State Management

Upphovsrätt för detta verk

Detta verk är framtaget i anslutning till kursen ASP.NET MVC vid Linnéuniversitetet.

Du får använda detta verk så här:

Allt innehåll i detta verk av Mats Loock, förutom Linnéuniversitetets logotyp och symbol samt ikoner, bilder och fotografier, är licensierad under:

Creative Commons Erkännande-IckeKommersiell-DelaLika 2.5 Sverige licens.
http://creativecommons.org/licenses/by-nc-sa/2.5/se/

Det betyder att du i icke-kommersiella syften får:

- kopiera hela eller delar av innehållet
- sprida hela eller delar av innehållet
- visa hela eller delar av innehållet offentligt och digitalt
- konvertera innehållet till annat format
- du får även göra om innehållet

Om du förändrar innehållet så ta inte med Linnéuniversitetets logotyp och symbol samt ikoner och fotografier i din nya version!

Vid all användning måste du ange källan: ”Linnéuniversitetet – ASP.NET MVC” och en länk till https://coursepress.lnu.se/kurs/aspnet-mvc och till Creative Common-licensen här ovan.

Vad händer med mina objekt och värden?

- Varje gång webbservern kör din sida initieras allt och från början, objekten som representerar sidan på servern skapas, renderas ut till klienten och till sist ”förstörs” alla objekt och variabler.
- Detta leder till problem när variablers värden ska behållas mellan ”postbacks” eller över flera sidor i en webbapplikation.
- Problemet har flera lösningar, som passar olika bra beroende av sammanhanget.

View State

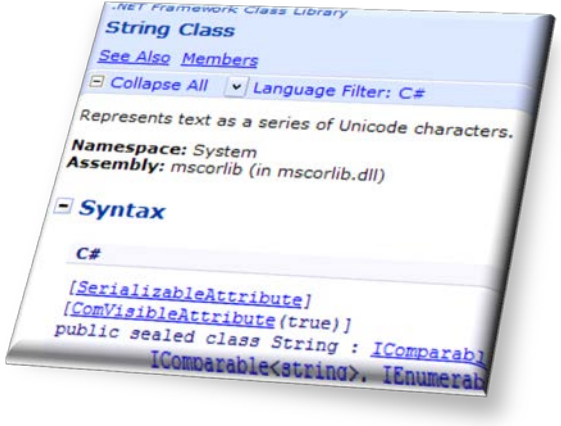
- ✓ Webbkontroller använder sig av ”view state” för att bevara status mellan ”postbacks”.
- ✓ Du kan också använda dig av en sidas ”view state” att spara data i.

```
// Sparar undan ett värde i en sidas "view state".
ViewState["myValue"] = 10;
```

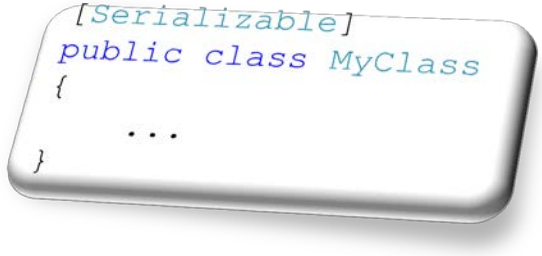
```
// En variabel som tar emot värdet i sidans "view state".
int myValue = 0;

// Kontrollera alltid att värdet finns i sidans "view state",
// innan du försöker hämta värdet (eller använd en try-catch-sats).
if (ViewState["myValue"] != null)
{
    myValue = (int)ViewState["myValue"];
}
```

Spara objekts data - serialisering

- För att kunna spara objekts data i en sidas ”view state” måste objektet gå att serialisera.



- Vilka klasser som är serialiseringsbara hittar du enklast i online-hjälpen.



- Klasser du skriver själv gör du serialiseringsbara med hjälp av attributet `Serializable`.

Fördelar och nackdelar med ”view state”

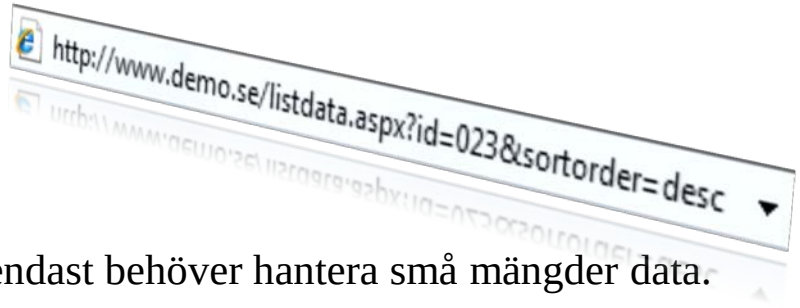
- ✓ Fördelar
  - Enkelt att använda.
  - Tar inte upp något minne på server.
  - Finns ingen tidsbegränsning.

- ✓ Nackdelar
  - Ingen säker lösning.
  - Kan inte användas över flera olika sidor, bara ”postbacks”.
  - Ingen bra lösning när du behöver hanterar stora datamängder.

ViewState och säkerhet

- Det är lätt att tro att datat i ”view state” är krypterad när du ser den. Så är dock inte riktigt fallet. Vill du använda kryptering finns två lösningar:
  - Använda ”hash code”.
    - `<%@ Page EnableViewStateMAC=”true” %>`
      Använder ”checksum” som försvårar att en ”view state” manipuleras.
    - Kan dock knäckas.
    - Använder maskinspecifik nyckel för att beräkna ”checksum”.
    - Är aktiverad som standard.
  - Använda ”view state encryption”.
    - `<%@ Page ViewStateEncryptionMode=”always” %>`
    - Använder en maskinspecifik nyckel för krypteringsalgoritmen.

Query String



- ✓ Fördelar
  - Oerhört enkelt att skapa när man endast behöver hantera små mängder data.
  - Belastar inte servern.
  - Datat tillgängligt över flera sidor (inte bara i samband med ”postbacks”).
- ✓ Nackdelar
  - Bara enkla URL-tillåtna strängar fungerar.
  - Informationen syns väl och är lätt att manipulera.
  - Många webbläsare har en begränsning på hur långa strängarna kan vara, men minst 2 kbyte data kan överföras enligt standarden.

Att använda ”querystring”

```
// Lägga till data.
int id = 10;
Response.Redirect("getData.aspx?id=" + id.ToString());
```

```
// Hämta data.
int dataID = Convert.ToInt32(Request.QueryString["id"]);
```

- URL Encoding
  - Det är viktigt att säkerställa att strängen som skickas innehåller tecken som kan tolkas av alla webbläsare.

```
// Lägga till data
string str = "sträng med mellanslag";
Response.Redirect("show.aspx?str=" + Server.UrlEncode(str));
```

Cookies

- Användande av ”cookies” kräver mer kod, bl.a. kan du ange hur länge datat ska sparas på klienten.

```
// Skapa ett cookieobjekt
HttpCookie myCookie = new HttpCookie("personal");

// Tilldela kakan ett värde
myCookie["shoeSize"] = "42";

// Måste lägga till det till "Response"-strömmen.
Response.Cookies.Add(myCookie);

// Hur länge ska cookien finnas hos användaren?
myCookie.Expires = DateTime.Now.AddMonth(2);
```

```
// Läsa cookie.
HttpCookie myCookie = Request.Cookies["personal"];

// En variabel som lagrar värdet cookien har.
int shoeSize = 0;

// Kontrollera alltid att cookien finns innan du försöker
// hämta värdet (eller använd en try-catch-sats).
if (myCookie != null)
{
    shoeSize = Convert.ToInt32(myCookie["shoeSize"]);
```

⚠ Klassen HttpCookie finns i namnrymden System.Web, inte System.Net som det felaktigt står på sidan 256 i kurslitteraturen.

Fördelar och nackdelar med cookies

- ✓ Fördelar
  - Kan spara data över längre tid än en session.
  - Enkelt att använda i vilken sida som helst i applikationen (märks inte hos användaren).
- ✓ Nackdelar
  - Vissa användare blockerar cookies.
  - Enkla att snappa upp och läsa av, samt ändra.
  - Inget bra sätt att spara större mängder av data.
  - (Fungerar kanske inte i mobila enheter.)

Session State

- Sessionsvariabler är en lite mer avancerad teknik för att lösa problemet med bevarande av data mellan ”postbacks”.
- Hanteringen av sessionsvariabler påminner syntaxmässigt mycket om hur ”view state” fungerar, men datat sparas på servern, och det går att komma åt datat från (nästan) alla delar av applikationen.
- En användare har ett unikt sessionsid, som sessionsvariablerna är kopplade till.
  - Vanligt i samband med kundvagn på e-handelswebbplatser.
- Nackdelen är att det tar upp minne på webbservern.
- Det är möjligt att spara alla typer av objekt i en sessionsvariabel.

Session – så fungerar det

- ✓ Eftersom denna teknik inte finns i HTTP-protokollet är det ASP.NET som har hand om det.
- ✓ Lösningen är ett 120-bitars ID-nummer som är unikt för varje användare och så pass slumpartat så att det inte går att gissa sig till.
- ✓ Detta ID är det enda som skickas mellan klient och server.
  - Antingen via ”cookie” eller modifierad URL.
- ✓ Sessionsdata kan sparas på flera olika sätt:
  - In-Process – sparas i primärminnet på den server där applikationen körs. (standard)
  - StateServer – sparas i primärminnet på en separat server.
  - ASPState Database – sparas i en SQL Server.

Hur använda och livslängd

```
// Spara i en sessionsvariabel.
ArrayList myArrayList = new ArrayList();
myArrayList.Add(1);
myArrayList.Add(12);
myArrayList.Add(35);
Session["myArray"] = myArrayList;

// Hämta efter postback eller på en annan sida.
ArrayList newArrList = (ArrayList)Session["myArray"];
```

- Sessionsdata försvinner när…
  - …det sker en time-out (20 minuter i regel).
  - …användaren loggar ut från webbapplikationen (om användaren loggat in).
  - …`Session.Abandon()` exekveras.
  - …applikationsdomänen går ner eller startas om (ifall man inte sparar sessionsdata i databas eller i extern process).

Några egenskaper och metoder

- `Count`
  - Antalet objekt som finns i sessionsobjektet.
- `Mode`
  - Talar om hur sessionen ska sparas (off, in-proc, stateserver, sqlserver).
- `SessionID`
  - En sträng med det unika sessionsid:et.
- `TimeOut`
  - Antalet minuter innan sessionen gör timeout.
- `Abandon()`
  - Stänger och avsluta sessionen.
- `Clear()`
  - Tömmer all data men behåller sessionen.

Säkerhet

- Sessionsdata är, i stort, säkert eftersom det sparas på servern.
- Dock kan elaka personer stjäla ett sessionsid (”session hijacking”) och på så sätt få tillgång till sessionen.
- Enda sättet du kan skydda användare mot detta är ”Sequre Socket Layer” (SSL) / HTTPS där trafiken mellan klient och server krypteras.

Application

- Sparar globala objekt som man kan komma åt från alla sidor i applikationen.
- Påminner om hur sessionsvariabler i sättet det sparas, syntax och typer av objekt man kan spara.
- Används inte så ofta p.g.a. att det är tämligen ineffektivt.

Använd Cache istället!

Sammanfattning

	View State	Query String	Cookies	Session State	Application
Tillåtna datatyper	Alla serialiserings-bara	URL-korrekta textsträngar	Textsträngar	Alla serialiserings-bara (andra om in-proc)	Alla datatyper
Lagring	type=”hidden”	Webbläsarens URL	Textfiler på användarens dator	Serverns minne	Serverns minne
Livstid	Permanent för ”postbacks”	När webbläsaren stängs	Sätter programmeraren eller när användaren tar bort dem	Timeout	Applikationens livstid
Scope	Bara samma sida	Den sida URL:en pekar på	Hela applikationen	Hela applikationen	Hela Applikationen, alla användare
Säkerhet	Osäkert	Osäkert	Osäkert	Ganska säkert	Säkert
Kapacitet	Stora datamänger negativt för prestandan	Försumbar påverkan av prestandan vid små datamängder	Försumbar påverkan av prestandan vid små datamängder	Stora datamänger negativt för prestandan	Mycket data kan försämra serverns prestanda
Användning	Sidspecifika inställningar	Skicka t.ex. ID för att hämta ut data	Personliga inställningar för en sida	Kundvagn	Lagra global data

Cross-Page Posting

- Det går att skapa en ”postback” till en annan sida än den som skapade sidan.
- När den andra sidan stöter på `PreviousPage` skapar ASP.NET om hela den sidan men avbryter strax innan `PreRender()`. Vidare ignoreras alla `Response.Write()`.
- `Page_Load`, `Page_Init` och eventuella hanterarmetoder körs.

```
<!-- i Page1.aspx -->
<asp:Button id="Button1" Text="Spara" runat="server" PostBackUrl="Page2.aspx" />
```

```
// I code-behind till Page2.aspx.
if (PreviousPage != null)
{
    string oldPageTitle = PreviousPage.Header.Title;
}
```