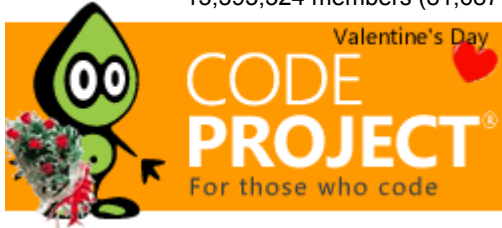



13,393,324 members (81,687 online)

Sign in 

[home](#) [articles](#) [quick answers](#) [discussions](#) [features](#) [community](#)
[help](#)

Articles » Web Development » Wiki.ASP.NET articles » General



__doPostBack function



ASP.NET Community, 12 Oct 2013



4.57 (19 votes)

Rate this:



Vote!

Hi everyone. Today I am going to talk about __doPostBack function, because there is some confusion in using __dopostback. You can see this

Editorial Note

This articles was originally at wiki.asp.net but has now been given a new home on CodeProject. Editing rights for this article has been set at Bronze or above, so please go in and edit and update this article to keep it fresh and relevant.

Hi everyone.

Today I am going to talk about the __doPostBack function, because there is some confusion with using this function. You can see this __doPostBack function in your ASP.NET generated HTML code.

The function takes the following two arguments:

eventTarget - This contains the ID of the control that caused the post back.

eventArgument - This contains any additional data associated with the control.

In any ASP.NET page the two hidden fields: __EVENTTARGET and __EVENTARGUMENT are automatically declared. When a page is posted back to the server ASP.NET inspects __EVENTTARGET and __EVENTARGUMENT values and this way it can decide which of the controls caused the page to be posted back and what is the event that has to be handled.

The value of the parameters eventTarget and eventArgument are stored in the hidden fields. The two hidden variables can be accessed from the code behind using the forms or params collection.

If we inspect the code of the __doPostBack function, we can see that it first sets the values of two hidden fields with the two parameters passed to the function. After this, the page is submitted back to the server. The ID of the control which causes the postback is stored in the __EVENTTARGET hidden field, so you can find the control which caused the postback.

```
<a id="LinkButton1" href="javascript:__doPostBack( 'LButton3','' )">LinkButton</a>
```

You can see the function call `__doPostBack('LButton3','')` in the href and the argument passed for `eventTarget` is "LButton3" which is the id of the link button control (EventSource)

Example

1. Add two hidden fields inside the form.

```
<input type =hidden name ="__EVENTTARGET" value ="">
<input type =hidden name ="__EVENTARGUMENT" value ="">
```

2. Add javascript under the Head tag.

```
<script>
function __doPostBack( eventTarget, eventArgument )
{
    document.Form1.__EVENTTARGET.value = eventTarget;
    document.Form1.__EVENTARGUMENT.value = eventArgument;
    document.Form1.submit();
}
</script>
```

3. Add two controls.

```
<a id="LButton3" href="javascript:__doPostBack('Button2','')">LinkButton</a>
<asp:Button ID="Button2" runat="server" onclick="Button2_Click" Text="Button" />
```

4. Add function in your cs page.

```
protected void Button2_Click(object sender, EventArgs e)
{
    Response.Write("Welcome to Student Academic Blog");
}
```

5. You also need some code in the code behind to capture the postback and fire the event. In the PageLoad method add.

```
if (Request.Form["__EVENTTARGET"] == "Button2")
{
    // Fire event
    Button2_Click( this, new EventArgs( ) );
}
```

This will capture the posted variable `__EVENTTARGET` and cause it to fire the event "Button2_Click". You can also pass an event argument along with the target in case you need to pass something to your code behind:

```
__doPostBack( "Button2", '<event argument here>' )
```

This would be captured in the code behind as `Request.Form["__EVENTARGUMENT"]`

So this is how you can use `__doPostBack`

Enjoy it

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)