

Connecting an ASP.NET-form to a database

Connecting an ASP.NET form created with SpreadsheetConverter to a database is very easy. We will do it in 3 steps:

1. [Calculate and save the form contents into a database.](#)
2. Retrieve previous entered data from the database, show it in the form and let the user edit it and recalculated and save it again.
3. [Show all submitted entries so that we can click on them to edit them.](#)

[You can read the 1st part here.](#)

Part 2: Presenting and updating data in the form

Filling the form with data from the database is almost as easy as saving the data. The only complication is that there must be a way to tell the ASP.NET-form which entry to display. We have chosen the standard solution where each entry has a unique that is used to find the entry.

Design

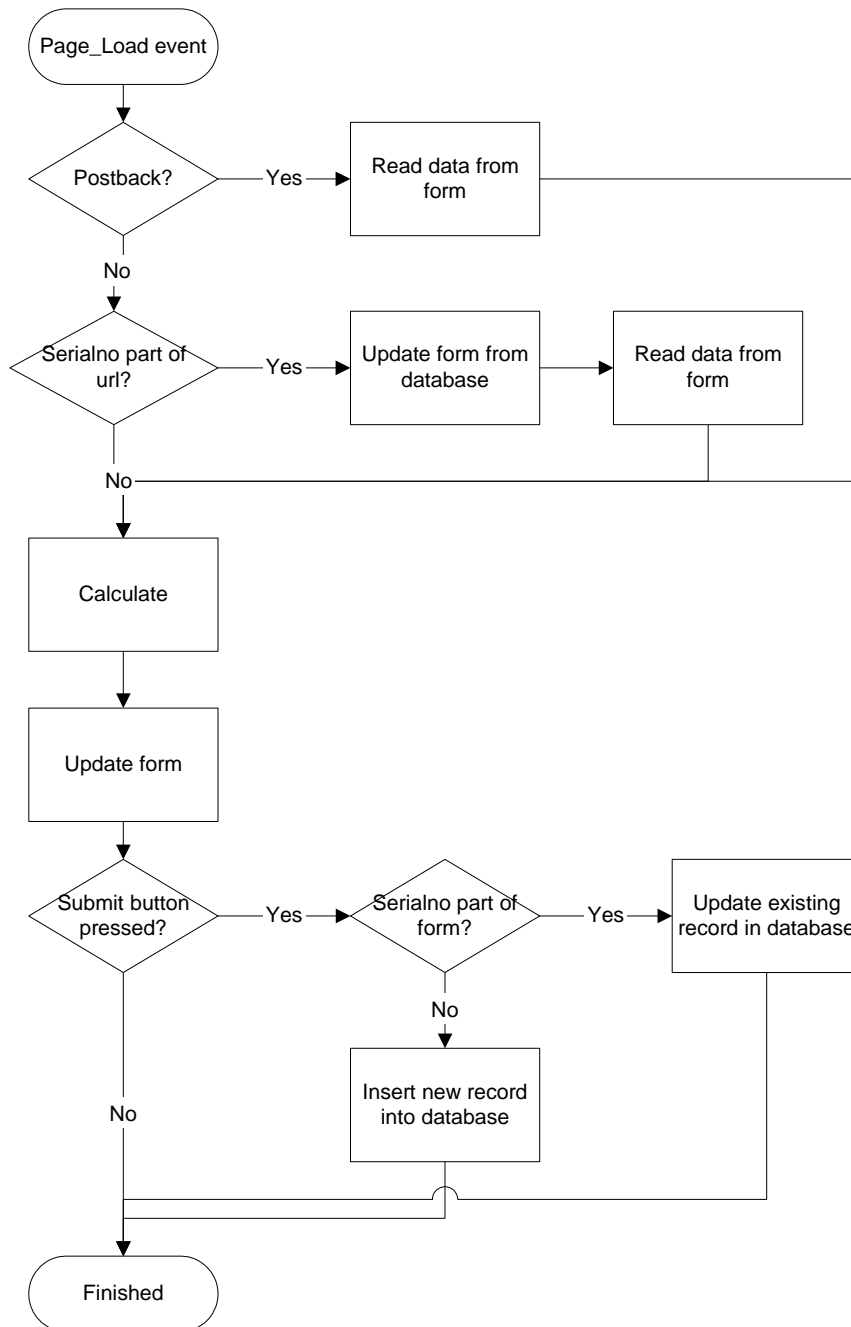
Each entry in the database as a unique serial number column, called serialno. The database automatically fills in this column.

The normal url to our ASP.NET-form is http://localhost/FillFromDB/time_report.aspx. When the web page is called the first time, it returns an empty form, when the web page is called after that, the calculated fields are automatically recalculated.

In order to handle updating of old data from the database, we add a HTTP Query Parameter to the url, which tells which row in the database to edit. The url with the query parameter will look like http://localhost/FillFromDB/time_report.aspx?serialno=14

If the contents of the form is read from the database, we do not want to add a new record to the database when the user presses Save/Submit button.

Whenever the user wants to save the form, we look at the querystring for serialno. If it is there, then we should update an existing row in the database. If serialno is not set, then this is a new row and we insert the data into the database.



SQL-statement for finding an old time report.

As described in part 1, we should always use parameterized SQL-statements. The structure of the SELECT-statement, which is the one that reads an old entry is
`select * from arrival Where serialno = @serialno`

SQL-statement for updating an old time report

The structure of the UPDATE statement which replaces the old values of arrival, departure, hours, name,today2... with new values is:

```
Update Arrival Set
name=@name,today2=@today2,arrival=@arrival,departure=@departure,lunch=@
lunch,experience=@experience,hours=@hours where serialno=@serialno
```

SQL-statement for inserting a new time report

The structure of the INSERT statement hasn't changed. Note that serialno is not mentioned. The database handler will set that column automatically, since it is an identity type.

```
Insert into
Arrival(name,today2,arrival,departure,lunch,experience,Hours)
values (@Name,@Today2,@Arival,@Departure,@Lunch,@Experience,@Hours)
```

Displaying values to different controls

To display values to different controls, we have declared global variable (outside method), like below

```
// variable declaration equivalent to the columns
string name = string.Empty;
string today2 = string.Empty;
int arrival;
int departure;
int lunch;
int experience;
int hours;
```

& will be using them to display values to different html elements using asp.net inline expression “<%=inline expression %>”. Search each element tag & place the appropriate attributes.

- 1) For Input(TextBox) control having name ”name” :

Set the value using `value='<%= name %>'` as

```
<input value='<%= name %>' ../>
```

- 2) For Calendar control having name ”today2” :

Set the value using `value='<%= today2 %>'`

- 3) For Drop down control having name ”arrival”:

For drop down control we can't use value attribute directly. We have a helper method named `string GetDropDownHelper(string currentValue, string selectedValue)`

and will be using to display selected value for the select element. This method accepts first parameter as the current value of the option element and second value is the value retrieved

from database to be selected in drop down control & finally returns ”selected” if they matches. So associate each option element with `GetDropDownHelper` as below:

```
<select name='arrival' id='arrival' class='ee102'
style='width: 100%' tabindex='3'

onchange="recalc_onclick('arrival')" size='1'>
  <option value='1' <%=
GetDropDownHelper("1",arrival.ToString()) %> >1</option>
```

```

        <option value='2' <%=
GetDropDownHelper ("2",arrival.ToString()) %> >2</option>
        <option value='3' <%=
GetDropDownHelper ("3",arrival.ToString()) %> >3</option>
        <option value='4' <%=
GetDropDownHelper ("4",arrival.ToString()) %> >4</option>
        <option value='5' <%=
GetDropDownHelper ("5",arrival.ToString()) %> >5</option>
</select>

```

4) For Slider control having name "departure":

Set the value using `value='<%= departure %>'`

5) For Radio buttons control having name "lunch":

For this control also, there is a helper method named "string GetRadioButtonHelper (string currentValue, string selectedValue)" which compares the current element value with the value retrieved from database & returns "checked='checked'" for the matched value. First remove the "checked='checked'" from the first input & associate each input with "GetRadioButtonHelper" as below:

```

<input style="vertical-align: middle;" type='radio'
id='lunch$1' name='lunch' value='1'
        onclick='recalc_onclick(lunch$1)' <%=
GetRadioButtonHelper ("1",lunch.ToString()) %> />

<input style="vertical-align: middle;" <%=
GetRadioButtonHelper ("2",lunch.ToString()) %>
        type='radio' id='lunch$2' name='lunch' value='2'
onclick='recalc_onclick(lunch$2)' />

```

Place this for all the input element having name "lunch".

6) For Rating control having name "experience":

For this control we will need to use some javascript code to display the selected value. So place the code

```

<script type="text/javascript">
    $(document).ready(
        function() {
            SetRatingStart('<%= experience %>', 5,
                'experience', 'StarA');
            recalc_onclick('experience')
        }
    );
</script>

```

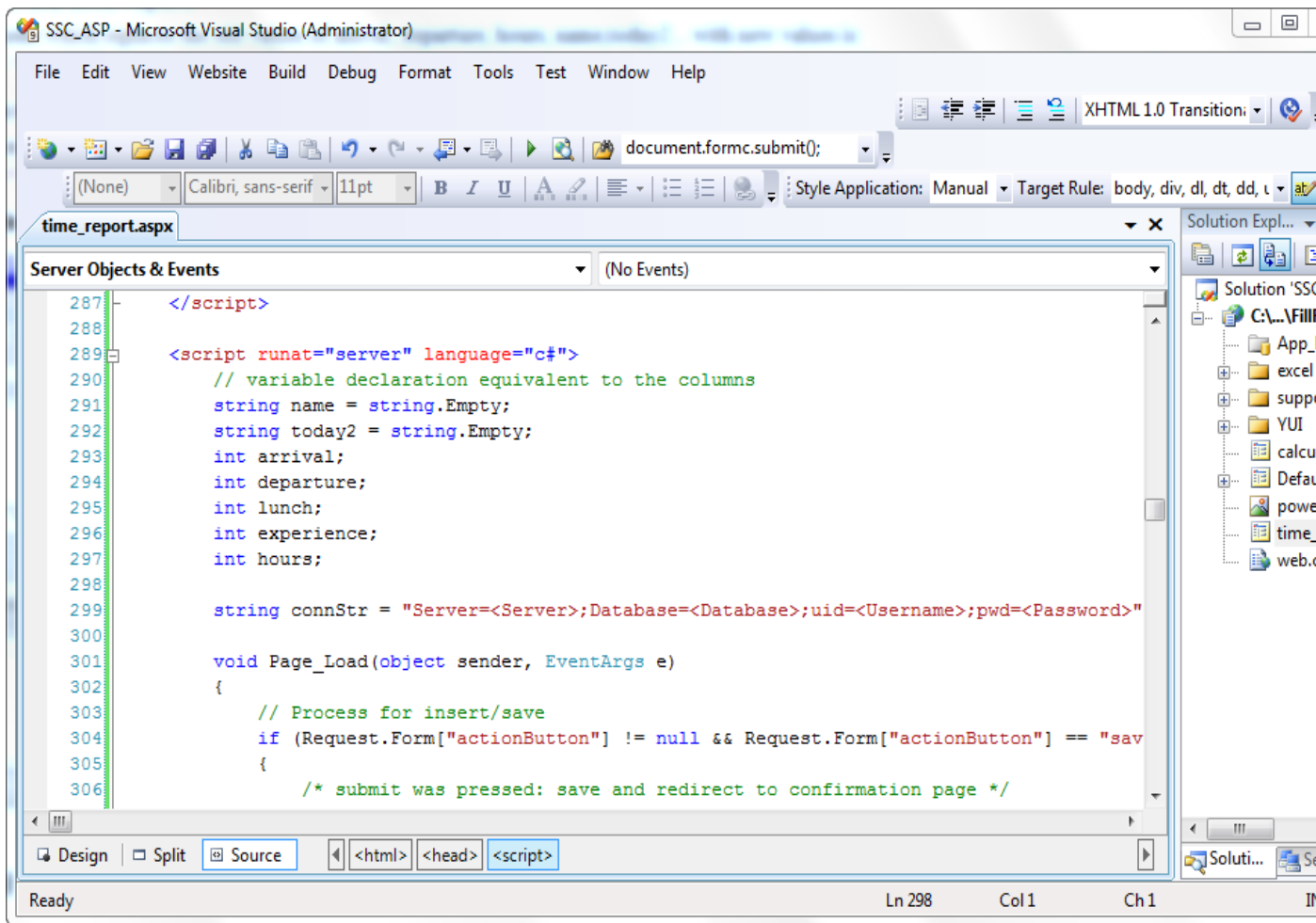
just above the rating element. We are calling the javascript function named "SetRatingStart" that will be called while selecting/clicking some rating value. It accepts value to be selected as first parameter (we are setting this value as `<%= experience`

%>), total number of rating as second parameter, name of input element to be passed to server as third element & the initial name of span element in the rating group.

The structure of the INSERT statement hasn't changed. Note that serialno is not mentioned. The database handler will set that column automatically, since it is an identity type.

The complete server-side-code

[The complete source code is a bit long. Click here to view it.](#)



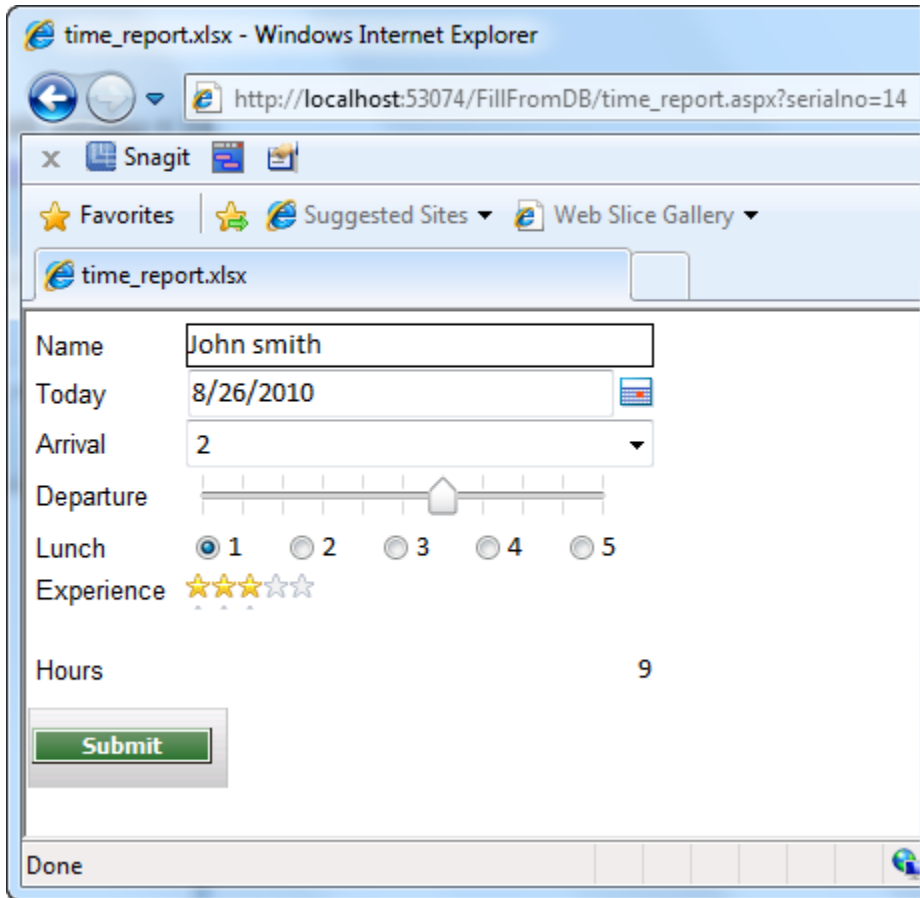
```
287 </script>
288
289 <script runat="server" language="c#">
290     // variable declaration equivalent to the columns
291     string name = string.Empty;
292     string today2 = string.Empty;
293     int arrival;
294     int departure;
295     int lunch;
296     int experience;
297     int hours;
298
299     string connStr = "Server=<Server>;Database=<Database>;uid=<Username>;pwd=<Password>"
300
301     void Page_Load(object sender, EventArgs e)
302     {
303         // Process for insert/save
304         if (Request.Form["actionButton"] != null && Request.Form["actionButton"] == "sav
305         {
306             /* submit was pressed: save and redirect to confirmation page */
```

Note: Remember to change the variable "connStr" with appropriate values to point to the database created in step1.

Testing updating an old value

I looked into my database and saw that I had an entry with serialno = 14.

The link to edit that row is http://localhost/FillFromDB/time_report.aspx?serialno=14. Clicking on that URL, the form opens with

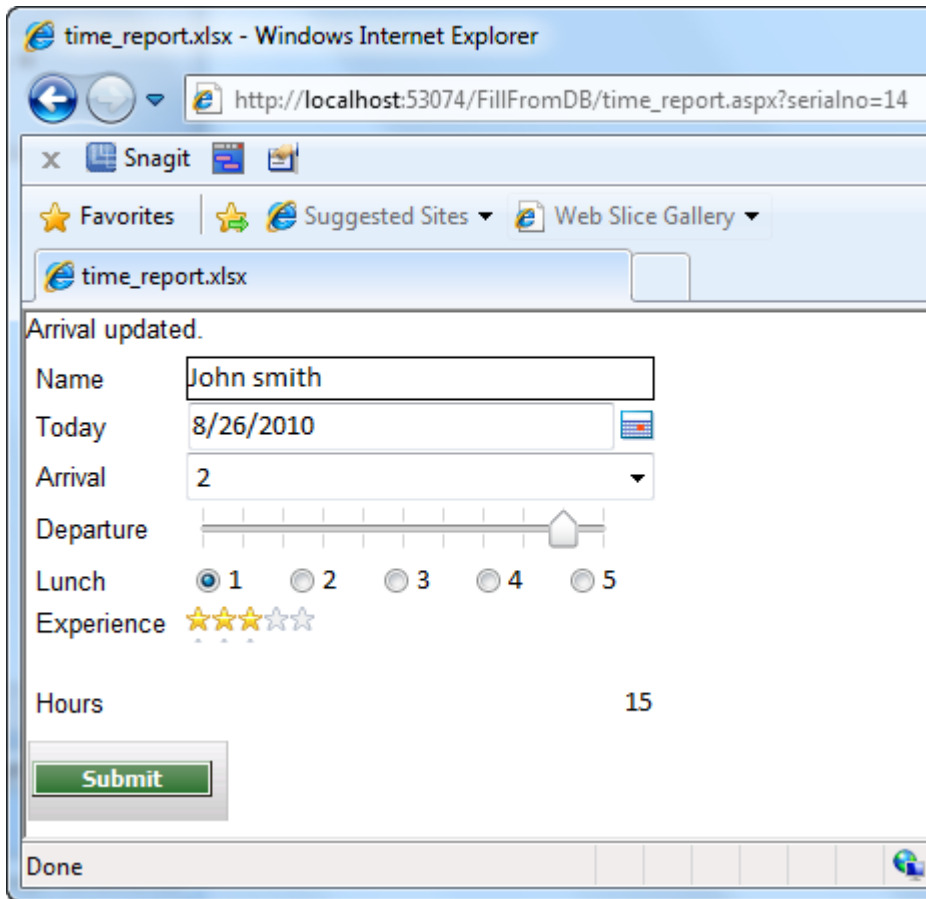


The screenshot shows a Windows Internet Explorer browser window titled "time_report.xlsx - Windows Internet Explorer". The address bar contains the URL "http://localhost:53074/FillFromDB/time_report.aspx?serialno=14". The browser's address bar shows "time_report.xlsx". The form contains the following fields and controls:

- Name:
- Today: (with a calendar icon)
- Arrival: (with a dropdown arrow)
- Departure: A slider control with a house icon in the center.
- Lunch: 1 2 3 4 5
- Experience:
- Hours: 9
- Submit:

The status bar at the bottom of the browser window shows "Done".

When changing departure to 18 and pressing save , we get



Looking into the database, I see that the rows has changed.

Conclusion

We added code that reads an old entry, and code that updates an old entry. Although the code is rather long, mostly depending on the fact that we have to use parameterized SQL-statements to get a secure site, it is simple & code to manipulate different controls is little tricky. In order to adapt this to your form, all you need to use is cut-and-paste and rename the column names.

Accessing old entries with the URL

http://localhost/FillFromDB/time_report.aspx?serialno=14 is not very nice, and in the next part we will use the GridView to create a clickable list of the existing entries.