



School of Information Technology

Practical 04: State Management

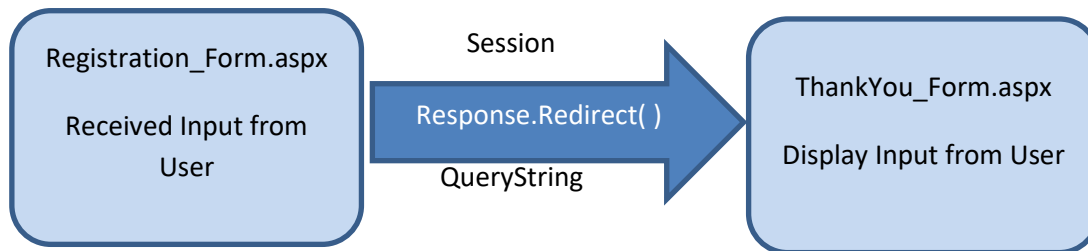
OBJECTIVES:

By the end of this Practical students should be able to:

- Extract data from Textbox, Radio Button, Checkbox, Dropdownlist
- Make use of Session and Query String to pass the extract data from one page to another.

Introduction

For this practical exercise, you will use Microsoft Visual Studio to create a server based Web application with 2 Web Forms – Registration_Form and ThankYou_Form. We will create a Registration Web application that allows user enter registration information in Registration_Form.aspx and send it to the ThankYou_Form.aspx for display. We will experience the use of both state management techniques (Session and Query String) to pass information from one form to the other.



State Management Web Application

HTTP is a stateless protocol. Once the server serves any request from the user, it cleans up all the resources used to serve that request. These resources include the objects created during that request, the memory allocated during that request, etc. There is no way the server could rely on objects and member variables alone to keep track of the current state of the application.

If we have to track the users' information between page visits and even on multiple visits of the same page, then we need to use the State management techniques provided by ASP.NET. State management is the process by which ASP.NET let the developers maintain state and page information over multiple requests for the same or different pages.

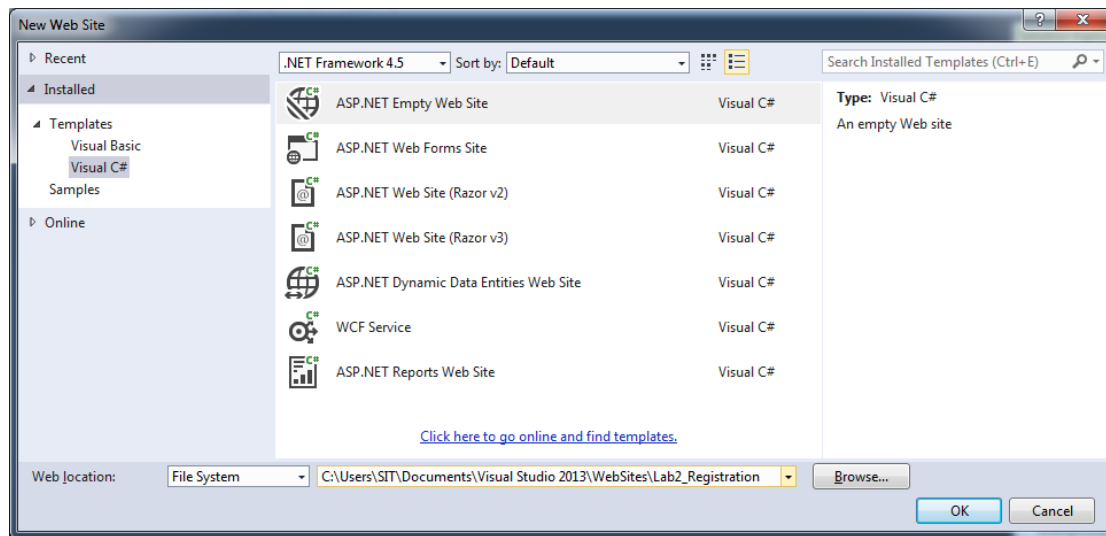
To overcome this inherent limitation of traditional Web programming, ASP.NET includes several options that help you preserve data on both a per-page basis and an application-wide basis. These features are as follows:

- View state
- Control state
- Hidden fields
- Cookies
- Query strings
- Application state
- Session state

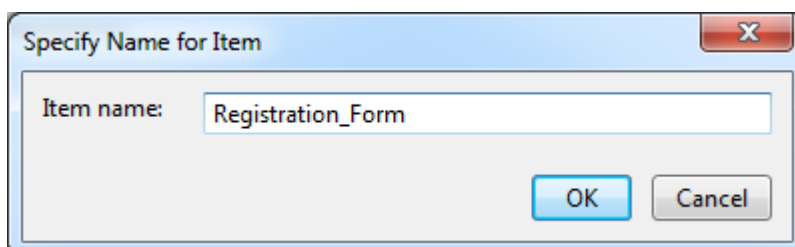
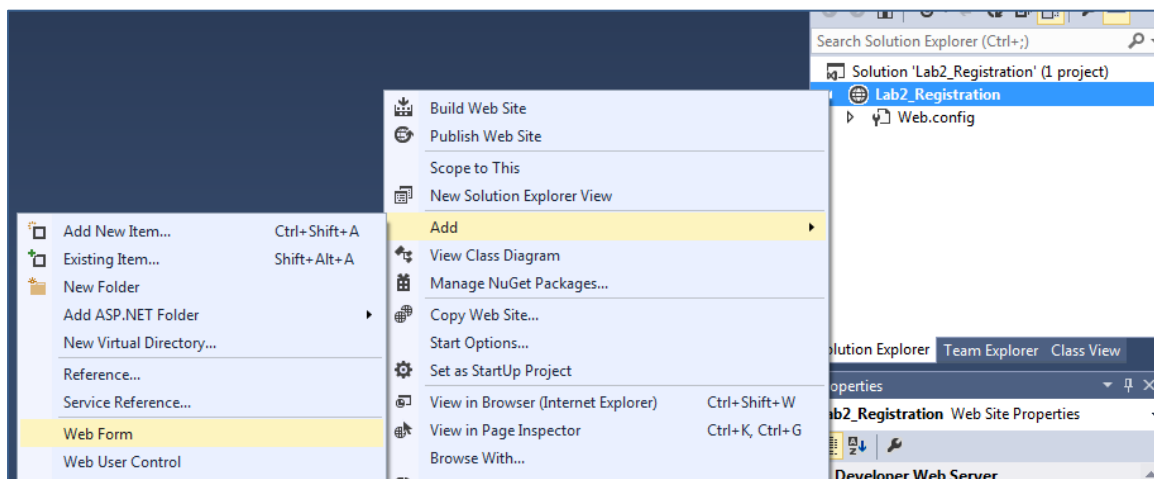
However, in this practical we will only focus on “Query Strings” and “Session state” as state management techniques.

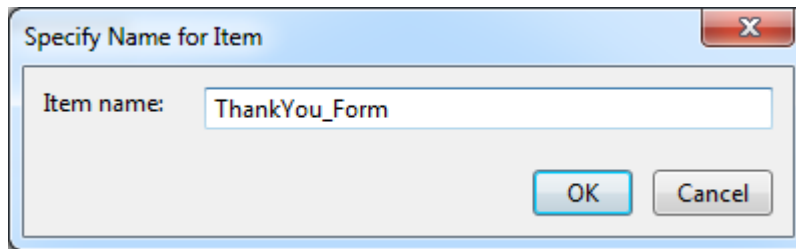
Exercise 1: Using the Query Strings Technique

1. Create a new Web site, Lab2_Registration → ASP.NET Empty Web Site.



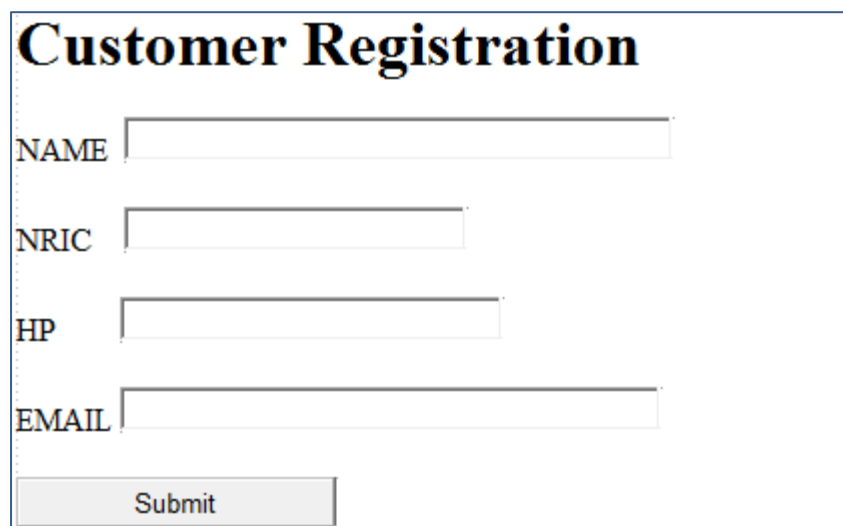
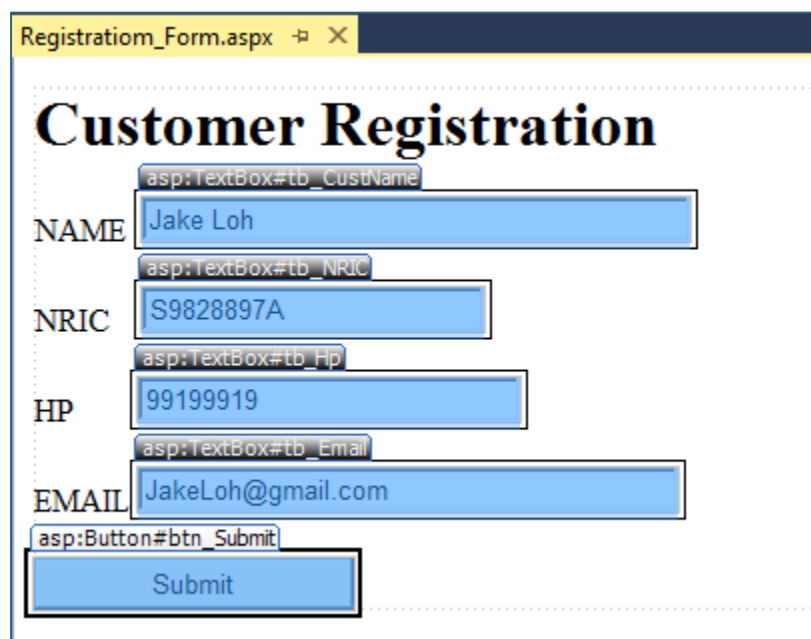
2. Right-Click "Lab2_Registration" website and choose "Add" → Web Form.
3. Create 2 forms – Registration_Form.aspx and ThankYou_Form.aspx.





4. In Registration_Form, create the following UI.
5. Use proper naming convention for your textboxes and buttons.

Create the following GUI



6. Double-click on the “submit button” to access the code behind.
7. Enter the codes into the button click event.

8. The query string should look something like this :

`http://url/ThankYou_form.aspx?CustName=Jake%20Loh&NRIC=S9828897A&Hp=99199919&Email=JakeLoh@gmail.com`

Each value pair is separated by the '&' character.

Example of a value pair : *CustName with value 'Jake Loh'*

```
public partial class Registration_Form : System.Web.UI.Page
{
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    0 references
    protected void btn_Submit_Click(object sender, EventArgs e)
    {
        //queryString - http://url/ThankYou_form.aspx?CustName=JakeLoh&NRIC=S9828897A&Hp=99199919&Email=JakeLoh@gmail.com
        //extract information from textbox and add to the query string
        string queryString = "?CustName=" + tb_CustName.Text + "&NRIC=" + tb_NRIC.Text + "&Hp=" + tb_Hp.Text + "&Email=" + tb_Email.Text;

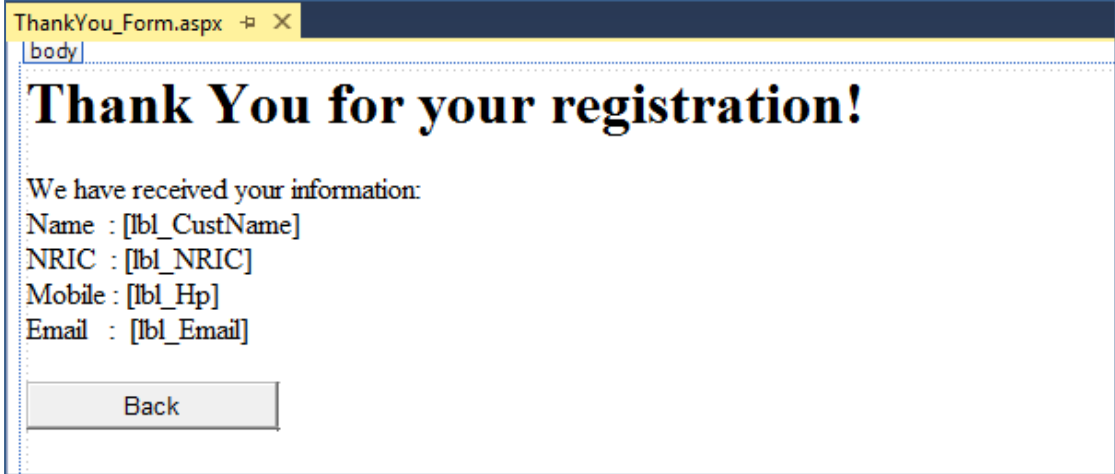
        //Redirect to Thank You Form for processing
        Response.Redirect("ThankYou_Form.aspx" + queryString);
    }
}
```

9. The partial query string would be something like that :

```
string queryString = "?CustName=" + tb_CustName.Text + "&" + "NRIC=" ...
```

10. Use `Response.Redirect(...)` to forward to another page.

11. In `ThankYou_Form.aspx`, create the following UI. Use proper naming convention.



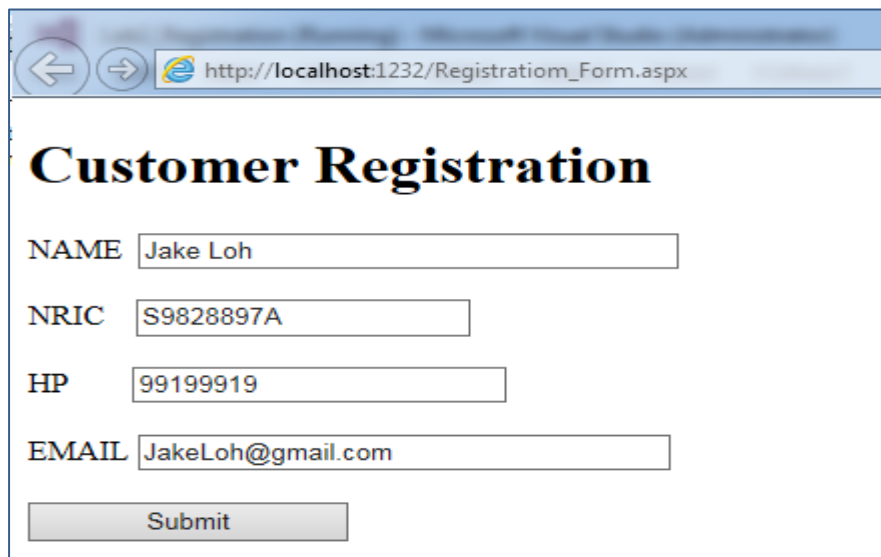
12. Go to the 'code behind' of the form and add the following codes.

13. Use `Request.QueryString[...]` to extract the info and save it to the label for display.

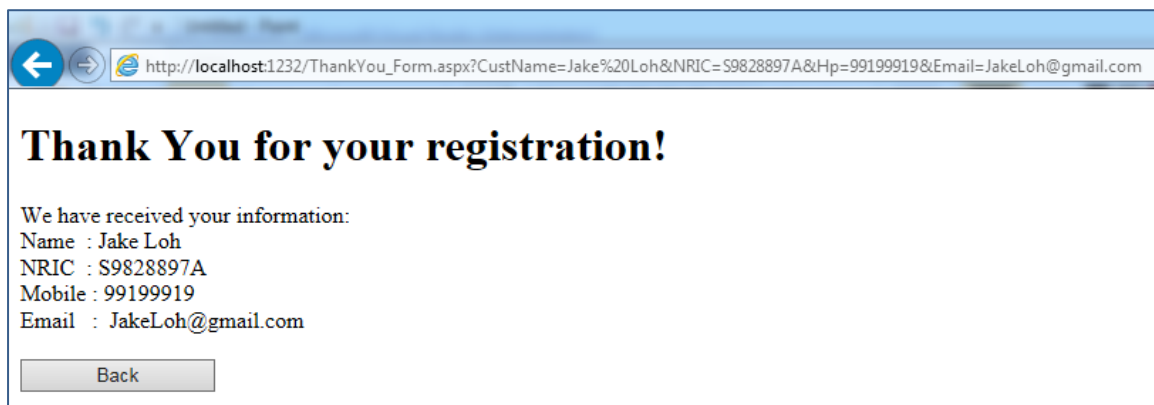
```
References
public partial class ThankYou_Form : System.Web.UI.Page
{
    References
    protected void Page_Load(object sender, EventArgs e)
    {
        //Extract data from QueryString - Request object and save it to the label
        lbl_CustName.Text = Request.QueryString["CustName"];
        lbl_NRIC.Text = Request.QueryString["NRIC"];
        lbl_Hp.Text = Request.QueryString["Hp"];
        lbl_Email.Text = Request.QueryString["Email"];
    }
    References
    protected void btnBack_Click(object sender, EventArgs e)
    {
        Response.Redirect("Registration_Form.aspx");
    }
}
```

14. Set the back button to point back to "Registration_Form.aspx".

15. Test run your application (press f5). Remember to set Registration_Form as the startup page.



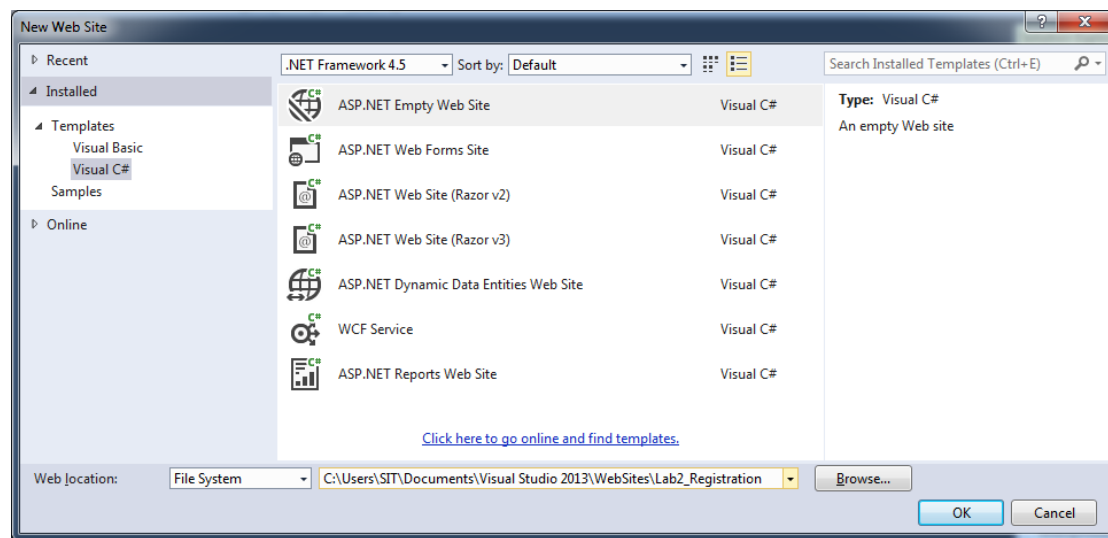
A screenshot of a web browser displaying the "Customer Registration" form. The browser's address bar shows the URL "http://localhost:1232/Registration_Form.aspx". The form has a title "Customer Registration" in a large, bold, serif font. Below the title are four input fields: "NAME" with the value "Jake Loh", "NRIC" with the value "S9828897A", "HP" with the value "99199919", and "EMAIL" with the value "JakeLoh@gmail.com". At the bottom of the form is a "Submit" button.



A screenshot of a web browser displaying the "Thank You for your registration!" page. The browser's address bar shows the URL "http://localhost:1232/ThankYou_Form.aspx?CustName=Jake%20Loh&NRIC=S9828897A&Hp=99199919&Email=JakeLoh@gmail.com". The page has a title "Thank You for your registration!" in a large, bold, serif font. Below the title, it says "We have received your information:" followed by a list of the user's details: "Name : Jake Loh", "NRIC : S9828897A", "Mobile : 99199919", and "Email : JakeLoh@gmail.com". At the bottom of the page is a "Back" button.

Exercise 2: Using the Session State Technique

1. Create a new Web site, Lab2_Registration_Session → ASP.NET Empty Web Site.



2. Right-Click “Lab2_Registration_Session” website and choose “Add” → Web Form.
3. Create 2 forms – Registration_Form.aspx and ThankYou_Form.aspx.
4. Alternatively, you may reuse the previous website for your practice.
5. Repeat the steps but use different codes for both the code behind for Registration_Form and ThankYou_Form.
6. Registration_Form.aspx code behind :

```

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

    }
    0 references
    protected void btn_Submit_Click(object sender, EventArgs e)
    {
        //extract info from textbox and save into session
        Session["CustName"] = tb_CustName.Text;
        Session["NRIC"] = tb_NRIC.Text;
        Session["Hp"] = tb_Hp.Text;
        Session["Email"] = tb_Email.Text;

        //Redirect to Thank You Form
        Response.Redirect("ThankYou_Form.aspx");
    }
}

```

7. Each data in save into the session with different key values (e.g “CustName”).

8. ThankYou_Form.aspx code behind :

```

References
public partial class ThankYou_Form : System.Web.UI.Page
{
    References
    protected void Page_Load(object sender, EventArgs e)
    {
        //Extract data from QueryString - Request object and save it to the label
        lbl_CustName.Text = (string)Session["CustName"];
        lbl_NRIC.Text = (string)Session["NRIC"];
        lbl_Hp.Text = (string)Session["Hp"];
        lbl_Email.Text = (string)Session["Email"];
    }
    References
    protected void btnBack_Click(object sender, EventArgs e)
    {
        Response.Redirect("Registration_Form.aspx");
    }
}

```

9. To extract the user input value, use the correct key value.

10. You might have already noticed that the above implementation does not take into account the possibilities of users not entering any value into the textboxes. If that happens, it might just crash your application. There are a few ways to prevent this :

- Ensure that users must enter something in the textbox (to be covered in later chapters on client side validation).
- Enforce server-side validation on all Session objects inside "Code behind". An example is shown below. You may use this method to implement your server-side validation for Session objects.

```

//Check whether session variable null or not
if (Session["UserName"] != null)
{
    //Retrieving UserName from Session
    lblWelcome.Text = "Welcome : " + Session["UserName"];
}
else
{
    //Do Something else
}

```


How about other controls?

RadioButton vs CheckBoxes

All subsequent GUI standards and the official W3C Web standards have included the same definition of these two controls:

1. Radio buttons are used when there is a list of two or more options that are mutually exclusive and the user must select exactly one choice. In other words, clicking a non-selected radio button will deselect whatever other button was previously selected in the list.
2. Checkboxes are used when there are lists of options and the user may select any number of choices, including zero, one, or several. In other words, each checkbox is independent of all other checkboxes in the list, so checking one box doesn't uncheck the others.

ToolBox – RadioButtonList

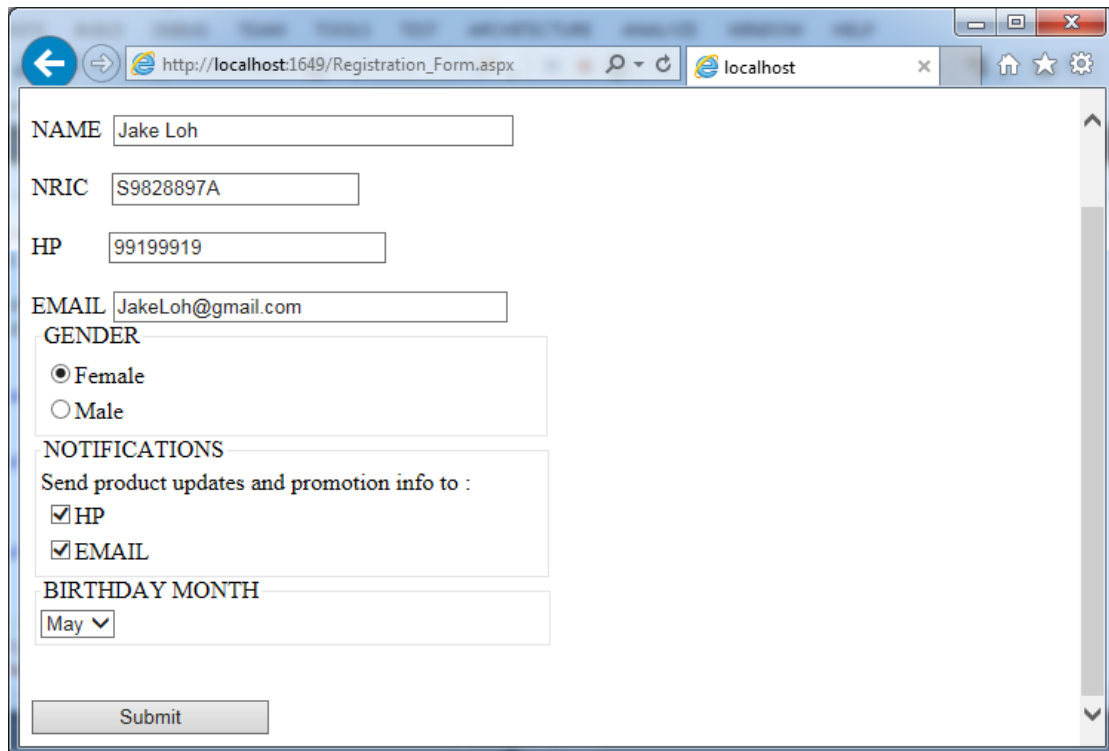
The RadioButtonList control is a single control that acts as a parent control for a collection of radio button list items. In ASP.Net radiobuttonlist control enable user to select an item from list. It can also be populated manually by input list item inside radiobuttonlist tag. radiobuttonlist is a single selection radio button group. radiobuttonlist have an items collection. we can determine which item is selected by test it's SelectedItem property.

ToolBox – CheckBoxList

The CheckBoxList control creates a multiselection check box group that can be dynamically generated using data binding. To specify items that you want to appear in the CheckBoxList control, place a ListItem element for each entry between the opening and closing tags of the CheckBoxList control. To determine the selected items in the CheckBoxList control, iterate through the Items collection and test the Selected property of each item in the collection.

Exercise 3: Using the Session State Technique for Checkbox, Radio Button and Dropdown list

Let's add 3 more additional UI to improve the registration process to make it more purposeful – Gender, Notification methods and Birthday Month.



NAME

NRIC

HP

EMAIL

GENDER

☒ Female

☐ Male

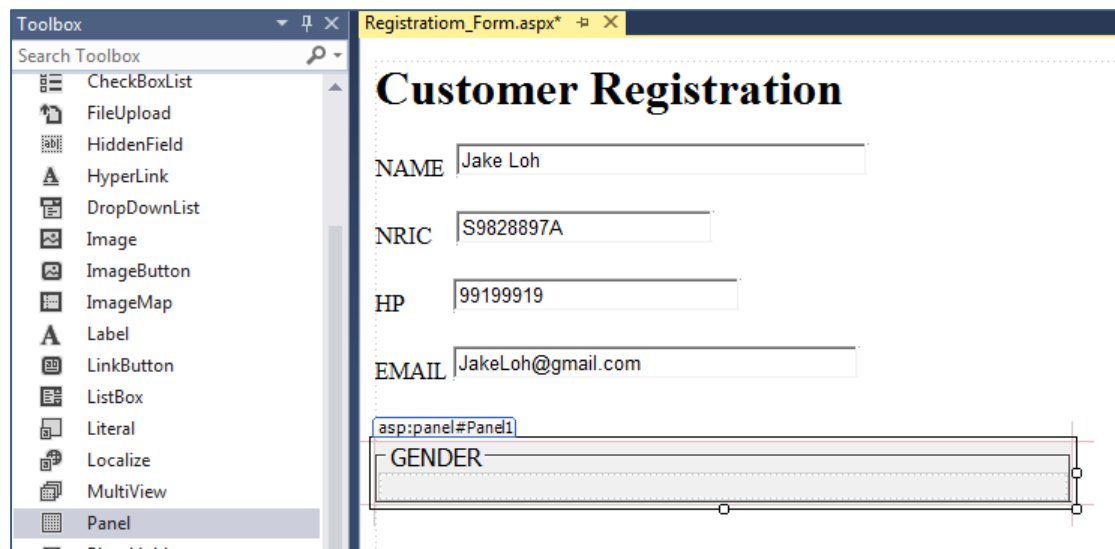
NOTIFICATIONS

Send product updates and promotion info to :

☒ HP

☒ EMAIL

BIRTHDAY MONTH



Toolbox

Search Toolbox

- CheckBoxList
- FileUpload
- HiddenField
- HyperLink
- DropDownList
- Image
- ImageButton
- ImageMap
- Label
- LinkButton
- ListBox
- Literal
- Localize
- MultiView
- Panel
- PlaceHolder

Registration_Form.aspx*

Customer Registration

NAME

NRIC

HP

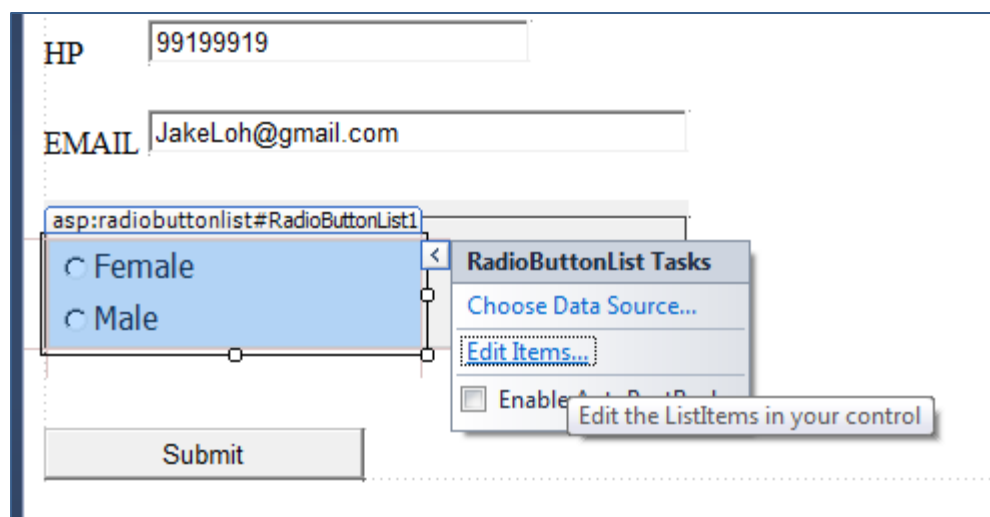
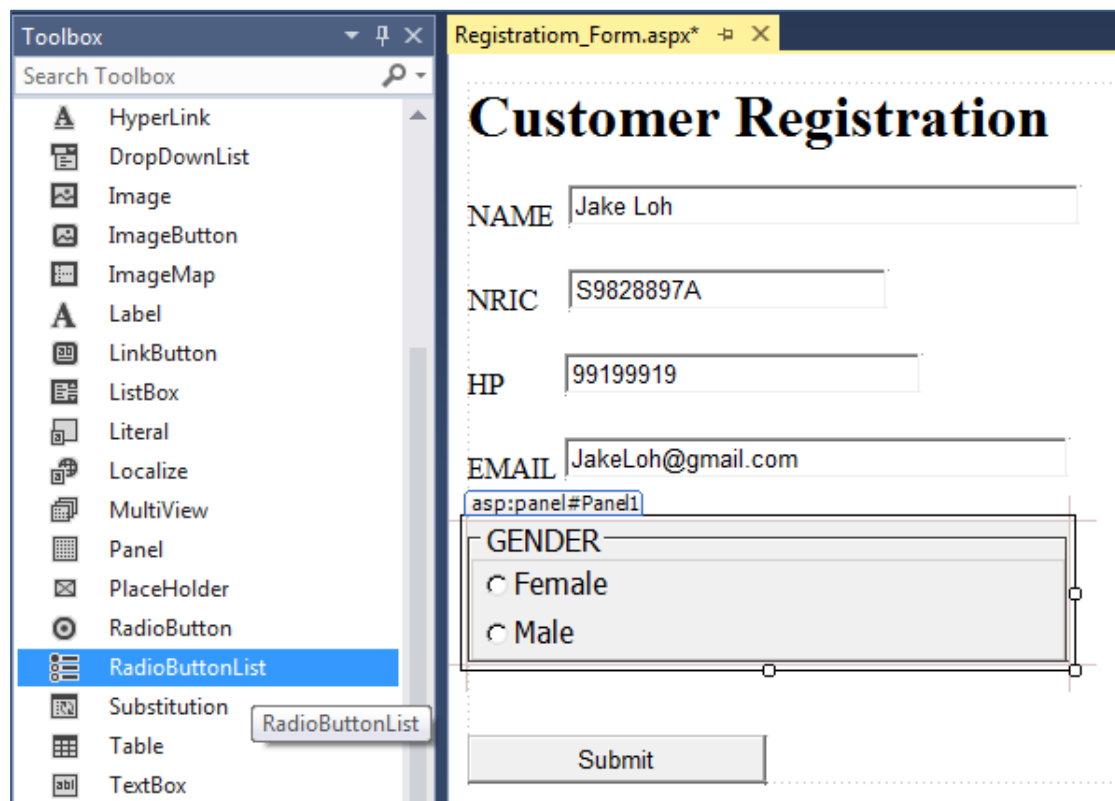
EMAIL

asp:panel#Panel1

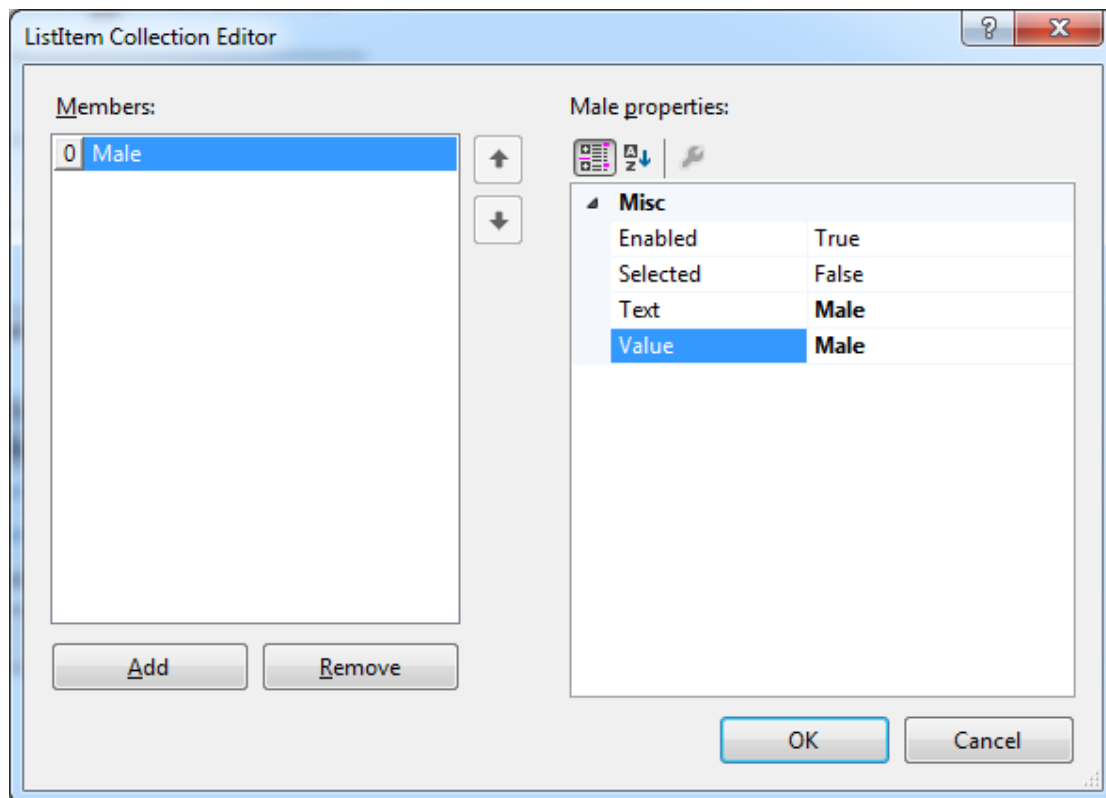
GENDER

16. Drag-drop "Panel" after the Email section. Set the Panel properties "GroupingText" as "GENDER"

17. Drag-drop "RadioButtonList" into the Panel. Set ID to "rbl_Gender".



18. Select the "Edit Items" to manually create 2 items – Male and Female.
19. Select "Add" → enter Text "Male" and Value "Male"



20. Repeat the above steps to create the Female Item. Done!

EMAIL

GENDER

☐ Female

☐ Male

NOTIFICATIONS

Send product updates and promotion info to :

☐ HP

☐ EMAIL

21. Create the "NOTIFICATIONS" section.

22. Drag-drop Panel followed by "CheckBoxList". Set ID to **cbl_Notifications**. Add 2 checkboxes for Notification purpose – HP and EMAIL.

23. Add items into the CheckBoxList (similar to the way RadioButtonList is being configured)

A screenshot of a web form with a light blue background. The form contains several input fields and sections. At the top, there are four text input fields: 'NAME' with 'Jake Loh', 'NRIC' with 'S9828897A', 'HP' with '99199919', and 'EMAIL' with 'JakeLoh@gmail.com'. Below these is a 'GENDER' section with two radio buttons: 'Female' and 'Male'. Next is a 'NOTIFICATIONS' section with the text 'Send product updates and promotion info to :' followed by two checkboxes: 'HP' and 'EMAIL'. Below that is a 'BIRTHDAY MONTH' section with a dropdown menu currently showing 'Jan'. At the bottom of the form is a 'Submit' button.

24. Create the "BIRTHDAY MONTH" section.
25. Drag-drop Panel followed by "DropDownList" – set ID to **ddl_BirthdayMonth**.
26. Add items into the CheckBoxList (Jan to Dec). Select "Edit Items" button.
27. Done !
28. Let move to "Code behind" and write some c# codes to interact with the controls you have just created.
29. The entire code for the "btn_Submit" is shown below.

```
protected void btn_Submit_Click(object sender, EventArgs e)
{
    //extract info from textbox and save into session with the keyword "CustName"
    Session["CustName"] = tb_CustName.Text;
    Session["NRIC"] = tb_NRIC.Text;
    Session["Hp"] = tb_Hp.Text;
    Session["Email"] = tb_Email.Text;

    //if rbl_gender is selected, the SelectedIndex will be more than -1.
    string gender = null;
    if (rbl_Gender.SelectedIndex > -1){
        gender = rbl_Gender.SelectedItem.Text;
        Session["Gender"] = gender;
    }

    //if cbl_Notifications is selected, the SelectedIndex will be more than -1.
    string notifications = null;
    if (cbl_Notifications.SelectedIndex > -1) {
        if (cbl_Notifications.Items[0].Selected)
        {
            notifications = notifications + cbl_Notifications.Items[0].Text + ";";
        }
        if (cbl_Notifications.Items[1].Selected)
        {
            notifications = notifications + cbl_Notifications.Items[1].Text + ";";
        }
    }
    Session["Notifications"] = notifications;

    string birthdayMonth = null;
    if (ddl_BirthdayMonth.SelectedIndex > -1) {
        birthdayMonth = ddl_BirthdayMonth.SelectedItem.Text;
    }
    Session["BirthdayMonth"] = birthdayMonth;

    //Forward your request to ThankYou.aspx for procesing
    Response.Redirect("ThankYou_Form.aspx");
}
```

30. Code examination for "rbl_Gender". To check if any of the Items in the radiobuttonlist is selected, inspect the "SelectedIndex" property which must be more than -1.

```
//if rbl_gender is selected, the SelectedIndex will be more than -1.
string gender = null;
if (rbl_Gender.SelectedIndex > -1){
    gender = rbl_Gender.SelectedItem.Text;
    Session["Gender"] = gender;
}
```

31. Code examination for "cbl_Notifications". Similarly, to check if any of the Items in the CheckBoxList is selected, check if the "SelectedIndex" property is more than -1. Access the value of the all selected Items by using the .Items[index] method.

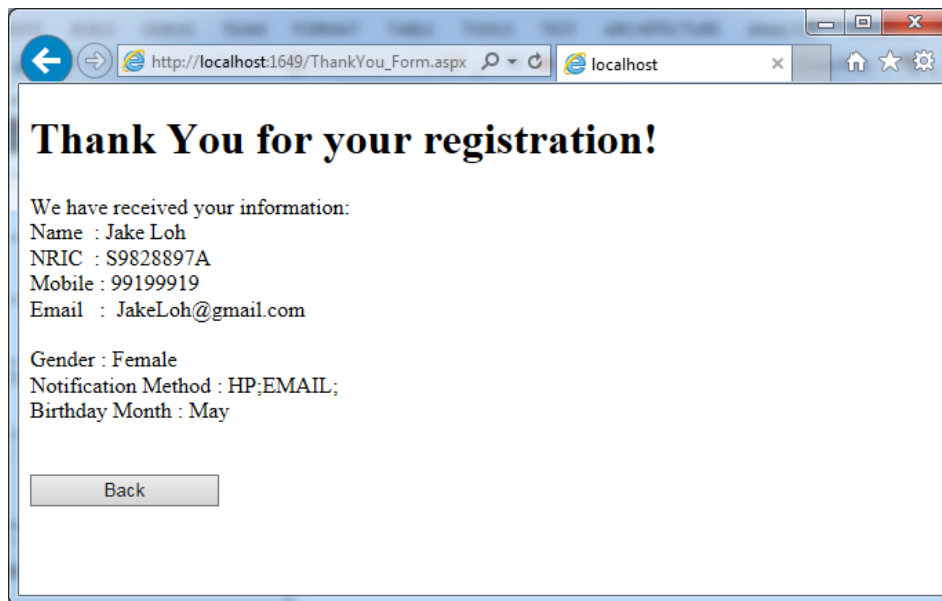
Question: Can you change the codes such that you do not need to alter the following codes when you add more items to the checkboxlist? Show your tutor your updated codes. Hint: Use for loop.

```
//if cbl_Notifications is selected, the SelectedIndex will be more than -1.  
string notifications = null;  
if (cbl_Notifications.SelectedIndex > -1) {  
    if (cbl_Notifications.Items[0].Selected)  
    {  
        notifications = notifications + cbl_Notifications.Items[0].Text + ";";  
    }  
    if (cbl_Notifications.Items[1].Selected)  
    {  
        notifications = notifications + cbl_Notifications.Items[1].Text + ";";  
    }  
}  
Session["Notifications"] = notifications;
```

32. Code examination for "ddl_BirthdayMonth". Similarly, to check if any of the Items in the DropDownList is selected, check if the "SelectedIndex" property is more than -1. Access the value of the selected Item by using the .SelectedItem property.

```
string birthdayMonth = null;  
if (ddl_BirthdayMonth.SelectedIndex > -1) {  
    birthdayMonth = ddl_BirthdayMonth.SelectedItem.Text;  
}  
Session["BirthdayMonth"] = birthdayMonth;
```

33. Next, make necessary changes to "ThankYour_Form.aspx" with the output shown below. The output is based on the input submitted by Registration_Form.aspx.



End of Practical