מבוא לניהול ה - State

כאשר הדפדפן גולש באינטרנט הוא למעשה פונה לשרת בבקשה (Request) לקבל דף. השרת מקבל את הבקשה, מעבד אותה ומחזיר תשובה (Response) לדפדפן. הבקשה והתשובה הינן הודעות בפרוטוקול בשם http. פרוטוקול זה מוגדר כ - stateless,משמעות הדבר היא שלאחר שהתשובה הגיעה אל הדפדפן, אין שום קשר בין הדפדפן ובין השרת. למעשה ברוב המקרים גם אם ננתק את הכבל המחבר אותנו אל האינטרנט עדיין נוכל לראות את הדף כל עוד לא נבקש בקשה חדשה מהשרת. מכיוון שהגלישה באינטרנט היא stateless ישנו קושי לדעת פרטים על דברים שקורים בין הצדדים.

לדוגמא: נרצה לספור (בצד השרת) כמה פעמים משתמש ניסה לבצע Login.

דוגמא נוספת: נרצה להציג רשימה של מוצרים ובלחיצה על מוצר נפנה את המשתמש לדף המציג את פרטי המוצר. הדף המציג את פרטי המוצר לא יודע על איזה מוצר המשתמש לחץ בדף המציג את רשימת המוצרים.

דוגמא נוספת: המשתמש מבצע Login לאתר ובמעבר בין הדפים השרת לא יודע מי המשתמש והאם הוא ביצע Login.

ניהול ה - State מספק לנו טכניקות וכלים המאפשרים לנו לפתור בעיות אלו. לכל טכניקה יש יתרונות וחסרונות כגון: אבטחה, כמה זמן הנתונים נשמרים וכד'. בשיעור זה נדון במספר אובייקטים, שהם הבסיס לניהול ה - State ובפרקים הבאים נכיר מספר טכניקות לניהול ה - State.

[01 Post Get Response Request]

האובייקט Request:

אובייקט המאפשר לשרת לקבל מידע לגבי הבקשה שהגיעה כגון: סוג הדפדפן, סוגי הקונטרולים ותוכנם, סוג בקשת ה HTTP- האם GET או POST. ברירת המחדל עבור הבקשה הראשונה היא GET כי היא מגיע דרך הכתובת. ברירת המחדל של הבקשות הבאות היא POST עבור דפי ASPX.

למשל:

קריאת נתונים משדה שנישלח ב POST:

```
Request.Form["txtName"]
```

קריאת נתונים משדה שנישלח ב GET:

```
Request.QueryString["txtName"]
```

האובייקט Response:

אובייקט המאפשר לשרת לשלוט בתגובה שתישלח ללקוח כגון: העברת הלקוח לדף אחר.

להלן דוגמה:

העברה לעמוד YNET:

```
Response.Redirect("http://www.ynet.co.il");
```

"שפיכה" של HTML לראש העמוד:

```
Response.Write("POST- from Form:" + item + "<br />");
```

QueryString

אובייקט שמכיל נתונים המועברים ע"י טכניקת GET. זו מאפשרת להעביר מידע בין דפים דרך כתובת ה - URL. הרעיון הוא שהדף השולח יעביר פרמטר (או כמה) לדף המקבל, והדף המקבל יקרא את הפרמטר שהועבר.

מבנה ה - QueryString:

http://.....aspx?p1=5&p2=10

כאשר כל מה שנמצא לאחר ה ? הוא חלק מה - QueryString ובדוגמא שלנו ישנם שני פרמטרים p1,p2 שהערכים שלהם 5,10 (בהתאמה).

מאפיינים של QueryString:

- חשוף לשינויים מצד הלקוח
- נשמר רק בין שני דפים
- מאפשר למשתמש לשמור את הכתובת כולל ה - QueryString כדי להגיע לדף באותו מצב בו המשתמש היה בו = bookmarking
- מוגבל בגודל

Form

אובייקט המכיל את הנתונים ומועברים ע"י הטכניקה POST המאפשרת להעביר מידע בין דפים בתוך הדף עצמו ולא דרך כתובת ה - URL. הרעיון הוא שהדף השולח יעביר פרמטר (או כמה) לדף המקבל, והדף המקבל יקרא את הפרמטר שהועבר.

מאפיינים של Post:

- מוחבא מהלקוח
- נשמר רק בין שני דפים
- אין אפשרות לעשות bookmarking
- לא מוגבל בגודל

View State

[02_1 ViewState]

View State הינו שדה hidden הנמצא בכל דף ומטרתו לשמור את מצב הפקדים בדף. למעשה בכל פעם שנעשה שינוי במאפיין של פקד מסוים מצד השרת הערך של המאפיין ישמר בשדה זה. המטרה של שדה זה היא לחסוך מאיתנו את הצורך לזכור בצד השרת את כל השינויים שעשינו.

השינויים שנישמרים בשדה הזה נשמרים בסוף התהליך שנעשה בשרת. השינויים הללו כוללים את כל ההבדל בין ה MARKUP למצב הנוכחי. ואז השדה הזה מוצפן ונשלח בגוף הדף. המשתמש עושה את השינויים שלו בצד הדפדפן ואז שולח חזרה את הדף. עכשיו השרת יודע מה הוא קיבל מהמשתמש וע"י השדה הזה אפשר לראות מה השתנה בין המצב הנוכחי למצב בו היה הדף כאשר הוא נשלח בפעם הקודמת.

ניתן גם לבטל את ה - View State לפקד מסוים, לכל הדף או ברמת כל האתר, באמצעות עדכון המאפיין EnableViewState של הפקד ל - false. דבר זה יגרום לכך שהשדה ה - hidden יקטן, אך שינויים שנעשה בפקדים (כגון שינוי צבע) לא יזכרו. שים לב שאם ה VIEWSTATEמופסק ברמת הדף אז לא ניתן להפעילו ברמת הפקד.

חשוב לדעת כי הפקדים הבאים ישמרו **חלק מערכם** גם כאשר בוחרים EnableViewState=false בכל מקרה המערכת שמה אותם ב VIEWSTATE (הם מממשים *IPostBackDataHandler*)!!! :

- CheckBox
- CheckBoxList
- DropDownList
- HtmlInputCheckBox
- HtmlInputFile
- HtmlInputHidden
- HtmlInputImage
- HtmlInputRadioButton
- HtmlInputText
- HtmlSelect
- HtmlTextArea
- ImageButton
- ListBox
- RadioButtonList
- TextBox

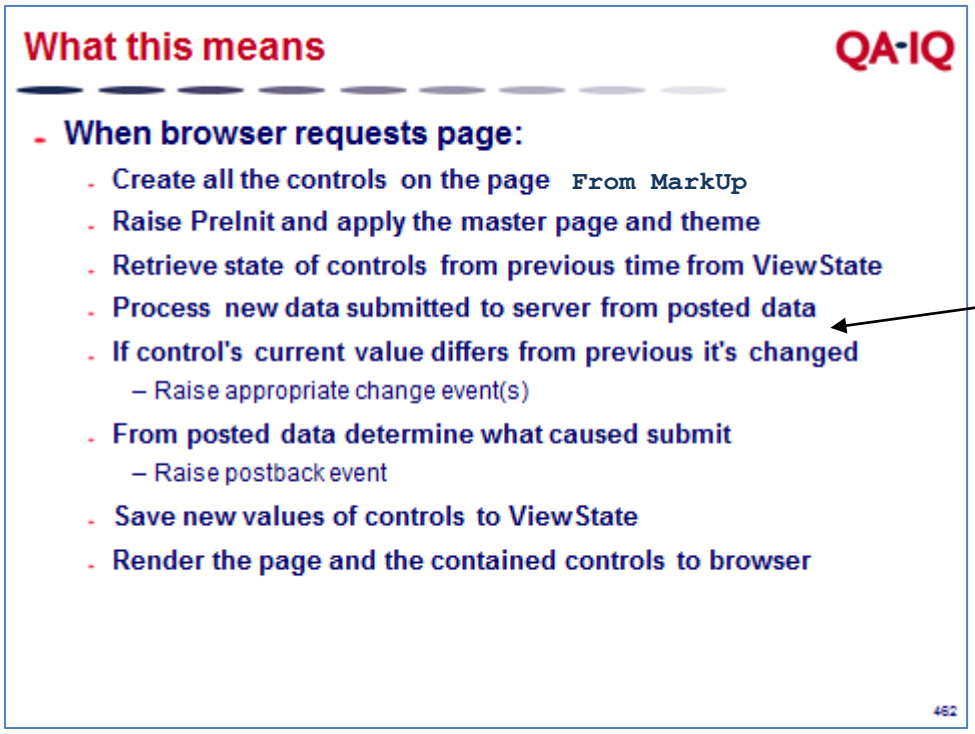
כתבתי **חלק מערכם** כי למשל את הצבע רקע הם לא ישמרו! נסתכל על הדוגמה האחרונה של השיעור בכדי להבין יותר לעומק את הנושא

04 ViewState Object / TextBox.aspx

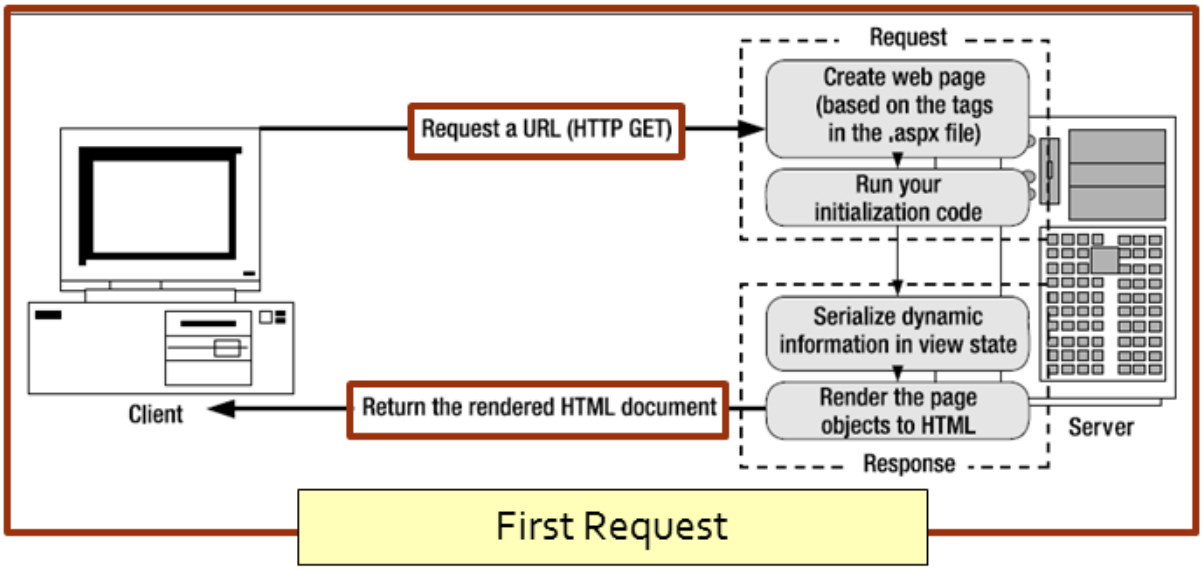
יתרון : כאשר מורידים את ה VIEWSTATE עבור פקד הדף יותר קטן ואז התקשורת יותר מהירה.
בדרך כלל זניח, אלה אם כן הדף מאוד עמוס בפקדים.

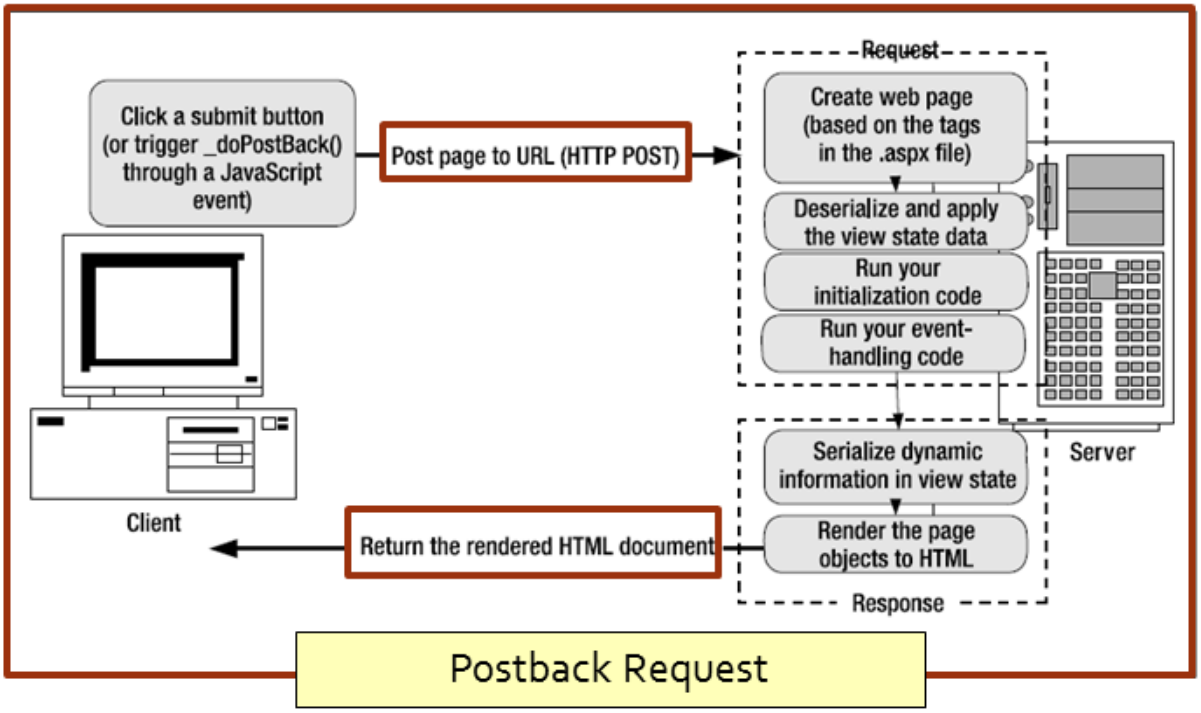
Life cycle

http://msdn.microsoft.com/en-us/library/ms972976.aspx



Load_page - after the viewstate and potdata are processed. Before the change events and postdata event


Request
Request a URL (HTTP GET)
Create web page (based on the tags in the .aspx file)
Run your initialization code
Serialize dynamic information in view state
Render the page objects to HTML
Return the rendered HTML document
Client
Server
Response
First Request


Request
Click a submit button (or trigger _doPostBack() through a JavaScript event)
Post page to URL (HTTP POST)
Create web page (based on the tags in the .aspx file)
Deserialize and apply the view state data
Run your initialization code
Run your event-handling code
Serialize dynamic information in view state
Render the page objects to HTML
Return the rendered HTML document
Client
Server
Response
Postback Request

❖ When **posted back** a page, ASP.NET follows these steps:

1. **ASP.NET re-creates the page and control objects based on its defaults**
   (as defined in the **.aspx** file).
   Thus, the page has the same state that it had when it was first requested.

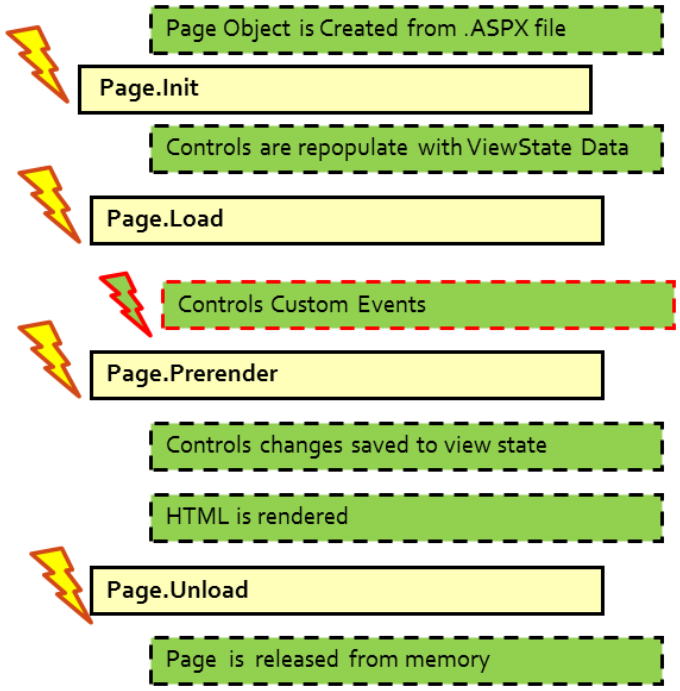
2. **Next, ASP.NET desterilizes the view state information and updates all the controls.** This returns the page to the state it was in before it was sent to the client the last time.

3. **Finally, ASP.NET adjusts the page according to the posted back form data.**
   For example, if the client has entered new text in a text box or made a new selection in a list box , that information will be in the Form collection and ASP.NET will use it to tweak the corresponding controls.
   After this step, the page reflects the current state as it appears to the user.

4. **Now your event-handling code can get involved. ASP.NET triggers the appropriate events**
   your code can react to change the page and Controls

❖ Here is the sequence of events that are raised whenever you request a page :

1. PreInit
2. Init
3. InitComplete
4. PreLoad
5. **Load**
6. LoadComplete
7. **PreRender**
8. PreRenderComplete
9. SaveStateComplete

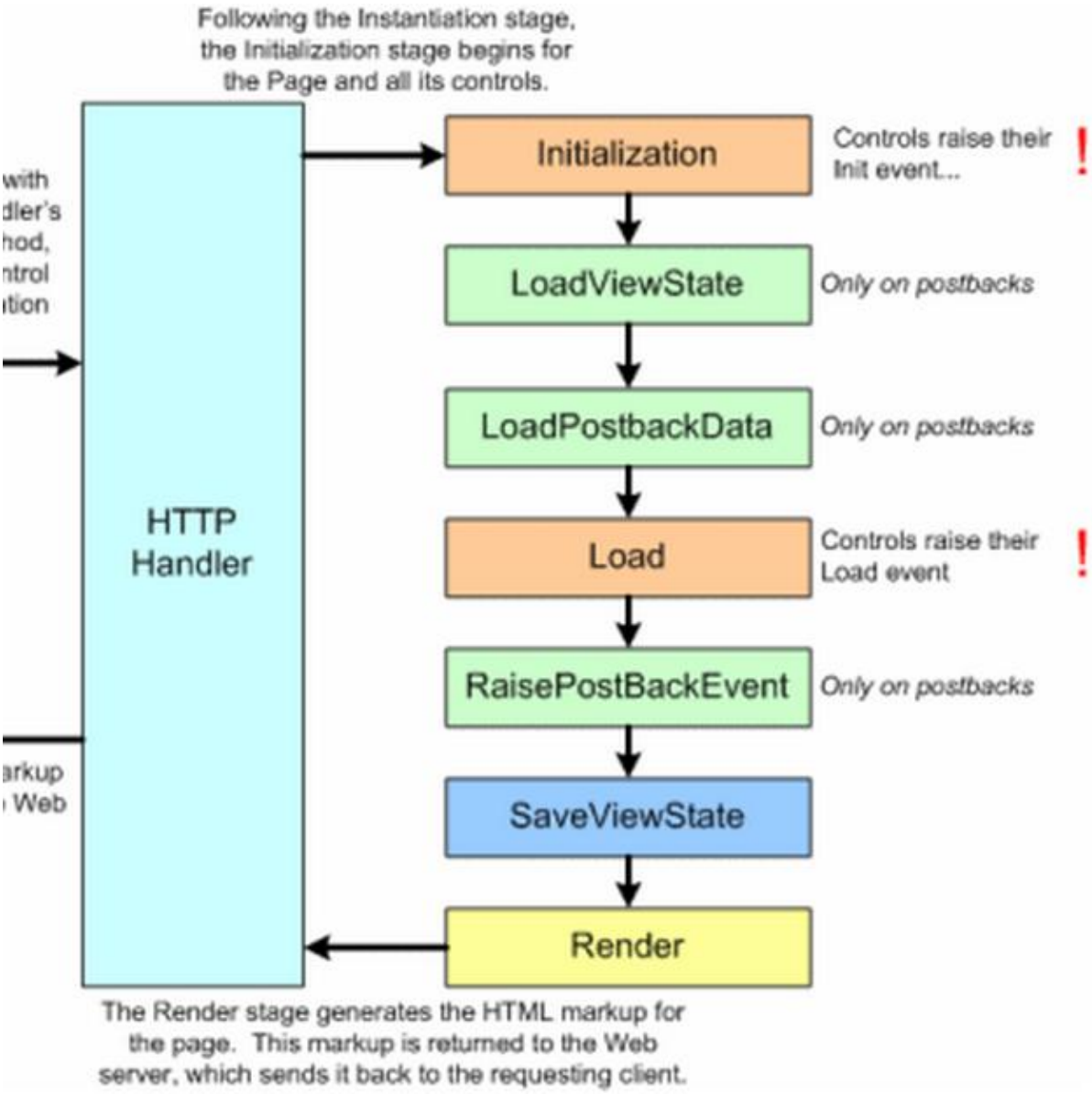


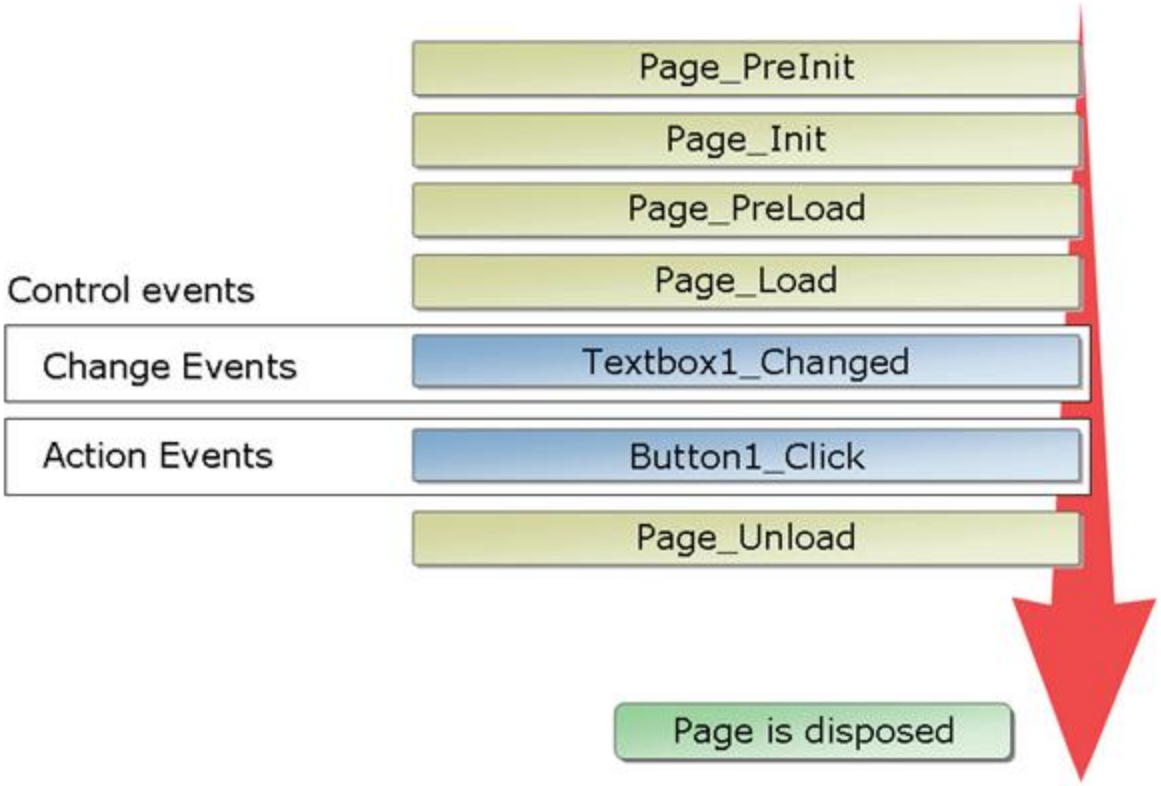
Why so many events? Different things happen and different information is available at different stages in the page execution lifecycle.

For example, **View State** is not loaded until after the **InitComplete** event.
Data posted to the server from a form control, such as a **TextBox** control, is also not available until after this event

Ninety-nine percent of the time, you won't handle any of these events except for the **Load** and the **PreRender** events.

The difference between these two events is that the **Load** event happens before any control events and the **PreRender** event happens after any control events.





[02_2 ViewState AND life cycle]

1. שימו לב לכך שתמיד הLOAD _PAGE רץ ראשון. בשלב הזה המידע שנימצא בפקדים הוא לאחר השיחזור של ה- VIEWSTATE וגם לאחר העידכון מה POSTBACK!
2. שימו לב לרצף האירועים:
   a. Page_load
   b. Change events
   c. Postback event – like click
3. סדר שלב .b לא ניתן לצפיה מראש ו\או לא כדאי לסמוך עליו!!!

אובייקט VIEWSTATE

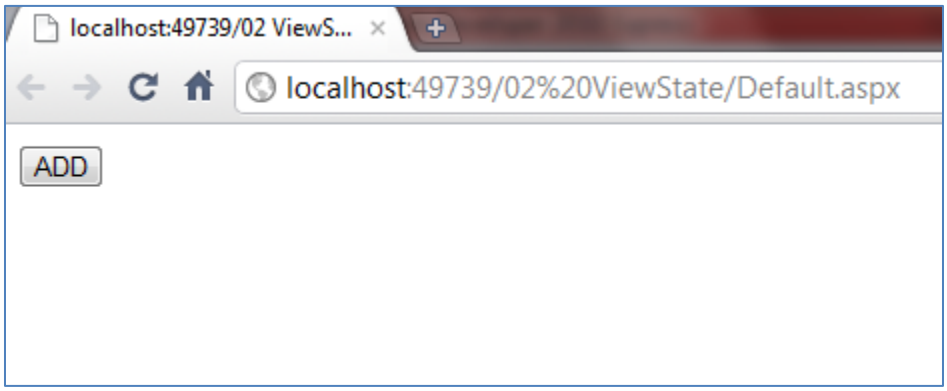
ניתן להשתמש באובייקט - ViewState כדי לשמור ערכים שלנו. לדוגמה: נרצה לדעת כמה פעמים משתמש לחץ על לחצן מסוים (נניח לחצן Login). שימו לב שלא ניתן לשמור משתנה בתוך הדף (כמו שעושים ב - Windows forms) מפני שבכל בקשה הדף נוצר מחדש והמשתנה מתאפס. שמירת המידע בתוך ה - ViewState מתבצעת כמו עבודה עם Dictionary (במבנה של key / value).

לא ניתן לשנות את המחרוזת עצמה של ה VIEWSTATE__ או לראות דרך האובייקט הזה את ערכי הקונטרולים. ניתן לערוך רק ערכים שאנחנו שמנו בעצמנו בתוך Dictionary.

מאפיינים של ViewState:

- נשמר כל עוד המשתמש לא עבר לדף אחר ולא סגר את הדפדפן
- נשמר בצד הלקוח ולכן חשוף

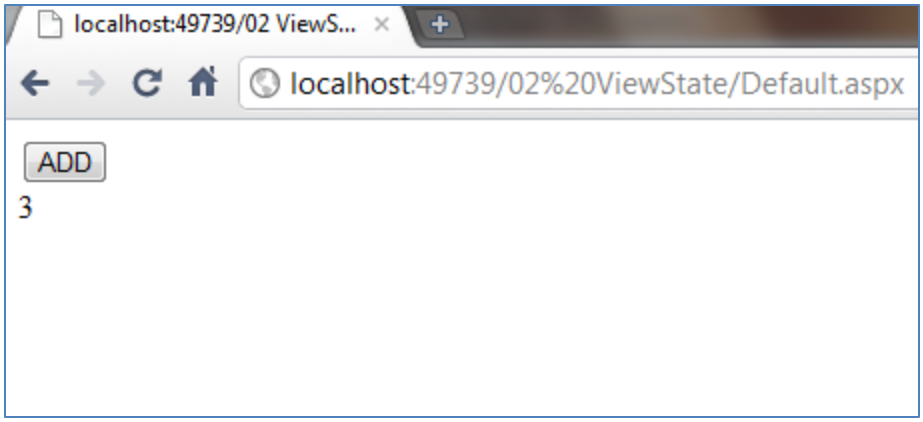
בדוגמא הבאה הלחצן Add יוסיף מספר לערך הנמצא ב - ViewState ולחיצה על Show תציג את הערך הנמצא ב - ViewState:



להלן הקוד:

```
   protected void btnAdd_Click(object sender, EventArgs e)
{
    if (ViewState["count"] == null)
    {
        ViewState["count"] = 1;
    }
    else
    {
        ViewState["count"] = (int)ViewState["count"] + 1;
    }
    lblRes.Text = ((int)ViewState["count"]).ToString();
}
```

כך נראה הדף לחיצה שלוש פעמים על Add ולאחר מכן על Show:



AutoPosBack

[05 AutoPostBack]

כאשר המאפיין הזה דולק הפקד נותן הוראת SUBMIT לאחר שעושים בו שימוש. הפעולה הזו נעשית ע"י קוד JSשהסביבה מוסיפה כ

```
function __doPostBack(eventTarget, eventArgument)
```

```
<select name="Dropdownlist2"
onchange="javascript:setTimeout(&#39;__doPostBack(\&#39;D
ropdownlist2\&#39;,\&#39;\&#39;)&#39;, 0)"
```