

# Chapter 12

## Managing ASP.NET State

*Everything is in a state ...*

Marcus Aurelius, *Meditations*,  
ix. 19

PRENTICE  
HALL

### core

# INTERNET

APPLICATION DEVELOPMENT  
WITH **ASP.NET 2.0**



▼ Get up to speed quickly with numerous real-world walkthrough exercises, code listings, in-depth examples, and snippets

▼ Learn valuable design principles and best practices of ASP.NET Web application development

▼ Gain in-depth knowledge of many key ASP.NET 2.0 topics, including controls, data binding, security, Web services, and a sneak peek at ASP.NET AJAX

RANDY CONNOLLY

---

# Overview

- This chapter is about state management in ASP.NET. State management is essential to any Web application, because every Web application has information that needs to be preserved from page to page.
- It covers:
  - Client-based state
  - Server-based state

## 2 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Why is state necessary?

- The vast majority of Internet applications operate using the HTTP protocol.
  - From an application developer's perspective, it does have one significant drawback:
    - HTTP is *stateless*.
  - This means that each request for a page is treated as a new request by the server.
    - As a result, information from one request is generally not available to the next request.
  - There are times, however, when it is quite helpful, or even necessary, for information from one request to be available to another.
- For this reason, most server-side dynamic Web technologies include some mechanism for managing and preserving state.

## 3 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# ASP.NET State Features

- ASP.NET includes a number of features for managing state on both a per-page basis and an application-wide basis:
  - Client-Stored
    - View state
    - Control state
    - Hidden fields
    - Querystrings
    - Cookies
  - Server-Stored
    - Profile properties
    - Application state
    - Session state

## 4 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# View State

- ASP.NET preserves the state of the page's controls between postbacks by packaging and encoding it within a hidden HTML `<input>` element with the name `__VIEWSTATE`.
- This view state feature can also be used programmatically to preserve additional information between postbacks for the same page

```
ViewState["RequestCount"] = count;
```

```
int count = (int)ViewState["RequestCount"];
```

## 5 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007  
[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# View State

- The information in the view state collection is serialized by the runtime into XML.
  - **Serialization** refers to the process of converting an object into a series of bytes in order to persist it to memory, a database, or a file.
  - The reverse process is called **deserialization**.
- You can add any custom type that can be serialized into XML to the view state, as well as any string, primitive number, Boolean, array, `ArrayList` object, or `Hashtable` object.

## 6 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Serialization

- To make a class serializable, you simply need to add the `Serializable` attribute to the class definition.
  - As well, all of its data members must be serializable.
  - Thus, any child objects stored in the class will also have to be marked with the `Serializable` attribute.

```
[Serializable]
public class Category
{
    private int _num;
    private string _name;
    private Market _market;
    ...
}
```

```
[Serializable]
public class Market
{
    ...
}
```

## 7 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007  
[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Uses of View State

- The view state collection is ideal for storing page-specific information of limited size.
  - Because the information is stored entirely within the view state sent to the client, it consumes no server memory.
  - The information in it is generally not meaningful to most users and hence is a bit more hidden than if it was in an HTML hidden field or in a querystring parameter.
  - However, even though the view state information is encoded, it is not encrypted and because it is still on the client, it can be tampered with by curious or malicious users.
    - It is possible though to encrypt it.

## 8 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Misuses of View State

- The main potential issue with view state is not really its security.
- Rather, the problem with view state is the fact that it can significantly add to the download size of the page.
  - You should be cautious with further increasing the downloaded page size by programmatically storing large amounts of additional data in the view state.
  - It is possible to disable view state for a page by using the `EnableViewState` attribute within the `Page` directive.

## 9 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Query Strings

- A querystring is the information that is appended to the end of a URL request.
- It begins with the ? character and is followed by name=value parameters separated by the & character.
- The querystring is **URL encoded**.
  - This means that special characters are replaced by either a % character followed by its two-digit hexadecimal representation, or a special reserved symbol (e.g., spaces are encoded with +).

`http://www.whatever.com/file.aspx?page=resum%E9&name=John+Locke`

## 10 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

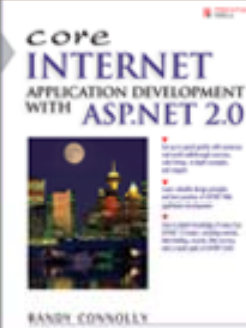
[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Query Strings

- A querystring can be used to submit data back to the same page or to another page through the URL.
  - Thus, querystrings provide a simple, universally supported but limited way of maintaining *some* state information.
  - The maximum allowable length of a querystring varies from browser to browser.
    - Internet Explorer limits the entire URL (which includes the domain address as well as the querystring) to about 2K, so clearly querystrings cannot be used to store anything more than a few simple primitive values.

## 11 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Query Strings

- A serious potential problem with querystrings is that the information contained within them is completely visible to the user in the browser's address bar.
- Some users quite commonly experiment by modify the querystring when making requests, so it is important to verify any information retrieved from a querystring.

```
string url = "productPage.aspx?id=" + id;  
Response.Redirect(url);
```

productPage.aspx

```
int id = Convert.ToInt32(Request["id"]);  
// use this id to construct SQL ...
```

## 12 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Cookies

- Cookies are a client-side approach for persisting state information.
- They are name-value pairs that are saved within one or more files that are managed by the browser.
- These pairs accompany both server requests and responses within the HTTP header

## 13 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Cookies

## 14 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007  
[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

Core  
INTERNET  
APPLICATION DEVELOPMENT  
WITH ASP.NET 2.0



RANDY CONNOLLY

# Cookies

- Cookies are not associated with a specific page but with the page's domain, so the browser and server exchanges cookie information no matter what page the user requests from the site.
- The browser manages the cookies for the different domains so that one domain's cookies are not transported to a different domain.

## 15 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Cookies

- Although cookies can be used for any state-related purpose, they are principally used as a way of maintaining continuity over time in a Web application.
  - One typical use of cookies in a Web site is to “remember” the visitor, so that the server can customize the site for the user. Some sites use cookies as part of their shopping cart implementation so that items added to the cart remain there even if the user leaves the site and then comes back.

## 16 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007  
[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Cookies

```
HttpCookie cookie = new HttpCookie("Name",txtName.Text);  
  
// Set expiry date to 1 day, 12 hours from now  
cookie.Expires = DateTime.Now + new TimeSpan(1, 12, 0, 0);  
Response.Cookies.Add(cookie);
```

```
HttpCookie cookie = Request.Cookies["name"];  
labCookie.Text = cookie.Value;
```

## 17 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Application State

- Application state is a server-stored state mechanism that allows you to store global data in a dictionary-style collection that is accessible from all pages in the Web application.
- It is stored in server memory, but unlike session state (covered shortly), which is specific to a single user browser session, application state applies to all users and sessions of the Web application.
  - Thus, application state is ideal for storing relatively small, frequently used sets of data that do not change from user-to-user or request-to-request.

```
Application["SiteRequestCount"] = 0;  
HttpContext.Current.Application["SiteName"] = "www.site.com";  
...  
int count = (int)Application["SiteRequestCount"];  
string name = (string)Application["SiteName"];
```

## 18 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Application State

- Items placed in application state remain there until the application restarts (for instance, because of a change to the Web.config file or due to ASP.NET's process recycling).

## 19 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Application State

- When should items in application state be initialized?
  - we generally cannot predict the first page to be requested in a Web application
- Two approaches:
  - use *lazy initialization* before you access any application variable.

```
if (Application["SiteRequestCount"] == null) {  
    Application["SiteRequestCount"] = 0;    // lazy initialization  
}  
  
int count = (int)Application["SiteRequestCount"];  
count++;  
Application["SiteRequestCount"] = count;
```

- This can result in synchronization problems if multiple requests execute this code simultaneously.

```
Application.Lock();  
if (Application["SiteRequestCount"] == null) {  
    Application["SiteRequestCount"] = 0;    // lazy initialization  
}  
  
int count = (int)Application["SiteRequestCount"];  
count++;  
Application["SiteRequestCount"] = count;  
Application.Unlock();
```

## 20 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Application State

- Because application state is instantiated when the application starts, it might make more sense to populate it at this point in the application lifecycle.
  - You can do this via the Application\_Start event handler in the Global.asax file.

```
<%@ Application Language="C#" %>
<script runat="server">
    void Application_Start(object sender, EventArgs e)
    {
        // Code that runs on application startup
        Application["SiteRequestCount"] = 0;
        Application["SiteName"] = "www.somesite.com";
    }
    ...
</script>
```

## 21 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Session State

- Session state is a server-based state mechanism that allows you to store data in a dictionary-style collection.
- It is scoped to the current browser session.
  - That is, each user or browser session has a different session state.
- Session state only lasts for a limited finite amount of time (usually around 20 minutes).
  - Thus, if a user leaves the application and then returns later, the second user session has a different session state from the first.

## 22 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Session State

- Session state is typically used for storing information that needs to be preserved across multiple requests by the same user.
- Because each user session has its own session state collection, it should not be used to store large chunks of information
  - this consumes very large amounts of server memory as loads increase.

## 23 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007  
[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Session State

- Because session information does eventually timeout, you should always check if an item retrieved from session state still exists before using the retrieved object.
- If the session object does not yet exist (either because it is the first time the user has requested it or because the session has timed out), one might generate an error, redirect to another page, or create the required object using lazy initialization.

```
if (Session["Cart"] == null) {  
    Session["Cart"] = new ShoppingCart();  
}  
ShoppingCart cart = (ShoppingCart)Session["Cart"];
```

## 24 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# How Does Session State Work?

- By default, a session ID value is transmitted between the browser and the server in a cookie.
- However, you can configure the application to use cookieless sessions via a setting in the Web.config file.
  - When cookieless sessions are enabled, the session ID is inserted into the URL.

`http://.../Ch11/(S(h1xh3ibe2htriazpxdne3b55))/Sample.aspx`

## 25 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007  
[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# How Does Session State Work?

- So what happens besides the generating or obtaining of a session ID after a new session is started?
- For a brand new session, the SessionStateModule assigns an initially empty dictionary-style collection that can be used to hold any state values for this session.
- When the request is finished processing, the session state is saved to some type of state storage mechanism, called a **state provider**.
- Finally, when a new request is received for an already existing session, the SessionStateModule fills the session's dictionary collection with the previously saved session data from the state provider.

## 26 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

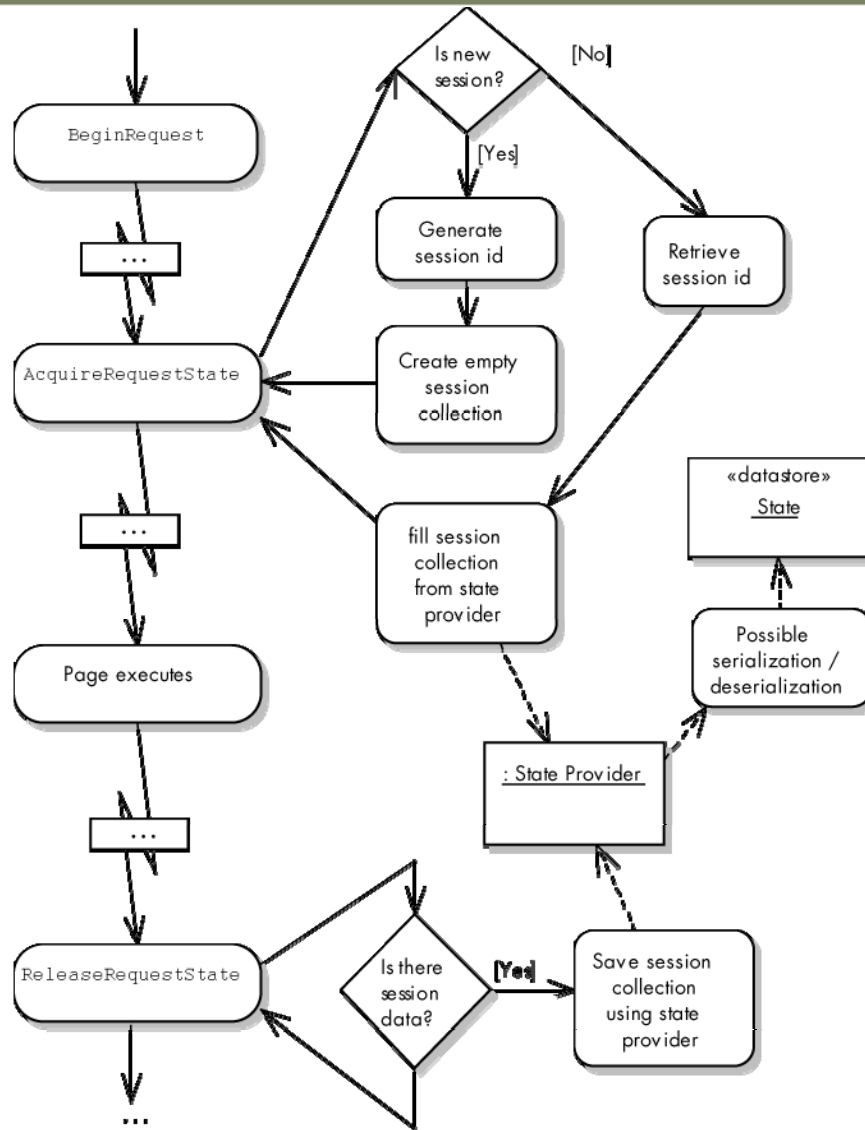
Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# How Does Session State Work?

## 27 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007  
[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

Core  
INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0



RANDY CONNOLLY

# State Providers

- ASP.NET supports three different session state providers, as well as the infrastructure for creating a custom state provider.
  - You can specify which state provider is to be used in your application via the mode attribute in the sessionState element in the Web.config file.
- They are:
  - *In-Process Session Provider*
  - *State Server Session Provider*
  - *SQL Server Session Provider*

## 28 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# In-Process Session Provider

- By default, state data is saved on the Web server within the Web application's in-memory cache.
  - This default state provider is called the InProc (or in-process) provider, because it is run in the same process as the Web application itself.
- InProc provides the fastest access to session state.
  - If your site is going to run on a single Web server and there is a fairly modest number of concurrent sessions, the in-process state provider provides the best performance.

## 29 State Management

COPYRIGHT © 2007 RANDY CONNOLLY

core  
INTERNET  
APPLICATION DEVELOPMENT  
WITH ASP.NET 2.0

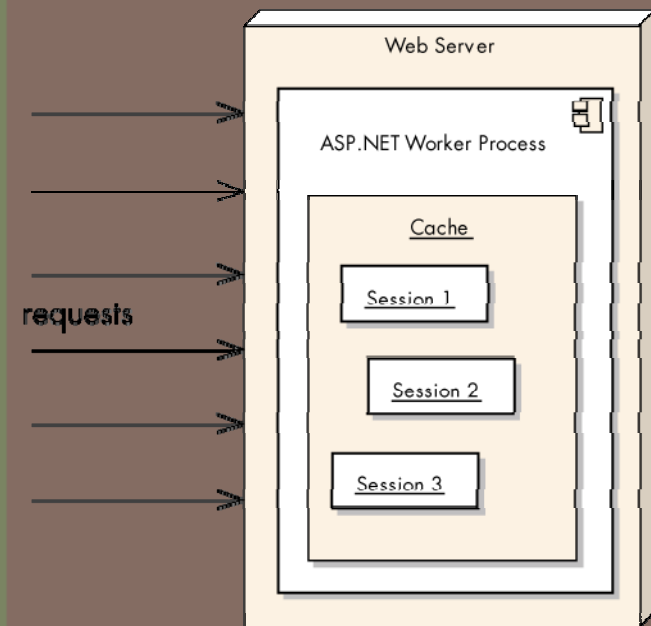


RANDY CONNOLLY

CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)



## Other Session Providers

- There are situations in which in-process state provider is not an ideal choice.
  - If there are a large number of concurrent sessions, the Web application may run out of memory because in-process session state uses the same memory as the application.
  - As well, in-process session state is lost whenever the application process is recycled (recall that the ASP.NET worker process is periodically restarted).
  - Finally, higher-volume Web applications often run on a multiprocessor Web server in which each processor handles a separate worker process (sometimes called a *Web garden*) or on multiple Web servers (also called a *Web farm*).

### 30 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

## Other Session Providers

- For these situations, in-process session state does not work, because one processor or server may service one request for a particular session, and then a completely different processor or server may service the next request for that session.
- The other state providers store the session's state data in the out-of-process Win32 state service (StateServer provider), or in a special SQL Server database (SQLServer provider).

### 31 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# State Server Session Provider

- The StateServer provider uses the session state service, which is a Win32 service (aspnet\_state.exe) that must be running on the Web server or on a remote server (generally preferable) that is accessible by all the Web servers in the Web farm.
- Because the session state service runs as a separate process, it is insulated from IIS failures or to ASP.NET process recycling

## 32 State Management

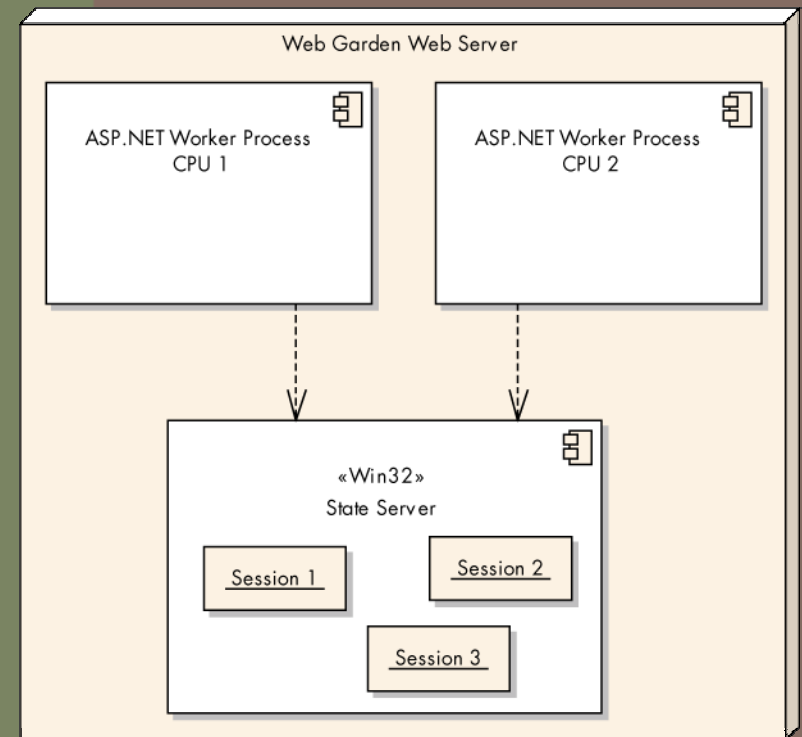
COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)



# SQL Server Session Provider

- The capability to store session state within a special SQL Server database provides a very reliable alternative to the session state service.
  - Because the session data is stored in a database table, it is preserved even if the Web server crashes.
  - However, because the session is being persisted to a database, it certainly will be slower than the other two state providers

## 33 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

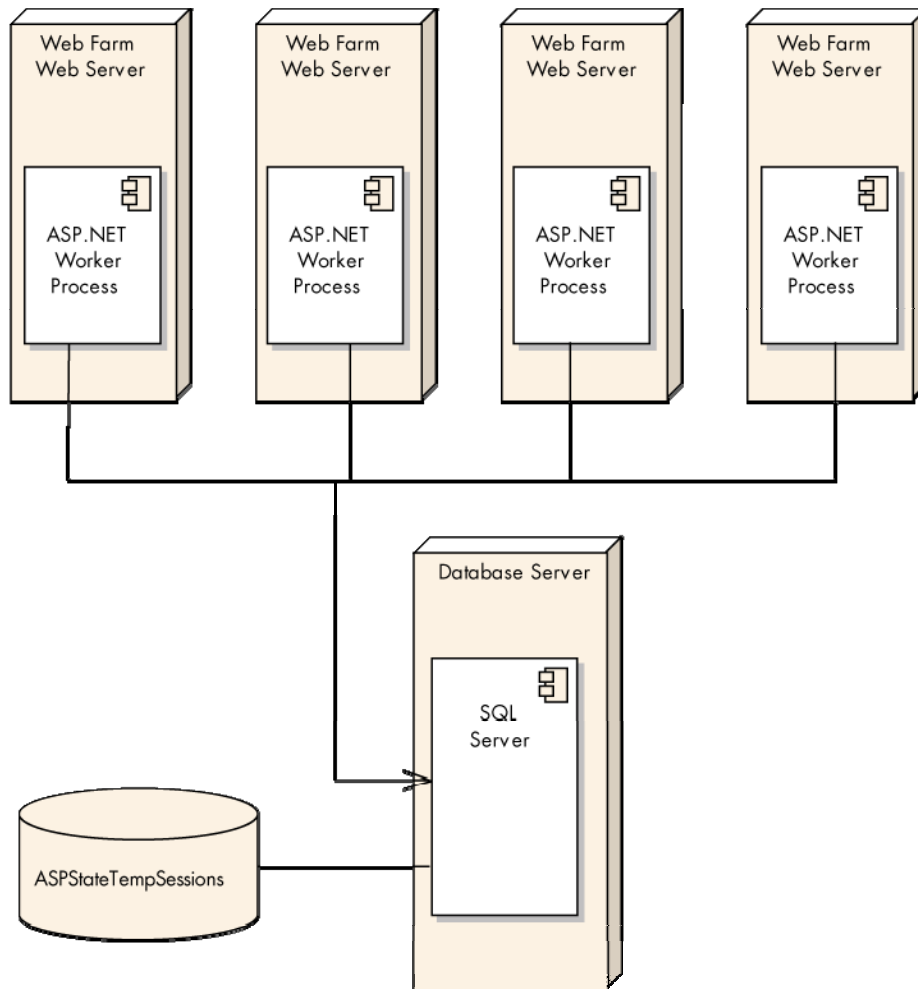
Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# SQL Server Session Provider

34 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007  
[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

Core  
INTERNET  
APPLICATION DEVELOPMENT  
WITH ASP.NET 2.0



RANDY CONNOLLY

# ASP.NET Cache

- The ASP.NET cache is contained within the memory space of a single ASP.NET worker process.
  - It is a dictionary-style collection that can contain any type of object.
- The cache is different than the application state mechanism in that application state data exists as long as the process for the application domain is running while cache data items have a more variable lifespan.

## 35 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Cache

- The cache is of flexible size and eliminates items that are less recently used.
  - To this end, items within the cache are time stamped and removed after a certain time period (although an item can be configured to have no expiry time).

## 36 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)

# Cache

- There are two different caching mechanisms in ASP.NET.
  - *application data caching*,
    - which allows you to programmatically cache data.

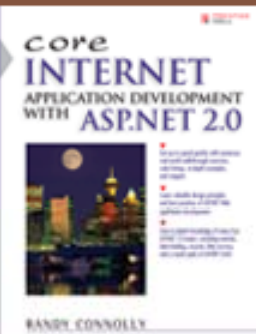
```
BookCatalogLogic bcl = new BookCatalogLogic();  
publishers = bcl.GetAllPublishers();  
  
Cache["PublisherKey"] = publishers;
```

- *page output caching*,
  - which saves the rendered output of a page or user control and reuses the output instead of reprocessing the page when a user requests the page again.
  - This allows ASP.NET to send a page response to a client without going through the entire page processing lifecycle again.
  - is especially useful for pages whose content does not change frequently, but requires significant processing to create.

```
<%@ OutputCache Duration="60" VaryByParam="None" %>
```

## 37 State Management

COPYRIGHT © 2007 RANDY CONNOLLY



CORE INTERNET  
APPLICATION DEVELOPMENT  
WITH  
ASP.NET 2.0

Prentice Hall, 2007

[www.randyconnolly.com/core](http://www.randyconnolly.com/core)