HTML Forms and Java Servlets

HyperText Markup Language (HTML) is used to create web pages that can be viewed with a browser. There are a number of things you can do with it, including changing the color of the page, adding Javascript, Java applets, or VB Script, and arranging information in lists and tables. There are also a different kinds of forms that can be used to gather data from the client. There include radio buttons, check boxes, and list boxes as well as the text boxes we used previously.

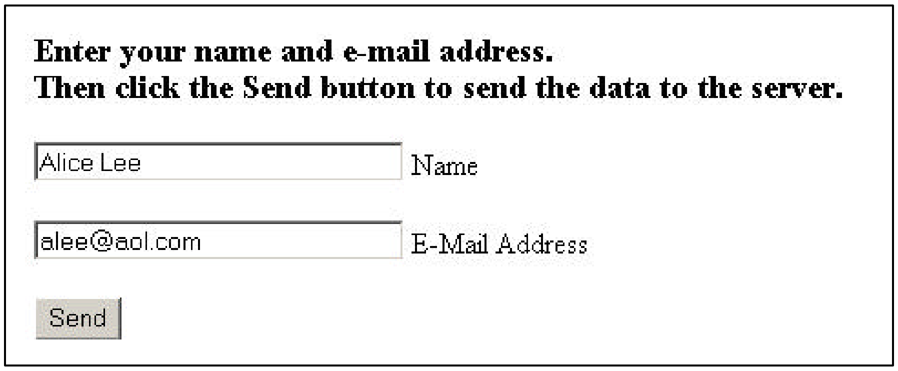
Forms in html begin with the <form> start-tag and close with the </form> end-tag. Forms are to be filled out by the user. They do everything from sending in address information to helping a buyer choose the size or color of a product. In addition, they tell the server where to find the servlet or JSP file that is to be used to process the request.

Text Boxes and Buttons

A simple form displays some input boxes and a button to submit the data to the server. The example below shows a complete html page that creates a form to send the name and e-mail address to the server.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
	<head><title>E-Mail Form</title></head>
	<body>
		<h3>Enter your name and e-mail address.
		<br />Then click the Send button to send the data to the server.</h3>
		<form method = "get" action="http://127.0.0.1/servlet/client_server.EmailServlet">
			<p><input type = "text" name = "name" value = "" size = 30 /> Name </p>
			<p><input type = "text" name = "email" value = "" size = 30 /> E-Mail Address </p>
			<p><input type = "submit" value = "Send" /></p>
		</form>
	</body>
</html>
```

Here the *head* tags only supply a title that will be shown at the top of the browser when the page is loaded. The *body* of the document contains a message to the user to enter data and click on the send button. The *form* first supplies the *method* that the server will use to process the data and the *action* information that tells the server what program to use for the processing. The form displays two *input* text boxes and a *submit* button. The *type* information is used to tell the browser what kind of object to display. A text box displays a box where the user can type in data. Its initial *value* is the empty string. But after data is entered, the value of the box will be whatever was typed in. When the type is submit, the browser displays a button with a caption given by the value attribute.


Enter your name and e-mail address.
Then click the Send button to send the data to the server.

Alice Lee Name

alee@aol.com E-Mail Address

A Java program that processes this request follows:

```
package client_server;

/**
 * EmailServlet processes a request from a web page.  It responds to the
 * request by sending back a web page listing the name and email address.
**/
import java.sql.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class EmailServlet extends HttpServlet
{
	public void doGet (HttpServletRequest request, HttpServletResponse response)
	{
		try
		{
			// Get a PrintWriter object and respond to the request.
			PrintWriter out = response.getWriter ();
			String name = request.getParameter ("name");
			String email = request.getParameter ("email");

			Page.createHeader (out, "Addresses");
			out.println ("<h3>Hello.</h3>");
			out.println ("<h3>" + name+ "</h3>");
			out.println ("<h3>Your email address is " + email + "</h3>");
			Page.createFooter (out);
		} catch (ClassNotFoundException e){System.out.println ("Class Not Found exception.\n");}
		  catch (SQLException e){System.out.println ("SQL Exception");}
		  catch (IOException ex) {System.out.println ("IO Exception.");}
	} // doGet
} // EmailServlet
```

Radio Buttons

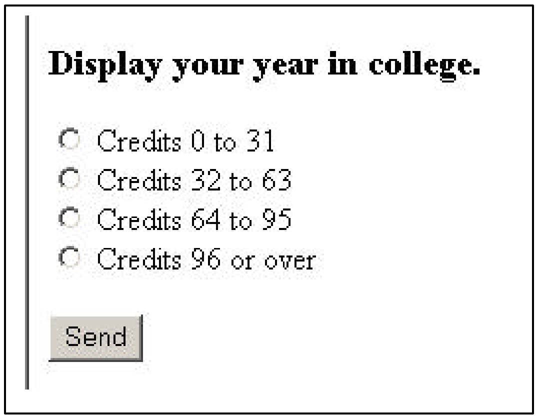
Another way to get information from a form is to use radio buttons. A set of radio buttons lets the user choose one option from a set. The example below asks a user to indicate his or her year in college.

```
<body> <h3>Display your year in college.</h3>
<form method = "get" action="http://127.0.0.1/servlet/client_server.CollegeYearProcessor">
	<input type="radio" name = "year" value = "First Year" /> Credits 0 to 31
	<br /><input type = "radio" name = "year" value = "Second Year" /> Credits 32 to 63
	<br /><input type = "radio" name = "year" value = "Third Year" /> Credits 64 to 95
	<br /><input type = "radio" name= "year" value = "Fourth Year" /> Credits 96 or over
	<p><input type = "submit" value = "Send" /></p>
</form> </body>
```

Note the URL string that the browser prepares for this form:

```
	GET/ client_server.CollegeYearProcessor?year=Fourth+Year HTTP/1.1
```

Only one value is sent to the server.



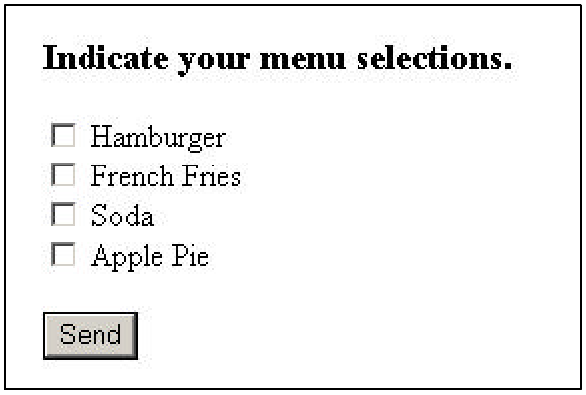
A Java program that processes this request follows:

```
package client_server;

/**
 * CollegeYearServlet processes a request from a web page.  It responds to the
 * request by sending back a web page listing the student's year in college.
**/
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class CollegeYearServlet extends HttpServlet
{
	public void doGet (HttpServletRequest request, HttpServletResponse response)
	{
		try
		{

			// Get the request parameter with type year.
			String year = request.getParameter ("year");

			// Get a PrintWriter object and respond to the request.
			PrintWriter out = response.getWriter ();
			Page.createHeader (out, "College Class");
			out.println ("<h3>Hello.</h3>");
			out.println ("<h3>Your year in college is " + year + "</h3>");

			Page.createFooter (out);
		} catch (ClassNotFoundException e){System.out.println ("Class Not Found exception.\n");}
		  catch (SQLException e){System.out.println ("SQL Exception");}
		  catch (IOException ex) {System.out.println ("IO Exception.");}
	} // doGet
} // CollegeYearServlet
```

Check Boxes

Check boxes are very similar to radio buttons. But they are individually named rather than all having the same name. You also may select more than one choice with a check box. The following is an example where that would be appropriate.

```
<body> <h3>Indicate your menu selections.</h3>
<form method = "get" action="http://127.0.0.1/servlet/client_server.CheckBoxProcessor">
        <input type="checkbox" name = "menu" value = " Hamburger" /> Hamburger
        <br /><input type = "checkbox" name = "menu" value = " French Fries" /> French Fries
        <br /><input type = "checkbox" name = "menu" value = " Soda" /> Soda
        <br /><input type = "checkbox" name= "menu" value = " Apple Pie" /> Apple Pie
        <p><input type = "submit" value = "Send" /></p>
</form>
</body>
```



Since it is possible to make several selections at once, you cannot use getParameter (…) to get the value. Instead Java servlets use getParameterValues (…). The following is an example.

```
      String [] choices = request.getParameterValues ("menu");
```

Note that getParameterValues (…) returns an array, not a single value. Since some of the choice may be empty, use an *if* statement to test for this.

```
      for (int count = 0; count < choices.length; count ++)
      {
            if (choices [count] != null) out.println ("" + choices [count] + ", ");
      }
```

This example only displays the results. Certainly other things can be done with the parameters as well.

```
package client_server;

/**
 * CheckBoxServlet processes a request from a web page that contains check boxes. It responds to the
 * request by sending back a web page listing the menu choices selected.
**/

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

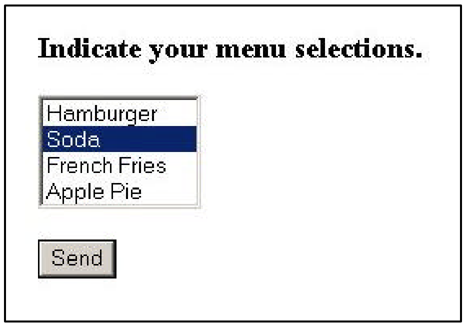
```
public class CheckBoxServlet extends HttpServlet
{
	public void doGet (HttpServletRequest request, HttpServletResponse response)
	{
		try
		{
			// Get the choices, an array of Strings.
			String [] choices = request.getParameterValues ("menu");

			// Get a PrintWriter object and respond to the request.
			PrintWriter out = response.getWriter ();
			Page.createHeader (out, "Menu Choices");
			out.println ("<h3>Hello.</h3>");
			out.println ("<h3>Your choices are ");
			// Print out the non-null values in the array.
			for (int count = 0; count < choices.length; count ++)
			{
				if (choices [count] != null)
					out.println ("" + choices [count] + ", ");
			}
			out.println ("</h3>");

			Page.createFooter (out);
		} catch (ClassNotFoundException e){System.out.println ("Class Not Found exception.\n");}
		  catch (SQLException e){System.out.println ("SQL Exception");}
		  catch (IOException ex) {System.out.println ("IO Exception.");}
	} // doGet
} // CheckBoxServlet
```

List Boxes

List boxes are similar to check boxes. They have a list of options that the user may choose from.

```
<body> <h3>Indicate your menu selections.</h3>
<form method = "get" action="http://127.0.0.1/servlet/client_server.ListBoxProcessor">
	<select name = "menu" size = "4" multiple = "multiple" >
		<option /> Hamburger
		<option /> Soda
		<option /> French Fries
		<option /> Apple Pie
	</select>
	<p><input type = "submit" value = "Send" /> </p>
</form> </body>
```

The size attribute determines how many items will be shown. If there are more options than the number given by size, a scroll bar is added. If you want to allow the selection of more than one item at a time, include the *multiple* attribute. (In order to select several items, users have to hold down the control key when making the selection.)



```
package client_server;

/**
 * ListBoxServlet processes a request from a web page.  It responds to the
 * request by sending back a web page listing the menu choices selected.
**/
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;


public class ListBoxServlet extends HttpServlet
{
	public void doGet (HttpServletRequest request, HttpServletResponse response)
	{
		try
		{
			// Get the choices, an array of Strings.
			String [] choices = request.getParameterValues ("menu");

			// Get a PrintWriter object and respond to the request.
			PrintWriter out = response.getWriter ();
			Page.createHeader (out, "Menu Choices");
			out.println ("<h3>Hello.</h3>");
			out.println ("<h3>Your choices are ");
			// Print out the non-null values in the array.
			for (int count = 0; count < choices.length; count ++)
			{
				if (choices [count] != null)
					out.println ("" + choices [count] + ", ");
			}
			out.println ("</h3>");

			Page.createFooter (out);
		} catch (ClassNotFoundException e){System.out.println ("Class Not Found exception.\n");}
		  catch (SQLException e){System.out.println ("SQL Exception");}
		  catch (IOException ex) {System.out.println ("IO Exception.");}
	} // doGet
} // ListBoxServlet
```