Table of Contents

It's likely a Dynamic Control Issue, not ViewState!

- ***InfinitiesLoop***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 897004 Thursday, November 09, 2006 9:11 PM by InfinitiesLoop

To Everyone -- pretty much 100% of the questions I get about ViewState issues aren't really ViewState problems at all. It usually boils down to a dynamic control issue.

Just keep this in mind -- if you add a control to the tree dynamically, you must add that control again on the next request; it won't be there automatically. ViewState is responsible for the data in the control, not the existence of the control in the first place.

But don't confuse that with initializing the control. When you add a control dynamically you should initialize its "initial" values before adding it to the control tree. But the values you give it SHOULD NOT be dynamic! For example... there's a dynamically added label on your page that starts off saying "ABC". A button on the page changes its text to "XYZ" in the click handler. When you initialize the label just before adding it to the tree, you should always set its text to "ABC"! The fact the text was changed is being maintained in the label's internal ViewState -- no reason for you to try and remember it yourself. Rest assured, the label will say "XYZ" if that button was clicked during a previous postback.

- ***Gail Lampinen***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1342857Sunday, December 31, 2006 7:37 PM by Gail Lampinen

In response to a submit button, which is capturing a variable parameter on the page, I am dynamically creating (multiple) xmldatasources and dynamically creating gridviews using the xmldatasources, which have a checkbox item in one of the columns. I am querying multiple distributed databases to build the data, and the number of datasets returned varies, which is why I build the gridviews (and their underlying xmldatasources) dynamically. However, all is lost on a postback. I need to save these dynamically created controls to the viewstate (these are not created onload or oninit, but in response to a submit button ...) When a user presses another button, I need to access the checked records and use information from them, however the gridviews, along with their checked checkboxes, are lost on the postback. Is there an easy way to save the viewstate of the dynamically created gridviews?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1342897Sunday, December 31, 2006 7:47 PM by InfinitiesLoop

… First of all, ViewState of dynamically created controls is saved automatically. The fact the control exists on the page is NOT. There's an important difference there... because the issue you are running into has nothing to do with viewstate at all.

Second... just because you have a variable number of items to create, doesn't mean you have to do it dynamically. That's what a repeater is for! Why not put your xmldatasource and gridview inside the item template of a repeater? All you would need to do is databind that

repeater, and volia... all your gridviews and datasources would be created and automatically maintained by the repeater on your behalf.

Give it a shot, if you need help working out the markup details, I will need a bit more detail. Contact me through email if you need help.

- ***Reynold***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1414471Monday, January 15, 2007 12:07 PM by InfinitiesLoop

Reynold -- As long as you are loading the controls each request, they should retain their viewstate automatically. The fact they are loaded dynamically does not mean their viewstate is not maintained, it is. There's probably some other issue, unrelated to viewstate. Read the first paragraph of my article on understanding dynamic controls :)

See if you can boil down the issue into a really simple page and user control or whatever. Get it down to the minimum set of code that still duplicates the problem. Doing that may reveal the problem to you (because you'll remove some code and it will suddenly and mysteriously start working). If not, then you'll have a nice compact "repro" of the problem that you can easily post here or send to me via the contact form.

Either way, we'll figure it out...

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1453155Monday, January 22, 2007 10:00 PM by Reynold Tucan

Thanks for your help. After stripping my app down to the bare minimum, I noticed that the EnableViewState flag for the instance of the user control in the default page was set to false. The obvious things are always the hardest to find and debug! After setting this to true everything works as expected.

- ***mehmetserif***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 2047486Saturday, March 17, 2007 11:03 AM by mehmetserif

This is the property that i made to count the button clicks and it'll create file uploads at run-time according to its click numbers

```
public int ButtonClick
  {
     get
     {
        return (int)ViewState["ButtonClick"];
     }
     set
     {
        ViewState["ButtonClick"] = value;
     }
  }
```

and then i create the fileupload controls at run-time

```
protected void lnkEkle_Click(object sender, EventArgs e)
  {
     ++ButtonClick;
     for (int i = 0; i < ButtonClick; i++)
     {
        FileUpload myControl = new FileUpload();
        myControl.ID = "upload_" + ButtonClick.ToString();
        this.panel1.Controls.Add(myControl);
     }
     lblSonuc.Visible = true;
  }
```

then when i want to upload files by those auto-created fileupload controls i get an nullReferenceException

```
protected void Button1_Click(object sender, EventArgs e)
  {
        for (int i = 1; i <= ButtonClick; i++)
        {
           myUpload = ((FileUpload)pnlUpload.FindControl("upload_" + i.ToString()));
           if (myUpload.HasFile)
            {
              upload(myUpload.PostedFile.FileName);
              lblSonuc.Text = "basari";
            }
        }
  }
```

So i don't know what to do?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 2048719Saturday, March 17, 2007 4:56 PM by InfinitiesLoop

mehmetserif -- the problem is you are only creating the fileupload controls when the user clicks on the button to add one. If they click on any other button you do not create them, therefore they do not exist. You should read my articles on understanding dynamic controls.

To see what I mean about them not existing, add another button to your page that doesn't do anything. Add some upload controls a few times, then click that button. They will vanish. Controls created dynamically must be added every request, and you're only doing it as long as they click the add button. You will need to override LoadViewState and create them there. The 'add' button then will just add a single control as well as increment the counter.

Must Recreate Dynamic Controls on Postbacks

- ***Bryan***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 734192Thursday, October 26, 2006 6:10 PM by Bryan

I have a question regarding dynamic "Web User Controls" that contain Databound DropDownLists.

I have one UserControl being loaded dynamically depending on the number of times the user has pressed a button (could add infinite user controls to the page). Now, the above "user control" calls a second "user control" and loads it into a PlaceHolder on the first "user control" depending on the dropdownlist within the first user control. This can happen infinite number of times for each "1st User Control" on the page. The "second user control" has databound DropDownlists in it.

Now, I have the enableviewstate for all of these controls set to "True". I know, I know... bad me. But the funny thing is, the "1st user control" keeps its state. The "second user control" does not keep its state on postback, even though viewing the page trace, the viewstate for that particular dropdown control is the correct selectedvalue (it resets itself).

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 737716Friday, October 27, 2006 11:37 AM by InfinitiesLoop

Bryan -- I'm not 100% sure I understand the scenario. But it sounds like you must have some way of tracking which controls have been created. Remember that when you add a control dynamically, that control will vanish from the page on the next postback unless you add it back again. So you need to keep track of which selected values the user has clicked on, and then rebuild the list of user controls on postbacks.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 738178Friday, October 27, 2006 12:14 PM by Bryan

I'm having trouble trying to "track" the controls state. I'm kinda a newb at this type of stuff, and I get confused when something isn't quite straight forward. I do end up re-adding all of the dynamic controls to the page.

Here is a piece of code that I have the can iterate through all of my controls that have been added to the page (to tell me what I'm seeing). Maybe I could use something like this to help me track the control value's?

```
  Protected Sub ParseControls(ByVal c As Control)
     For Each child As Control In c.Controls
        If InStr(child.ClientID, "DropDownList", CompareMethod.Text) > 0 Then
           Dim RuleDropDownList As New DropDownList
           RuleDropDownList = child
           Response.Write(child.ClientID & " " & RuleDropDownList.SelectedValue &
"<br>")
        End If
        ParseControls(child)
```

```
    Next
End Sub
```

Maybe it would help you understand my situation if I sent you some examples of what I'm trying to do?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 746024Saturday, October 28, 2006 4:49 AM by InfinitiesLoop

Bryan,

So you are navigating through the controls in order to detect dropdown lists, and then writing out their selected value.

I don't know what problem you are specifically having, but I'll try to help as best I can if you give me more details. Start by sending me mail about the high level goals you have. Its likely you can avoid using dynamic controls altogether, and that greatly simplifies things. Honestly though I'm quite busy lately so I can't promise a speedy response :)

Understanding Binding

Retaining Data Changes through a Rebinding

- ***Rob***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4585234Monday, October 15, 2007 3:26 PM by Rob

Great Article but I must have missed something because I can't seem to get my viewstate to retain the data after postback

I have a repeater with some form fields in there and an add button to add a extra repeater item, If I press the add button the page reloads and add a extra repeater line but it will repopulate the data of the first line after postback any ideas?

```
<%@ Page Language="VB" EnableViewState="true" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"www.w3.org/.../xhtml1-transitional.dtd">
<script runat="server">
  Dim number_of_Parcels As ArrayList = New ArrayList()
  Protected Sub form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles form1.Load
    If Not Page.IsPostBack Then
      Session("PCount") = 1
      Create_Parcels()
    End If
  End Sub
  Protected Sub Add_Parcel_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Add_Parcel.Click
    Session("PCount") = Session("PCount") + 1
    Create_Parcels()
  End Sub
  Protected Sub Create_Parcels()
    For a = 1 To CInt(Session("PCount"))
      number_of_Parcels.Add(1)
    Next
    Repeater1.DataSource = number_of_Parcels
    Repeater1.DataBind()
  End Sub
</script>
<html xmlns="www.w3.org/.../xhtml">
<head runat="server">
  <title>Untitled Page</title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
  <asp:Repeater ID="Repeater1" runat="server" EnableViewState="true">
    <HeaderTemplate>
```

```
        <table style="width: 100%;">
        <tr>
          <td>Packaging Type</td>
          <td>Length x Width x Height</td>
          <td>Weight:</td>
          <td>Number of Parcels:</td>
        </tr>
      </HeaderTemplate>
      <ItemTemplate>
        <tr>
          <td><asp:DropDownList ID="Packaging" runat="server">
            <asp:ListItem>Your Packaging</asp:ListItem>
            <asp:ListItem>Express Envelope</asp:ListItem>
            <asp:ListItem>Plastic Pack</asp:ListItem>
            <asp:ListItem>Small Box</asp:ListItem>
          </asp:DropDownList></td>
          <td><asp:TextBox ID="Length" runat="server" Columns="5"
EnableViewState="true" ></asp:TextBox> X <asp:TextBox ID="Width"
runat="server" Columns="5" />
             X <asp:TextBox ID="Height" runat="server" Columns="5" /></td>
          <td><asp:TextBox ID="Weight" runat="server" Columns="5"
/> <asp:DropDownList ID="WeightType" runat="server">
            <asp:ListItem>lb</asp:ListItem>
            <asp:ListItem>Kg</asp:ListItem>
          </asp:DropDownList></td>
          <td><asp:TextBox ID="Parcels" runat="server" Columns="5" /></td>
        </tr>
       </ItemTemplate>
       <FooterTemplate>
         </table>
       </FooterTemplate>
      </asp:Repeater>
      <asp:Button ID="Add_Parcel" runat="server" Text="Add Parcel" />
  </div>
  </form>
</body>
</html>
```

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4585429Monday, October 15, 2007 3:36 PM by InfinitiesLoop

Rob -- this is not an issue with retaining ViewState. When you databind a repeater, you're telling it to start fresh. It throws away everything and binds the new UI to the data in the datasource.

If you want to save what was changed in the repeater's items you'll have to get them out and put that data into the datasource you're binding to, then add your new item and rebind it.

- ***Rony Klachko***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4624733Thursday, October 18, 2007 4:12 PM by Rony Klachko

… Using the designer I created a Page that contains a GridView databinded to an XMLDataSource. One of the GridView columns is an TemplateItem containing a dropDownList binded to an ObjectDataSource. The selectedValue of the dropdownlist is binded to the value of one of the gridview's columns. All of this is done by the at design time.

I encountered some problems:

1. While ViewState was enabled my page has loaded with a very very big _VIEWSTATE attribute even before touched any of the controls.

2. To avoid this and as I have no real need for the viewstate I have disabled it on both GridView and dropDownList and both of the DataSources. But than encountered an Exception on my first page postback. The exception was caused by an empty XmlDataSource, and seemed to happen before reloading the page. I temporarily solved that by assigning an event handler for GridView_Init and inside it assigning the XML to the DataSource. (but this solution causes the pulling of this xml from the server excessively).

3. Both before and after disabling the viewstate I had problems preserving the dropdownlist selected value after posting.

I hope I was clear about what I have done and what are my problems, and hope even mroe, that you can shed some light on my issues and help me solve them.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4732956Wednesday, October 24, 2007 4:00 PM by InfinitiesLoop

Rony -- without viewstate, the grid is going to re-bind on every request. That means for sure the datasource needs to be available every request. One way or another, the grid needs to repopulate, either from viewstate or from the data itself.

Also the fact it will rebind on every request means the dropdown is rebinding, too. Rebinding means losing the existing user state, such as dropdownlist selected index or textbox values. The trick is to rebind before these controls load the user state (the postback data), by doing it from OnInit. So you'd just call GV.DataBind() from OnInit rather than allowing the GV to bind itself automatically.

Databinding is Recursive

- ***Robert***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4801569Sunday, October 28, 2007 11:01 AM by Robert

… I know that it is possible to rebind a dropdownlist on every request in the Init event and the ddl gets the selected index. But I have the problem that this does'nt work in the ASP:Wizard Control - Why?

I have only a problem with some big (130 rows) dropdownlist resulting in a large viewstate - all other works fine. I only want to "remember" the last selectedindex.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4807137Sunday, October 28, 2007 6:27 PM by InfinitiesLoop

Robert -- are you sure nothing else is re-binding it later on? Remember that databinding is recursive -- if the DDL is within the Wizard and something calls DataBind on the wizard, it's binding the DLL too.

Specific Controls

CheckBox

- ***John***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 714357Monday, October 23, 2006 2:09 PM by John

… I've run into a strange problem when I create checkboxes dynamically. The checked state of the checkboxes seem to be reloaded fine on the page, but not until after Page_LoadComplete. The checked state always seem to evaluate as false before Page_LoadComplete. Any insight into would be appreciated!

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 715438Monday, October 23, 2006 4:28 PM by InfinitiesLoop

John -- are you creating them during the OnLoad phase? If so, thats why. Posted data is processed in two passes -- one before OnLoad, and one after OnLoad. The one after OnLoad is so that controls that are dynamically loaded in OnLoad can still load their posted data. However, that also means their state won't be up to date until the 2nd pass, so basically in your PreRender phase.

If you can rework your solution so they are dynamically created during OnInit it will solve your problem since they will load their state before OnLoad. If OnInit is too early because you depend on form state, you could do it in LoadViewState -- which is just before form data is processed.

DropDownList

SelectedIndexChanged Problem

- ***Shad***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1392825Thursday, January 11, 2007 11:24 AM by Shad

I have a dropdown list which I bind on the OnInit page and its view state is turned off.

The AutoPostBack property of the dropdown list is set to true.

Am running into a very strange situation:

When I change the drop down index the SelectedIndexChanged event is fired (not strange at all till now).

Now if I click on any other button that causes a PostBack on the page the SelectedIndexChanged event is firing as well(very strange).

Any ideas why this is happening and how could it be avoided?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1392854Thursday, January 11, 2007 12:15 PM by InfinitiesLoop

Shad -- First, the event is meant to fire whenever the index changes and there's a postback, not just when autopostback is enabled. The dropdown uses ViewState to remember what the previously selected index was. That is how it knows whether the index has changed. Since you have viewstate off, it will always think the previous index was 0, so it will fire every postback.

May I ask why you are using the SelectedIndexChanged event? There's probably a different way you can handle it to work around this issue.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1394257Friday, January 12, 2007 3:27 AM by Shad

Thanx for that information. Well on the SelectedIndexchanged event am using the selected value to retrieve some information from the database and display it on the page.

After what you explained I guess the solution would be to create a custom control derived from the dropdown and override the LoadControlState and SaveControlState events and save the selected value in the ConstrolState of the dropdownlist.

That way it will always remember the previous selected value which will avoid firing the SelectedIndexChanged on each postback of the page. Still I have to implement it to see if it works!! What do you think?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1395073 Friday, January 12, 2007 12:21 PM by InfinitiesLoop

Shad -- that would work, and you'd be able to reuse it for other purposes, so that's a plus. If you were just looking for a quick workaround though, it would be simpler for you to just remember the selected index on the page itself by storing it in viewstate.

Something like this:

```
protected override OnLoad(...) {

  object o = this.ViewState["index"];
  if(o != null) {
    int oldIndex = (int) o;
    if (oldIndex != ddl.SelectedIndex) {
      // changed!
    }
  }
  this.ViewState["index"] = ddl.SelectedIndex;
```

}

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1414488Monday, January 15, 2007 12:12 PM by InfinitiesLoop

Shad --

Certainly caching the data would work, even if the data is constantly changing within the database. But I'm not so sure that's the best thing to do. Devs tend to be afraid of database access for some reason... sure, its best to minimize the calls to the database if you can, but sometimes an extra call is more appropriate, and I think this may be one of those scenarios. If you cached the query from the first databinding, you'll be retaining all the data in memory "just in case" you need it later. But if you don't need it later, it was a waste. If the typical postback is not due to the dropdownlist changing, then you will be typically retaining data that you don't use. An extra database call might make the request slightly slower for that one user, but the server as a whole is better off.

That's just how it is on paper though... in reality, it depends on a lot of factors, so I can't say.

Rebinding

- ***Mithu***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1458644Tuesday, January 23, 2007 4:24 PM by Mithu

… started fixing the view state issues. Now I am struck with this problem. The article says that the selected value does not get reset because of disabling the view state but because of the drop down is being bound again. But if I check the selected value in any event handler, it is empty. The drop down is not getting repopulated at this point of time. If I enable view state, the selected value miraculously appears. Am I missing something here?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1458745Tuesday, January 23, 2007 4:50 PM by InfinitiesLoop

Mithu -- I'm still not clear on what is going on. You must assign the DropDown's data source and call DataBind on it every request if you are disabling ViewState. Not in response to an event not in only certain conditions, but always. Perhaps you can post some code that illustrates your problem? Ideally, boil down the code to the minimum set of code and markup that you can while still illustrating your scenario.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1458949Tuesday, January 23, 2007 6:03 PM by Mithu

I have an user control , say ParentControl in my page. ParentControl contains the search control and some other controls.

The search control has a drop down, text box and two buttons(search and clear). The drop down value comes from the DB. So I thought of disabling viewstate and bind the drop down

every time. Because the only time the page gets posted is when the search or clear button is clicked.

I disabled view state on the page , ParentControl and SearchControl.

Parent control contains

```
private void Page_Load(object sender, System.EventArgs e)

{
if(!IsPostBack)
{
-- code retrieve value from DB.set to variable in user controls --
SearchControl1.BindData();
}
}
private void SearchControl1_Command(object sender, CommandEventArgs e)
{
-- code retrieve value from DB.set to variable in user control --
SearchControl1.BindData();
}
```

SearchControl contains

```
private void Page_Load(object sender, System.EventArgs e)
{
if(!IsPostBack)
{
BindData();
}
}
public void BindData()
{
-- code to bind data to drop down, text box --
}
private void btnSearch_Command(object sender, CommandEventArgs e)
{
-- code to bubble up event --
}
```

If I try to see the value of my drop down and text box in btnSearch_Command, text box still has the value but not the drop down. Why? If I enable view state, I can see the selected value here. I dont know if this is even related to view state.

I checked the selected value in the client side just before the postback, it is fine.

I hope all this makes sense.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1459056Tuesday, January 23, 2007 6:22 PM by InfinitiesLoop

Mithu -- you say you will bind it every request, but then you have:

if(!IsPostBack)

That means you are not rebinding the data on postbacks. Since ViewState is disabled, the DropDownList's data will vanish after a postback. You need to call DataBind on it every single request, and you need to do it in OnInit. If you do it in OnLoad, the original problem you quoted me on comes into play, where the selected value is reset each time.

Please also keep in mind that when you call DataBind on a control, you are also in turn databinding all of that control's child controls. So if you call DataBind on a user control which contains a DropDownList, you have just databound that DropDownList, too. Many people don't realize DataBind is a recursive call, and they end up databinding things multiple times. Sometimes all that causes is slower performance, but sometimes it causes problems.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1463124Wednesday, January 24, 2007 10:27 AM by Mithu

Thanks for the reply Dave!

I do call the data bind every postback. first time in page load then from the parent control after the event bubbled up all the way to the parent. And the user control is nothing fancy just a text box, drop down and buttons.

Anyway I couldn't bind the control in Init() , because the pages are designed in such a way to handle all the db calls, the user controls only bind the retrieved data from the page. The child controls Init fires before the parent control and pages, I couldn't bind the drop down in the child control on Init because I don't have the data available to child at that point of time. So I realized my best bet is to enable view state in this case, unless I am missing some other obvious solution.

I appreciate the help and the article! It gave me a very good understanding of what to do and what not to do with view state.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1463468Wednesday, January 24, 2007 1:45 PM by InfinitiesLoop

Mithu -- "first time in page load then from the parent control after the event bubbled up all the way to the parent". Sorry, I'm still confused :) What if the postback is caused by something other than the bubbled event, like a separate button on the parent page? You will lose it. You need to call databind every request, from the same code. Why complicate it by splitting it into two places?

The fact you depend on state info to perform the binding does mean you can't do it from OnInit. That does complicate things. Unless your viewstate on that page is unacceptably large I wouldn't try too hard to work around it, just surrender yourself to it :)

I say that because it's much easier than the solution -- you would have to write a custom control for this scenario, or make the dropdown list a dynamic control. That would allow you to bind it later in the request but still disable viewstate... its just, not worth the effort unless you have a big viewstate problem on that page.

I can help you with that if you want to pursue it, just send me mail through the contact form so we can take it offline.

DropDownList within Dynamically-Loaded User Control: Contents not Restored on Postback

- ***Andy Johnson***

… I have a problem with a dynamically loaded user control and viewstate.

I understand about how and where to dynamically load controls and fast forwarding through events when attached to the control tree (and yes I do need dynamically loaded user controls).

If my user control contains a dropdown list with viewstate enabled and populated with list items when first created, then I would expect the list contents to be restored on postback after the user control is recreated (without having to reload the list contents). This does not happen and tracking the page/control events indicates that LoadViewState is called ok before the user control is recreated in the page load but is never called after the user control is created and attached to the control tree. This, of course, could have many repercussions for other dynamically created controls.

I know that creating the user control in OnInit would solve the problem but this is not always possible. Is there a way to 'force' the loading of view state for the user control after the main LoadViewState has been called ?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1562898Tuesday, February 06, 2007 1:33 PM by InfinitiesLoop

Andy -- don't worry, it's not a hole in the framework. ViewState is loaded for dynamically created controls in OnLoad, I can assure you of that. You said you only see LoadViewState once, but are you referring to the LoadViewState on the page only? Each control has its own LoadViewState, so you should be looking for it happening within the UserControl.

If you aren't seeing that either then there's another problem going on. It could be lots of things, like maybe the id of the control is different on the postback. I would need to see some code. Feel free to post code here if you can summarize it tightly enough, or use the contact form to send it to me privately.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1569904Wednesday, February 07, 2007 5:33 AM by Andy Johnson

I was tracking page and control LoadViewState and I could see the page LoadViewState called but not the control LoadViewState for the dynamically loaded control.

However, I have solved the problem. By placing the user control creation in an override page LoadViewState function after a call to base.LoadViewState(savedState) it all works OK.

I think the base call allows the page ViewState to be loaded and is, therefore, accessible but the recursive child control LoadViewState is called after the page LoadViewState returns by which time the dynamic controls are created and the ViewState is loaded for all controls.

```
protected override void LoadViewState (object savedState)
{
    base.LoadViewState (savedState);
    SetupDynamicControls();  // which can use page viewstate
}
```

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1579370Wednesday, February 07, 2007 10:20 PM by InfinitiesLoop

Andy -- viewstate for dynamically loaded child controls will be loaded even after Page LoadViewState. That plays into the "catching up" with the event cycle feature. It's a primary scenario that ASP.NET expects, so if it's not working there must be something else wrong.

Setup a small test environment (with a completely new project, to avoid outside influence). Try to duplicate your problem by creating a page and control that does nothing but override LoadViewState. Put some text in it too so at least you know its loading.

Populating a Grid Control According to Item Selected in DropDownList

- ***kurt***

I want to use an editable grid control on a page. The grid rows will be based on the selection in a drop-down. When the page first loads, the drop-down will have a default value, and the grid will be populated based on the default value.

If the user selects another value in the drop-down, I want the grid rows to reload (via callback), taking the user's selection into account (basically rebuilding the grid with a new DataSet). When the user presses the "save" button on the page, how can I get the data to save to the correct new dataset?

I'm not sure if this ViewState article is the right place to ask this question, but any help you could give me would be awesome.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1747964Wednesday, February 21, 2007 1:37 PM by InfinitiesLoop

kurt - sounds like you want a two-way binding, which you can do pretty easily with a datasource control. You can even make it work against a custom business layer using the object datasource control. This definitely isn't my strongest area :) But there should be lots of examples out there. Sorry I'm not more helpful than that.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1751804Wednesday, February 21, 2007 10:57 PM by Verinder

Kurt--I have done the same thing with ObjectDataSource and With two way DataBinding . What you need to do is in Button Type CommandName="Update" and use two way dataBinding with ObjectDataSource. I have done this with FormView not with DataGrid though but concept is same

Avoiding Putting DropDownList Contents into ViewState

- ***Graeme Elliott***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1756336Thursday, February 22, 2007 6:30 AM by Graeme Elliott

… I have a UserControl with a property ClassificationName that uses ViewState as the backing. I also have a DropDownList that uses the ClassificationName as a parameter when DataBinding. DataBinding occurs through a Click event handler that sets the ClassificationName property then DataBinds the DropDownList.

I'd like to avoid having the DropDownList writing its contents into ViewState, I would prefer to DataBind the list BEFORE ViewState tracking on the DropDownList kicks in on subsequent requests after the Click event handler has been called.

The obvious problem is that I don't have access to my ClassificationName property until after the Init phase on the UserControl by which time its too late to do this.

Is there an elegent way to achieve this? My best guess is that I need to defer initialisation of the DropDownList control by dynamically adding it to the page after the Init phase is complete and I have access to the ClassificationName property.

Any thoughts would be appreciated.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1760333Thursday, February 22, 2007 2:00 PM by InfinitiesLoop

Graeme -- yes, you are right that you need to delay initialization of the dropdown by creating it dynamically. I wish there were a non-dynamic way of doing it, but its your best bet.

What you should do is use a viewstate key to store the classification name which you have databound. Then in LoadViewState, after calling base.LoadViewState, you check for that key and if it exists, you create the dropdown, bind it, and add it to the page.

Then, its possible the click event comes through again later on in the request. So you would have to be sure to remove the old dropdown and create a new one. Give each a specific ID that is the same every request, or make the ID based on the classification name. It should work like a charm.

SelectedIndexChanged Firing Unexpectedly

- ***Sada***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 3430234Tuesday, August 07, 2007 9:18 PM by Sada

… I am having a AddNew Button which shows a panel with following controls with a ddl1,ddl2, a textbox and a save button. When i click on Addnew I populate the ddl1 with values and based on a session variable i get the selected value of the ddl1 (which i set in Page_Prerender using ddl1.selectedvalue=_value) and populate the ddl2 based on the selectedvalue.

The page is displayed fine. The user selects a value from ddl2 , enters some value in the textbox and click on Save button.

At this point it calls the SelectedIndexChanged event for the ddl1 which causes the selected value of ddl2 to go away. So My Save_Click event fails at it is unable to get the Selectedvalue of ddl2.

I am wondering why the SelectedIndexChanged event is fired.

Also one more thing, the same code works fine in asp.net 1.1 and does not work in asp.net 2.0

I have compared the aspx and the .cs files of old and new (converted) sources but do not find anything changed with respect to the ddl.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 3525698Thursday, August 16, 2007 1:57 PM by InfinitiesLoop

Sada -- sorry for the long delay in a reply. Doesn't sound like the event should be firing for your scenario, so there must be something in particular about the code causing it. Do you mind sending me some sample code?

It's not ViewState's job to track changes to non-Form Elements

- ***Losing Earlier-Added Listbox Item on Postback***

- ***Anil***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4758728Friday, October 26, 2007 2:38 AM by Anil

… I have the Attachement.Ascx control in the Mail.Aspx page, Which is Loaded in PlaceHolder on the Page_Load, which is in the Formview

In Attachement.Ascx i have the ListBox were I am adding the Items to the listBox on the button click,

for the firstTime the item is added it is visible in the listBox but when I click the Button to add the secound Item into the ListBox the firstadded item is lost on, OnPostBack.

I have tried with True/False for EnableViewState,

On every Post back the listBox items is Null.

- ***sanjeev***

http://weblogs.asp.net/infinitiesloop/archive/2008/02/08/Truly-Understanding-Viewstate.aspx
Friday, February 08, 2008 9:15 AM by sanjeev

… I have a dropdownlist which i am getting filled using AJAX.

now the problem is that at the time of pageload and when the postback of the page occurs, the list gets empty and on clicking of a button only it gets filled again.

how can i maintain the viewstate of this list and how can i have the list populated at the page load by the default options.

http://weblogs.asp.net/infinitiesloop/archive/2008/02/08/Truly-Understanding-Viewstate.aspx
Tuesday, February 12, 2008 1:55 PM by InfinitiesLoop

sanjeev -- changes to the page client side are going to be lost when the page refreshes. That's the way it is. It isn't ViewState's job to remember client-side changes to non-form elements. DDL is a form element, but only its selected value is posted by the browser, not its entire list.

You'll have to store the items in a hidden field and use it to repopulate the list when the page reloads. Or just reload it automatically. That or don't do it client side. :)

GridView

- ***teatime***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 3814573Monday, September 10, 2007 10:06 AM by teatime

… If you ever get the chance it would be useful to read an article specific to the GridView and ViewState.

It would be useful to know what is required to have a GridView with ViewState disabled, that is bound to a datasource (in my case entityspaces) but still allow for extraction of datakeys, sorting, paging etc.

Also to throw a spanner in, the GridView is part of a custom control, which means no access to the preinit event...

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5437947Tuesday, December 11, 2007 10:21 AM by teatime

InfinitiesLoop can you recomend any articles on getting a GridView to work without ViewState enabled? By work I mean, Paging, Sorting, Editing and Selecting rows... to complicate matters, the GridView is embedded in a custom control and most respond to events on another control.

The example I am working on is with my page being split into a Master/Detail format. The Master section contains a custom control that has fields in a form layout. The detail section contains an AJAX tab control, with each tab containing a custom control in the form of a list, using a GridView and here is where my hell began...

ImageButtons, CheckBoxes and ViewState

ImageButtons

- ***masonator***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 2057234 Sunday, March 18, 2007 2:39 PM by Masonator

Thanks for a fantastic article. Really helped my understanding.

I have a question though. In the last part of the article when dealing with dynamically generated controls, you suggest adding them like this:

```
public class JoesCustomControl : Control {
  protected override void CreateChildControls() {
     Label l = new Label();
     l.Text = "Joe's label!";
     this.Controls.Add(l);
  }
}
```

Which works perfectly for most controls. However, when I try adding Imagebuttons or Checkboxes with this method the Viewstate starts growing. For exmaple,

```
public class JoesCustomControl : Control {
  protected override void CreateChildControls() {
     Label l = new ImageButton();
     l.Text = "Joe's label!";
     this.Controls.Add(l);
  }
}
```

causes the Viewstate to grow.

Can anyone explain why this happens, or how to avoid it, as this is causing a huge viewstate in my current project.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 2057394Sunday, March 18, 2007 3:18 PM by InfinitiesLoop

Masonator -- ImageButtons use a part of page viewstate to register themselves as postback controls. Unfortunately there's no way to turn that off. The reason it needs to do that is hard to explain in a comment... its basically because the name of the posted value coming from the control in the form fields isn't the same as its ID, so it tells asp.net to raise its postback event anyway. The only way around this I'm afraid would be to write your own image control that doesn't need to do this.

You shouldn't see the same thing for checkboxes, except perhaps if you are hooking into the check changed event.

CheckBoxes

- ***masonator***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 2057525Sunday, March 18, 2007 3:53 PM by Masonator

… Imagebuttons aren't so much the problem as checkboxes. If you try the code above but add a checkbox rather than an imagebutton, the viewstate still grows.

The most annoying thing is that im fairly sure viewstate isn't even needed for the checkchanged event to work, as i have managed to get this to fire correctly even when manually setting the viewstate to the empty string. But then, of course, all the imagebuttons on the page stop working.

Any ideas?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 2057784Sunday, March 18, 2007 5:05 PM by InfinitiesLoop

CheckChanged may fire without viewstate, but it won't work the way you think. It will only fire if the state isn't the declared value. If you hit it, then switch it back again, the changed event won't fire because it's the same as the original value. The checkbox will save its value in viewstate automatically if you subscribe to the check changed event (unless its disabled). Are you using that event?

If so consider changing your design not to require the event. Usually it isn't necessary -- why do you need to know when it CHANGES? Usually just knowing what it IS is enough. In all my years I've never needed that event.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 2061735Monday, March 19, 2007 7:42 AM by Masonator

I am using the checkchanged event to cause an update panel to refresh...but actually I've just realised I don't need that because all I really want is AutoPostBack=True....

But I was trying things out on a checkboxlist hardcoded in at design-time and viewstate was still increasing. Do I have to manually disable the event? Or is it enough to simply not speciify a handler?

Again, I really appreciate your help as this is the first asp.net site ive worked on, and I think it could be a really nice site if I can sort out this one issue. To see an example of what I am trying to achieve, see

http://www.bringmeatakeaway.com/MenuViewer.aspx?ShopID=13&CatID=78

and play around with the menu items.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 2063519Monday, March 19, 2007 4:40 PM by InfinitiesLoop

Ahhh, yum!

The checkbox knows whether anyone is hooked into the event, so just not hooking it should prevent your viewstate growth. If that's not what you are seeing then grab a viewstate decoder and see what the difference is with and without a checkbox on the form.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 2068789Tuesday, March 20, 2007 11:28 AM by Masonator

Ok...I think im beginning to understand the situation.

The viewstate decoder tells me that every listitem I add in the checkbox list is being added to an array in controlstate called "__ControlsRequirePostBackKey__" and this is whats causing viewstate to grow.

Do you know how I can stop this happening? From what ive read so far, im coming to the conclusion that its impossible to prevent this information being recorded even though, in my situation, its unnecessary.

Any thoughts?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 2069249Tuesday, March 20, 2007 1:38 PM by InfinitiesLoop

Ok... I hadn't realized this (thank you). CheckBox calls Page.RegisterRequiresPostback, which puts it into that collection. I'm afraid there's nothing you can do about that. One thing that could help reduce the viewstate size in this case though is to make your IDs short. If you have a repeater inside a datagrid inside a gridview and inside that is a checkbox list, then well the naming container'd IDs get pretty long. Keeping the ids 2 or 3 characters long could have a big impact.

I don't know precisely why CheckBox calls that register method. It may have other purposes, but the purpose I know about for calling that method is when the post data key your control posts doesn't match its unique ID. For example an image button with id 'img1', when clicked on, will post to the server two keys, img1.x and im1.y (representing the x/y coordinates of

where in the image you clicked on). Since asp.net can't find a control with that ID it doesn't call its LoadPostData method. But if the control registered that it requires postback, it does.

Checkboxes use their uniqueID as the name which is posted so I don't see why they need to do that. There's probably some scenario I'm not thinking about. I'll dig deeper and see what I can find out.

You could also consider writing your own checkbox control that doesn't do that and see what you get.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 2069528 Tuesday, March 20, 2007 2:58 PM by Masonator

… I think I understand the situation now and am just as confused as you as to why it requires this information.

Seems crazy that you have no option to turn it off, either.

Repeater

Repeater with ViewState Disabled

- ***Sabih***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1463063Wednesday, January 24, 2007 9:33 AM by Sabih

I have a problem. I have a repeater containing controls and data populated through datasource. The problem is if I don't set the datasource and rebind on page.Ispostback then repeater doesn't call item_command method.

Moreover repeater renders nothing on postback if I don't rebind data.

I think this is something with viewstate. Viewstate is not retained for repeater's child control.

I want to avoid loading data again on post back.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1463435Wednesday, January 24, 2007 1:37 PM by InfinitiesLoop

Sabih -- I'll need more details or code samples. It sounds like you have viewstate disabled on the repeater. Keep in mind that disabling viewstate for a control also disables it for all its children, so you could also possibly have viewstate disabled on the control the repeater is in, or it could be disabled for the entire page.

Textboxes and ViewState

- ***Rekna***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 468328Sunday, August 13, 2006 7:03 AM by rekna

"ViewState is not responsible for the population of values that are posted such as by TextBox" DOT.NET 1.1 Suppose I have 2 user controls (let's call them G and F) on a page one containing a gridview one containing some textboxes. G is visible , F is not. When linkbutton detail in grid is clicked, G becomes invisible, F becomes visible and loads data (using databinding) for the selected row. Click save and G becomes visible, F invisible. Click New button on G, then G becomes invisible, F visible, there is NO databinding, so I would have expected empty textboxes in F, but previous values show up. These values could not have come from POSTing data, as the F control was invisible, and thus was not rendered. So values of textboxes on a control that is invisible are stored in ViewState ?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 469408Monday, August 14, 2006 12:27 PM by InfinitiesLoop

rekna -- yes exactly. ViewState is important in a textbox if you want its value to be maintained even if the textbox is not visible for a while. It's also important if you want to use the TextChanged event since the way it knows whether the event should fire is by comparing its posted value to its viewstate value. Without ViewState it will always think it has changed (unless the value is blank) and so will always fire the event.

That being said, its still primarily the POST nature of the control that maintains its value. Thats why when you disable viewstate, it will still recall its value on a postback.

In 2.0 (not entirely sure if this is true of 1.1 but I think so) -- the textbox tries to be smarter about whether viewstate should be utilized. If you don't hook into the TextChanged event, the TextBox isn't "disabled" (to appear greyed out), and it is Visible, then there's no reason to pollute viewstate, and so it doesn't... automatically.

Wizard, Tab and MultiView Controls

Preserving ViewState in Wizard-like Sequences

- ***kmobarra***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 591929Thursday, September 28, 2006 4:29 PM by kmobarra

… I have a problem that sounds similar to one discussed in the comments, but it's a bit different and I still can't figure an easy way to tackle it.

I have a wizard-like part in my application. Each "page" is actually a user control that is loaded dynamically, depending on what the user wants to accomplish. While these user

controls may be loaded and unloaded with each postback of the page, they not only have to persist their own data across postbacks, but also alter a data structure that is basically the big picture that they all are building (currently stored in an xml-driven object). Now the problem is two-fold:

1- How exactly can I reload the right control on each postback, with its latest state?

2- How should I store the "big picture" structure, so they all can access and alter it?

Thursday, September 28, 2006 6:17 PM by InfinitiesLoop

kmobarra -- Just keep in mind you must load the user control on every request. If you load the "next" user control when the user clicks "next", that's fine. But then on the next postback the control is gone and won't process the results, unless you load it again. To do that you can just remember which step you are in within this wizard, and make sure you load the appropriate user control. So imagine you are on step 1, and Step1.ascx is loaded. The user clicks 'next'. In OnLoad or LoadViewState, you realize you're on step 1 still, so you load Step1.ascx again. Then the next_click event fires. You remove Step1.ascx from the page (you probably put it into a placeholder), then load Step2.ascx, and update the step you are on. The next postback that occurs, Step2.ascx will load instead.

If you follow this pattern the controls will maintain their state automatically. There's no magic to be done.

As for them sharing the "big picture"... the driver of this page, whatever it is that is loading the controls, can provide to them via a property or method the data structure. It could do this after ever time it loads them dynamically. That way the controls will always be given to them the structure they need. Between postbacks of course, you'll have to have a way to save this structure. How best to do that depends on the nature of the document -- if its compact enough, and easy enough to covert to xml, you could convert it to an xml string and store it in ViewState. Or if you have some way of saving it to a database, you could do that too.

Tab Control Problem

- ***Darren***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1687447Friday, February 16, 2007 1:36 PM by Darren

I have a page with a tab control (from Telerik - more about that in a bit) which according to the selected tab, loads a dynamic user control into an asp panel. The tab control has a property tabstrip.selectedtab.id which I read in Page_load and via a simple couple of if statements, load the apropriate user control into the panel.

The controls are assigned a unique ID before being added to the panel.

When I click the first tab, the control is loaded fine and I can work with it as expected (it has a grid and a form), the postbacks are handled ok because the tab control maintains it's selectedtab.id and the control is reloaded on each page_load. However if I click the second tab the app hits an exception with 'Failed to load viewstate. The control tree into which

viewstate is being loaded must match the control tree that was used to save viewstate during the previous request. For example, when adding controls dynamically, the controls added during a post-back must match the type and position of the controls added during the initial request.'

This happens at the point where the user control attempts to load.

I can understand why this may happen, in that the viewstate for the first control in the panel was set by the control that was previously loaded and so does not match with the new control loaded in its place. I just don't know what to do about it!

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1709977Sunday, February 18, 2007 3:27 PM by InfinitiesLoop

Darren -- are you giving each user control a unique ID that is always the same? You could assign an ID which is based on the view it is visible in, for example. UC1, UC2, etc. That way at least the ID of the first control and the 2nd control are different.

Now, I'm not exactly sure how asp.net 2.0 handles this scenario. It could be that viewstate is loaded by index, in which case having a different ID won't matter. But if not, then it would. Even if it does load by index I believe there's a way to set it back into loading by ID.

Another solution would be to always load the user control that was visible on the last request. If the selected index has changed, then you then remove the control and load the right one. That may sound like a hack, but I would recommend that approach if you must load them dynamically. Controls just were not designed to be stripped out of the lifecycle mid way like that. By loading the old control first, its viewstate is loaded, and the 2nd control is going to get a fresh start just as if it were a new page request.

If you don't have many views, you could also always load them ALL every request. Then you just make them visible/invisible based on the selected index. That has the advantage of them being able to maintain state. If someone switches from Tab A to Tab B and then back to Tab A again, all the controls on Tab A will still have all the data they did before. A definite plus if it contains form data that the user may have typed things into.

Suppressing Creation of Controls in Currently Invisible MultiView views

- ***Verinder***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1700400Saturday, February 17, 2007 8:27 PM by Verinder

…I have one Question though regarding MultiView Control and ViewState.

I have Multiple User control one per view in MultiView. If I switch between Views Those Views are Bind (Say i have a Repeater) and shown. Now When i switch back View A from Visible to Non Visible Mode ItemCreated Event is called for Repeater Even though that View is Not Visible.I want ItemCreated to be called in PostBack Only When View IS Visible . Problem is If it keeps on calling ItemCreated It is a Performance Bottleneck If i Have Lot OF

Views in MultiView on Server side. IN The ItemCreated Event I Create Dynamic Controls back for PostBack If DataItem==Null.

How To Avoid Other Views not to do anythng when they are Not Visible??? Please Answer I am IN Trouble...

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1709946Sunday, February 18, 2007 3:20 PM by InfinitiesLoop

Verinder -- You could load the user control which is in the multiview dynamically. You'd only load the user control which should be visible.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1710922Sunday, February 18, 2007 5:56 PM by Verinder

…First of loading the control Dyanmically is not a good solution.I Did that but does require lot of work to make it work. It defeats the purpose of Wizard or Multiview, Then i can write Multiple pages to escape from that coding. If i have a Wizard With lot of steps and Controls are Bidded as you Go from one Step to other. Suppose i moved from Step 1 to step 5. Now at Step 5 Wizard will have a perfromace hit because ItemCreated Event of all step 1-4 even though i am o Step 5. I don't know even what is the use of calling Page_Load of each step in Wizard or Multiview When Only One Control is Visible. Looks like a Flaw in Microsoft design.

Please tell me some elegant solution without Dynamic controls.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1711131Sunday, February 18, 2007 6:22 PM by InfinitiesLoop

Verinder -- and what if the user is on Step 5 but needs to go back to Step 1?

The fact all the controls continue to exist is by design, because it allows them to (1) continue to maintain their state, and (2) you can access them when the wizard is finished in order to process the information. Controls in general can't just assume they are not going to be visible either, since it is quite possible that you (user code) may make them visible at almost any time. Even so like I said, data in invisible controls is still accessible, and that’s a good thing. Visible=false doesn’t mean "this control does not exist", it means it isn't rendered into the html, nothing more!

Since you seem not to need the control anymore you must be using the data as soon as they move to the next step -- why? Gather it all at once when they are finished, it would be simpler. If there's reason why you need to do it the way you are, rather than try to remove the control or something you could always just clear the bound data from them, so there are no items to create (controls.Clear, or re-bind it with no datasource).

You are worried about 5 controls going through ItemCreated events? Surely there are more important bottlenecks in your app. A complex asp.net page can have hundreds upon hundreds of controls on it and still perform extremely well, so if thats your only bottleneck you're in good shape. Sorry, it just seems like you're worried about the wrong thing.

The design of the framework may not fit 100% your scenario here, but that's how frameworks are. They meet a great many needs and to some degree allow customization for specific scenarios.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1725302Monday, February 19, 2007 10:14 PM by verinder

You are right that ItemCreated Event doesn't cause much of time. But what i have seen is ViewState of Page Keeps Growing as you Move from one step to other. It contains the ViewState of all the Controls in each Wizard step once they are bound. So It does increase the Size of page passing through the Wire even though the UI which you are seeing in the step is very simple. I was able to get round this problem by Enabling the ViewState of View Which i am going to show using StepIndexChanged Event. That solves the problem.

Thanks a lot!! My Manager was very worried about having multiviews and Wizards inside multiview can cause the page to slow down. Based on what you said I removed all the code from Page_load and Bind the UI which i am going to see on StepIndexChanged Event of Wizard . Along with that i set the ViewState of all Views set to False other then what i am going to show. That solves most of the Issue.

My Company is very worried about using 10-15 views in one page with each view has its own user control. Based on your opinion i will go ahead and push this Wizard design forward. Other school of though is why can't we do separate pages and write Wizard like functionality using page logic. My though was well then what is the use of Wizard if we can't use that.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1725349Monday, February 19, 2007 10:25 PM by verinder

…I did see that ItemCreated Event is not very heavy processing wise. What i have noticed is ViewState of page keeps growing and it adds the viewstate of each control on the page as you move step forward, and hence will be heavy on wire even if the view which is showing up uses 10% of hidden field which is out there. I Used EnableViewState of WizardStep to set the ViewState f step i am going to be in and disable all others. THis clears that issue.

I will try Controls.Clear solution also.

My Company(Managers) were very concern about 10-15 controls in one page and all doing heavy lifing work. I solved the problem by removing everything from there Page_load and Call Binding when we go to that step. Also Using ObjectDataSource Declerative DataBinding DataBinding is called by itself when view shows up.

ViewState and ItemCreated solution i resolved by creating WizardBase which Captures StepIndexChanged Event to disable all view state other then active step.

Thanks for telling me that it won't be performance issue because we were thinking to write separate pages to mimic Wizard.

Controls – General Issues

Setting Default Values for Controls

Dynamic Controls

- ***GSR***

Wednesday, August 23, 2006 2:17 PM by GSR

Hi! I've read your article, very interesting, here is my problem: i have an user control, which generates controls programmatically and initialize them in the Page_Load of the User Control. Now, when i try to generate the controls once again in the postback, the new values that i set are lost, the old values persist, the ones that i set first. The only way i found not to lost the new values, is creating the controls on the PreRender Event. What can i do?

Sorry for my english!!

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 479102Wednesday, August 23, 2006 2:59 PM by InfinitiesLoop

GSR -- Are the controls you are creating textboxes or other form controls (like dropdownlist)? You have to realize that these controls are designed to maintain the value entered by the user. The value you assign the control before you add it to the control collection is only meant to be a "starting" value. This can be illustrated if you declare a TextBox statically on the form and assign it a Text value. When the page first renders you see the value you gave it declaratively. But if you change the value as a user, then do a postback, your value remains, not the declared value. That is what it is _supposed_ to do. It sounds like you want to force the textbox value to your own value, potentially overwriting whatever the user entered, is that correct?

If that's the case, I highly recommend you change to creating the controls in OnInit. Then in Load you can assign the value you want, just as if it weren't a dynamic control at all. This works because by creating it in OnInit, you give the textbox the opportunity to load its posted value in the natural way (at the same time all the other controls do, before Load). When you don't create the control until Load, it misses that opportunity, and participates in the 2nd and last chance to load posted values which is after Load. Since its after Load its overwriting your assigned value.

In general its always better to separate the creation of a dynamic control and dynamic changes to it, ala OnInit and Load.

I hope that solves your problem... sounds like it should.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 482372Thursday, August 24, 2006 9:51 AM by GSR

Thanks!! Problem solved. Your suppositions were right. I had three textboxes and I wanted to overwrite their values.

Don’t Overwrite Attributes that are provided Declaratively

- ***Rahul Patel***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 568705 Thursday, September 21, 2006 6:47 PM by Rahul Patel

In solution 2 for problem 4 you give the following code:

```
public class DateTimeLabel : Label
{
 public DateTimeLabel()
 {
   this.Text = DateTime.Now.ToString("MM/dd/yyyy HH:mm:ss");
 }
}
```

Could this also be done like this?

```
public class DateTimeLabel : Label
{
 protected override void OnInit(EventArgs e)
 {
   this.Text = DateTime.Now.ToString("MM/dd/yyyy HH:mm:ss");
   base.OnInit(e);
 }
}
```

Would overriding the OnInit method work?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 569025Thursday, September 21, 2006 9:01 PM by InfinitiesLoop

Rahul -- you could do that, and it would work fine. But you'd be overwriting any text value that a user of your control provides declaratively, because declared attributes are assigned before OnInit. Doing it in the constructor is a way of providing a default value that can be overwriten by any of the usual means.

ViewStates of Disabled Controls

- ***Pierre***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 536194Monday, September 11, 2006 5:49 PM by Pierre

… I have viewstate question:

I am right to think that viewstates of disabled controls are not serialized ?

I have a Page that implements the IPostBackEventHandler interface.

When the user clicks a button, a javascript function is called. This function calls GetCallbackResult() that fetches data in a database based on the value of a textbox. In the callback function, I populate a list with the result.

Because I don't want the user to enter new data in the textbox before the result of the request, I disable it in the Javascript function and I enable it in the callback function.

I noticed that viewstates (in fact control states) are not posted to the server when the control is disabled so I don't have the value of the textbox in GetCallbackResult().

Thanks.

By the way, I solved my problem by passing the value of the textbox in parameter (RaiseCallbackEvent()).

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 536382Monday, September 11, 2006 7:03 PM by InfinitiesLoop

Pierre --

ViewState is maintained (and serialized if there is any 'dirty' data) so long as EnableViewState is true and ViewState is not disabled for any of its parent controls.

I think you really had double trouble with what you had -- first of all, by themselves, callbacks do not post the values of all the controls on the page at the time the callback is made. Instead, they post the values of all the controls as they were at the time the page was rendered. So if a TB has value "a" when the page renders, then the user changes it to "b", and then a callback occurs -- the server side will still see the value as "a". The 2nd part of the problem is that even if it did update the data that is posted, since you were marking it as disabled, it wouldn't have sent the data _at all_.

There is a way to reinitialize the callback data so that posted values are up to date with the latest changes by the user. However, your solution of passing the textbox value as a parameter to the callback is a much better solution if you ask me. I'd stick with that. It decouples your callback handling logics from the rest of the page.

If you really want to ensure the latest data is posted, calling the JS function WebForm_InitCallback() will do that for you. If you call that and then disable the TextBox, I bet it will work. I'd recommend sticking with the parameter solution though :)

- ***rspaz16***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 3499070Tuesday, August 14, 2007 3:57 AM by rspaz16

…. I have one composite control with one dropdownlist. The control has a property Enabled, she sets Mybase.Enabled to True or False. If Enabled is true, the dropdownlist saves her value correctly on a Postback, but if Enabled is false, the value is lost. Do u know what can the reason be?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 3504720Tuesday, August 14, 2007 12:34 PM by InfinitiesLoop

rspaz16 -- do you mind providing a small sample showing the problem? I don't think disabled controls post their value because the browser doesn't include them in the postback data. But even so, I think the DD should still restore its last value from ViewState.

Populating a Control according to User-selected value of a previous control

- ***Lord Fetta Cheesabio IV of Denver***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 542606Thursday, September 14, 2006 2:05 AM by Lord Fetta Cheesabio IV of Denver

… I'm having a bit of an odd situation with viewstate and would appreciate any feedback from anyone else reading this. Im generating dynamic controls with a method that gets the list of controls to generate from a database, something like this:

```
makeTheLists()
{
 //all the calls here are loaded dynamically from the DB, there isnt
 // actually two seperate lines of hardcoded calls to cretae the lists
 createAndloadCountriesList();
 createAndloadStatesList( getCountryListsSelectedValue()  );
}
```

now i've added an autopostback value to the country drop down list but the problem is that the state list is populated and created right after the country list was created and it needs the selected value of the country id. If I call 'makeTheLists' from my pages Init() method I can correctly output the selected value of the dynamically created country list in my PageLoad for example, and the correct country is selected when the screen refreshes BUT when the state list is being populated it is COMPLETELY unaware of the selected country value !!!!

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 542704Thursday, September 14, 2006 3:10 AM by InfinitiesLoop

Dear Lord Fetta Cheesabio IV of Denver,

Generally its best to separate the creation of controls from the injection of dynamic data into them. You can't do both at the same time because when you are creating controls, they nor any of the other controls on the page have been updated with the dynamic state of the page. So in other words, _don't_ depend on the country SelectedValue during the creation of the State list.

The pattern you want to follow is to (1) create the controls as early as you can (OnInit is a good start) and then (2) separately, load/populate/initialize/whateveryoucallit as early as you

can. In your case the earliest you can create the controls is OnInit -- but you can't populate the state list yet since the country list hasn't been updated with its posted/selected value. That happens by OnLoad, so that is where you must populate the state list, using the selected value. But you must take care to only do this initialization when the country value changes (if you rebind it every time you'll revert its selected value to the first item). Since the list will use viewstate to remember the items that were databound to it, all is well. But going on --- if you are populating the list manually before adding it to the control collection, or if you have disabled viewstate on it, then it wont remember them all you'll be forced to populate it all the time. To avoid the problem of overwriting the posted/selected value in this case, avoid creating it until OnLoad as well. By delaying its creation you stall it from loading the posted value until after OnLoad. And finally -- the next problem you'd have is that in this case, after switching countries, the list will attempt to load a posted value which doesn't exist in the list and so will throw an exception. To avoid that, you give the list an ID that is based on the country list's selected value. That way when the country changes, the new state list has a new identity and won't try to load the previous lists' posted value.

Whew... I kind of rambled there but I hope it sheds some light on your situation. You picked a sort of complex scenario to have a problem with.

The real underlying problem if you ask me is that you are creating the controls dynamically. If you make them static again via a Repeater, all the issues I just rambled on about go away. Doing things dynamically when there's a static way to do it just makes things more complex than they need to be.

...And its problems like these that have motivated me to write the articles I'm working on about dynamic controls. If you have a problem like this or any problem with Dynamic Controls (even if you think the problem is with ViewState), please read them.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 545606Thursday, September 14, 2006 10:28 PM by Enlightened Lord Fetta Cheesabio IV of Denver

thanks for the comments... I ended up splitting the code a bit as you suggested and calling them manually from Page Load and oninit which made things easier to handle. The extremely flexible functionality I needed meant that I had to use dynamic controls (unless there's something about Repeaters I don't know).

Disabling ViewState on a Control disables it for all its Children

- ***Ruben Cordoba***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 975144Tuesday, November 21, 2006 2:23 AM by Ruben Cordoba

Because disabling viewstate on a control disables it for all its children too, I have changed the property EnableViewState from False to True on the GridView on page Animal.aspx.

However the dynamic user child control AnimalPictures.ascx doesn't persist the viewstate neither. An so, the DataList1_ItemCommand is still not fired.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 976202Tuesday, November 21, 2006 4:41 AM by InfinitiesLoop

Ruben -- it sounds like there's more going on than you describe. Is the AnimalPictures.ascx the control that contains the Grid View or the other way around? Are you saying viewstate is disabled on the usercontrol as well? You don't necessarily need viewstate enabled to do what you need to do -- itemcommand really has nothing to do with viewstate. Can you provide more details or code samples?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 994433Wednesday, November 22, 2006 2:23 AM by Ruben Cordoba

InfinitiesLoop -- You can find details and code samples about what I am talking on a previous comment in this blog on Saturday, November 11, 2006 5:35 PM by Ruben

By the way, you said that ItemCommand really has nothing to do with viewstate. I don't think so. I feel that viewstate is the cause of the problem and ItemCommand is not fired because viewstate is not persisted on the dynamic user control called AnimalPictures.ascx.

I have tried different combinations turning on and off viewstate on the GridView and the dynamic user child control. Neither of them work propertly. Let me know your opinion and suggestions.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 999329Wednesday, November 22, 2006 1:37 PM by InfinitiesLoop

Ruben -- you were correct that turning on ViewState for the GridView, because it applies to all children.

But now that you are utilizing ViewState on the GridView you should not databind it every request. What is happening is the controls are being recreated (automatically for you) when ViewState is loaded, but then you DataBind again, which recreates them all -- and the newly created controls have a different identity (autogenerated ids), so the 'button' you click on isn't the same button.

Put a !IsPostBack around the databind call, just like you have in the user control, and I don't see why it shouldn't work.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 999715Wednesday, November 22, 2006 2:34 PM by Ruben Cordoba

Putting a !IsPostBack around the databind call on Animal.aspx.vb doesn't work. The reason is the following:

If I use !IsPostBack around the GridView databind call, when the page is posted back, the GridView is persisted thanks to the viewstate is loaded. Until here, everything works property. However, because the GridView is not being re-created (automatically for me) the

GridView1_RowDataBound is not fired, and so, I don't have the opportunity to re-create the dynamic user child control called AnimalPicture.asc when the GridView is populating the first row. Because the dynamic user control is not re-created after postback, the Page_Load event handler on AnimalPictures.ascx.vb is not fired neither. And so, the DataList1_ItemCommand is still not fired.

If you have another idea, I am ready to test it.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 999892Wednesday, November 22, 2006 2:50 PM by InfinitiesLoop

ItemCreated is called every request regardless of whether it is being databound or not. That is the correct place to do things like what you are doing.

Curious -- why not simply declare the user control in the template rather than load it dynamically? Then you wouldn't even need this event. You could even set that HiddenField property on it using the proper Databinding syntax HiddenField='<% Eval("AnimalID") %>'

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1000081Wednesday, November 22, 2006 3:07 PM by Ruben Cordoba

You said: why not simply declare the user control in the template rather than load it dynamically?

I don't understand what you mean. Can you explain to me your idea with more details, please?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1000147Wednesday, November 22, 2006 3:15 PM by Ruben Cordoba

The reason to load the user control dynamically is because I need it only for the first populated row populated on the GridView. If I declare the user control in the template, I think, it is not very efficient to have a user control on each GridViewRow when it is only used once.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1001014Wednesday, November 22, 2006 5:11 PM by Ruben Cordoba

Infinities Loop -- As you suggested me, instead of using GridView_RowDataBound as the place to re-created the dynamic user control, I have used GridView_RowCreated this time. Debugging the code, I can see that when clicking on the LinkButton, the dynamic user control is re-created within the GridView_RowCreated property after postback, and also, the Page_Load event handler on AnimalPictures.ascx.vb is fired. However, the DataList1_ItemCommand is still not fired. I cannot understand why.

The GridView is using EnabledViewstate = True and I am putting a !IsPostBack around the databind call on Animal.aspx.vb. What wrong is happening?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1001540Wednesday, November 22, 2006 7:04 PM by InfinitiesLoop

Ruben -- the user control contains a DataList. I assume the button you are clicking is inside the datalist, no? The DataList will "consume" the click event and fire its own item command event -- and then that's where it stops. If you want it to keep on bubbling up you will have to add code to handle the DataList item command and then raise a Bubble event. Then the GridView will get the Bubble event and raise its ItemCommand.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1003951Thursday, November 23, 2006 2:41 AM by Ruben Cordoba

Infinities loop -- Let me recap several things:

I have created a label control for testing purposes inside the dynamic user control. I have assigned the value "Hello world" to the label control declaratively. After that, at the end of the Page_Load event handler I have changed the text property value to "Happy programming" programmatically to start tracking viewstate. When the page is posted back and the execution code is located at the beginning of the Page_Load event handler, I see that the text property value of the label control is "Happy programming". That means:

1. Using GridView_RowCreated as the place to load the dynamic user control and putting a !IsPostBack around the databind call on Animal.aspx.vb has been an improvement in the system. Now the viewstate is persisted inside the dynamic user control. Before, it wasn't.

2. You were correct and I was wrong. ItemCommand really has nothing to do with viewstate. ItemCommand is still not fired and viewstate is persisted inside the dynamic user control.

Answering your previous question: Yes, the user control contains a DataList and the LinkButton I am clicking is inside the DataList. You can look at the code I provided to realize about it.

What is happening now is the following:

When I click the LinkButton and the page is posted back, the Page_Load event handler inside the dynamic user control is fired. However, when the Page_Load event handler has finished to be executed, the execution code jumps to the detail page about the animal picture clicked on, and the DataList doesn't have the opportunity to "consume" the click event and fire its own item command event. Because of that, the DataList_ItemCommand is not fired inside the dynamic user control and I cannot raise a Bubble event. So, the GridView will not get the Bubble event and raise its ItemCommand.

Why the DataList_ItemCommand is not executed after finishing Page_Load? I don't know.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1006472Thursday, November 23, 2006 10:07 AM by Ruben Cordoba

Only removing the line !IsPostBack around the databind call on the dynamic user control I can do the DataList1_ItemCommand to be fired. The problem is that I need that line and it shouldn't be removed.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1007517Thursday, November 23, 2006 1:34 PM by InfinitiesLoop

Ruben -- Do you still have ViewState enabled on the GridView? If its enabled you should be able to keep !IsPostBack.

I notice you are using an image inside a linkbutton. Why not just use an ImageButton? It gives you the functionality you want, you can still set a command name and argument, etc.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1008977Thursday, November 23, 2006 3:25 PM by Ruben Cordoba

InfinitiesLoop - I have viewstate enabled on the GridView and dynamic user control. Also, I have !IsPostBack around the databind call on the GridView and dynamic user control. The dynamic user control is re-created inside the GridView_RowCreated.

Yes, you are right. I will change the implementation to use ImageButton instead of an image inside a linkbutton.

Anyway, I don't know what to do more to fire the DataList1_ItemCommand.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1009870Thursday, November 23, 2006 4:16 PM by Ruben Cordoba

I have changed the LinkButton for an ImageButton as follows.

<asp:ImageButton ID="ImageButton1" runat="server" CommandName="ImageButton1" CommandArgument='<%Eval("PictureID") %>' PostBackUrl="~/PictureDetails.aspx" OnCommand="ImageButton1_Command" />

Once the ImageButton1 is clicked on, what is supposed to be fired first, ImageButton1_Command or DataList1_ItemCommand? Neither of them is fired.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1012264Thursday, November 23, 2006 5:38 PM by InfinitiesLoop

The CLICK (not COMMAND) event should fire on the image button. If it isn't that explains why item command isn't. Does the button still exist after the postback? Are you doing anything else peculiar besides what you've outlined?

Check this... in the page put a break point in OnLoad, PreRender and in the GridView_ItemCreated event. On the postback where you click on the image button, what happens first, ItemCreated or PreRender?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1013730Thursday, November 23, 2006 6:21 PM by Ruben Cordoba

Before posting back the page, the order the events are executed is:

Page_Load

GridView1_RowCreated

Page_PreRender

After posting back the page, the order is:

GridView1_RowCreated

Page_Load

Page_PreRender (is not fired)

ImageButton1_Click (is not fired)

The ImageButton still exists after the post back:

DataList1.Controls.Count -> 4

DataList1.Controls(0).ID -> Nothing

DataList1.Controls.Controls(0).ID -> Nothing

DataList1.Controls.Controls(1).ID -> ImageButton1

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1024930Friday, November 24, 2006 12:43 AM by InfinitiesLoop

There's no way Page Prerender doesn't fire unless there's a redirect occuring or it's a callback or atlas partial update. There must be more going on than you've described. Do you want to send me your project? I may not be able to run it without your middle tier or backends but I can still tell a lot more.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1040950Saturday, November 25, 2006 1:41 AM by Ruben Cordoba

InfinitiesLoop -- There are several points I would like to comment to you:

1. After clicking on the ImageButton and the page posting back, why GridView1_RowCreated is executed before Page_Load? Is that the normal flow? I think Page_Load should be always be executed before GridView1_RowCreated. Am I correct?

2. Why Page_PreRender is so important for you to determine if things go good or wrong? I never use Page_PreRender in my project.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1041014Saturday, November 25, 2006 2:14 AM by InfinitiesLoop

Ruben -- (1) No... RowCreated will happen whenever the rows are recreated. That isn't necessarily at a particular point in the lifecycle. It so happens that the rows are recreated immediately after viewstate is loaded, which is before page load. (2) I was only using

PreRender as a measure of when the event fires -- certain things are too late if they happen after prerender. No reason to get into the details. You should definitely always get a page_prerender event though, unless again, the postback is a partial update in atlas, a callback, or a redirect or response.end occurs before hand.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1073059Wednesday, November 29, 2006 3:31 AM by Ruben Cordoba

InfinitiesLoop -- Have you received my comments about the Gridview simplified full version? Do you think I found an ASP.NET bug? All my experiments about getting LinkButton_RowCommand to be fired inside a dynamic user control inside a GridView row template were failed.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1077263Wednesday, November 29, 2006 3:25 PM by InfinitiesLoop

Ruben -- I sent you an email asking for you to attach the files. Perhaps you didn't get the email, or I didn't get your reply. Send me an email through the 'contact' form on this blog, then we can communicate privately instead of through this clunky comment form.

Why Even Default Values must be Persisted

- ***H Canberger***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4334087Thursday, October 04, 2007 10:43 AM by H Canberger

… After trying to optimize view state usage on my custom controls I realized that the pattern for properties persisting its data in the view state, used on most controls in System.Web.UI.WebControls, is far from optimal. If you look at your TextBox's ReadOnly property it's implemented like this:

```
public virtual bool ReadOnly
{
  get
  {
    object obj =ViewState["ReadOnly"];
    if (obj != null)
    {
      return (bool) obj;
    }
    return false;
  }
  set
  {
    ViewState["ReadOnly"] = value;
  }
}
```

Much like your Text property example above.

If you set ReadOnly=true after the TextBox control has started tracking changes to its view state, the value will go into the hidden __VIEWSTATE field. Just like it should.

If we set it to ReadOnly=false it will do the same. Why? This time it's unnecessary to persist the false value in the hidden field, since the get accessor will default to false if the value is not in the ViewState statebag.

So, if we set the property to its default value, according to what get returns, there's no need to persist it to the hidden field and therefore we can set the dirty flag to false.

In the example below I'm using a constant for the default value, and a shorter get-accessor than above, besides the changes of the set accessor.

```
private const bool _Default_ReadOnly = false;
[DefaultValue(_Default_ReadOnly)]
public virtual bool ReadOnly
{
 get
 {
   object obj =ViewState["ReadOnly"];
   return obj!=null ? (bool) obj : _Default_ ReadOnly;
 }
 set
 {
   ViewState["ReadOnly"] = value;
   //When setting the default value, there's no need
   //to persist it in the viewstate field
   if(IsTrackingViewState && value == _Default_ReadOnly)
   {
     ViewState.SetItemDirty("ReadOnly", false);
   }
 }
}
```

This way no unnecessary data will go into __VIEWSTATE.

One might say that setting properties will take longer. Sure, it will, but in total it must be shorter than serializing and deserializing the data.

Dave, any thoughts? Is this pattern safe to use? I haven't seen this pattern on the net so I'd be happy for any feedback.

Thursday, October 04, 2007 1:41 PM by InfinitiesLoop

H Canberger --

I have thought the exact same thing in the past. But there is one scenario where it's important to store the value even though it is the default. Imagine you have:

<asp:Foo runat="server" Visible="false" />

Visible is true by default. Since this is just the declared value, it won't be saved into __VIEWSTATE. Now dynamically set Visible to true.

foo.Visible = true;

If the Visible property said hey, that's my default value, so just remove the viewstate entry, then what do you think would happen on the next postback?

<asp:Foo runat="server" Visible="false" />

Sets it to false... and then since there's no entry in the dirty ViewState for it, it stays false.

So it's important to store the value even if it's the default, because it also tracks changes from the natural, declared state of the control.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4347660Friday, October 05, 2007 3:22 AM by H Canberger

Thanks for your comments. I knew the pattern was to good to be true. :)

Events

Using OnInit

Binding with/without ViewState Disabled

- ***Jouni Heikniemi***

An interesting article! … One thing I'd like to challenge you on is your advice on using OnInit to avoid dirtying the ViewState when initializing the States example (chapter 3 of the misuse part). Right after that in chapter 4 you state that initializing the formatted DateTime in OnInit is too late, since the label is already tracking its ViewState. There is apparent controversy between these two chapters. What are your thoughts on this, and what have you intended to say in chapter 3?

Monday, August 14, 2006 12:33 PM by InfinitiesLoop

Jouni -- …

OnInit -- actually there's no controversy really. In the Dropdown example not only did I suggest binding in OnInit, but I said to disable viewstate on the control. ViewState for child controls is indeed tracking in OnInit. It is not tracking for your OWN viewstate though. You can see it for yourself if you put a break point in OnInit and do a watch on something like:

this.ViewState.IsTrackingViewState // false

this.Controls[0].IsTrackingViewState // true

In the DateTime example I tried to initialize a label in OnInit, but _without_ disabling its viewstate. I hope that clears it up.

Thanks for the comment :)

Wasteful Rebinding?

- ***Brandon Croft***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4144698Tuesday, September 25, 2007 2:44 PM by Brandon Croft

… The problem with simply binding on every init is this: Many times we have to rebind to allow postback events to be raised from dynamic controls. After the event is raised, it's even more common to bind yet again to reflect changes or leave the page entirely. Either way, it's a waste of a database trip when you could have just examined the form collection to begin with. This is the fundamental flaw of ASP.NET: event-ignorant pages.

Hooking into the Init event of a static Control

- ***Daniel Bailiff***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4218748Thursday, September 27, 2007 4:36 PM by Daniel Bailiff

You mention that we can hook into the Init event of a static control to dynamically set its properties. Such as:

<asp:Label id="Label2" runat="server" OnInit="lblDate_Init" />

However, I'm having problems writing the event handler. The key issue being that the sender object is not initialized so I can't do anything with it!

```
public void MyObject_OnInit (object sender, EventArgs e)
{
((MyObjectType)sender).SomeProperty = "foo";
}
```

In the above example, "sender" is null. Can you please provide a code example on how to do this? Thanks!

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4218836Thursday, September 27, 2007 4:43 PM by InfinitiesLoop

Daniel -- works for me. Sender is the control. Perhaps its caused by something else on the page -- are you using a master page? Can you duplicate this with just a simple page with nothing else on it?

Using OnPreInit

OnPreInit Problems with Master Page

- ***Calle Arnesten***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 471850Friday, August 18, 2006 5:26 AM by Calle Arnesten

In your text you say "Also for ASP.NET 2.0 developers, there's OnPreInit. That is actually a great place to initialize child control properties programmatically because it occurs before the child control's OnInit (and therefore before it is tracking ViewState) and after the controls are created. "

This is not working for me. If I override OnPreInit on a page the controls have not been created. They are all null. …

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 472144Friday, August 18, 2006 4:05 PM by InfinitiesLoop

Calle -- are you using a master page? Try calling base.OnPreInit, then check the controls, they should be there.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 474431Monday, August 21, 2006 2:27 AM by Calle Arnesten

I tried without a master page and then it worked. Why will it not work with at master page?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 474703Monday, August 21, 2006 11:59 AM by InfinitiesLoop

Master pages cause the controls to be shuffled around a bit during initialization. Did you try calling base.OnPreInit first?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 475742Tuesday, August 22, 2006 6:47 AM by Calle Arnesten

Yes. Here is the code:

```
protected override void OnPreInit(EventArgs e)
{
   base.OnPreInit(e);
   Label1.Text = "Test"; // Crashes here with NullPointerException
}
```

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 479105Wednesday, August 23, 2006 3:01 PM by InfinitiesLoop

Calle -- I haven't had a chance to test this out. I know I've done it that way before, but perhaps there's something different about your master page setup that is yielding different behavior. May I ask why you need to initialize the control there? There's usually another/better way.

- ***Paul***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5439960 Tuesday, December 11, 2007 6:42 PM by Paul

… Thanks for the article. I want to set some Control attributes that are basically static. I do not want these properties tracked by ViewState because that would be a huge waste. So, I do something like:

```
protected override void  OnPreInit( EventArgs e )
{
  base.PreInit(e);
  userName.Attributes["onfocus"] = "getHelp('userName','Help1')";
  email.Attributes["onfocus"] = "getHelp('email','Help1')";
  ...
}
```

This doesn't work, .net complains since userName is null. MasterPages seem to be blameworthy. However, it works if I do something like this:

```
protected override void  OnPreInit ( EventArgs e )
{
  MasterPage master = Master;
  base.PreInit(e);
  userName.Attributes["onfocus"] = "getHelp('userName','Help1')";
  email.Attributes["onfocus"] = "getHelp('email','Help1')";
  ...
}
```

Seems that Master.get somehow makes everything initialized. I'm assuming that it somehow forces construction of the MasterPage object and reshuffles the controls/contentholders.

So: What should I do. Referencing this.Master on all my my pages in preinit seems like a hack. The only other option I can think of is to perform this option in Render() before calling base.Render(). I suppose a good option would be to share a subclass of Page that has an OnRender() stub.

Any Ideas?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5440605Tuesday, December 11, 2007 8:56 PM by InfinitiesLoop

Paul -- indeed I have seen that Master Page problem. I'm afraid that's just how it is, OnPreInit as far as I know was only intended for dynamically setting the master page (one reason why it isn't a control event).

But actually I would probably do it in Render like you suggest anyway. That way the data can be up to date if something during the processing of the form needs to change it. Render is safe as far as viewstate goes obviously since it's after it's already been saved with SaveViewState.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5446771Wednesday, December 12, 2007 8:09 PM by Paul

Thanks infinity,

I want all of my "life cycle handlers" to look consistent, therefore, I did the following.
Hopefully this helps someone later on.

```
class MyPage {
 ..
protected override void Render( HtmlTextWriter w )
{
  base.Render(w);
}
protected virtual void OnRender( )
{
}
```

```
 ..
}
```

I don't actually name it "MyPage", but this is the base class of all my pages. If you are inheriting from Page and not a subclass, then I recommend refactoring since a lot can be accomplished at this level.

My subclasses, then, look like:

```
class AwesomePage {
 ...
 protected override void  OnRender( )
 {
 userName.Attributes["onfocus"] = "getHelp('userName','Help1')";
 email.Attributes["onfocus"] = "getHelp('email','Help1')";
 base.OnRender() // Not necessary, but good habit
 }
 protected override void OnInit( EventArgs e )
 {
 ...
 }
...
}
```

Now, I just need to figure out why my password-mode textbox's value is not being persisted through multiple postbacks. I have a MultiView and I want to hold onto a password entered on the first view. I have ViewState enabled. It is odd how ViewState acts one way with some controls, and another way with other controls (MasterPages, PasswordMode TextBoxes)

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5446875Wednesday, December 12, 2007 8:26 PM by InfinitiesLoop

In password mode TextBox does not remember the password on purpose. Whether you like it or not, that's just how it was written. I think that’s a fair feature though, you wouldn’t normally want that password to hang around and have to be cleared manually.

If you look at its implementation of AddAttributesToRender you'll see what I mean. I suppose you can just derive from it, override AddAttributesToRender, and make sure the Value attribute is added. ViewState still remembers the value of course, but since the TB is empty after 1 post, the 2nd post clears it out, even in ViewState (just as if the user cleared it themselves).

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5465728Monday, December 17, 2007 1:19 PM by Paul

Hum, I guess I will just hold on to the password in session state and clear it out after submitting the form. Thanks.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5465744Monday, December 17, 2007 1:25 PM by InfinitiesLoop

Paul -- don't use session state, just use viewstate.

ViewState["Password"] = txtPassword.Text;

But even that isn't needed if you implement what I said with AddAttributesToRender.

ViewState's Relationship to Event Handlers

Tutorial Needed?

- ***Scott***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1602077Friday, February 09, 2007 5:12 PM by Scott

Great article. I would like to reiterate a comment left earlier:

It would be useful if you could extend the article to clarify how viewstate is associated with event handlers e.g. the dropdownlist control's SelectedIndexChanged event handler.

How to Access ViewState in Page_Load

- ***venkatesh***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1580343Thursday, February 08, 2007 12:28 AM by venkatesh

how to get the view state value in page load? could you please help me?

LoadViewState Event

- ***Abey***

Thursday, October 04, 2007 8:46 AM by Abey

I had a problem few days back. I have couple of user controls which has a method (say method A) which returns the values for the controls (text boxes and dropdowns) inside the user control. I am dynamically loading the user control in my aspx page (by default one user control will be loaded). I have a dropdown in my aspx and when i change the dropdown, the respective user control is loaded after removing the previously loaded user control. But before i load the new user control, i call the method A on the currently loaded user control to get the values of text boxes of user control. I am getting the values properly for the default user control which is loaded. But the problem starts when it tries to call the method A after changing the drop down (ie after a new user control is loaded in dropdown selected index

changed method.) The method doesn't return the values which i have entered in the controls. I tried a bit of googling to find out the reason and solution for this.

Some say that its because i am loading the user control in dropdown, your view state doesn't store that user control values, because what ever controls which is dynamically loaded after the LoadViewState method of the page, view state doesn't seems to store the values of those controls. I tried loading the user controls in different events which actually fires before the loadViewState, but no use.

Though i didn't get a proper solution to the problem, i partially solved it by using the request object in method A of user control. For eg: you can use Request["your textbox id"] , which will give you the values you have entered.

This one i actually posted in my blog

blogs.ittoolbox.com/.../user-control-not-persisting-the-value-19410

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4335510Thursday, October 04, 2007 1:35 PM by InfinitiesLoop

Abey -- you say you don’t get the values you want when you call A() on the newly loaded user control. But when exactly are you calling it. You make it sound like you're calling it right after loading it, but then the user hasn't had a chance to enter anything yet.

A control created after LoadViewState will still load its viewstate, so whoever told you that is wrong :) There's no reason this shouldn't work, if it's done the right way. The main thing would be to make sure you're always loading the control that previously was loaded, and the best place to do that would be from LoadViewState. If you follow the pattern I describe in Part 4 of my article on understanding dynamic controls it should work.

Design Considerations

Response.Redirect vs. Transfer

- ***TBD***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 589101Wednesday, September 27, 2006 4:13 PM by TBD

… I'm using ASP.NET 1.1. I have a login.aspx site that contains two textboxes and one button; tbUsername, tbPassword and bLogin. When pressing the bLogin button it logs in with an API library (which keep me logged in until i logout manually). The problem here is that later when I trace the target page (default.aspx), immediately after logging in (login.aspx), I can see tbUsername, tbPassword, bLogin as well as their values in clear text! I want to remove these values or putting them into the encoded viewstate (to prevent password from being visible). Since I'm using an API that keeps me logged in, the values don't need to be posted.

How can I do this?

Thanks in advance.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 589634Wednesday, September 27, 2006 7:56 PM by InfinitiesLoop

TBD -- I'm not sure what you mean you can see the values in clear text. You said you are on an entirely different page. How are you transferring from login.aspx to default.aspx? Is it a Response.Redirect or Transfer? If it's a transfer, the target page can access the posted values from the previous page, because it's all done within one request. If you don't like that, do a redirect instead. It's got nothing to do with ViewState, unless I'm not understanding your problem. Let me know.

Using a Custom/Personal Statebag

- ***Mark Olszowka***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 591358Thursday, September 28, 2006 11:57 AM by Mark Olszowka

Maybe just slightly off topic but....

Any thoughts/comments on use of a "custom" or "personal" statebag.

There have been many times that I wanted to store one or more custom values in "viewstate" but did not want to turn viewstate on.

I realize you can store these values in hidden form fields, but it would be nice if you dealt with this custom data the same way as if you were placing it in the page's builtin viewstate, ie serialization, encryption, tamperproof, etc.

I tend to keep viewstate turned OFF unless it is absolutely necessary (and with the new ControlState, that is almost never the case.

I see that the statebag object in flagged as Not Inheritable.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 593276Thursday, September 28, 2006 6:12 PM by InfinitiesLoop

Mark -- Even if you could inherit from StateBag, I don't think it would help.

There's a couple of ways you could do it... one, you could NOT turn off viewstate. Instead, wrap all your controls in a PlaceHolder which does have viewstate disabled. Then you can just use the page's viewstate (or the user controls, whichever you are) to store stuff in ViewState, but still have viewstate essentially disabled on all the controls on the page or control (disabling viewstate on a control disables it for all its children too).

Another way is you could use ControlState. You'd have to write a custom control to let you do it though. It would be like a ControlStateHelper control, could be useful for other purposes. Not trivial to write though.

It's good that you turn off ViewState by default, but I actually wouldn't recommend that practice, at least not if it's only because you want to avoid the fluff. A well written, correctly written page will only use viewstate when it really needs to, even if it's enabled. So turning it off just allows you to be "lazy" and do things that wouldn't be good if it were on. You're letting your ViewState muscles get atrophy :(

Private Member Variable vs. Direct Use of ViewState

- ***Matt***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 674526Tuesday, October 17, 2006 6:51 AM by Matt

Excellent article thanks. Here's a question though: an alternative to having properties that are directly put into/obtained from ViewState (in get() and set()) is to store the property to a private member variable and explicitly save to/obtain from ViewState in the SaveViewState() and LoadViewState() methods.

What are the pros and cons of this?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 675446Tuesday, October 17, 2006 11:35 AM by InfinitiesLoop

Matt -- that works fine, but you have to be careful not to stuff the default value into viewstate when saving -- and, you have to be careful not to overwrite the existing value during load if viewstate for that value is empty. Much easier just to use the statebag directly.

Basically, just loading your control or page for the first time (!IsPostBack) shouldn't result in any persisted viewstate at all, unless your control or page has loaded dynamic data and plans on letting viewstate maintain it.

Manipulating SetItemDirty - Pro and Con

- ***Cat***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 752249Sunday, October 29, 2006 3:10 AM by Cat

Great article! Before I read this article, I was confused by why the value of IsDirty property is always true, and only in a few situation it's set to be false by SetItemDirty method.

When deriving from an existing control, you can set any property at any time without it being saved to __VIEWSTATE, as long as you know which StateItem it uses and remember to reset its IsDirty property to false. Here's a new DateTimeLabel with its Text property set in OnLoad:

```
public class DateTimeLabel : Label {
  protected override void OnLoad(EventArgs e) {
     this.Text = DateTime.Now.ToString("MM/dd/yyyy HH:mm:ss");
     this.ViewState.SetItemDirty("Text", false);
     base.OnLoad(e);
  }
}
```

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 756818Sunday, October 29, 2006 2:35 PM by InfinitiesLoop

Cat -- SetItemDirty is a trick you can use if you must, but in general I wouldn't recommend it. This DateTimeLabel doesn't behave like most controls, and that inconsistency is going to be frustrating for those who use it. Maybe in this situation it would be alright, since it isn't likely a dev would want to set the text property (since its specifically designed to show the date). But if they did, you'd be overwriting the value with your own.

Disable Submit Button until Postback Completes

- ***Nash***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 773799Tuesday, October 31, 2006 2:10 PM by Nash

I'm posting to discuss an interesting symptom of ViewState. I've read your article, which is very informative, but I am still not quite certain as to which characteristic is causing the aforementioned symptom. In any case I think you may find interest in this.

In a nutshell the symptom is simply that an arraylist is inserted into the ViewState upon a textbox change event and after a certain sequence, it will not be persisted in a secondary event of a button click. Previous ViewState entries are persisted, but the latest entry is missing.

Here is the description:

The page setup consists of three simple dynamic text boxes populated by database. When the user changes the text values each is saved to ViewState until the user is ready to save changes to the database. The error arises from an interaction between the postback of the textbox change and the postback of the button click that is used to save the data (SaveButton).

For example, if a user enters values into each three text boxes {"A","B","C"], each is posted back in turn as the next textbox is selected. But, upon the last entry of "C" when the SaveButton is clicked before anything else, the ViewState symptom may occur, based on the event firing.

The result is that both text change and button click events do fire in order. The text change correctly updates the ViewState with "C", but stepping through the SaveButton the ViewState contains all inserts accept those from the "C" text change event.

Here is the interesting difference. Perhaps, you understand why it is causing the disjunct ViewState symptom. Upon the down click of the SaveButton a wired javascript function pops up and allows the user a typical "Are you sure you want to save" message.

Sometimes this action only calls the waiting onchange event; which causes a refresh and works. Sometimes this action calls both the onchange event and the button click event; which causes the symptom. When the latter happens, the wired javascript handler displays the yes/no alert, the text change fires and populates the correct values into ViewState, but as the user selects 'yes' and the button click event is run…the ViewState is still antiquated. That is to say, ViewState hasn't been updated with the most recent addition of "C" in the fired text change event.

The work around is easy but I am not sure why this is the case. I would be interested if you know the intricacies of why this occurs.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 786532Wednesday, November 01, 2006 7:06 PM by InfinitiesLoop

Nash --

Why are you storing data in response to TextChanged? Why not just calculate your data when you need it, such as in the button click event? That way it doesn't need to be in viewstate at all, and you don't need postbacks between each textbox.

Assuming you can't change that design... something else you said concerns me. You have a button that opens a javascript confirm dialog. You seem to be saying that there is a postback occurring _while_ this dialog is open? I'm kind of lost after that -- the dialog is a client-side operation, there shouldn't be anything at all going on server side at that point, not until they user dismisses the dialog.

Just a shot the dark here, but since you are doing postbacks between each textbox, its quite possible the user is clicking on the button before the last postback is finished. So you'd see the postback of 'C' occuring correctly, but then another postback would come in afterwards, and that postback would not contain the updated viewstate since it was posted from the same form the first one was. If you get rid of the postbacks like I stated in the first paragraph, your form will be much zippier and won't have this problem. Otherwise, you'll have to disable the submit buttons when the postback starts, or the user may click them too quickly.

Defer Populating Data-bound Control until DataBind() Event

- ***kmobarra***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 894688 Thursday, November 09, 2006 5:09 PM by kmobarra

I have a bit of a catch 22 here that I hope you (as the only person I know that understands ViewState :) can help me with: I have created a tree usercontrol that is populated dynamically, after its container control sets its data source. Note that the control is not loaded dynamically by the container, but just populated. If I could populate the usercontrol in its page_init event, the controls would get properly registered with the container page's viewstate and when a postback happened, the control would get populated in init, before the container page started to assign viewstate and by the time the control's Page_load would fire, I had access to the controls' last state before postback. The problem is that again, the usercontrol's datasource is determined in the container page and therefore it's not available at the time page_init happens, but later in Page_Load, which is too late. Can you suggest a way out of this dilemma? Thanks, Kathy

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 896981Thursday, November 09, 2006 9:03 PM by InfinitiesLoop

kmobarra -- It shouldn't have to matter when you populate the control -- before, or after viewstate is loaded. The fact it does matter kind of indicates a design issue. Instead of populating as soon as the datasource is assigned, why not populate in DataBind() just like other databound controls do? The containing page would set the datasource then call databind -- but it would only do this on !IsPostback. Then in your tree control, you have to rebuild the control tree on LoadViewState. So you will need to save whatever information you need into ViewState at the time DataBind is called to be able to do that.

That being said, let me make sure I understand the problem. The page is assigning the datasource during page_load, and I presume it is doing that on every request? So your problem is, the page loads with the first set of data -- then the user makes some client side changes and performs a postback. The controls viewstate is loaded and it repopulates itself. Then page_load comes along again and the data is reassigned, causing the control to be repopulated and the values changed by the user are lost. Am I right? Or, perhaps the issue is that the posted values of input controls like textboxes are not updated by page_load (because the controls were added during page_load, they won't get posted values until the 2nd pass a loadpostdata which is after page_load)?

I still think that rethinking the design a bit like I said in the first paragraph is the right way to go... but a quick workaround for the situation I described is for the page itself to dynamically load the control. This is assuming your issue is with viewstate... it won't help you with the post data. In page_load, you create the control, assign the datasource, and then add it to the control collection (add it to a placeholder more than likely). This lets the page get a chance to assign data to the control before its Init event, since Init won't happen in the control until it is added to the control tree. It's a hack solution though... your control aught to behave like every other databound control. The page that uses it should be able to decide when the data is refreshed and how often.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 900875Friday, November 10, 2006 8:27 AM by kmobarra

I take your advice and change the design of the tree control. Thank you. Kathy

Slow Page Loads – ViewState may not be the Culprit

- ***Dave***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4783365Saturday, October 27, 2007 1:10 PM by Dave

I am investigating a performance nightmare we are having and I think it could have something to do with viewstate. We were getting a lot of viewstate errors and the coders have gotten those to stop, but our performance is dismal. We spent the last month optimizing our code for SQL, but it's made no change in performance. Your viewstate madness graphic matches our source view EXACTLY. Nutty looking cat upchuck of characters. I am not a coder (many years designing though and good knowledge) but I have spent a TON of money (measured in gold bullion) on trying to figure out the performance problem. The db has 400K records, but we have broken things into smaller tables. Nothing seems to help. I am not real sure the coder knows some of these advanced/proper uses of viewstate for instance.

Looking at our view source of one of our sites, www.sarasotaguide.tv I am not sure if you can tell if viewstate is our problem, but after reading this article it smells like it?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4785308Saturday, October 27, 2007 3:11 PM by InfinitiesLoop

Dave -- the viewstate size on your main page is about 25kb. That's pretty big, but it's not so big that you'd have a "nightmare".

What I noticed though was that your page took like 4 minutes to finally come through. And the size of the HTML of the front page is about 160kb. Thats huge and sure to drive dial-up users away. Interesting the bulk of the size of your HTML seems to be that tree view on the left. You're obviously using Telerik Rad Controls. Those controls are great but their richness comes with a cost.

Still all of that doesn't explain why it took 4 minutes to respond. To figure that out it would take some debugging. Sounds like the server is doing much more than it needs to, like executing a lot of SQL queries where it could cache the results or consolidate them into one query. Some SQL Profiling would help with that. Send me a private mail and we can dig deeper.

Don't use ViewState at all?

- ***Jakub Anderwald***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5636122Tuesday, January 22, 2008 1:41 PM by Jakub Anderwald

… I'm just new to ASP.NET, learning everything I can. I used to write in PHP. The more I read about Viewstate, I'm getting just one thing on my mind: don't use it, remove completely. It creates a lot of issues / problems. It throws tons of data to the user and back. But this is not that bad, when used properly. The worst thing is that it makes your server application depend on some encoded / hidden data being sent by user.

I find it unacceptable for a client / server architecture application to depend on the client to work properly. I think it's unsecure by design, no matter how many compensating controls we put on it, it's unreliable, and it's throwing a lot of overhead for nothing.

To me any application that depends on keeping configuration data on client-side is, sorry for the word, lame. I believe that every data should be repopulated on server-side on every client request. If it's from a database query or an internal application cache, it doesn't matter - it's just a matter of how much code would you write for sake of performance and database profiling.

Some really expensive to get data can be cached either in session (when they're per-user by nature) or in some application-wide cache (when they're global), but NOT sent to user and asked to be sent back. I really don't understand why Microsoft is forcing this faulty method.

If I'm just wrong with some basic stuff and reading the whole thing from the wrong side, please correct me. But it doesn't make any sense to me.

Plus, I'm a bit shocked by the lack of understanding of client/server architecture in some of the developers commenting here. To me, always, a developer was a master of the technology he was using - he had not only to use it, he was supposed to tell it what to do. And here I find people who are devleopers just because they can click some fancy controls in IDE.

I just hope they won't be writing my next-gen car computer software ;)

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5636175Tuesday, January 22, 2008 1:54 PM by InfinitiesLoop

… As a rule in a lot of ways I agree with you -- its far better to repopulate data on every request than to have it stored on the client.

But...

Storing data on the client... that is not what ViewState is all about. That is what it does, but what data you put there is up to you, if you use it correctly.

Take for instance an accordion control, common to many menu systems. You click a category and it expands to show a sub-menu. Ok, now over on the other side of the page you type in your email address and click submit in order to add yourself to the site's monthly newsletter. Oops... now the accordion has collapsed again. That's not what the user wanted. Better if that menu could have remained in the same state. That is the kind of thing ViewState is great for. It's transient data that is only useful within the context of this single page for this single user, and only while the user is on this exact page. Imagine if you had to store that piece of data in the user's session. It seems harmless -- but with millions of users, that adds up to a lot of data. Data which will hang around for the user's entire session length, even though it's only used for a few seconds! ViewState in this case is a _far_ more efficient way to maintain that kind of state information.

Microsoft is not forcing anything. In my opinion it would have been better if ViewState was always off by default, or if you had to write some explicit code to get data in and out of viewstate. But it being on by default is a long way from 'forcing' anything.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5636510Tuesday, January 22, 2008 3:16 PM by Jakub Anderwald

Two things here:

1/ I had a look at the Accordion and I think it should be expanded at the current navigation level, ie. where you currently are. And as far as I saw, it's quite easy to see in ASP.NET where you are thanks to the web sitemap etc. So I see no reason to store it somewhere.

I see your point here: keep only some not critical data, and let your application gracefully degradate when that data is not available. But it still "doesn't speak to me", as we say in Poland ;) See - if I read your post correctly, you have to take care about deleting that data after it's been used, while in session it would simply expire. And RAM is quite cheap these days, while amount of data flowing in/out of your connection is not.

Plus, POST is not the only way to interact with your page - there are regular GET requests, there are F5-refreshes, there are users who can manually type / paste a link to another part of your site - I would loose the whole ViewState then. Still, I can't see any valid reason to use it.

2/ My friend was playing with VS2008 and ViewState / controls and he was not able to remove it completely. Turning it off in web.config and on control level didn't remove it completely, just made it shorter. I think we'll use the decoder to see what's inside.

He found just one way to remove it at all - overwrite some default functions, so that they return null. But even then, you get the input field in your form.

So yes, it looks like forcing to me ;)

If we're wrong at what we're doing - we're just learning, you know - please correct me. I'll ask him to comment here as well detailing what he did.

Also - a great note about the note that FORMs populate themselves with client-provided data without Viewstate and that ViewState it's not about that. When I asked a coworker who wrote a couple of ASP.NET sites what ViewState is really useful for, he said that it allows the site to repopulate all the data in a form that a client submitted just before.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5636560Tuesday, January 22, 2008 3:31 PM by InfinitiesLoop

An accordion was just a contrived example. If you can make it do the right thing without storing data, great. But not all accordions are used for menuing. What would you do then?

Yes ram is cheap, but server resources are precious. You have to think about not only the amount of data you are saving, but how long it is resident for. If a session is 20 minutes and you are storing 100 bytes, that’s 2000 byte-minutes, regardless of how long the user is actually on that particular page. If you store that same 100 bytes in viewstate, it only actually consumes server memory when the user actually posts the form. A user isn't likely to sit there on the same page posting it over and over again for 20 minutes are they? You'll get far less byte-minutes on the server. It seems trivial, but image 1000's of users on the site at the same time. That can quickly add up to many megabytes worth of data for something so trivial and simple.

You can't completely get rid of viewstate. It is absolutely required in some situations by the framework. You might scoff at the notion that it is required for anything, but it is, and it has good reasons. And better to use a single hidden field for the required data -- yet somehow I doubt people would complain about it if this 'required' data were stored in a differently named field or fields. Read my post on what Page.RegisterRequiresPostBack is used for as an example.

Tuesday, January 22, 2008 4:32 PM by Jakub Anderwald

What most of the websites do - create menus. Using (X)HTML / CSS styled list. That's all that's required to create a menu ;)

Is ViewState required to be sent to user, or is it enough to be active during the page generation and processing? I'm thinking of getting rid of it totally from the generated content. I don't mind sending an empty __VIEWSTATE field, but I don't like the idea of having elements in my application, that are "absolutely required" to be sent to client and then sent

back via POST. What happens when we use simple links or AJAX to fetch more data? Would the "absolutely required" data be there? Not really.

But - I am open to read more ;) There have been times when I thought of something as totally weird, unnecessary, wrongly designed, and then I found out that it's actually OK in some circumstances. Can you guide me somewhere?

Thanks a lot.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5636865Tuesday, January 22, 2008 4:49 PM by InfinitiesLoop

>> What happens when we use simple links or AJAX to fetch more data? Would the "absolutely required" data be there? Not really. <<

Sure it will. If you use an UpdatePanel. And if you don't use an UpdatePanel, then there isn't any update from the server anyway, and there is no viewstate to update.

>> But - I am open to read more ;) There have been times when I thought of something as totally weird, unnecessary, wrongly designed, and then I found out that it's actually OK in some circumstances. Can you guide me somewhere? <<

The RegisterRequiresPostBack article I mentioned explains the gist of it. If you want to use the page and control model, certain things have to be known across postbacks. That's the way it is. It isn't like it's just the way it was implemented -- no, it couldn't be implemented differently without quirks and bugs creeping in. For a checkbox, as mentioned in that article, it's due to a flaw (IMO) in how checkboxes are handled by browsers during posts.

I suggest you look at the MVC framework. Sounds right up your alley. There's no ViewState there. weblogs.asp.net/.../search.aspx

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5636867Tuesday, January 22, 2008 4:50 PM by InfinitiesLoop

You didn't understand what I mean about the accordion. I know there are nice CSS ways of creating menus. But it's got nothing to do with my point :)

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5638800Wednesday, January 23, 2008 5:24 AM by Jakub Anderwald

Thanks for all the help and patience.

It might be I'm making unnecessary noise about some things, but the idea of sending information to / from user to make my application work properly gives me the creeps.

I will give it more read, maybe I'll find some sense in it.

Miscellaneous Questions

Unintended Double PostBack

- *Steve*

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 508556Thursday, August 31, 2006 2:51 PM by Steve

…I have an interesting problem that I'm hoping you can shed some light on. I have a Page that implements the IPostBackEventHandler interface. This is used to get a post back after performing a client side action (which is a popup). The event handler receives information indicating if the popup was successful and there is information I would like to store in the viewstate at this point. All seems well, however, there seems to be another postback or recursion of the page processing. Following is the sequence:

Page_Init(object sender, EventArgs e)

Page_Load (object sender, EventArgs e)

RaisePostBackEvent(string eventArgument)

info saved in ViewState and correct

Page_PreRender (object sender, EventArgs e)

Page_PreRenderComplete(object sender, EventArgs e)

----- now comes the unexpected -----

Page_Init(object sender, EventArgs e) *** data in ViewState reverted ***

Page_Load (object sender, EventArgs e)

Page_PreRender (object sender, EventArgs e)

Page_PreRenderComplete(object sender, EventArgs e)

I'm guessing, based on the info in your article that the viewstate was updated, but not re-written so when the next Page_Init comes in it loads based on the Request object hence losing the change just made. That leads me to a couple of questions:

1. Is my guess correct, or is there something I'm missing?

2. What is a good way around this? One thing I thought of was to override the viewstate and save serverside...one of the articles I read mentioned that...it would provide control over when the data is updated and ensure things aren't unexpectedly overwritten.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 508635Thursday, August 31, 2006 3:06 PM by InfinitiesLoop

Steve,

Thanks for the comment. I think I need more context about your problem. You have a page that is opening an additional popup window (to another page I presume), and upon doing so it posts back? Maybe some code samples would make it more clear.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 508738Thursday, August 31, 2006 3:54 PM by Steve

Sorry for the lack of details... I'm not sure which code samples might be beneficial so here goes.

A Main Page contains a variety of info and controls, one button is uxTestEmailButton which is an asp:button. The following code is used to establish the PostBack & handle the popup...

```
string emailInfoScript = string.Format("ShowEmailInfo()");

string emailInfoUrl = Page.ClientScript.GetPostBackEventReference(this, "");

uxTestEmailButton.OnClientClick = emailInfoUrl.Replace("'", emailInfoScript);
```

ShowEmailInfo is javascript which launches the popup window with the new page. The popup page gathers User specific information to be included in the "TestEmail". When a save / close button is selected on this popup, the ShowEmailInfo function captures the information and returns it.

The RaisePostBackEvent(string eventArgument) is received when ShowEmailInfo function returns and the eventArgument contans the return value from the ShowEmailInfo.

```
function ShowEmailInfo()
{
  var invokeString = '../Dialogs/EmailInfo.aspx';
  var returnValue = window.showModalDialog(invokeString, '', 'dialogHeight: 700px;
dialogWidth: 800px; resizable: yes; scroll: yes; status: no;');
  if (returnValue == null)
  { returnValue = ''; }
  else
  { returnValue = 'EMAIL:' + returnValue; }
  return returnValue;
}
```

Sequence of events:

Main Page

Click TestEmail

ShowEmailInfo javascript function runs

Poup comes up

Fill in info in popup

click button on popup

popup closes

ShowEmailInfo javascript gets return info from popup

ShowEmailInfo javascript returns

Main page postback received

Hope this provides a bit more context.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 508773Thursday, August 31, 2006 4:18 PM by InfinitiesLoop

Steve, I think you are getting two postbacks because of the way you are hooking up that click event. An ASP:Button does a postback naturally, because its a type="submit" button. But you've hooked up client script that runs when it is clicked, which also does a postback via the PostBackEventReference you are using. Hence, clicking the button causes two postbacks. I haven't tested it to see if that's the case (one would think the first postback would stop page processing, but I'm not sure if that's true).

Whenever you need a link or button that just runs script and nothing else consider using a non-webcontrol like an HtmlButton control, which don't cause postbacks naturally. Or, you can "cancel" the natural postback caused by clicking the button by returning false in the client click script.

In other words, go ahead and set the OnClientClick code, but be sure and end it with a "return false". Like this...

uxTestEmailButton.OnClientClick = emailInfoUrl.Replace("'", emailInfoScript) + ";return false;";

If that's the problem then great. But it goes to show what a scapegoat viewstate has become... everyone assumes their problem is a viewstate issue. I assure you, viewstate works perfectly. :)

If that doesn't help then we're figure it out... but it still won't be a viewstate problem. Let me know ;)

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 508816Thursday, August 31, 2006 5:12 PM by Steve

That resolved my issue. It makes perfect sense, but then again hindsight is 20/20.

Thanks again, the article really helped and even more so for the latest thought.

Callbacks

- ***Kalpesh***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 498825Monday, August 28, 2006 1:34 PM by Kalpesh

Can you please tell how to manage view state in callback events?

Because life cycle of the page is not same as PostBack!

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 503067Tuesday, August 29, 2006 4:04 PM by InfinitiesLoop

Kalpesh -- the life cycle is similar. It does get cut short because there's no point in PreRender or Render (there's nothing to render to). Viewstate and postback data are all still processed as normal though.

The one thing that's different about a callback is that the data which is posted is representative of the form at the time it was rendered on the browser. For example, if a textbox has value 'foo', the page will load with 'foo' in the box. If the user changes the value to 'bar', and then before doing another postback a callback occurs, the value posted in the callback data is 'foo'. There's a way to update the data so it represents the current state of the form instead if thats what you require. What issue are you having?

- ***Jack***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 760106 Sunday, October 29, 2006 9:09 PM by Jack

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 760106 On August 29th, you state:

"For example, if a textbox has value 'foo', the page will load with 'foo' in the box. If the user changes the value to 'bar', and then before doing another postback a callback occurs, the value posted in the callback data is 'foo'. There's a way to update the data so it represents the current state of the form instead."

What is that way to get the latest values when you make a callback instead of postback?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 765255Monday, October 30, 2006 2:59 PM by InfinitiesLoop

Jack --

Normally you perform a callback like this:

WebForm_DoCallback(...);

To 'update' the form data that is posted so it is up to the minute, you must do this just prior:

__theFormPostData = "";

__theFormPostCollection = [];

WebForm_InitCallback();

Problem with Multiple Simultaneous Users

- ***Big Red***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 648232Tuesday, October 10, 2006 4:23 PM by Big Red

I am having problems with a bulk delete I have in a gridview.

The gridview is nested within a repeater. The gridview has a checkbox for each row and a single 'Delete' button. When this button is pressed, it goes to its onclick function where it looks for checked boxes, and initializes a delete.

This all seems to work fine and dandy, except when two users are working simultaneosly. If user a deletes rows 1,2 and 3, and user B adds 5 rows at the same time, none of the rows are deleted, and the checkboxes continue to be checked. (Although, if one of the 5 rows is inserted and displays between 1 and 2 or 2 and 3, the first 3 rows are checked.)

I can't seem to figure out how to delete these rows before ASP.NET checks to see if the gridview has changed.

Cross-page Postbacks

- ***ChRoss***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1569730Wednesday, February 07, 2007 5:10 AM by ChRoss

Hi Dave, what if the postback target is to another page?

For example PageA.aspx dynamically create a user control, let's say ucC, and I add it in a placeholder plhD (declared), and the postback target of the form is PageB.aspx.

How PageB.aspx can retrieve the values from ucC, because I can load the user control, but cannot add to the control tree. I use PreviousPage property to access PageA.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1579376Wednesday, February 07, 2007 10:23 PM by InfinitiesLoop

ChRoss -- that's a great question! It's great because I've actually never even tried that scenario before.

The value is null probably because the page doesn't exactly go through its typical processing. But it is a page instance, so you should be able to find the control using FindControl.

You should have access to the placeholder you mentioned... call FindControl ('id') on it, where 'id' is the id you give to the UserControl you dynamically loaded. It would work assuming the PreviousPage goes through the life cycle up to OnLoad, which I honestly have no idea about. So I'd love to hear how it goes.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1581370Thursday, February 08, 2007 1:35 AM by ChRoss

Hmmm... I made a further test.

My user control contains a textbox txtC

Case 1

In PageA.aspx, I have textbox txtA, and also a user control created dynamically on the page load. When submit to PageB.aspx, I can retrieve both textboxes value (txtA and txtC), so all working fine.

Case2

In PageA.aspx, I have textbox txtA and a button. When this button is clicked, I create the user control. When click another button to submit to PageB.aspx, I cannot find the user control (using FindControl). it shows null. I check on the placeholder containing the user control, placeholder.Controls.Count return 0.

So, only if the user control dynamically created after postback, then postback to another page, I cannot retrieve the value.

I'm not sure about the life cycle of the PreviousPage, but how do I "load" again the user control? to retrieve the value.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 1589507Thursday, February 08, 2007 6:16 PM by InfinitiesLoop

…ChRoss -- controls have to be loaded every request for them to continue to exist. If you only create a control when a button is clicked, it's not going to exist on requests where the button was not clicked, and that includes cross page postbacks. What you'll have to do is when you load the user control (in response to the button click), store something in ViewState (with this.ViewState["foo"] = "bar"). The thing you store is for your benefit, because you're now going to override LoadViewState. After calling base, you check for that same viewstate value, and if it exists, and using any context information you may have stored in it, you now

recreate the usercontrol. One more thing though -- if the user happens to have been clicking the button during this request, you'd end up loading it a 2nd time. So in response to the button click, you first clear the control collection of the placeholder you are adding it to. That or you find it and remove it -- or you hold on to the instance you created from LoadViewState and use that to remove it. Doesn't matter how, just get it out of there. And finally -- you'll want the IDs of the removed control and the new control to be the same, so don't allow the framework to auto-assign an ID -- give it a specific ID yourself.

Page.RegisterRequiresPostback

- ***jatin***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4743757Thursday, October 25, 2007 11:29 AM by jatin

I read an article "Understanding ASP .Net ViewState" on MSDN. However I have one query regarding the statement in this article.I hope you will help me to understand.

Under section “Parsing the View State".

It says “The SavePageViewState() method starts by creating a Triplet that contains the following three items:

1. The page's hash code. This hash code is used to ensure that the view state hasn't been tampered with between postbacks. We'll talk more about view state hashing in the "View State and Security Implications" section.

2. The collective view state of the Page's control hierarchy.

3. An ArrayList of controls in the control hierarchy that need to be explicitly invoked by the page class during the raise postback event stage of the life cycle. “

I have understood point 1 and 2 however in point no.3, I failed to understand which controls are needed to be explicitly invoked and why are they needed to be mentioned in viewstate ?? how are they getting populated ? who is populating them ? Does array list of controls mentioned in Point no. 3 is in context of dynamic controls?

I have analyzed view state of the DataGrid using Viewstate parser and the third node of triplet appears to be null and same thing is observed when I checked it for sample application with button , textbox and dropdown list on forms.

msdn2.microsoft.com/.../ms972976.aspx

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 4745466Thursday, October 25, 2007 1:37 PM by InfinitiesLoop

Jatin -- an excellent question. One the deserves its own blog entry, I'd say! Check it out! Thanks for the idea for a post.

weblogs.asp.net/.../understanding-what-page-registerrequirespostback-does.aspx

How Page State is retained across Postbacks

- ***Raoul***

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5510901Friday, December 28, 2007 1:09 AM by Raoul

[MCTS Self-paced training kit (Exam 70-528), Microsoft Press, Page 501

As you might have already noticed, if a user clicks a button to submit an ASP.NET page, the page retains all its values and settings. For example, if you modify the text on a label and the user clicks a button, the modified text is still displayed when the page reappears. This happens because ASP.NET has a client-side state management technique built in: ViewState.]

So this is totally incorrect statement? Thanks.

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5518908Saturday, December 29, 2007 4:47 PM by InfinitiesLoop

Raoul -- no that's completely correct. Why do you think not?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5540610Thursday, January 03, 2008 4:01 AM by Raoul

Thanks for coming back. Your statement at the beginning of the article [Then there's this W3Schools article on ViewState that seems to indicate that posted form values are maintained via ViewState, but that's not true.]

seems to contradict

[MCTS Self-paced training kit (Exam 70-528), Microsoft Press, Page 501 - the page retains all its values and settings. This happens because ASP.NET has a client-side state management technique built in: ViewState]

?

http://weblogs.asp.net/infinitiesloop/archive/2006/08/03/Truly-Understanding-Viewstate.aspx - 5542831Thursday, January 03, 2008 2:33 PM by InfinitiesLoop

Raoul -- you're painting a broad brush. Both statements are true. ViewState is only one way controls maintain values across postbacks. Regular good ole HTML FORMS play a role, too. For example, disable viewstate on a textbox, and it will still maintain its value, because it is POSTING the value with the form. Make that TextBox invisible then do a post, and the value is lost. That's where ViewState helps, which would allow it to maintain the value even if it's invisible.

__theFormPostCollection, 63 __theFormPostData, 63