**Preface**

A CAPTCHA (an [acronym](http://en.wikipedia.org/wiki/Acronym) for "Completely Automated Public [Turing test](http://en.wikipedia.org/wiki/Turing_test) to tell Computers and Humans Apart") is a type of [challenge-response](http://en.wikipedia.org/wiki/Challenge-response_authentication) test used in [computing](http://en.wikipedia.org/wiki/Computing) to determine whether or not the user is human.

CAPTCHAs are used to prevent [bots](http://en.wikipedia.org/wiki/Internet_bot) from using various types of computing services or collecting certain types of sensitive information. Applications include preventing bots from taking part in online polls, registering for free [email](http://en.wikipedia.org/wiki/Email) accounts (which may then be used to send [spam](http://en.wikipedia.org/wiki/Spamming)) and collecting email addresses. CAPTCHAs can prevent bot-generated spam by requiring that the (unrecognized) sender pass a CAPTCHA test before the email message is delivered, but the technology can also be exploited by spammers by impeding [OCR](http://en.wikipedia.org/wiki/Optical_character_recognition) detection of spam in images attached to email messages. CAPTCHAs have also been used to prevent people from using bots to assist with massive downloading of content from multimedia websites. They are used in online [message boards](http://en.wikipedia.org/wiki/Message_boards) and [blog](http://en.wikipedia.org/wiki/Blog) comments to prevent bots from posting spam links as a comment or message.

**Why we needed our own Captcha?**

The problem with using Recaptcha is:

1. Limited customization
2. Complex images and no-way to reduce complexity of image.

So, we needed a Captcha that is fully customizable and which frees us from depending upon the third party

.

**Proposed solution**

Solution was simple to create our very own Captcha that overcomes the major flaws with the Recaptcha. The main features of this Captcha are as following:

1. Customizable width and height.
2. Customizable Hardness level (character wise/ image distortion wise) .
3. Customizable length of Captcha characters.
4. Easily customizable Captcha design.

**Integration**

There are two ways our Captcha can be integrated in your site.

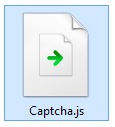
1. Using Captcha as a Service
2. Using Captcha as a User Control

To make Captcha image show up on your Page you need to follow these steps.

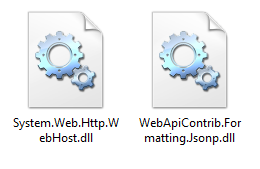
1. **Using Captcha as a Service**

For the simple client side pages the Captcha widget can be integrated using following steps.

Include the following files in your project.



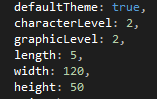
Apart from this the standard WebApi .dll’s must also be added to the references.



Now, in order to create the Captcha widget on your page, you will need to call the ‘**SetupCaptcha()’** on the element where you want to insert Captcha as follows.

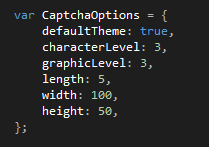


This will load up the Captcha widget with default settings which are as follows:



|  |  |
| --- | --- |
| defaultTheme | ‘true’ or ‘false’. True if you want the default css theme to be applied. |
| characterLevel(2) | {1, 2, 3}. Character wise hardness level of the Captcha.   1. Only numbers 2. Numbers and upper case alphabets 3. Alphanumeric (both uppercase and lowercase) |
| graphicLevel(2) | {1, 2, 3}. Level of image distortion   1. Plain colored background 2. Patterned background with characters on the same level 3. Patterned background with characters on different levels and background color and fore color of the same range. |
| length(6) | {4 to 8}. Number of characters present in the Captcha. |
| width(120 - 240) | Width of the Captcha image in pixels |
| height(50-100) | Height of the Captcha image in pixels |

If you want to customize the Captcha as per your requirements you can define the properties as given below as pass the parameter to the **SetupCaptcha(parameter)** method.

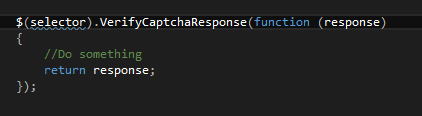




**NOTE:** In case the user passes invalid parameter values the Captcha widget will be generated with default values rather than producing error message.

The verification of the response will happen on the service side only and it will return true/false on correct/incorrect answer respectively.

For the response verification the user will have to attach the following function on the element.

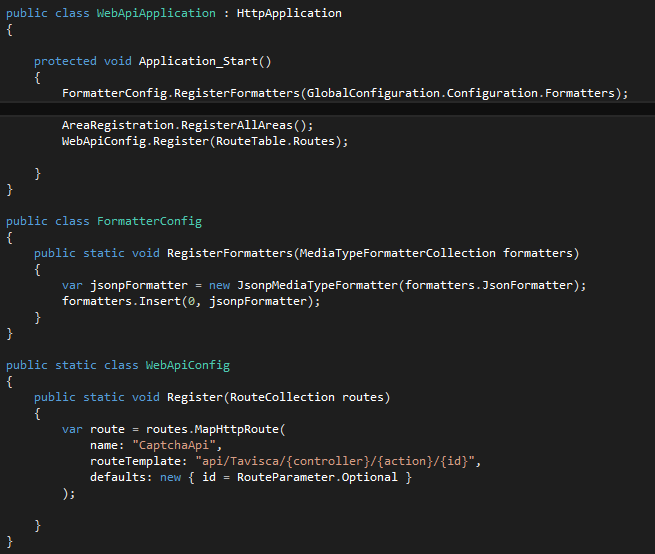
****

**NOTE:**  You need to define a callback function and pass it as a parameter to the ‘VerifyCaptchaResponse’ function.

‘response’ parameter of the callback function is either true or false based on whether Captcha validation failed or passed respectively.

Apart from this the project requires the Global.asax file for the Captcha service. If there’s already an .asax file present in the project then you can simply add the missing functions.

First thing that needs to be added in the global.asax is the ‘JsonpMediaTypeFormatter’ that will add Jsonp support in the webapi. Second is registering the route of the Captcha WebApi in WebApiConfig. The required changes can be easily referred from sample PlaygroundCaptcha.

****.

1. **Using Captcha as a user control**

For the server side pages the Captcha can be integrated as a user control.

Then integrating the Captcha widget is same as integrating the user control. First, register the control,



Second, define the control as follows.



For the response verification the following function needs to be called from the code behind.

