

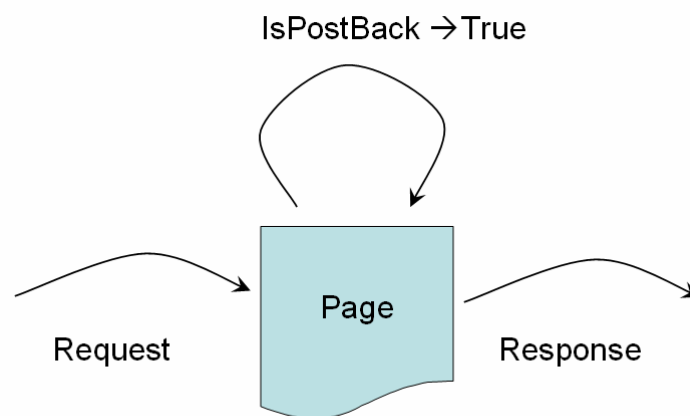
Accesso protetto ad una applicazione web

Può capitare, ed anzi capita spesso, di aver sviluppato un'applicazione web in cui l'utente si autentica in una pagina di login e poi naviga le pagine a cui viene indirizzato, magari diverse a seconda della sua identità e/o ruolo.

Il problema è che senza specifici accorgimenti gli indirizzi delle pagine e le corrispondenti query rimangono memorizzate nella lista degli indirizzi del browser. Segue che salvandosi ad esempio un collegamento ad una pagina interna, cliccandoci magari il giorno dopo si accede direttamente all'applicativo senza passare dalla login.

Un modo per evitare comportamenti di questo tipo è utilizzare le cosiddette Sessioni di ASP.NET. In pratica una volta eseguita l'autenticazione si dichiara una opportuna variabile (es. LoggedIn) che viene mantenuta nel passaggio da una pagina all'altra. Tutte le pagine eseguiranno le loro attività solo se LoggedIn=True riportando alla login se LoggedIn=NULL.

Qui di seguito riporto un esempio pratico in proposito. Molto sinteticamente (e semplicisticamente) giova ricordare che questo è il ciclo di una pagina ASP.NET:



L'oggetto Page si identifica con il contesto corrente http (`HttpContext.Current`).

Utilizzo delle Sessioni : un esempio pratico

Costruiamo un sito di esempio rispondente all'indirizzo <http://www.pippo.com/Prova> costituito da due pagine, ad esempio:

- Default.aspx (*Start Page*)
- Form3.aspx

...posizionate sulla root del sito. I nomi chiaramente sono casuali : c'è form3 e non 1 e 2 perché queste le ho utilizzate per altre prove e non sono significative in questo contesto. Nel Web Config mettiamo qualcosa del genere:

```
<?xml version="1.0"?>
<configuration>
  <appSettings/>
  <connectionStrings/>
  <system.web>
    <compilation debug="true"/>
  </system.web>
</configuration>
```

```
<sessionState mode="InProc" timeout="30"></sessionState>
</system.web>
</configuration>
```

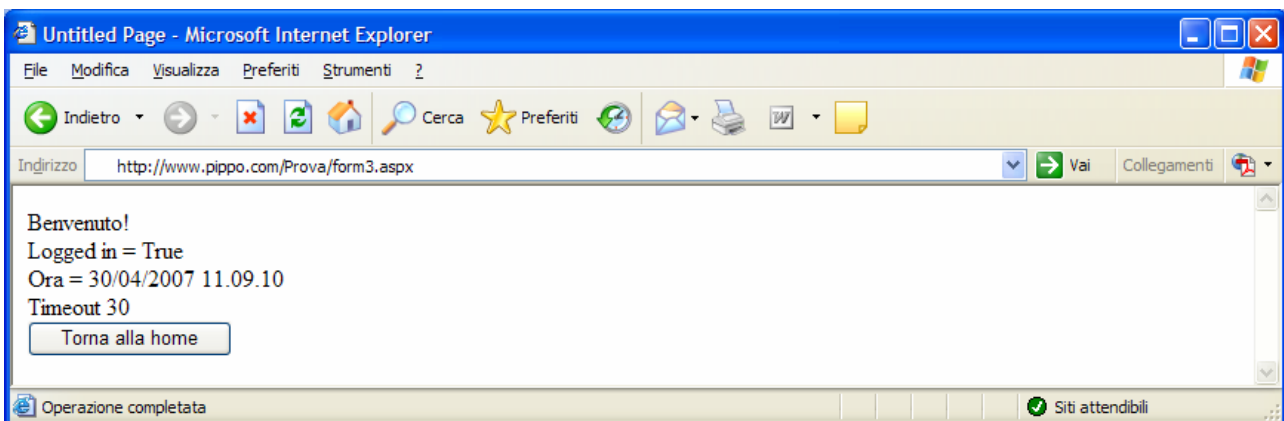
..ovvero abilitiamo nella nostra applicazione la gestione degli stati della sessione. Nella Default.aspx mettiamo poi un bottone e questo codice:

```
protected void Button2_Click(object sender, EventArgs e)
{
    Page.Session.Add("LoggedIn", "True");
    Response.Redirect("http://www.pippo.com/Prova/form3.aspx");
}
```

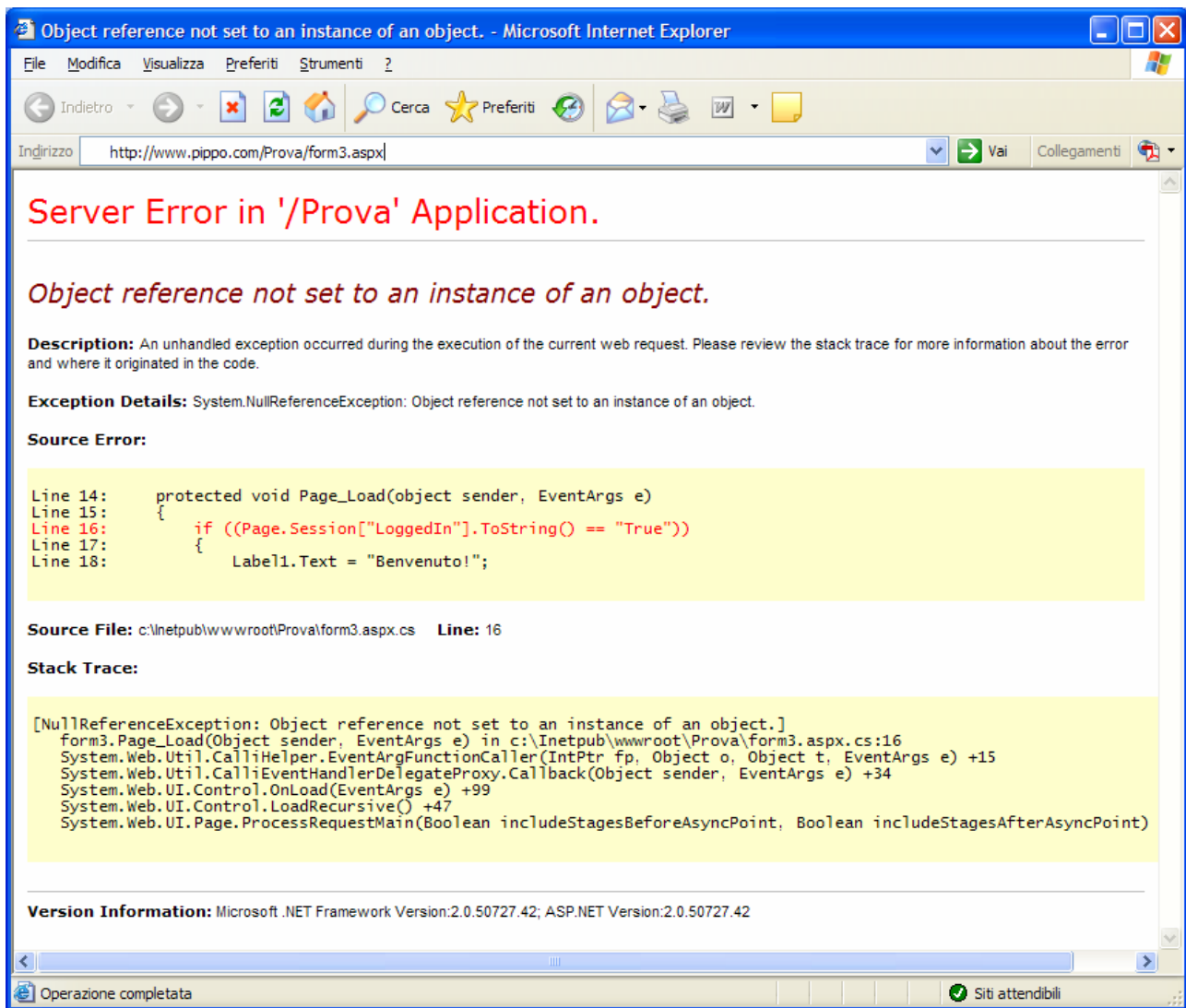
Nella Form3.aspx mettiamo dei Label (in questo caso 1,2,3,4) per evidenziare le informazioni ricevute (e magari qualche altra se ci serve): Per farlo inseriamo il seguente codice:

```
protected void Page_Load(object sender, EventArgs e)
{
    if ((Page.Session["LoggedIn"].ToString() == "True"))
    {
        Label1.Text = "Benvenuto!";
        Label2.Text = "Logged in = " +
HttpContext.Current.Session["LoggedIn"].ToString();
        Label3.Text = "Ora = " + DateTime.Now.ToString();
        Label4.Text = "Timeout " +
HttpContext.Current.Session.Timeout.ToString();
    }
    else
    {
        Response.Redirect("http://www.pippo.com/Prova/Default.aspx");
    }
}
```

Lanciamo ora la Default, ovvero digitiamo nel browser l'indirizzo <http://www.pippo.com>. Premendo in nostro bottone viene visualizzata la Form3.aspx con i dati della sessione:



Se proviamo a chiamare direttamente la form3 otteniamo il seguente errore:



..perchè LoggedIn=null. Quindi, in sintesi la Default apre una sessione e consente l'accesso alle altre pagine. L'accesso diretto a queste è impossibile.

Nota : le sessioni sono definite a livello di dominio. Quindi se il sito è sotto pippo.com e all'interno di questo ambito le pagine vengono chiamate (come nel mio esempio) attenzione al fatto che quando fate il debug Visual Studio non lanci <http://localhost/Prova> perchè il cambio di dominio da localhost a pippo.com causa la perdita della variabile stato sessione.