Highcharts for ASP.NET MVC General Documentation

[System Requirements](#h.14vyrblc20m8)

[Server-Side System Requirements](#h.bqji4vklxuxa)

[Client-Side System Requirements](#h.igm6x64sxxbi)

[Installation](#h.w18hsdv5e9n7)

[Working with the demo](#h.4vi646sm1fbi)

[Adding Highcharts for ASP.NET MVC to your project](#h.914v2kwmm2wc)

[Visual Studio .NET](#h.l6xifewr7c55)

[Other IDEs / Web Projects](#h.xosb9kbd5qlj)

[Your First Chart](#h.2rquf6jndrw7)

[Adding Reference to the Chart in your View](#h.vl075ty4jvvo)

[Including the required Javascript files](#h.tgvunwfuj7nh)

[Defining the Chart in your View](#h.ywn7bir9k9x6)

[Getting Data for your Chart](#h.irhgrxk01p7r)

[Passing Data From the Controller to the Chart in the View](#h.d0pxywvcsivn)

[Using Fixed Data In The View](#h.vaexoilyc4vx)

# System Requirements

## Server-Side System Requirements

The **Highcharts for ASP.NET MVC** product has the following requirements:

* ASP.NET 4.x +
* ASP.NET MVC 4 +

If you wish to use the demos that ship with the product and open them directly in Visual Studio .NET, you need:

* Visual Studio .NET 2012 +

If you run the project in Visual Studio 2013 or 2015, just go through the auto-upgrade wizard. Versions prior to Visual Studio .NET 2012 are not supported.

## Client-Side System Requirements

Highcharts and Highstock work in all modern browsers including mobile devices and Internet Explorer from version 6. Standard browsers use [**SVG**](http://www.w3.org/Graphics/SVG/) for the graphics rendering. In legacy Internet Explorer (IE8 and before) graphics are drawn using [**VML**](http://www.w3.org/TR/NOTE-VML).

Highcharts and Highstock run on any server that supports HTML. You can even run Highcharts locally from a filesystem, since all the rendering is done locally in a browser.

**BROWSER COMPATIBILITY**

We test our software on many browsers using the latest versions. Knowing that Internet Explorer users have a tendency not to upgrade we also systematically test older versions of that browser. Highcharts runs on the following browser versions:

|  |  |
| --- | --- |
| **Brand** | **Versions supported** |
| **Internet Explorer** | 6.0 + |
| **Firefox** | 2.0 + |
| **Chrome** | 1.0 + |
| **Safari** | 4.0 + |
| **Opera** | 9.0 + |
| **iOS (Safari)** | 3.0 + |
| **Android Browser** | 2.0 + \*) |

\*) Android browser 2.x has limited support, see [**below**](http://www.highcharts.com/docs/getting-started/system-requirements#android).

**JAVASCRIPT FRAMEWORKS**

Our products rely on external JavaScript frameworks to do DOM manipulation, animation and event handling. We try to keep the interface between our library and the JavaScript framework as small as possible, the adapter classes that serve as a bridge to the JavaScript frameworks are about 10 methods long.

We have built-in support for jQuery and we bundle adapter classes for use with Mootools and Prototype. The strategy we have chosen is to always test the latest versions (possibly with a small delay) and to also include the latest patch on earlier versions. Currently our test-framework runs unit tests for the following versions of javascript libraries:

|  |  |  |
| --- | --- | --- |
| **jQuery** | **Mootools** | **Prototype** |
| 1.8.2+ | 1.4.5 | 1.7 |
| 1.7.2 | 1.3.2 |  |
| 1.6.2 | 1.2.5 |  |
| 1.5.2 |  |  |
| 1.4.4 |  |  |
| 1.3.2 |  |  |

Other versions may still work even though they are not tested.

**RENDERING ENGINES AND PERFORMANCE**

Different browsers support different rendering technologies, modern browsers have support for SVG but older versions of Internet Explorer do not, here we use VML to draw the graphs. This table describes the technologies we use in various browsers:

|  |  |  |
| --- | --- | --- |
| **Browser version** | **Rendering technology** | **Rendering performance** |
| Microsoft Edge | SVG | Excellent |
| Internet Explorer 9, 10, 11 | SVG | Excellent |
| Internet Explorer 8 | VML | Ok |
| Internet Explorer 7 | VML | Slow |
| Internet Explorer 6 | VML | Slow |
| Firefox | SVG | Excellent |
| Chrome | SVG | Excellent |
| Safari | SVG | Excellent |
| Opera | SVG | Excellent |
| iOS Safari | SVG | Ok |
| Android 3+ | SVG | Ok |
| Android 2.x | Canvas | Slow |

## 

**ANDROID 2.X**

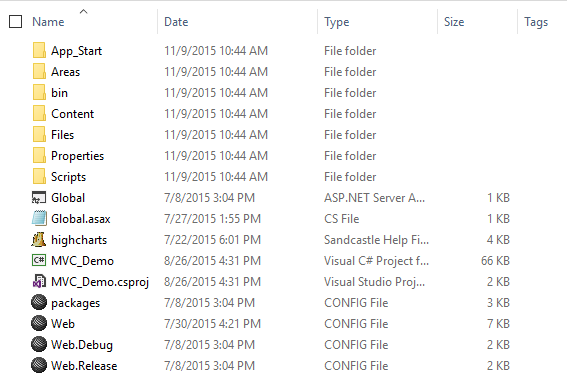
Android 2.x doesn't have SVG support built in, so we have created a separate renderer based on the canvg library for this system. This solution has some limitations:

* Shared tooltip is always enabled.
* During first render, the canvg renderer + rgbcolor.js + canvg.js (concatenated to one file) will be downloaded from code.highcharts.com This is configurable with the [**global.canvasToolsURL**](http://api.highcharts.com/highcharts#global.canvasToolsURL) option.
* Chart and series animation is turned off.
* Show/hide series from the legend is not enabled.
* Series and point touch events are not enabled.
* Zooming is not enabled.
* Using the Renderer API directly to add shapes to the charts is not supported.

# 

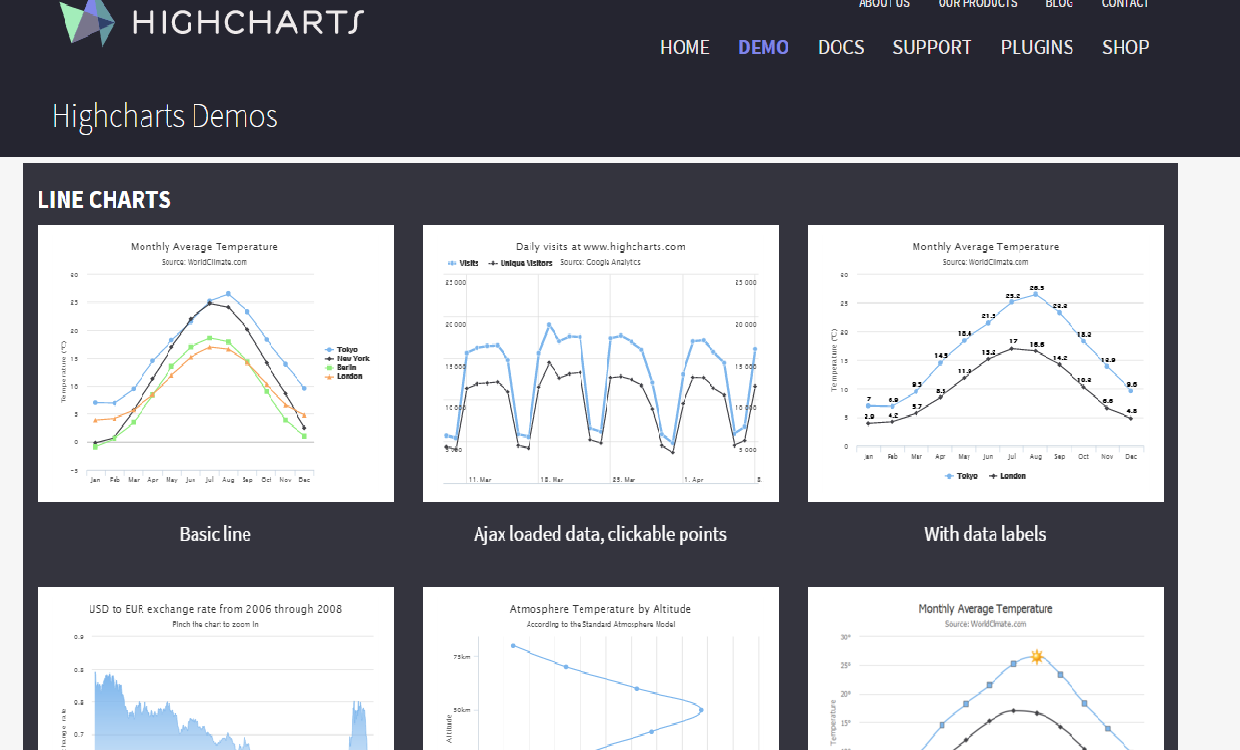
# Installation

Download and unzip the Highcharts trial .zip file (e.g. **Highcharts\_ASPNET\_MVC\_4\_1\_7\_0.zip**) to any folder on your local drive. The resulting file structure is the following:



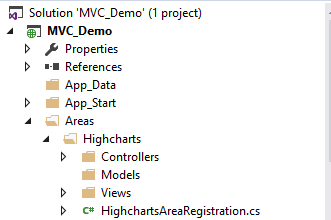
This is an actual ASP.NET MVC project that you can open directly in Visual Studio.NET. Just open the **MVC\_Demo** project file and it will open automatically in Visual Studio .NET. The project itself is a VS 2012 project, but it also supports both VS 2013 and VS 2015. Versions prior to VS 2012 are not supported.

After you load the project in Visual Studio, you can simply run it by pressing F5. The resulting screen is the start screen from our demo framework with more than 80 chart examples which you can run locally.

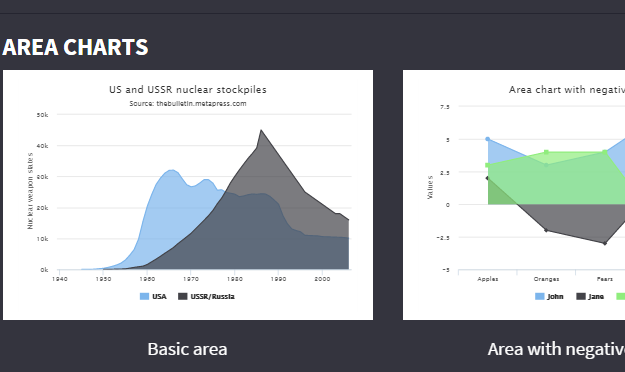


# Working with the demo

The demo contains all 80 examples from the demo framework. The actual location of the Demos source files (Controllers and Views) is under the ~/Areas/Highcharts folder.



There is one Controller in the Areas/Highcharts/Controllers/Shared folder and one View in the Areas/Highcharts/Views/Shared folder for each example. A typical example for that is the following sample chart - when you run the demo - one of the examples is the **Basic Area** example



The Controller and View files for the “Basic Area” example are located in the the following folders:

**Controller** - ~Areas\Highcharts\Controllers\Shared\AreaBasicController.cs

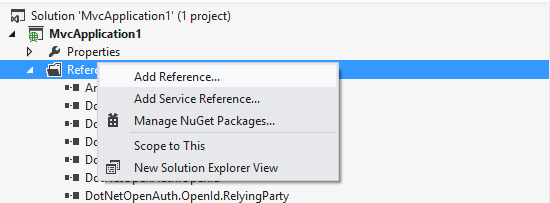
**View** - ~\Areas\Highcharts\Controllers\Shared\AreaBasic.cshtml

You can view the code for the example in both the Controller and the View and if you want you can change it locally to experiment with settings and custom code.

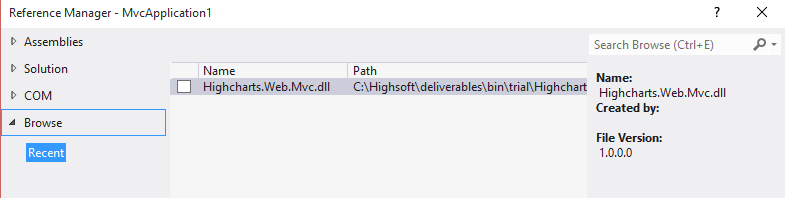
# Adding Highcharts for ASP.NET MVC to your project

## Visual Studio .NET

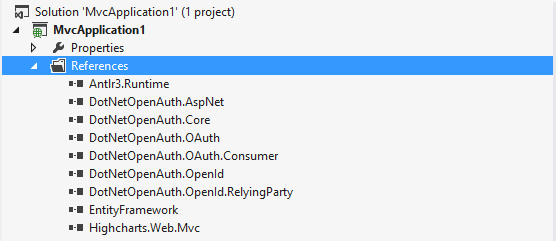
The first thing to know is that Highchatrs for ASP.NET MVC has both server-side (DLL) and client-side (Javascript) code files and both are needed for the proper functioniong of the chart. To add the server-side DLL library to your project, create a new ASP.NET MVC project and right-click the “References” section of the project, then select “Add Reference..”



After that, navigate to the **/bin** folder of the location where you have unzipped the Highcharts for ASP.NET MVC and add the **Highcharts.Web.Mvc.dl**l file to your project



After finishing the process, the Highcharts.Web.Mvc file should be included in your references.



## Other IDEs / Web Projects

If you are using IDEs different from Visual Studio .NET, make sure you find the equivalent of the “References” section described above or simply copy the **Highcharts.Web.Mvc.dll** file directly to your project’s /bin folder.

# Your First Chart

After you have added the correct reference to **Highcharts.Web.Mvc.dll** file to your project and the file is now available in the **/bin** folder of your application, you are now ready to construct your first sample chart.

## Adding Reference to the Chart in your View

The first step is to add a reference to the chart in the View of your sample page. This is done by referencing the Highcharts assembly using the following line at the top of the View (make sure you have added the reference to the assembly before that as described in the previous section)

*...*

*@using Highsoft.Web.Mvc.Charts*

*…*

## Including the required Javascript files

Highcharts for ASP.NET MVC is dependent on ***jQuery***, so you need to add a reference to the jQuery library to your project. There are many ways to do that - you can either download the jQuery library directly from [www.jquery.com](http://www.jqeury.com) or reference a global CDN that hosts it freely, e.g. Google or Microsoft. The example below refers to Google’s CDN

...

*<script src="*[*https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js*](https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js)*"></script>*

*...*

Once you have added the jQuery reference, you need to add the highcharts javascript reference to your View as well. Make sure that the jQuery reference comes first, as the Highcharts javascript code is dependent on it. To refer to the latest Highcharts version, you can directly reference it from Highsoft’s website, e.g.

...

*<script src="*[*http://code.highcharts.com/highcharts.js*](http://code.highcharts.com/highcharts.js)*"></script>*

*...*

Or you can download it locally and reference it from there. The current javascript files for the particular version are also available in the **/js** folder of the trial installation .zip file.

Some of the advanced chart types and features in Highcharts are not defined in the main “highcharts.js” file and are available as separate modules. Depending on the chart you create, you may need to add one or several of them to your page. Here is a complete list of the additional modules that are available:

...

*<script src="http://code.highcharts.com/modules/data.js"></script>*

*<script src="http://code.highcharts.com/highcharts-3d.js"></script>*

*<script src="http://code.highcharts.com/modules/heatmap.js"></script>*

*<script src="http://code.highcharts.com/highcharts-more.js"></script>*

*<script src="http://code.highcharts.com/modules/funnel.js"></script>*

*<script src="http://code.highcharts.com/modules/exporting.js"></script>*

*<script src="http://code.highcharts.com/modules/heatmap.js"></script>*

*<script src="http://code.highcharts.com/modules/treemap.js"></script>*

*<script src="http://code.highcharts.com/modules/drilldown.js"></script>*

*<script src="*[*http://code.highcharts.com/modules/solid-gauge.js*](http://code.highcharts.com/modules/solid-gauge.js)*"></script>*

*...*

Make sure you add the module you need after the jQuery and highcharts.js definitions, as they are dependent on both.

## Defining the Chart in your View

The chart itself can be defined in any place inside the View’s HTML definition, based on your page layout. To start defining the charts, use the following syntax:

*...*

*@(Html.Highsoft().Highcharts(*

*new Highcharts*

*{*

*Chart = new Chart*

*{*

*Width = 1087,*

*Height = 400,*

*Type = ChartType.Area*

*}*

*...*

*, "chart")*

*...*

If you are using Visual Studio .NET, you will get automatic auto-complete for all the options that are available to the chart. There are many available options - for setting the legend, tooltip, axis and various other aspects of your chart. Full documentation on all this options can be found online in the ***API reference*** of the chart or you can simply explore by using Visual Studio’s autocomplete.

The second parameter of the *Html.Highsoft().Highcharts()* directive is the ID of the chart - in our example “chart”, but this can be any valid identifier. This can be used to obtain a reference to the chart on the client-side and manipulate it with javascript, if needed. More info and sample code on the can be found online in several of our demos, for example the “Ajax Loaded Data, Clickable Points” example.

## Getting Data for your Chart

Once you have added a reference to the chart, added the required javascript references and have defined the visual options, you need to set data for the chart so that it can display the respective chart based on the datasource. This is done in the Series section of the chart View definition, for example:

*...*

*@(Html.Highsoft().Highcharts(*

*new Highcharts*

*{*

*...*

*Chart = new Chart*

*{*

*Width = 1087,*

*Height = 400,*

*Type = ChartType.Area*

*},*

*...*

*Series = new List<Series>*

*{*

*new AreaSeries*

*{*

*Name = "USA",*

*Data = @ViewData["usaData"] as List<AreaSeriesData>*

*},*

*new AreaSeries*

*{*

*Name = "USSR/Russia",*

*Data = @ViewData["russiaData"] as List<AreaSeriesData>*

*}*

*}*

*}*

*, "chart")*

*…*

In the example above, we have defined a chart with height 400 pixels, width 1087 pixels, the chart type is “area” and the data defined in the series comes directly from the Controller, using the **ViewData** ASP.NET MVC mechanism (you can also use **ViewBag** or define the data directly in the View). An example how to pass data from the Controller can be found below.

## Passing Data From the Controller to the Chart in the View

The last step is to actually get the data and pass it to the ***View*** definition of the chart. This is typically done in the ***Controller*** that is responsible for the ***View*** using the chart. Most typically, this would involve database query, but for the sake of simplicity, we are using fixed data to demonstrate how this can be done.

Just define the Controller for the View and make sure you create a List<AreaSeriesData> with values corresponding to the points you need. If you are using other chart types, e.g. “Line” instead of “Area”, the collection becomes List<LineSeriesData>. You can add any number of points to the collection and then set the ViewData for the respective item to the instance of the collection you need. This will later be used by the View and the chart itself to get the data, for example:

*...*

*public ActionResult AreaBasic()*

*{*

*List<double?> usaValues = new List<double?> {*

*null, null, null, null, null, 6, 11, 32, 110, 235, 369, 640,*

*1005, 1436, 2063, 3057, 4618, 6444, 9822, 15468, 20434, 24126,*

*27387, 29459, 31056, 31982, 32040, 31233, 29224, 27342, 26662,*

*26956, 27912, 28999, 28965, 27826, 25579, 25722, 24826, 24605,*

*24304, 23464, 23708, 24099, 24357, 24237, 24401, 24344, 23586,*

*22380, 21004, 17287, 14747, 13076, 12555, 12144, 11009, 10950,*

*10871, 10824, 10577, 10527, 10475, 10421, 10358, 10295, 10104 };*

*List<double?> russiaValues = new List<double?> {*

*null, null, null, null, null, null, null, null, null, null,*

*5, 25, 50, 120, 150, 200, 426, 660, 869, 1060, 1605, 2471, 3322,*

*4238, 5221, 6129, 7089, 8339, 9399, 10538, 11643, 13092, 14478,*

*15915, 17385, 19055, 21205, 23044, 25393, 27935, 30062, 32049,*

*33952, 35804, 37431, 39197, 45000, 43000, 41000, 39000, 37000,*

*35000, 33000, 31000, 29000, 27000, 25000, 24000, 23000, 22000,*

*21000, 20000, 19000, 18000, 18000, 17000, 16000 };*

*List<AreaSeriesData> usaData = new List<AreaSeriesData>();*

*List<AreaSeriesData> russiaData = new List<AreaSeriesData>();*

*usaValues.ForEach(p => usaData.Add(new AreaSeriesData { Y = p }));*

*russiaValues.ForEach(p => russiaData.Add(new AreaSeriesData { Y = p }));*

*ViewData["usaData"] = usaData;*

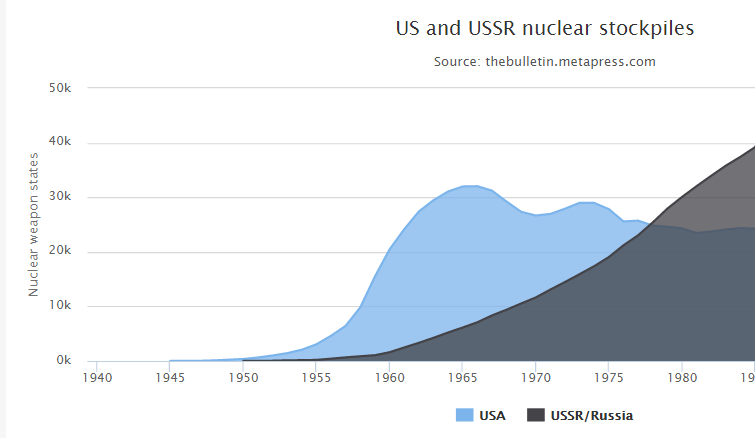
*ViewData["russiaData"] = russiaData;*

*return View();*

*}*

*...*

And this is the chart that is produced by this View and Controller:



## Using Fixed Data In The View

Using the Controller to pass chart data for the chart series to the View is the most common method of populating the chart with data, however you can also define data directly in the View, for example:

*...*

*@(Html.Highsoft().Highcharts(*

*new Highcharts*

*{*

*Chart = new Chart*

*{*

*Width = 1087,*

*Height = 400,*

*Type = ChartType.Column*

*},*

*Series = new List<Series>*

*{*

*new ColumnSeries*

*{*

*Name = "Brands",*

*ColorByPoint = true,*

*Data = new List<ColumnSeriesData>*

*{*

*new ColumnSeriesData { Name = "Microsoft Internet Explorer", Y = 56.3, Drilldown = "Microsoft Internet Explorer" },*

*new ColumnSeriesData { Name = "Chrome", Y = 24.03, Drilldown = "Chrome" },*

*new ColumnSeriesData { Name = "Firefox", Y = 10.3, Drilldown = "Firefox" },*

*new ColumnSeriesData { Name = "Sfari", Y = 4.77, Drilldown = "Safari" },*

*new ColumnSeriesData { Name = "Opera", Y = 0.91, Drilldown = "Opera" },*

*new ColumnSeriesData { Name = "Proprietary or Undetectable", Y = 0.2, Drilldown = null }*

*}*

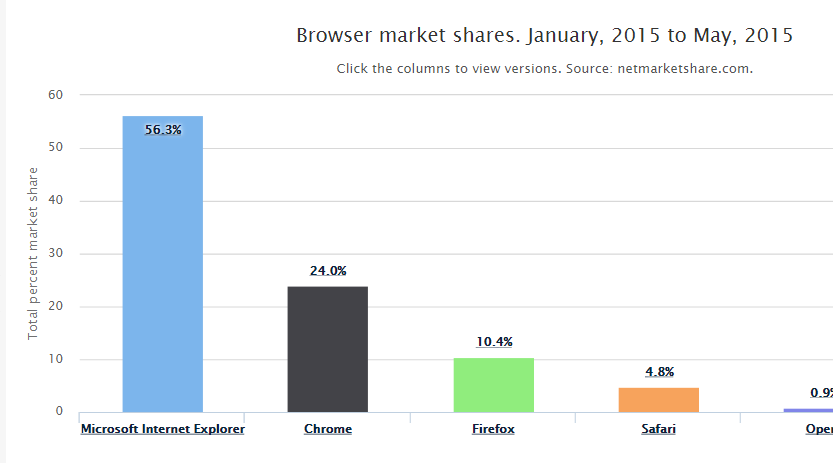
*}*

*}*

*...*

In this case we are creating a column chart with six columns, representing browser market share - each column with its name and value.

And this is the chart that is produced by this View:



# 

# Trial and Licensed Versions

The trial is a fully functional and feature complete version of Highcharts for ASP.NET MVC that expires after 30 days of usage. If you need additional time for testing, you can contact our sales department for trial extension. The licensed version of Highcharts for ASP.NET MVC is identical in every way to the trial with the exception that it never expires. The only thing that is needed to replace the Trial with Licensed version is to replace the trial Highcharts.Web.Mvc.dll file in your project with the licensed one.