ASP.NET Callback Control – How to Use the Callback Control

The ASPxCallback Control allows you to easily add AJAX-based callback technology to your website. The Callback class represents a non visual component that allows callbacks to be sent from the client side and processed both on the server and the client.

In this video, you’ll learn how to how to use the ASPxCallback control to retrieve and display the server time without initiating a postback.

1. I’ll start with an ASP.NET application.

2. I’ve already created a table and added a few controls to the page.

3. The first row contains an ASPxTextBox control.

4. The second row has standard ASP.NET TextBox control.

5. These will be used to demonstrate that the Callback Control works with both Developer Express as well as standard ASP.NET components.

6. Next, there is a button control which will be used at runtime to retrieve the server time and display it inside the text boxes.

7. Now let’s take a look at the page’s source.

8. Here you can see the markup for the ASPxTextBox.

9. In the next row is the standard textbox.
10. Note that since the standard controls do not provide a powerful Client-Side scripting mechanism like their DevExpress counterparts, we need to set an id for the table cell so that it can be accessed during runtime.

11. And finally, the code for the ASPxButton Control.

12. I switch back to design view.

13. From the Toolbox, I drag and drop an ASPxCallback component onto the page.

14. Before moving on, let’s take a look at the Client-Side members exposed by the ASPxClientCallback type.

15. For this demo, we’ll be using the `PerformCallback` method to raise the server-side `Callback` event.

16. We’ll use this event to retrieve the server’s time.

17. The data is then sent back to the client side using the `Result` property of the event.

18. Then the `CallbackComplete` event is raised to process the data that was returned from the server-side.

19. This data can be retrieved using the `result` property of the event.

20. Let’s return to Visual Studio.

21. I select the ASPxCallback component and specify a `ClientInstanceName` for it.

22. I’ll also do this for the ASPxTextBox control on the page.
23. I select the button control and disable its **AutoPostBack** property.

24. From its smart tag, I’ll launch the ASPxButton Client-Side Events Editor.

25. Inside the **Click** event handler, I make a call to the callback control’s client-side **PerformCallback** method.

   ```
   callback.PerformCallback();
   ```

26. To be able to process any type of information on the server, I select the Callback control and create a handler for its **Callback** event.

27. Here I’ll paste the following code.

   ```
   string time = DateTime.Now.ToString("T");
   ASPxCallback1.JSProperties["cpTime"] = time;
   StandardTextBox.Text = time;
   ASPxCallback1.JSProperties["cpTextBox2"] =
       ASPxCallback.GetRenderResult(StandardTextBox);
   ```

28. The first line retrieves the server’s current time in string format.

29. The time value is then passed on to the client using the **JSProperties** property.

30. This property enables you to declare temporary client properties. Once declared, these can be accessed on the client-side.

31. I switch to the ASP.NET page.

32. I invoke the callback control’s Client-Side Events Editor.
33. I’ll add the following code to the control’s client-side **CallbackComplete** event.

```javascript
document.getElementById('StandardTextBoxContainer').innerHTML = s.cpTextBox2;
aspxTextBox.SetText(s.cpTime);
```

34. This will first set the text of the standard text box control using the “StandardTextBoxContainer”
id that we specified for the table cell.

35. Then the ASPxTextBox control’s value is set.

36. I click OK to close the editor and run the application to see the results.

37. Upon clicking the “Show Server time” button, you can see that both textboxes are populated
with the current time of the server without the need to perform a postback.

Thanks for watching and thank you for choosing DevExpress!