**How postback works in ASP.NET**

**Def :** *a postback is an HTTP POST to the same page that the form is on. In other words, the contents of the form are POSTed back to the same URL as the form , in ASP.NET with the \_\_doPostBack() function and an application model that allows a page to perform validation and processing on its own form data. Postbacks are commonly seen in edit forms, where the user updates information in a form, hits save or submit, and the page is refreshed with a new form populated with the recently updated information.*

**we will take a closer look at how ASP.NET pages post back to themselves, and how to customize this feature in our web applications.**

***function \_\_doPostBack(eventTarget, eventArgument)***

**One of the most important features of the ASP.NET environment is the ability to declare controls that run on the server, and post back to the same page. Remember the days of classic ASP? We would create a form which would accept the user's input, and then we would most probably have to create another page that would accept all those inputs, either through HTTP GET or POST, and perform some kind of validation, display and action. Sometimes, even a third page was necessary to perform our actions. This wasted a lot of time and complicated things when you had to make a change. But of course, this is not necessary any more with ASP.NET. There is no need to create second pages that accept the inputs of the first, process them and so on. Form fields and other controls can be declared to run on the server, and the server simply posts the page back to itself and performs all the validation, display and actions.**

**When a control is declared to run on the server, a VIEWSTATE is created which remembers the ID of that control, and the method to call when an action is performed. For example, let's say we input this HTML on a page:**

***<html>***

***<body>***

***<form runat="server" id="myForm">***

***<asp:linkbutton id="Test" runat="server" text="Create Text file" onclick="Test\_Click" />***

***</form>***

***</body>***

***</html>***

**This is a very simple page. We declare only one web control, a linkbutton, to run on the server, with an ID of Test and we assign a method called Test\_Click to run when the link is clicked on the page. The linkbutton has to be wrapped inside a form that runs on the server as well. We save the above as an ASPX page and then we browse to it. The HTML that gets created looks like this:**

*<html>*

*<body>*

*<form name="myForm" method="post" action="test.aspx" id="myForm">*

*<input type="hidden" name="\_\_EVENTTARGET" value="" />*

*<input type="hidden" name="\_\_EVENTARGUMENT" value="" />*

*<input type="hidden" name="\_\_VIEWSTATE"*

*value="dDwtMTAwOTA0ODU1NTs7PsllCK1DzvZsK0J6OQ2dxKqmEwVs" />*

*<script language="javascript">*

*<!--*

*function \_\_doPostBack(eventTarget, eventArgument) {*

*var theform = document.myForm;*

*theform.\_\_EVENTTARGET.value = eventTarget;*

*theform.\_\_EVENTARGUMENT.value = eventArgument;*

*theform.submit();*

*}*

*// -->*

*</script>*

*<a id="Test" href="javascript:\_\_doPostBack('Test','')">Create Text file</a>*

*</form>*

*</body>*

*</html>*

**Our link calls the javascript function \_\_doPostBack when clicked (that's 2 underscore symbols in front of it). You do not write this function, instead it is generated by the ASP.NET engine and automatically included in our page. It submits the form to the same page, and accepts 2 arguments:**

**1 - eventTarget: the control doing the submission**

**1 - eventArgument: any additional information for the event**

**For example, our generated link button, is telling the javascript function that the control submitting the form is the one with ID=Test, and no further information is needed for the second argument. The javascript function is actually setting 2 hidden form fields with these 2 arguments: \_\_EVENTTARGET and \_\_EVENTARGUMENT. When the form is submitted back to the server, the server reads these 2 hidden form fields and decides what submitted the form and performs the necessary action. In this case, the server will determine that the linkbutton performed the action, and will execute the Test\_Click method.**

**Lastly, I should point out that at least one control needs to be set to Visible, for the server to generate the \_\_doPostBack function in our page. Even if we have numerous web controls declared to run on the server, but they are all set to Visible=false, then the javascript function will not be included and we will not be able to perform any actions. You can test this out, by changing the linkbutton source code to this and looking at the source code generated when it runs**

**This Lecture will be discussed at 11 march 2012 with god will**