What do you do when you can't use ViewState?

Asked 16 years, 3 months ago Modified 12 years, 8 months ago

Viewed 3k times

7

I have a rather complex page that dynamically builds user controls inside of a repeater. This repeater must be bound during the Init page event before `ViewState` is initialized or the dynamically created user controls will not retain their state.

This creates an interesting Catch-22 because the object I bind the repeater to needs to be created on initial page load, and then persisted in memory until the user opts to leave or save.

Because I cannot use `ViewState` to store this object, yet have it available during Init, I have been forced to store it in Session.

This also has issues, because I have to explicitly null the session value during non postbacks in order to emulate how `ViewState` works.

There has to be a better way to state management in this scenario. Any ideas?

Edit: Some good suggestions about using `LoadViewState`, but I'm still having issues with state not

being restored when I do that.

Here is somewhat if the page structure

Page --> UserControl --> Repeater --> N amount of UserControls Dynamicly Created.

I put the overridden `LoadViewState` in the parent `UserControl`, as it is designed to be completely encapsulated and independent of the page it is on. I am wondering if that is where the problem is.

asp.net webforms viewstate page-lifecycle

Share

Improve this question

Follow

edited Apr 26, 2012 at 5:55

Charles Menguy

41.4k ● 18 ● 96 ● 117

asked Sep 5, 2008 at 3:16

FlySwat

175k ● 75 ● 248 ● 314

7 Answers

Sorted by: Highest score (default)

4

The LoadViewState method on the page is definitely the answer. Here's the general idea:

```
protected override void LoadViewState( object savedSta
  var savedStateArray = (object[])savedState;

  // Get repeaterData from view state before the norma
```

```
occurs.
  repeaterData = savedStateArray[ 0 ];

  // Bind your repeater control to repeaterData here.

  // Instruct ASP.NET to perform the normal restoratio
  // This will restore state to your dynamically creat
  base.LoadViewState( savedStateArray[ 1 ] );
}
```

SaveViewState needs to create the savedState array that we are using above:

```
protected override object SaveViewState() {
  var stateToSave = new List<object> { repeaterData, b
  return stateToSave.ToArray();
}
```

Don't forget to also bind the repeater in Init or Load using code like this:

```
if( !IsPostBack ) {
  // Bind your repeater here.
}
```

Share Improve this answer

Follow

answered Nov 9, 2008 at 20:26

William Gross

2,103 ● 2 ● 17 ● 35

1

> This also has issues, because I have to explicitly null the session value during non postbacks in order to emulate how ViewState works.

Why do you **have** to explicitly null out the value (aside from memory management, etc)? Is it not an option to check Page.IsPostback, and either do something with the Session variable or not?

Share Improve this answer

Follow

answered Sep 5, 2008 at 3:22

Greg Hurlman

17.8k ● 6 ● 55 ● 87

1

I have always recreated my dynamic controls in the LoadViewState event. You can store the number of controls needed to be created in the viewstate and then dynamically create that many of them using the LoadControl method inside the LoadViewState event. In this event you have access to the ViewState but it has not been restored to the controls on the page yet.

Share Improve this answer

Follow

answered Sep 5, 2008 at 7:11

DancesWithBamboo

4,156 ● 1 ● 20 ● 20

0

1) there's probably a way to get it to work... you just have to make sure to add your controls to the tree at the right moment. Too soon and you don't get ViewState. Too late and you don't get ViewState.

2) If you can't figure it out, maybe you can turn off viewstate for the hole page and then rely only on querystring for state changes? Any link that was

previously a postback would be a link to another URL (or a postback-redirect).

This can really reduce the weight of the page and make it easier to avoid issues with ViewState.

Share Improve this answer

Follow

answered Sep 5, 2008 at 14:32

Ben Scheirman

40.9k ● 21 ● 103 ● 139

0

```
protected override void LoadViewState(object savedStat
{
    // Put your code here before base is called
    base.LoadViewState(savedState);
}
```

Is that what you meant? Or did you mean in what order are the controls processed? I think the answer to that is it quasi-random.

Also, why can't you load the objects you bind to before Page_Load? It's ok to call your business layer at any time during the page lifecycle if you have to, with the exception of pre-render and anything after.

Share Improve this answer

Follow

edited Sep 5, 2008 at 17:05

answered Sep 5, 2008 at 16:59

Daniel Auger

12.6k ● 5 ● 53 ● 73

0

When creating dynamic controls ... I only populate them on the initial load. Afterwords I recreate the controls on postback in the page load event, and the viewstate seems to handle the repopulating of the values with no problems.

Share Improve this answer

Follow

answered Sep 7, 2008 at 1:05

mattruma
16.7k • 36 • 108 • 174

> I have to explicitly null the session value during non postbacks in order to emulate how ViewState works.

I'm still foggy as to why you can't store whatever object(s) you are binding against in session. If you could store that object in session the following should work:

1. On first load bind your top user control to the object during OnPreInit. Store the object in session. Viewstate will automatically be stored for those controls. If you have to bind the control the first time on Page_Load that is ok, but you'll end up having two events that call bind if you follow the next step.
2. On postback, rebind your top user user control in the OnPreInit method against the object you stored in session. All of your controls should be recreated before the viewstate load. Then when viewstate is restored, the values will be set to whatever is in viewstate. The only caveat here is that when you bind again on the postback, you have to make 100% sure that the same number of controls are created again. The key to using Repeaters, Gridviews etc... with dynamic controls inside of them is that they **have** to be rebound on every postback before the viewstate is loaded. OnPreInit is typically the best

> place to do this. There is no technical constraint in the framework that dictates that you must do all your work in Page_Load on the first load.

This should work. However, if you can't use session for some reason, then you'll have to take a slightly different approach such as storing whatever you are binding against in the database after you bind your control, then pulling it out of the database and rebinding again on every postback.

Am I missing some obvious detail about your situation? I know it can be very tricky to explain the subtleties of the situation without posting code.

EDIT: I changed all references to OnInit to OnPreInit in this solution. I forgot that MS introduced this new event in ASP.NET 2.0. According to their page lifecycle documentation, OnPreInit is where dynamic controls should be created/recreated.

Share Improve this answer Follow

edited Sep 7, 2008 at 15:02

answered Sep 6, 2008 at 21:50

Daniel Auger

12.6k ● 5 ● 53 ● 73