What is Ajax

AJAX stands for **A**synchronous **J**avaScript**A**nd **X**ML.

Ajax is not a technology or a product,

Ajax is a client-side script that communicates to and from a server/database without the need for a postback or a complete page reload.

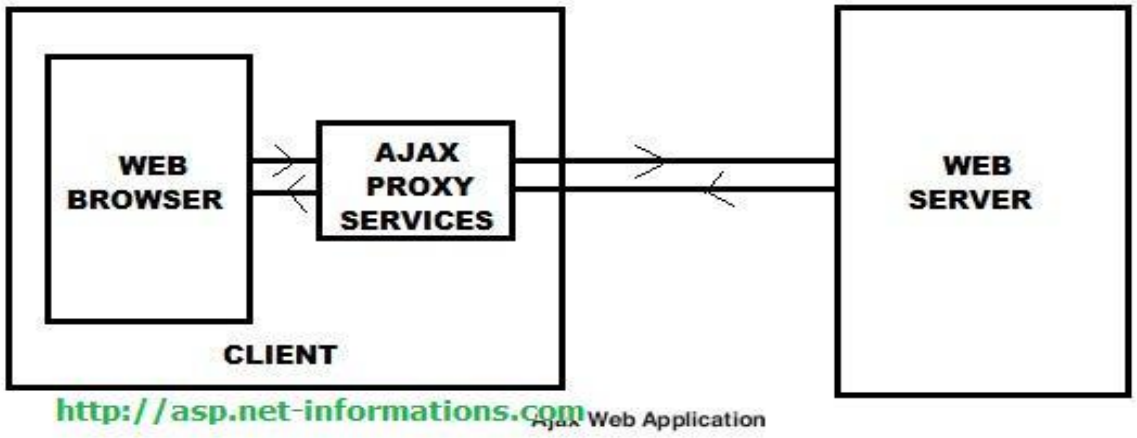
AJAX is "the method of exchanging data with a server, and updating parts of a web page without reloading the entire page."

Ajax itself is mostly a generic term for various JavaScript techniques used to connect to a web server dynamically without necessarily loading multiple pages.

In a more narrowly-defined sense, it refers to the use of XmlHttpRequest objects to interact with a web server dynamically via JavaScript.

It is just a way to refer a set of existing standards used together for developing faster interfaces, improve the user experience and better response times.

Ajax support for the most popular web browsers, which includes Microsoft Internet Explorer, Mozilla Firefox, Google Chrome etc. The public use of Ajax can be found worldwide such as Google Maps, Google Suggest, Live.com etc.



Ajax introduces a new approach to WebPages that update the portion of the page by a techniques called Partial-page rendering.

Partial-page rendering eliminate the traditional way of doing a full postback (full page refresh) to the web server and updates only the necessary portion of the web page. That is, only the necessary data is transferred back and forth between the client and the web server.

Using this techniques you can cut down on the network load, bandwidth usage and moreover the web page is not locked during the server processing. It helps in creating faster, better, dynamic and interactive web applications.

Following term shows meaning of AJAX

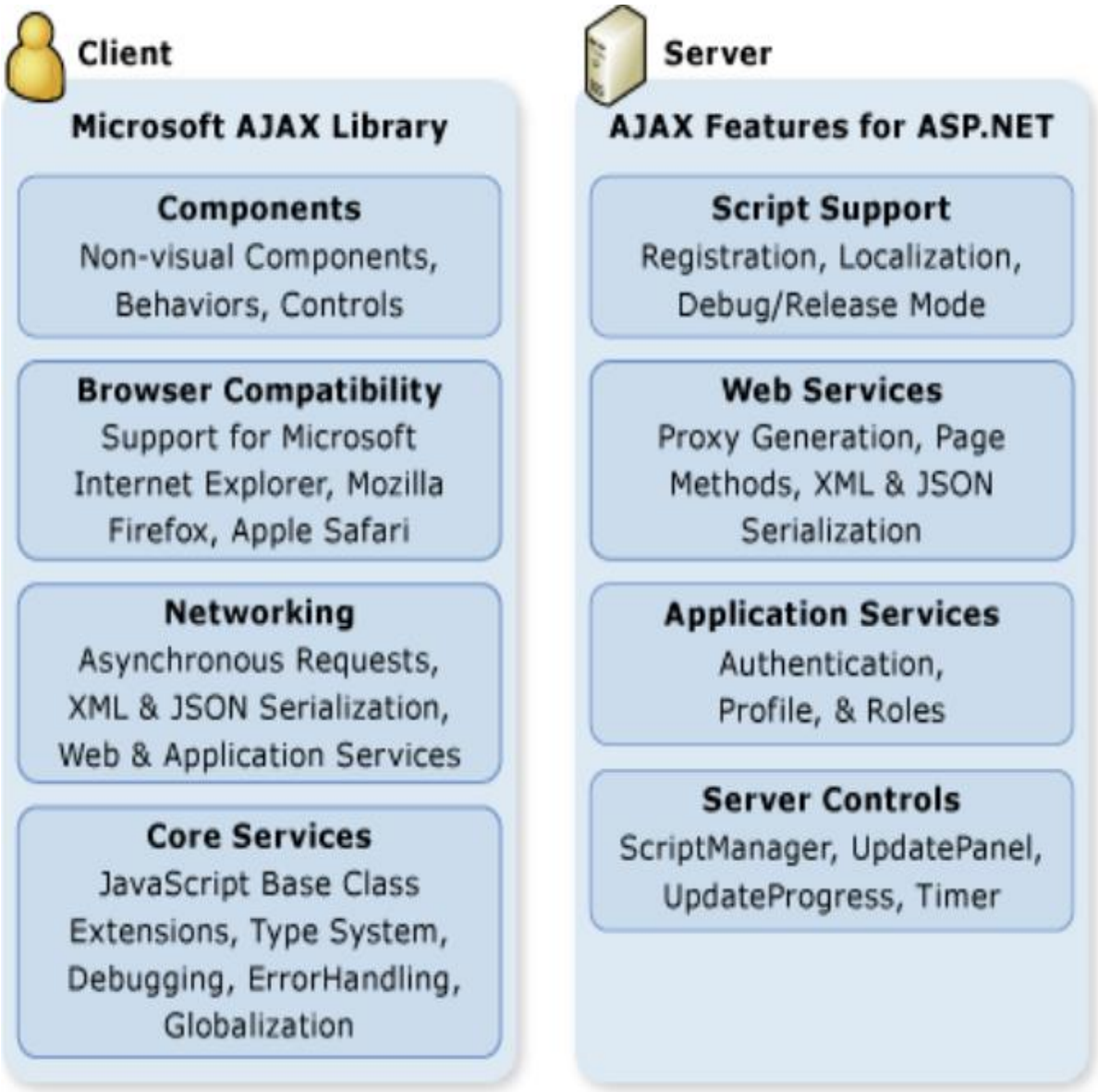
- **Asynchronous communication:**
  It is the ability of an application to send multiple requests and Receive responses from the server simultaneously.

- **JavaScript:**
  It is a scripting language used to generate client side scripts. Itforms an Integral part of the AJAX – enabled web applications. Many requests are Processed on the client side saving the turnaround server time.

- **XML:**
  It is a standard used for transferring data between applications. It is the base for manipulating the data received from the server.

AJAX Architecture

In AJAX enabled web application, the page is loaded for the first time when requested by a user. Next whenever a user requests for the data from the server, the requests are sent to the AJAX engine as JavaScript calls. The AJAX engine is a mediator between the client and the server. It sends only small page bits that are to be updated without reloading the entire page.



Advantages of AJAX

- **Speed**
  Reduce the server traffic in both side request. Also reducing the time consuming on both side response.
- **Interaction**
  AJAX is much responsive, whole page(small amount of) data transfer at a time.
- **Asynchronous calls**
  AJAX make asynchronous calls to a web server. This means client browsers are avoid waiting for all data arrive before start the rendering.

- **Form Validation**
  This is the biggest advantage. Form are common element in web page. Validation should be instant and properly, AJAX gives you all of that, and more.
- **Bandwidth Usage**
  No require to completely reload page again. AJAX is improve the speed and performance. Fetching data from database and storing data into database perform background without reloading page.

Disadvantages of AJAX

1. AJAX application would be a mistake because search engines would not be able to index an AJAX application.

2. View source is allowed and anyone can view the code source written for AJAX which not secure.

3. ActiveX requests are enabled only in Internet Explorer and newer latest browser i.e. older browser does not support AJAX.

4. JavaScript disabled browsers cannot use the application.

5. Security is less in AJAX applications as all the files are downloaded at client side.

ASP.NET Ajax ServerControls

ASP.NET Ajax server controls consist of server and client code that integrate to produce rich client behavior. The following list describes the most frequently used Ajax server controls.

- ScriptManager Control
- UpdatePanel Control
- UpdateProgress Control
- Timer Control

ScriptManagerControl :

The ScriptManager control is the essential core for the AJAX Extensions that makes ASP.NET AJAX possible by managing the JavaScript for you behind the scenes. There are a number of client script libraries available to ASP.NET developers, and when a ScriptManager is used, only the relevant ones are downloaded to the client. Moreover ScriptManager control supports the UpdatePanel controls to handles the asynchronous postbacks and refreshes only the regions of the page that have to be updated (partial-page rendering). The ScriptManager control will be visible in design view but will not be visible when the web site is run.

UpdatePanelControl :

UpdatePanel control is used to visually design the part of the page that is used in partial-page rendering. Placing controls inside an UpdatePanel control enables you to refresh selected parts of the webpage instead of refreshing the whole page with a postback to the webserver. When the UpdatePanel control performs an asynchronous post, it adds a custom HTTP header. There is no limit to the number of UpdatePanel controls that can be on a page, and they can be nested. Each UpdatePanel is updated individually and asynchronously, without affecting one another or anything else on the page.

UpdateProgressControl :

UpdateProgress control shows the status information for the progress of the download occurring in the UpdatePanel. The page can contain multiple UpdateProgress controls. Each one can be associated with a different UpdatePanel control. Alternatively, you can use one UpdateProgress control and associate it with all the UpdatePanel controls on the page.

Timer Control :

The Timer control enables you to perform postbacks at a specified interval. It can be used to trigger automatic updates to an area of the page that is wrapped with an UpdatePanel.

The ScriptManager Control

The ScriptManager control is the most important control and must be present on the page for other controls to work.

It has the basic syntax:

```
<asp:ScriptManagerID="ScriptManager1"runat="server">
</asp:ScriptManager>
```

If you create an 'Ajax Enabled site' or add an 'AJAX Web Form' from the 'Add Item' dialog box, the web form automatically contains the script manager control. The ScriptManager control takes care of the client-side script for all the server side controls.

First Asp.Net Ajax Program

Ajax introduces a new approach to WebPages that update the portion of a webpage by a techniques called Partial-page rendering. In the following example you can learn the difference between How a Partial-page rendering and a traditional way of page update works in a same web page.

The ScriptManager control is the essential core for the ASP.NET AJAX Extensions. We have to use a ScriptManager Control in just about every Ajax application. In the following example we placed a ScriptManagercontrol , one UpdatePanel Control, two Buttons named Ajax-Button and Non-Ajax-Button and two label controls in the form.

The first Button (Ajax-Button) is placed inside the UpdatePanel control area and the second Button (Non-Ajax-Button) placed outside the UpdatePanel control area. When you run this application and clicked first button (Ajax-Button), then you can see the first label control only updated the current server time. When you clicked on second Button (Non-Ajax-Button) you can see both label control will update the current server time. This is happening because the first Button(Ajax-Button) and label control we placed inside the UpdatePanel Control area and it behaves like Partial-page rendering. That means it will update only the portion of the webpage (between UpdatePanel Control area) while clicking first Button. When you clicked on the second Button (Non-Ajax-Button) you can see both labels updated with current server time ,because second Button (Non-Ajax Button) is placed outside the UpdatePanel area and it behaves like ordinary web application and updated

both labels. That means a full page refresh or full postback to the web server is happening while clicking second Button.

```
<%@PageLanguage="C#"AutoEventWireup="true"CodeFile="Default.aspx.cs"
Inherits="_Default"%>

<!DOCTYPEhtml>

<htmlxmlns="http://www.w3.org/1999/xhtml">
<headrunat="server">
<title></title>
</head>
<body>
<formid="form1"runat="server">
<div>
<asp:ScriptManagerID="ScriptManager1"runat="server"></asp:ScriptManage
r>
<asp:UpdatePanelid="up1"runat="server">
<ContentTemplate>
<asp:LabelID="Label1"runat="server"Text="Label"></asp:Label>
<asp:ButtonID="Button1"runat="server"Text="Ajax -
Button"onclick="Button1_Click"/>
</ContentTemplate>
</asp:UpdatePanel>
<asp:LabelID="Label2"runat="server"Text="Label"></asp:Label>
<asp:ButtonID="Button2"runat="server"Text="Non Ajax -
Button"onclick="Button2_Click"/>
</div>
</form>
</body>
</html>
```

Default.aspx.cs

```
usingSystem.Web;
usingSystem.Web.UI;

public partial class _Default : System.Web.UI.Page
{
```

```
        protected void Page_Load(object sender, EventArgs e)
        {

        }
        protected void Button1_Click(object sender, EventArgs e)
        {
                Label1.Text = DateTime.Now.ToString();
                Label2.Text = DateTime.Now.ToString();
        }
        protected void Button2_Click(object sender, EventArgs e)
        {
                Label1.Text = DateTime.Now.ToString();
                Label2.Text = DateTime.Now.ToString();
        }
}
```

The UpdatePanel Control

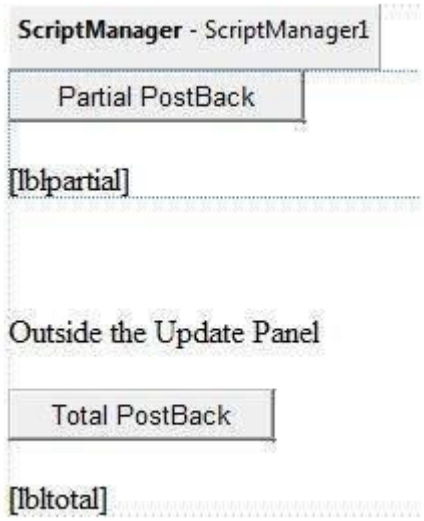
The UpdatePanel control is a container control and derives from the Control class. It acts as a container for the child controls within it and does not have its own interface. When a control inside it triggers a post back, the UpdatePanel intervenes to initiate the post asynchronously and update just that portion of the page.

For example, if a button control is inside the update panel and it is clicked, only the controls within the update panel will be affected, the controls on the other parts of the page will not be affected. This is called the partial post back or the asynchronous post back.

Example

Add an AJAX web form in your application. It contains the script manager control by default. Insert an update panel. Place a button control along with a label control within the update panel control. Place another set of button and label outside the panel.

The design view looks as follows:



The source file is as follows:

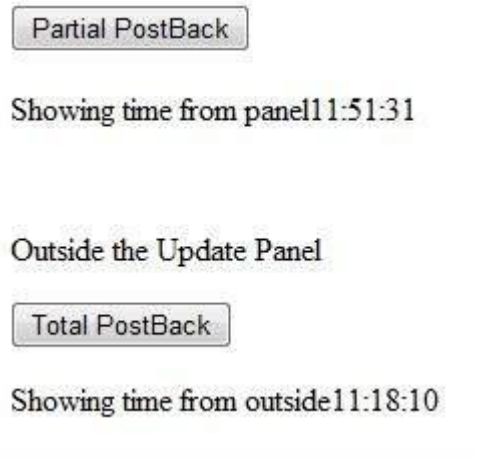
```
<formid="form1"runat="server">
<div>
<asp:ScriptManagerID="ScriptManager1"runat="server"/>
</div>

<asp:UpdatePanelID="UpdatePanel1"runat="server">
<ContentTemplate>
<asp:ButtonID="btnpartial"runat="server"onclick="btnpartial_Click"Text="Pa
rtial PostBack"/>
<br/>
<br/>
<asp:LabelID="lblpartial"runat="server"></asp:Label>
</ContentTemplate>
</asp:UpdatePanel>

<p> </p>
<p>Outside the Update Panel</p>
<p>
<asp:ButtonID="btntotal"runat="server"onclick="btntotal_Click"Text="Total
PostBack"/>
</p>

<asp:LabelID="lbltotal"runat="server"></asp:Label>
```

```
</form>
```

Both the button controls have same code for the event handler:

```
string time =DateTime.Now.ToLongTimeString();
lblpartial.Text="Showing time from panel"+ time;
lbltotal.Text="Showing time from outside"+ time;
```

Observe that when the page is executed, if the total post back button is clicked, it updates time in both the labels but if the partial post back button is clicked, it only updates the label within the update panel.



A page can contain multiple update panels with each panel containing other controls like a grid and displaying different part of data.

When a total post back occurs, the update panel content is updated by default. This default mode could be changed by changing the UpdateMode property of the control. Let us look at other properties of the update panel.

ASP.NET Ajax UpdateProgress

UpdateProgress control shows the status information for the progress of the download occurring during the partial-page rendering in the UpdatePanel. The page can contain multiple UpdateProgress controls. Each one can be associated with a different UpdatePanel control. Alternatively, you can use one UpdateProgress control and associate it with all the UpdatePanel controls on the page.

You can place UpdateProgress control either inside or outside the UpdatePanel controls. The following Asp.Net program shows how to an UpdateProgress control waits for completion of the task and during this waiting time it shows a .gif file as waiting message.

Default.aspx

```
<%@PageLanguage="C#"AutoEventWireup="true"CodeFile="Default.aspx.cs"
Inherits="_Default"%>

<!DOCTYPEhtml>

<htmlxmlns="http://www.w3.org/1999/xhtml">
<headrunat="server">
<title></title>
</head>
<body>
<formid="form1"runat="server">
<div>
<asp:ScriptManagerID="ScriptManager1"runat="server">
</asp:ScriptManager>
<asp:UpdateProgressID="UpdateProgress1"runat="server">
<ProgressTemplate>
        <imgalt="ajax-progress"src="OIPL65Z6Q2E.jpg"></img>
</ProgressTemplate>
</asp:UpdateProgress>
        <asp:UpdatePanelID="UpdatePanel1"runat="server">
        <ContentTemplate>
```

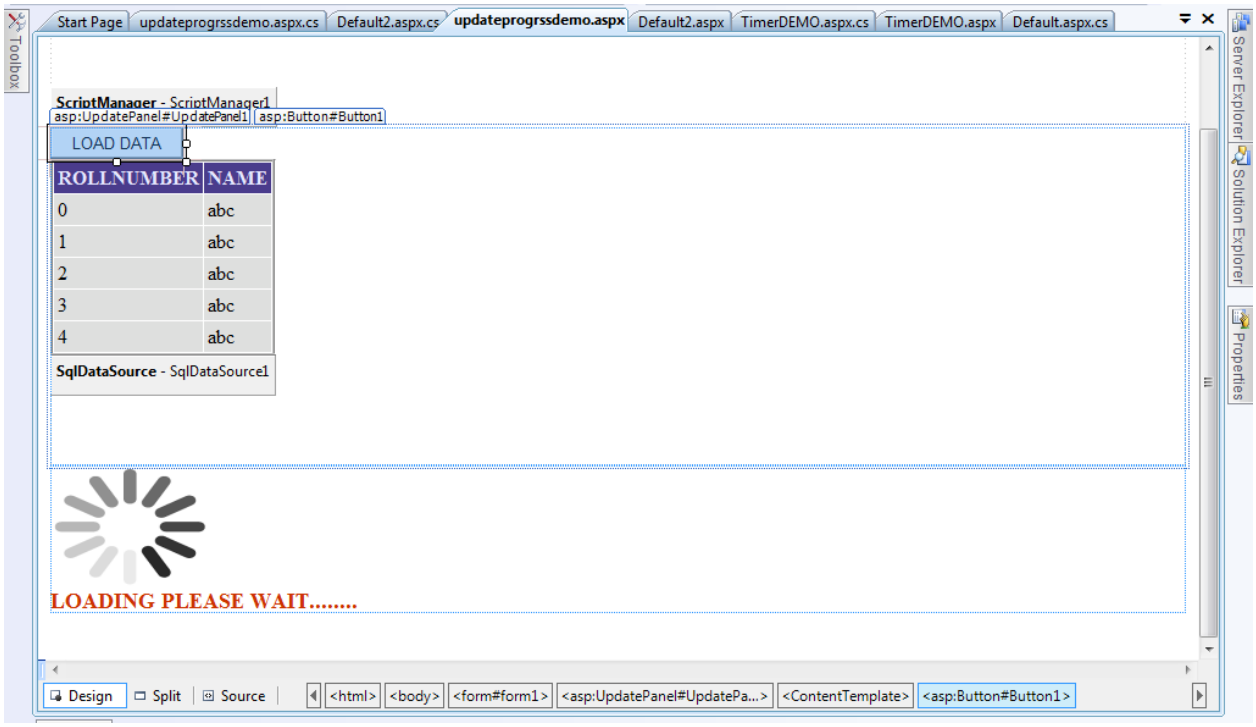
```
            <asp:LabelID="Label1"runat="server"Text=""></asp:Label>
                <br/>
            <asp:ButtonID="Button1"runat="server"Text="Button"
onclick="Button1_Click"/>
        </ContentTemplate>
        </asp:UpdatePanel>
</div>
</form>
</body>
</html>
```

Default.aspx.cs

```
using System;
usingSystem.Web.UI;

public partial class _Default : System.Web.UI.Page
{
        protected void Page_Load(object sender, EventArgs e)
        {

        }
        protected void Button1_Click(object sender, EventArgs e)
        {
                System.Threading.Thread.Sleep(5000);
                Label1.Text = "Server Time :  " + DateTime.Now.ToString();
        }
}
```

DEMO- 2 UpdateProgress AJAX Control----

DESIGN PAGE



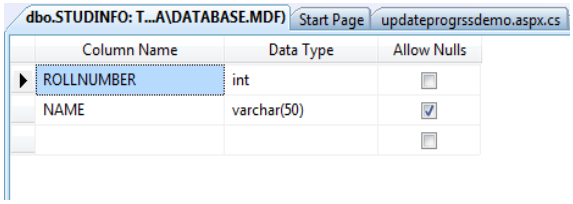
SOURCE PAGE-

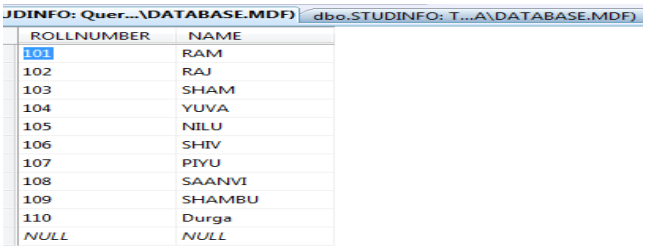
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="updateprogrssdemo.aspx.cs"
Inherits="updateprogrssdemo" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    <asp:Label ID="Label1" runat="server" Font-
Size="Large"
        Text="UPDATE PROGRSS AJAX DEMO"></asp:Label>
    <p>
```

```
         </p>
    <asp:ScriptManager ID="ScriptManager1"
runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <asp:Button ID="Button1" runat="server"
Text="LOAD DATA"
                onclick="Button1_Click" />
            <br />
            <asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False"
                BackColor="White" BorderColor="White"
BorderStyle="Ridge" BorderWidth="2px"
                CellPadding="3" CellSpacing="1"
DataSourceID="SqlDataSource1" GridLines="None"
                Visible="False">
                <RowStyle BackColor="#DEDFDE"
ForeColor="Black" />
                <Columns>
                    <asp:BoundField
DataField="ROLLNUMBER" HeaderText="ROLLNUMBER"
                        SortExpression="ROLLNUMBER" />
                    <asp:BoundField DataField="NAME"
HeaderText="NAME" SortExpression="NAME" />
                </Columns>
                <FooterStyle BackColor="#C6C3C6"
ForeColor="Black" />
                <PagerStyle BackColor="#C6C3C6"
ForeColor="Black" HorizontalAlign="Right" />
                <SelectedRowStyle BackColor="#9471DE"
Font-Bold="True" ForeColor="White" />
                <HeaderStyle BackColor="#4A3C8C" Font-
Bold="True" ForeColor="#E7E7FF" />
            </asp:GridView>
            <asp:SqlDataSource ID="SqlDataSource1"
runat="server"
                ConnectionString="<%$
ConnectionStrings:ConnectionString %>"
```

```
                SelectCommand="SELECT * FROM
[STUDINFO]"></asp:SqlDataSource>
            <br />
            <br />
            <br />
        </ContentTemplate>
    </asp:UpdatePanel>
    <asp:UpdateProgress ID="UpdateProgress1"
runat="server">
    <ProgressTemplate>

        <asp:Image ID="Image1" runat="server"
ImageUrl="~/loading.png" Height="101px"
            Width="130px" />
        <br />
        <asp:Label ID="Label2" runat="server"
Text="LOADING PLEASE WAIT........."
            Font-Bold="True" Font-Size="Large"
ForeColor="#CC3300"></asp:Label>
    </ProgressTemplate>
    </asp:UpdateProgress>
    </form>
</body>
</html>
```

DATABASE TABLE DESING AND DATA-

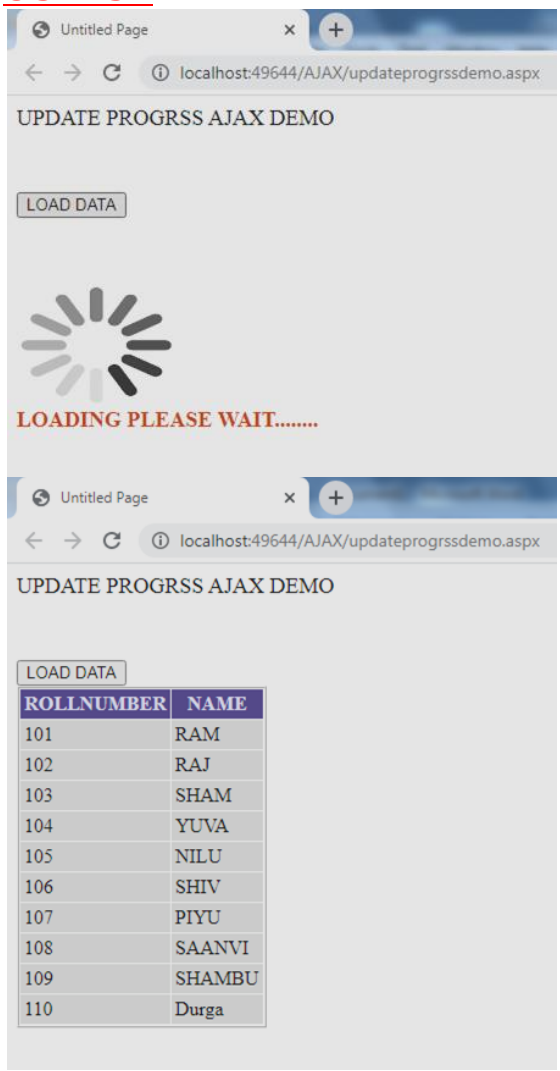




CODE BEHIND PAGE-

```
13
14  public partial class updateprogrssdemo : System.Web.UI.Page
15  {
16      protected void Page_Load(object sender, EventArgs e)
17      {
18
19      }
20      protected void Button1_Click(object sender, EventArgs e)
21      {
22          System.Threading.Thread.Sleep(5000);
23          GridView1.Visible = true;
24      }
25  }
26
```

OUTPUT



ROLLNUMBER	NAME
101	RAM
102	RAJ
103	SHAM
104	YUVA
105	NILU
106	SHIV
107	PIYU
108	SAANVI
109	SHAMBU
110	Durga

ASP.NET Ajax Timer

The Timer control can work as a trigger to an area of the page that is wrapped with an UpdatePanel Control. It performs postbacks at defined intervals. The Interval property is defined in milliseconds, so that setting the Interval property to 5000 milliseconds will refresh the UpdatePanel control every 5 seconds.

The following asp.net program placed a Timer and label controls inside the UpdatePanel control area and one label control outside the UpdatePanel control area. When you run this program first label shows the current server time as page loaded time. After five seconds the second Label inside the UpdatePanel control shows the current server time. After that each five seconds you can see the second label only updating the current server time , because partal page updating is happening at defined intervals.

Default.aspx

```
<%@PageLanguage="C#"AutoEventWireup="true"CodeFile="Default2.aspx.cs"Inherits="D
efault2"%>

<!DOCTYPEhtml>

<htmlxmlns="http://www.w3.org/1999/xhtml">
<headrunat="server">
<title></title>
</head>
<body>
<formid="form1"runat="server">
<div>
<asp:LabelID="Label1"runat="server"Text="Label"></asp:Label>
                <br/><br/>
<asp:ScriptManagerID="ScriptManager1"runat="server"/>
<asp:TimerID="Timer1"runat="server"interval="5000"ontick="Timer1_Tick"/>
<asp:UpdatePanelID="UpdatePanel1"runat="server">
        <ContentTemplate>
        <asp:LabelID="Label2"runat="server"Text="Label"></asp:Label>
        </ContentTemplate>
        <Triggers>
        <asp:AsyncPostBackTriggercontrolid="Timer1"eventname="Tick"/>
```

```
        </Triggers>
</asp:UpdatePanel>


</div>
</form>
</body>
</html>
```

Default.aspx.cs

```
using System;
usingSystem.Web;

public partial class _Default : System.Web.UI.Page
{
        protected void Page_Load(object sender, EventArgs e)
        {
                Label1.Text = "Page loaded time : " +
DateTime.Now.ToString();
        }

        protected void Timer1_Tick(object sender, EventArgs e)
        {
                Label2.Text = "Current Server time : " +
DateTime.Now.ToString();
        }
}
```