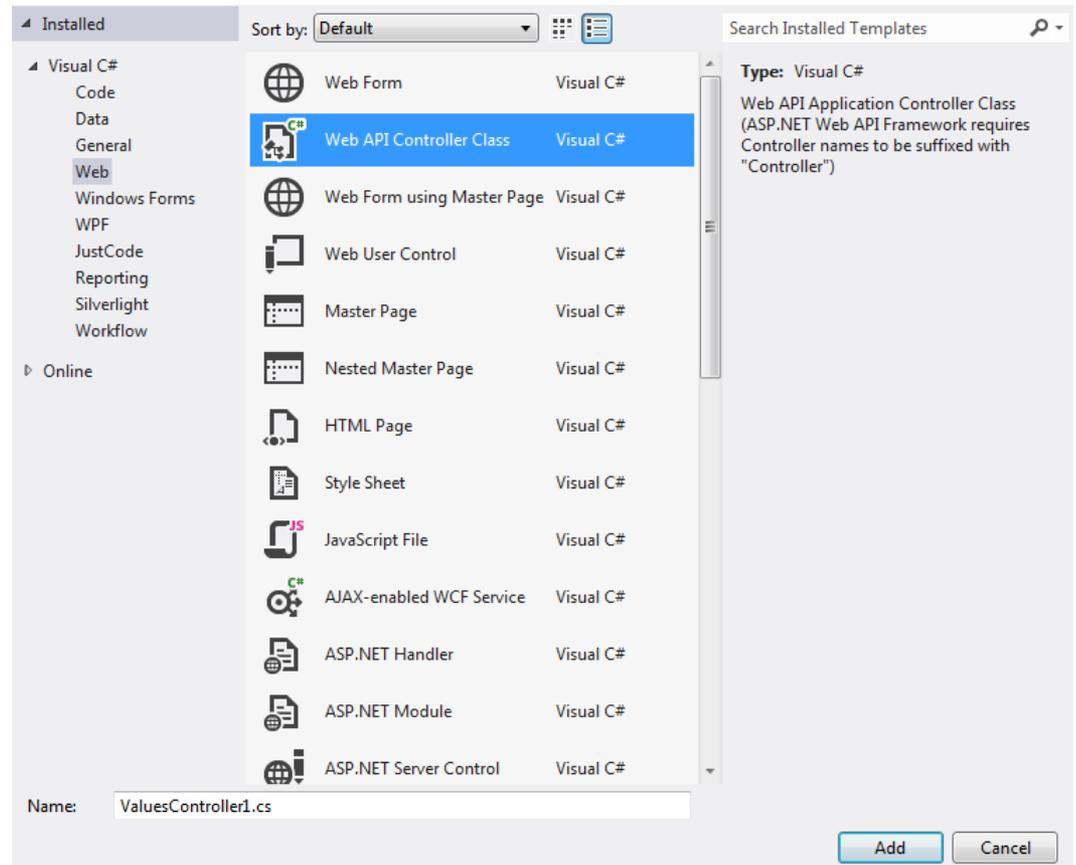
```
RouteTable.Routes.MapHttpRoute(  
    name: "DefaultApi",  
    routeTemplate: "api/{controller}/{id}",  
    defaults: new { id = System.Web.Http.RouteParameter.Optional });
```

**Note:** To keep Visual Studio from yelling at you, and to make the project compile, you will need to add using statements for: System.Web.Http, and System.Web.Routing.



If you have worked with ASP.NET MVC, this route should look very familiar. Basically, this route says: when a request comes in that starts with '/api', send it to {controller}, and if there is an id in the url, go ahead and pass it along too. With the route set up, it is time to add a controller to handle the specified route! In the solution explorer right click on the asp.net project, and select **Add > New Folder**, and name the new folder **Api**. Next right click on the new folder, and select **Add New Item**.

In the **Add New Item** dialog, select **Web API** controller, give the controller a name, and click **Add**.





Now create an entity to expose through the service. For this example, a basic customer entity was created:

```
public class Customer
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
}
```

**Note:** Make sure the object you want to expose has a parameter-less constructor! Otherwise there will be serialization issues.

In the controller make a new action method which returns an `IQueryable<Customer>` as shown here:

```
public IQueryable<Customer> Get()
{
    return repository.Get();
}
```

When an action method exposes the entity set as `IQueryable<T>`, WebAPI automatically allows the dataset to be queried using some of the OData operators. At this point the service is ready to be consumed in any application!

## The Bottom Line

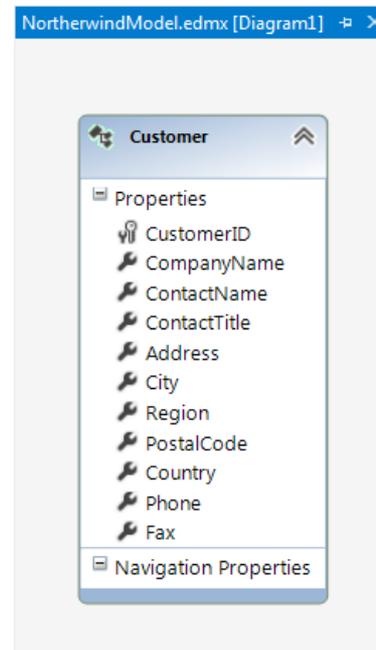
WebAPI is the perfect choice for modern single page applications, and other applications where resources need to be exposed over HTTP. It provides easy to configure routing, model binding, content negotiation, and querying via OData conventions. For a full walkthrough, make sure to take a look at this [introduction blog](#), and this follow up [blog](#).

## 2 STRONGLY TYPED DATA BINDING FOR RUNTIME CONFIDENCE

### • Why Strongly Typed?

Web Forms controls have always had the ability to be bound to fields found within the underlying data source in which they are bound. However this was, and still is, primarily done utilizing the Bind() and Eval() statements which rely on strings in order to do the binding, leaving the binding loosely typed. With the introduction of ASP.NET 4.5 there is a new ability to define exactly what type of object or data item the control is bound to, allowing these bindings to now provide intellisense for field and property names, in both one and two-way binding scenarios. This allows developers to not have to maintain an endless collection of strings or misspell field names when binding and overall reduce errors and issues that occur within the binding process.

For the examples below the following Entity will be utilized, which is simply a representation of the Customer table in the Northwind database.



## • Strong Binding

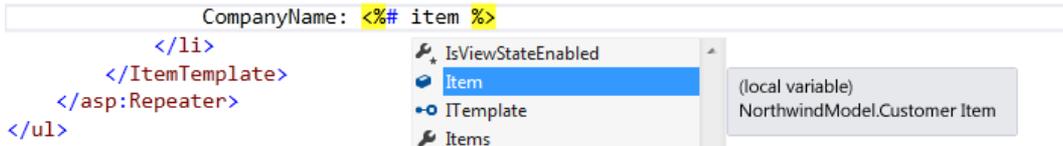
To utilize one-way binding, or any strongly typed binding for that matter, the new ItemType property of the databound controls (the Repeater or GridView for example) will be utilized. This property will be set to the specific entity of interest, so in our case it would be Customer. Keep in mind that if the web.config has not been updated to include the namespace of the application, this would be ApplicationName.Customer. This is the step that actually makes the control strongly typed, and the binding statements just take use of this prep work.

Once this has been completed the usage of the preferred binding statement can be placed anywhere within the control where the old Eval() and Bind() statements would be.

However, the entire statement will not need to be replaced as <%# still indicates the start of the binding statement, and %> ends it. For both one- and two-way binding it is very simple to bind to a field.

In one-way binding the Item object contains a representation of the current item, or object, that is being bound to the control.

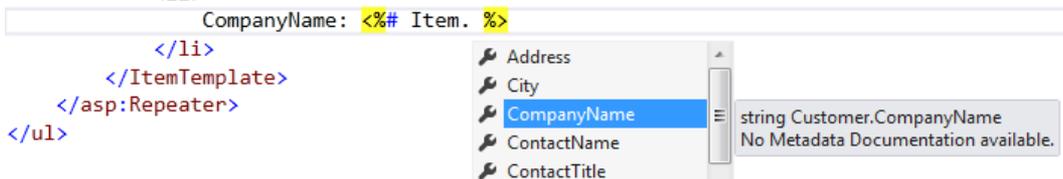
```
<ul>
  <asp:Repeater ID="myRepeater" runat="server" ItemType="NorthwindModel.Customer">
    <ItemTemplate>
      <li>
        CompanyName: <%# item %>
      </li>
    </ItemTemplate>
  </asp:Repeater>
</ul>
```



Thanks to intellisense it is easy to ensure that the Item object is properly accessed. As seen above, the additional details that are provided when hovering over Item indicates what type this object is.

Once the period key has been pressed intellisense will kick in once more and display the available fields off of the object being bound.

```
<ul>
  <asp:Repeater ID="myRepeater" runat="server" ItemType="NorthwindModel.Customer">
    <ItemTemplate>
      <li>
        CompanyName: <%# Item. %>
      </li>
    </ItemTemplate>
  </asp:Repeater>
</ul>
```



With two-way binding the approach is exactly the same, except that BindItem is used instead of Item.

The usefulness of this should immediately be apparent. Traditionally Eval() and Bind() statements were utilized, which would contain misspelled words and additionally keeping track of what fields were available to what models could also be a chore. ASP.NET 4.5 and Visual Studio 2012 solve these issues by providing a strongly typed control implementation.

## The Bottom Line

The new strongly typed controls allow developers to reduce runtime binding errors in their web applications. This leads to faster development, with a significant reduction in runtime errors. For a step-by-step tutorial on the strongly typed data control features take a look at this [blog](#)

### 3 LAZY REQUEST VALIDATION PREVENTS YoD

In previous versions of ASP.NET, if html and script tags were submitted as part of a form, ASP.NET would throw a so called "Yellow Screen of Death."

For example, if a form was created and submitted like this:

And a developer attempted to access the value like this:

```
var val = Request.Form["myTextBox"];
```

ASP.NET would become very unhappy, and respond with:

#### Server Error in '/' Application.

*A potentially dangerous Request.Form value was detected from the client (myTextBox="<script>").*

**Description:** ASP.NET has detected data in the request that is potentially dangerous because it might include HTML markup or script. The data might represent an attempt to compromise the security of your application, such as a cross-site scripting attack. If this type of input is appropriate in your application, you can include code in a web page to explicitly allow it. For more information, see <http://go.microsoft.com/fwlink/?LinkId=212874>.

**Exception Details:** System.Web.HttpRequestValidationException: A potentially dangerous Request.Form value was detected from the client (myTextBox="<script>").

#### Source Error:

[No relevant source lines]

**Source File:** c:\Windows\Microsoft.NET\Framework\v4.0.30319\Temporary ASP.NET Files\root\08b06130\88ea5be\app\_Web\_ftizwhj4.0.cs **Line:** 0

#### Stack Trace:

```
[HttpRequestValidationException (0x80004005): A potentially dangerous Request.Form value was detected
System.Web.HttpRequest.ValidateString(String value, String collectionKey, RequestValidationSource
System.Web.<>c__DisplayClass5.<ValidateHttpRequestValueCollection>b__3(String key, String value) +18
System.Web.HttpValueCollection.EnsureKeyValidated(String key) +9648053
System.Web.HttpValueCollection.Get(String name) +17
System.Web.UI.WebControls.TextBox.LoadPostData(String postDataKey, NameValueCollection postCollect
System.Web.UI.WebControls.TextBox.System.Web.UI.IPostBackDataHandler.LoadPostData(String postDataK
```



When developers wanted to allow HTML to be posted to the server, they were forced to tell ASP.NET to skip request validation altogether as shown here:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="Default"
ValidateRequest="false" %>
```

This had the undesirable side effect of leaving the page vulnerable to a variety of attacks.

ASP.NET 4.5 now allows developers to delay request validation until a posted value is actually accessed, and even then, only the requested item is validated.

In the above example developers can allow myTextBox to send HTML to the server by adding this to the web.config:

```
<httpRuntime requestValidationMode="4.5"/>
```

This tells ASP.NET to delay request validation until a request parameter is actually accessed. To access the value of myTextBox on the server ASP.NET now provides the Unvalidated collection on the request object:

```
var val = Request.Unvalidated.Form["myTextBox"];
```

This will bypass request validation, and retrieve the raw posted value from the request object!

## The Bottom Line

The new delayed request validation features, coupled with access to unvalidated request data, allows developers to protect their pages from vulnerability, but it gives the flexibility to get raw values when needed.



**Carl Bergnhem**  
@carlberghem



**#VS11 Awesomeness: Prevent those pesky yellow screens of death with #VS2012 lazy request validation [tldr.it/MBfMqZ](http://tldr.it/MBfMqZ)**

[Reply](#) [Retweet](#) [Favorite](#)

Liked this?  
 Tweet it!



## CATCHING BUGS BEFORE THEY CATCH YOU

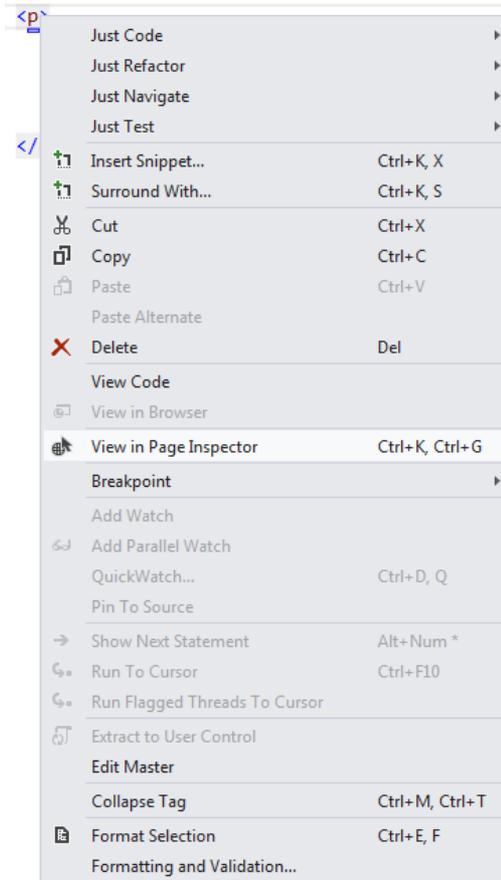
### 1 PAGE INSPECTOR PREVENTS LAYOUT HEADACHES

The Page Inspector is a new feature found in Visual Studio 2012 that bridges the gap between the development environment and the development tools found within modern day browsers. The inspector allows developers to see a rendered version of their page without the need of opening it in a browser first. Some of the tools provided include help with DOM navigation, finding specific elements, modifying the CSS of elements on the fly – and it is all done within Visual Studio! While this is not a replacement for browser developer tools it does grant developers additional aid within their IDE.

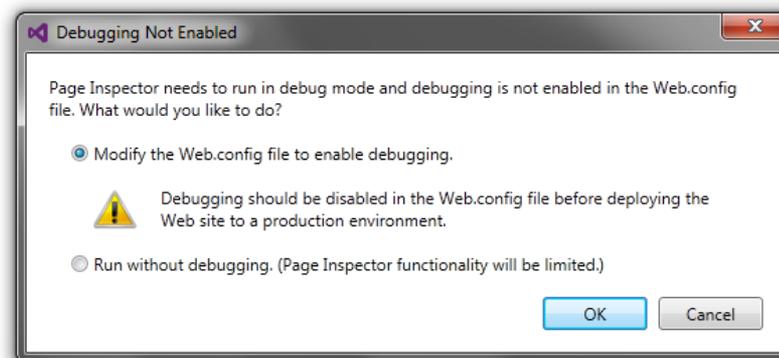
#### • Inspect Away

In order to start using the Page Inspector the easiest approach is to right-click on a particular element inside any ASP.NET WebForms page.

From the context menu that appears “View in Page Inspector” should be selected.

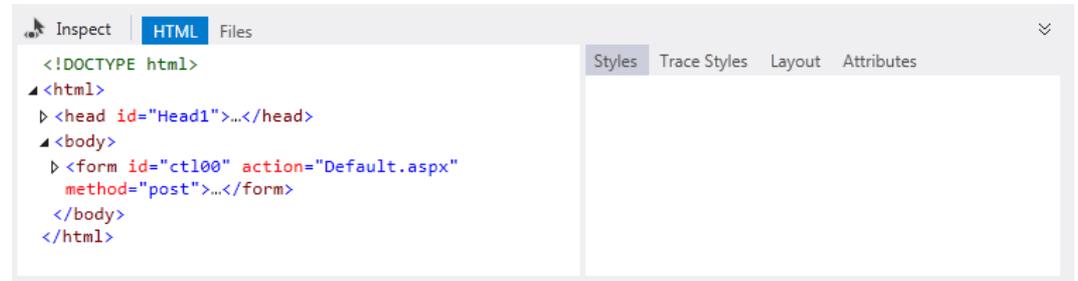


If this is the first time using the Page Inspector in a project, this dialogue will appear:





Once debugging has been enabled, the page will be displayed in a new dock within Visual Studio 2012 and will contain the Page Inspector Tools at the bottom



As the screenshot above shows, the inspector shares many features that can be found in any modern browser's developer tools. The left-hand side provides a view of the rendered DOM of the page, while the right-hand contains related CSS and attribute options related to any element that may be selected.

## The Bottom Line

Debugging layout and style issues in web applications is one of the leading causes of migraines for web developers. The new Page Inspector aims to reduce this migraine inducing problem by making it easy for developers, and designers, to troubleshoot HTML and CSS issues in their web applications without additional tooling. For a deeper dive in to all the awesome capabilities the Page Inspector provides check out this [blog post](#)



**Carl Bergenhem**  
@carlbergenhem



**#VS11 Awesomeness: Page inspector is just one of the migraine preventing features in #VS2012. See more at [tlrk.it/MBfMqZ](http://tlrk.it/MBfMqZ)**

[Reply](#) [Retweet](#) [Favorite](#)

Liked this?  
 Tweet it!



SO MANY FEATURES,  
SO LITTLE TIME!

This list just barely begins to scratch the surface on all the great new features Microsoft has packed into the latest release of ASP.NET, and Visual Studio 2012.

As you can tell this looks to be the best release yet!

## WANT TO SAVE EVEN MORE TIME IN VS 2012 AND ASP.NET 4.5.?

### Try RadControls for ASP.NET AJAX for 30 days!

The toolset offers 70+ feature-rich controls, 20 built-in skins (including one for mobile devices!) and productivity add-ons, all backed by Telerik's legendary support.

Telerik's controls can help you easily create awesome-looking projects, without having to worry about cross-browser compatibility, standards compliance or touch-device support - it's all taken care of! The result: You spend less time developing common functionality and more time focusing on the business logic of your app.

See for yourself by downloading a free 30-day trial  
with unlimited dedicated support.



 **Please don't print this Ebook**  
unless it's necessary.