CREATE A FORM

Adding a form to a Web page allows you to gather data from users who visit the page. A form can be placed anywhere between the `<body>` and `</body>` tags in an HTML document. The body of your Web page can include as many forms as you need.

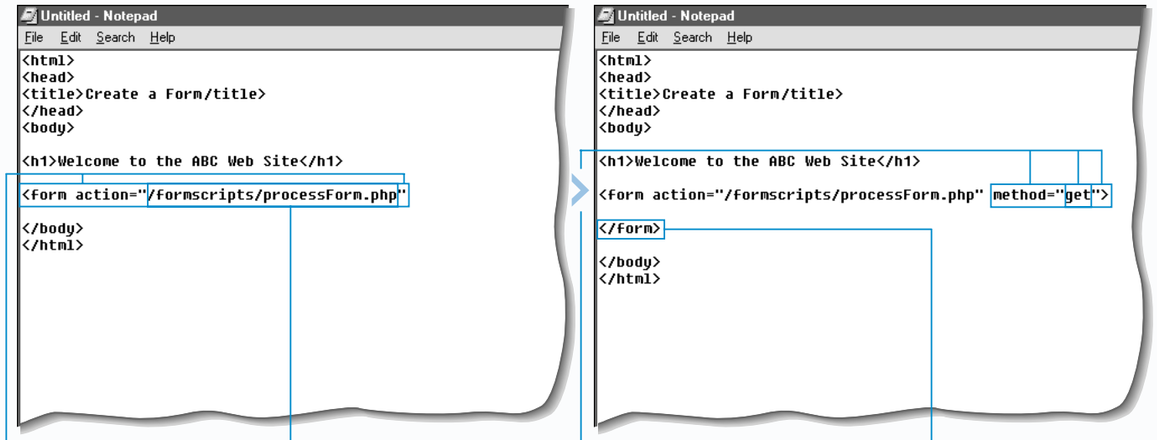
You use the `<form>` tag to create a form and the `action` attribute to specify the location and name of the PHP page that will process the data entered into the form. If the PHP page is stored in the same directory as the Web page containing the form, you have to specify only the name of the PHP page. If the PHP page is not stored on the same Web server as the Web page containing the form, you must specify the full URL of the PHP page.

You must also specify which method the form will use to pass data to the PHP page. There are two methods the form can use—`get` and `post`. The method you should use depends on the amount of data that will be passed. The `get` method sends data to the PHP page by appending the data to the URL of the page. The `post` method sends the data and the URL separately. The `get` method is faster than the `post` method and is suitable for small forms. The `post` method is suitable for large forms that will send more than 2000 characters to a PHP page.

Unlike many other technologies used to process form information, PHP can automatically determine whether a form is submitting data using the `get` or `post` method and then retrieve the information accordingly.

For information about creating a PHP page that processes data from a form, see page 140.

CREATE A FORM



1 Type **<form action=""** where you want to add a form to a Web page.

2 Between the quotation marks, type the location and name of the PHP page that will process the data entered into the form.

3 Type **method="">**.

4 Between the quotation marks, type the method the form will use to pass data to the PHP page.

5 Type **</form>** where you want to end the form.

■ You can now add elements to the form.

ADD ELEMENTS TO A FORM

Elements are areas in a form where users can enter data and select options. The most commonly used element is a text box, which allows users to enter a single line of data into a form. Text boxes are often used for entering names, addresses and other short responses.

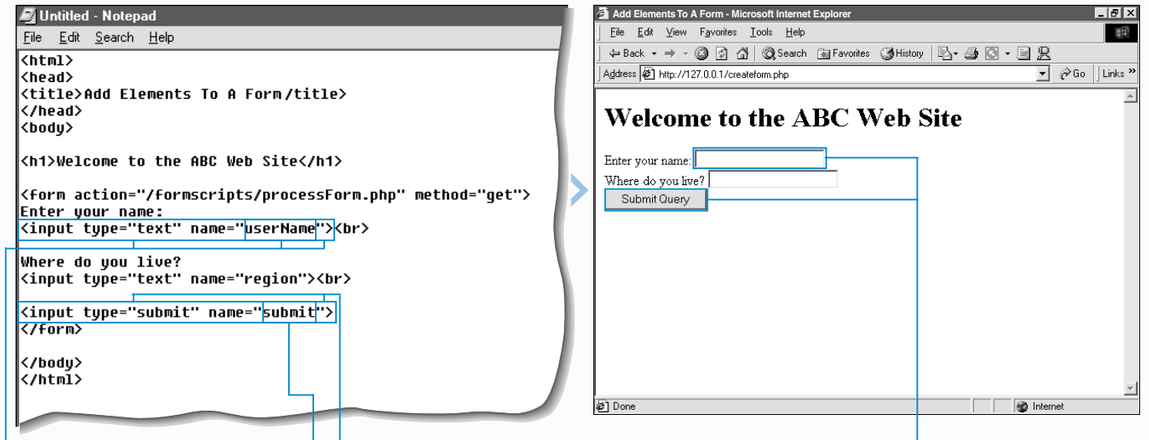
Elements you add to a form must be placed between the `<form>` and `</form>` tags. A form can contain as many elements as you need.

There are many different types of elements you can add to a form, such as text areas and check boxes. Text areas allow users to enter several lines or paragraphs of text, while check boxes let users select options on a form. For information about commonly used elements, see page 138.

Each form element has attributes, such as `name`, `type` and `size`, which offer options for the element. The `name` attribute allows you to provide a name for an element. The name you specify is used by the PHP page that processes the form to identify the element and access the information in the element. A name can contain letters and numbers, but should not contain spaces or punctuation. If you want to include spaces in a name, use an underscore character (_) instead.

You must add a submit button to every form you create. The submit button allows users to send the data they entered into the form to the Web server. When the Web server receives data from a form, the server transfers the data to the PHP page that will process the data. The PHP page can then perform an action with the data, such as storing the data in a database or displaying the information in a Web browser.

ADD ELEMENTS TO A FORM



1 To add a text box to a form, type **<input type="text" name="">** between the `<form>` and `</form>` tags.

2 Between the quotation marks, type a word that describes the text box.

3 To add a submit button to the form, type **<input type="submit" name="">**.

4 Between the quotation marks, type a word that describes the button.

5 Display the Web page in a Web browser.

■ The Web browser displays the text box and submit button.

FORM ELEMENTS

An element is an area in a form where users can enter data or select options. There are several different types of elements you can add to a form.

Most elements require you to specify attributes that determine how the element will appear on a Web page.

You can find more information about form elements and attributes at the www.w3.org/TR/1999/REC-html401-19991224/interact/forms Web site.

COMMONLY USED ATTRIBUTES

Type

The `type` attribute allows you to specify the kind of element you want to use.

Name

The `name` attribute allows you to specify a name for an element. The PHP page that will process data from the element uses the `name` attribute to identify the data. Element names can contain more than one word, but should not contain spaces or special characters.

Value

The `value` attribute allows you to specify a value for an element. If an element displays a button, you can use the `value` attribute to specify the text that will appear on the button.

Maxlength

The `maxlength` attribute allows you to restrict the number of characters a user can enter into an element.

Size

The `size` attribute allows you to specify the width of an element.

Checked

The `checked` attribute allows an element to display a selected option by default.

COMMONLY USED ELEMENTS

Password Box

A password box allows users to enter private data. When a user types data into a password box, an asterisk (*) appears for each character, which prevents others from viewing the data on the screen. A password box does not protect the data from being accessed as it is transferred over the Internet. You must set the `type` attribute to `password` and use the `name` attribute to create a password box. You may also want to use the `value`, `maxlength` and `size` attributes.

```
Password Please <input type="password"
name="secretWord" value="password" maxlength="20">
```

Password Please ********

Drop-Down List

The `select` element displays a drop-down list that allows users to select an option from a list of several options. For example, a drop-down list can be used to allow users to select one of three shipping methods. You must use the `name` attribute to create a drop-down list. You use the `<option>` tag with the `value` attribute to add options to the list.

```
How would you like your products shipped?
<select name="shipMethod">
<option value="air">Air</option>
<option value="land">Land</option>
<option value="sea">Sea</option>
</select>
```

How would you like your products shipped? Air

Air
Land
Sea

COMMONLY USED ELEMENTS (CONTINUED)

Text Box

A text box allows users to enter a single line of text, such as a name or telephone number. You must set the `type` attribute to `text` and use the `name` attribute to create a text box. You may also want to use the `maxlength` and `size` attributes.

```
First Name <input type="text" name="firstName" maxlength="20">
```

First Name

Text Area

The `textarea` element displays a large text area that allows users to enter several lines or paragraphs of text. A large text area is ideal for gathering comments or questions from users. You must use the `name` attribute to create a text area.

```
Questions? <textarea name="userQuestions"></textarea>
```

Questions?

Check Box

Check boxes allow users to select one or more options. For example, check boxes can be used to allow users to specify which states they have visited. You must set the `type` attribute to `checkbox` and use the `name` and `value` attributes to create a check box. You may also want to use the `checked` attribute.

```
Which states have you visited in the past year?<br>
New York <input type="checkbox" name="states" value="New York" checked>
California <input type="checkbox" name="states" value="California">
Texas <input type="checkbox" name="states" value="Texas">
```

Which states have you visited in the past year?
New York ☑ California ☐ Texas ☐

Radio Button

Radio buttons allow users to select only one of several options. For example, radio buttons can be used to allow users to specify if they are male or female. You must set the `type` attribute to `radio` and use the `name` and `value` attributes to create a radio button. You may also want to use the `checked` attribute.

```
What is your gender?<br>
Female <input type="radio" name="gender" value="female" checked>
Male <input type="radio" name="gender" value="male">
```

What is your gender?
Female ◉ Male ○

Submit Button

A submit button allows users to send data in the form to the PHP page that will process the data. You must add a submit button to each form you create. You must set the `type` attribute to `submit` to create a submit button. You may also want to use the `name` and `value` attributes.

```
<input type="submit" name="submit" value="Submit Now">
```

Submit Now

Reset Button

A reset button allows users to clear the data they entered into a form. A user cannot redisplay data that has been cleared. Reset buttons are commonly used in forms that have many text boxes. You must set the `type` attribute to `reset` to create a reset button. You may also want to use the `value` attribute.

```
<input type="reset" value="Click to Reset">
```

Click to Reset

PROCESS DATA FROM A FORM

After creating a form on a Web page, you can create a PHP page that will process data submitted in the form. A PHP page can use information it accesses from a form to perform tasks such as displaying the data or storing the data in session variables or a database.

PHP makes it easy to process data from a form. When a PHP page receives form information, the page automatically converts the names of form elements to PHP variables and assigns the data entered in the elements to the variables. To process data entered in a form element, you simply access the variable for the element by prefixing the name of the element with a dollar sign ($). The name of a form element is specified when the element is created on the Web page containing the form. For information about creating elements on a form, see page 137.

The PHP page that processes a form also automatically identifies the method the form is using to transfer information. This allows you to create a PHP page that processes form data without having to specify whether the form uses the `get` or `post` method. For more information about the methods a form can use to transfer information, see page 136.

After creating and saving a PHP page that processes data submitted in a form, you should review the code for the Web page that contains the form to verify that the `action` attribute displays the correct filename and location for the PHP page. You should also verify that each form element has a unique name to ensure that the PHP page will create a unique variable for each form element. As with all variable names in PHP, the names of variables that store form data are case sensitive.

Extra

In addition to using forms, data can also be passed to a PHP page by a query string. A query string is one or more name-value pairs appended to the URL of a PHP page. To create a query string, you enter the URL of the PHP page in a Web browser, followed by a question mark. You then enter a name followed by an equal sign and a value for the name. To enter multiple name-value pairs, separate each pair with an ampersand (&). A query string should not exceed 2000 characters. The PHP page that receives the data will create variables using the names you specify and will assign the specified values to the variables.

Example:

```
http://www.abccorp.com/processform.php?userName=Ernest&region=USA
```

As a security precaution, a PHP page that processes form data should not be used to display secure information based on the data submitted in the form. For example, you should not display troubleshooting or administrative information when a user name such as admin is entered in a text box. Form data can be easily manipulated, allowing unauthorized users to access the secure information.

The `empty` function can be used to verify whether a user entered data in a form element. If a user leaves the form element blank, the `empty` function will return a value of true.

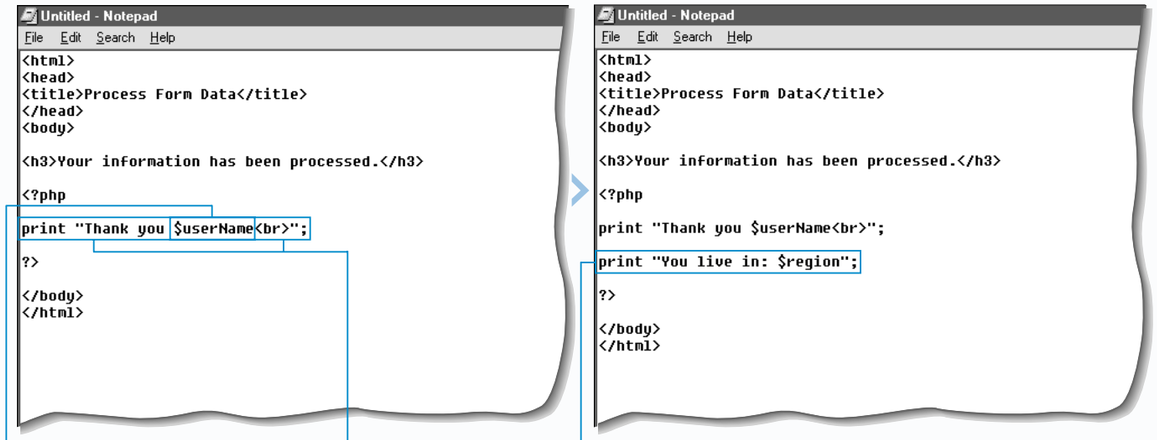
TYPE THIS:

```
if(empty($userName))
{
      print "You did not enter a user name.";
}
```

RESULT:

You did not enter a user name.

PROCESS DATA FROM A FORM

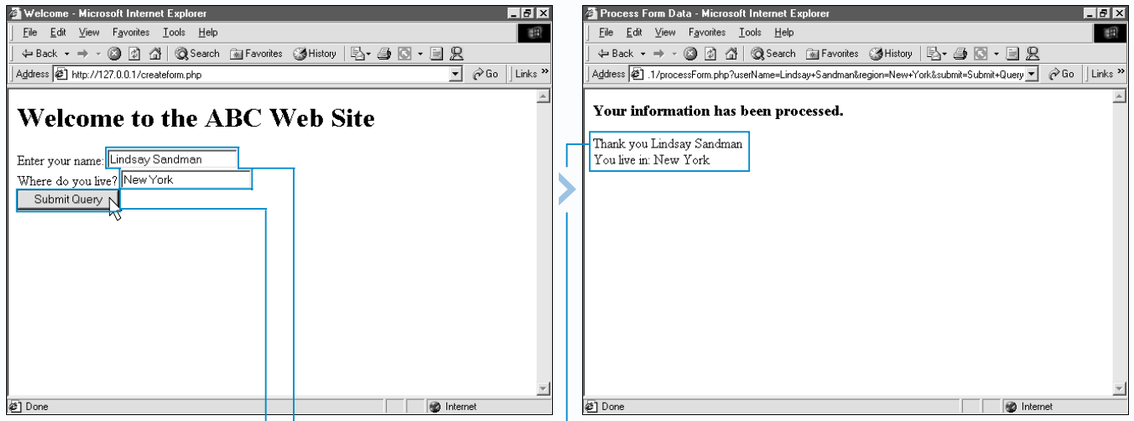


SET UP THE PHP PAGE

1 Type **$** followed by the name of a form element you want to access.

2 Type the code that uses the data from the form element.

3 Repeat steps 1 and 2 for each form element you want to access.



PROCESS FORM DATA

1 In a Web browser, display the Web page containing the form you want to process.

2 Enter data into the form.

3 Click the submit button to pass the data in the form to the PHP page that will process the data.

■ The Web browser displays the result of processing data from a form.

PROCESS MULTIPLE FORM SELECTIONS

Most elements in a form are used to submit a single item of information, such as a name or telephone number, to a page that processes the information. Some form elements, however, allow users to choose more than one option at once. For example, you can create a form that allows a user to select one or more items from a list.

The elements most often used to pass multiple selections in a form are the drop-down list and the check box element. For more information about form elements, see page 138.

When creating a form that can pass multiple selections for one element to a PHP page, you must indicate that the PHP page should process the data as an array of values, rather than as a single variable. To specify that the data should be passed as an array, you include a pair of brackets `[]` after the name of the element in the form, such as <select name="products[]">.

In the PHP page that processes the form data, you can access all the values selected for an element by using a `for` loop. For information about creating a `for` loop, see page 54. To access the values in the `for` loop, you prefix the name of the element with a dollar sign (`$`), just as you would access a single item of information from an element. Using the `count` function to determine the number of values passed by the element allows you to specify when the `for` loop should end.

After creating and saving the PHP page, you should review the code for the Web page that contains the form to verify that the `action` attribute displays the correct filename and location for the PHP page.

Extra

When you use the check box element to pass multiple selections in a form, you must include a pair of brackets after the name of the element for each option to ensure that PHP will store the selected options in an array.

Example:

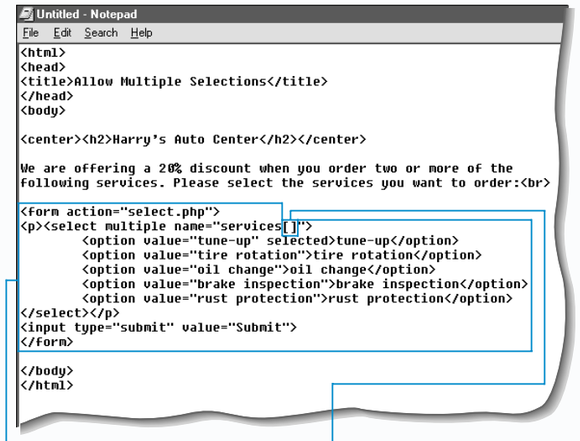
```
<form action="select.php">
<input type="checkbox" name="services[]"
      value="tune-up" checked>tune-up<br>
<input type="checkbox" name="services[]"
      value="tire rotation">tire rotation<br>
<input type="checkbox" name="services[]"
      value="oil change">oil change<br>
<input type="submit" value="Submit">
</form>
```

When processing data from a form, you should include code in the PHP page that checks the validity of data a user submits in the form. For example, if you want users to select at least two options from a list, you can add error-checking code that ensures two selections were made.

Example:

```
if (count($services) > 1)
{
      for ($x = 0; $x < count($services); $x++)
            print "$services[$x]<br>";
}
else
{
      print "Please select at least 2 items.";
}
```

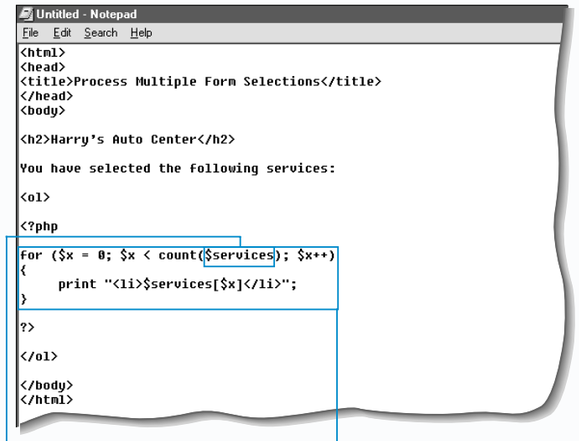
PROCESS MULTIPLE FORM SELECTIONS



CREATE THE FORM

1 Type the code that creates a form with an element that allows users to select multiple options.

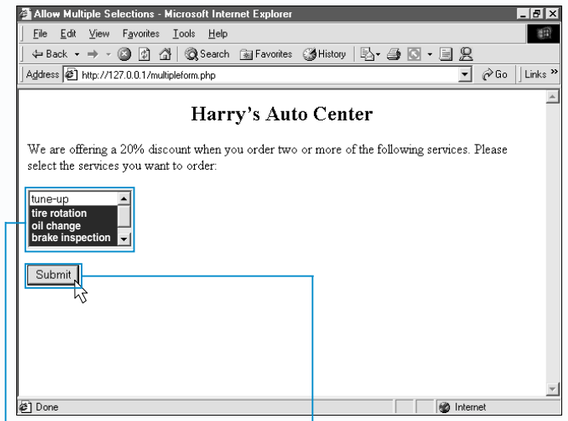
2 To indicate that the information from the element should be stored in an array, type **[]** after the name of the element.



SET UP THE PHP PAGE

1 In the PHP page, type **$** followed by the name of the form element that allows users to select multiple options.

2 Type the code for the `for` loop that will access each piece of data from the form element.



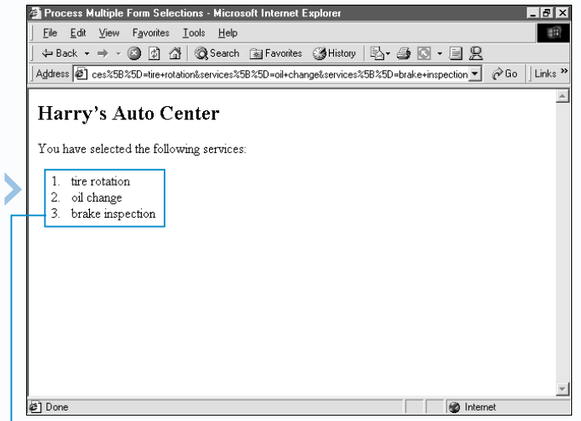
PROCESS MULTIPLE SELECTIONS

1 In a Web browser, display the Web page containing the form you want to process.

2 Select options in the form.

Note: To select multiple options from a drop-down list, hold down Ctrl as you click each option.

3 Click the submit button to pass the data in the form to the PHP page that will process the data.



■ The Web browser displays the result of processing multiple form selections.

CREATE A FORM TO UPLOAD A FILE

A form can be used to allow users to upload a file to a Web server. Allowing users to upload files is useful for collecting information that is best displayed in a separate file, such as a résumé or an order form. Users can send plain text files or binary files, such as images and compressed files. The files users send are usually stored on your Web server.

The `action` attribute of the `<form>` tag should specify the PHP page where the file is to be sent and processed. In the `<form>` tag, you should also use the `enctype` attribute with the `multipart/form-data` value to ensure that the files users send will transfer in the proper format. The form should also use the `post` method when transferring a file.

To indicate that a form element will be used to upload a file, the `type` attribute of the `<input>` tag should have a value of `file`. This form element will allow users to type the name of the file they wish to send or use a Browse button to select a file from their computer. The `name` attribute of the `<input>` tag should indicate the name that will be used to identify the file that a user sends.

As with others forms, a submit button must be added to the form, which allows users to start the process of transferring the file to the Web server.

You may also want to provide users with an alternate method of transferring files, such as using an FTP server. This is especially helpful in cases where a user's Web browser is not able to upload files.

After the file has been sent to a Web browser, a PHP page may be used to retrieve and process the uploaded file. For information about using a PHP page to process a file submitted using a form, see page 146.

Extra

You may incorporate other form elements into a form that allows users to upload files. This is useful for transferring other information, such as the sender's name and a file description, which you may want to use when processing the file that has been sent.

Example:

```
<form action="savefile.php" enctype="multipart/form-data" method="post">
Enter File Name: <input type="file" name="uploadedFile"><br>
Your Name: <input type="text" name="username" size="30"><br>
File Description: <input name="fileDetails" type="text" size="50"><p>
<input type="submit" value="Upload File">
</form>
```

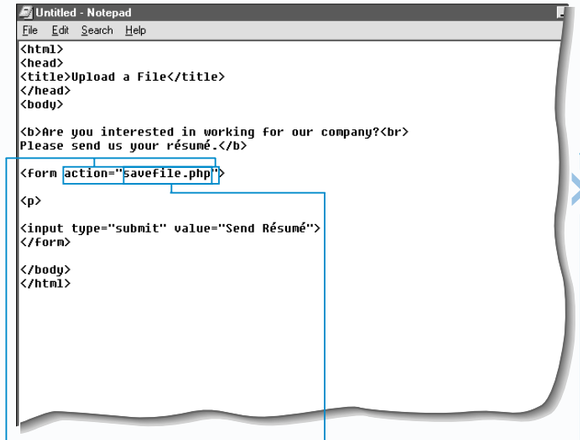
You may add a hidden field to your form to specify the maximum size of the file that may be sent to the Web server. The value `MAX_FILE_SIZE` is assigned to the `name` attribute, and the `value` attribute is used to specify the maximum size, in bytes. This hidden field must be placed before the file input field.

Example:

```
<form action="savefile.php" enctype="multipart/form-data" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="1000000">
Enter File Name: <input type="file" name="uploadedFile"><p>
<input type="submit" value="Upload File">
</form>
```

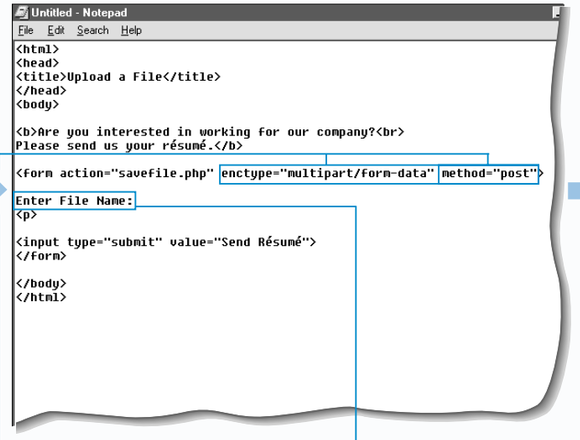
The ability to upload files using forms is not strictly a PHP feature. It is an HTML specification and can be used to upload files to Web servers using technologies other than PHP. For more detailed information about how files are transferred using forms, you may refer to the World Wide Web Consortium (W3C) Web site at www.w3.org/TR/device-upload.

CREATE A FORM TO UPLOAD A FILE



1 In the `<form>` tag, type **action=""**.

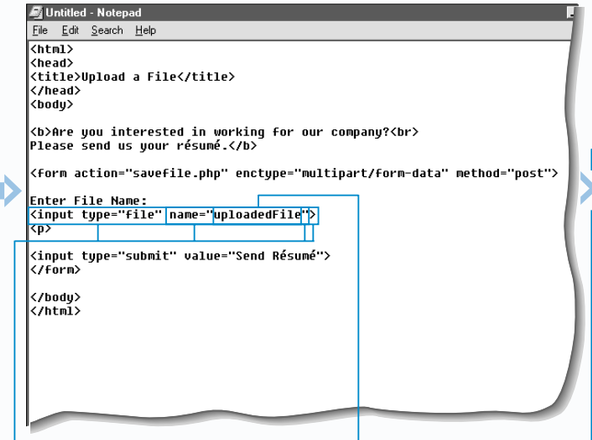
2 Between the quotation marks, type the name of the PHP page that will process the file sent by the form.



3 Type **enctype="multipart/form-data"** to ensure that the files users send will transfer in the proper format.

4 Type **method="post"** to specify the method used to transfer the file.

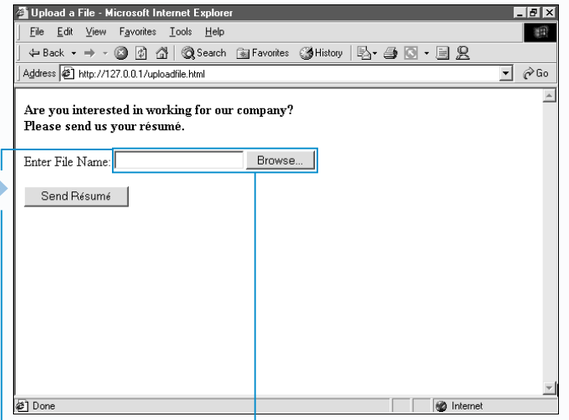
5 Between the `<form>` and `</form>` tags, type the text you want to appear beside the area that will allow users to send files.



6 To add the element that allows users to send a file, type **<input type="file">**.

7 In the `<input>` tag, type **name=""**.

8 Between the quotation marks, type the name that will be used to identify the file.



9 Save the page with the .html extension and display the HTML page in a Web browser.

- The Web browser displays an area that allows users to send a file.
- Users can click in the box and type the location and name of the file they want to send.
- Users can also click the Browse button to open a dialog box that will help them locate the file they want to send.

PROCESS AN UPLOADED FILE

A PHP page can be used to retrieve and process a file that was uploaded to a Web server by a user.

When a file is uploaded to a Web server using a PHP page, it is first temporarily stored using a filename assigned by PHP. This temporary file will be deleted after the PHP page has finished processing. You can use the `copy` function to save a permanent copy of the uploaded file. You need to pass two arguments to the `copy` function—the temporary filename and the filename to be used to save the file, which can be the original name of the file. After the file has been permanently stored, you can access the file as you would access any other file on the Web server.

To determine the temporary filename of the uploaded file, you must access the $HTTP_POST_FILES array, which is an associative array that stores information about the uploaded file. When accessing information from the $HTTP_POST_FILES array, you need to specify the name used to identify the file. This is the same name that was specified in the `name` attribute of the `<input>` tag in the form used to send the file. You must also specify the index name of the information you want to access. You use the `tmp_name` index name to access the temporary filename assigned to the uploaded file.

You can use other index names to access additional pieces of information about the file from the $HTTP_POST_FILES array. For example, use the `name` index name to determine the original filename of the uploaded file. The `size` index name returns the file size, in bytes. To determine the file type, you specify the `type` index name. Some Web browsers may not allow you to determine the file type.

Extra

The php.ini file contains configuration settings that allow you to change the temporary directory used for storing uploaded files and the maximum size of files that can be uploaded. You can also disable the ability to upload files. Open the php.ini file on the Web server and look for the following section of code.

```
;;;;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;;;;
file_uploads         = On      ; Whether to allow HTTP file uploads
upload_tmp_dir       = c:\temp ; temporary directory for HTTP uploaded files
upload_max_filesize  = 2M      ; Maximum allowed size for uploaded files
```

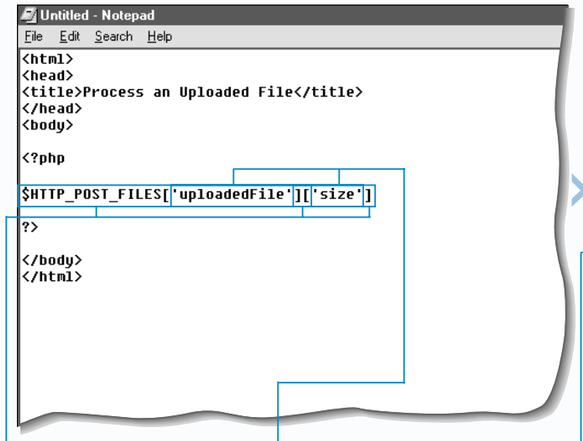
Before you can process uploaded files using a PHP page, you must make sure that file upload has been enabled and the temporary directory specified in the php.ini file exists on the Web server.

When an uploaded file is permanently saved on a Web server, by default, the file is saved in the same directory where the PHP page is located. You may want to save the file in a different directory by appending the directory path to the name of the file. To avoid errors, you should make sure that the specified directory exists on the Web server before attempting to save the uploaded file.

Example:

```
$tempFile = $HTTP_POST_FILES['uploadedFile']['tmp_name'];
$destination = "c:\\uploads\files\\" .
     $HTTP_POST_FILES['uploadedFile']['name'];
copy($tempFile, $destination);
```

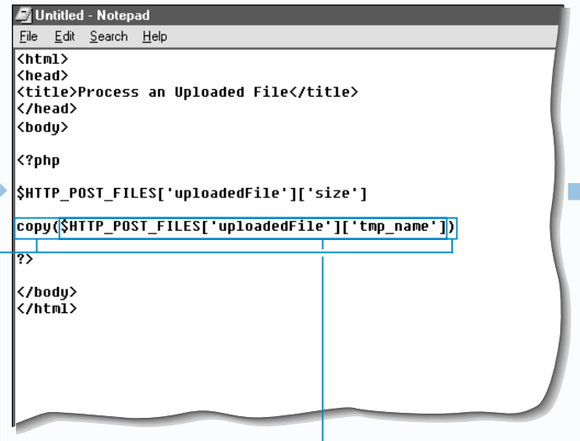
PROCESS AN UPLOADED FILE



1 To access information about a file uploaded to the Web server, type **$HTTP_POST_FILES[][]**.

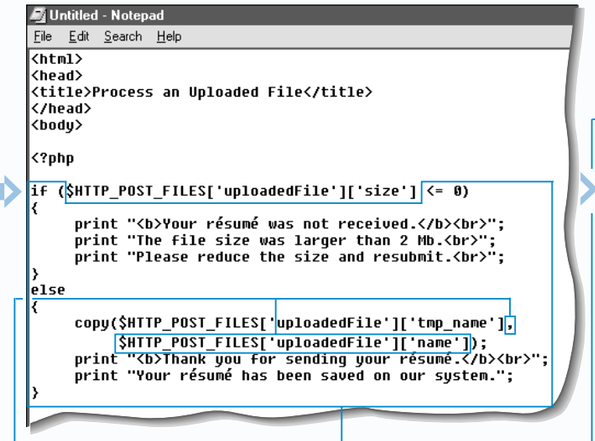
2 Between the first set of brackets, type the name used to identify the uploaded file enclosed in single quotation marks.

3 Between the second set of brackets, type the index name of the information you want to access enclosed in single quotation marks.



4 To save a permanent copy of the uploaded file, type **copy()**.

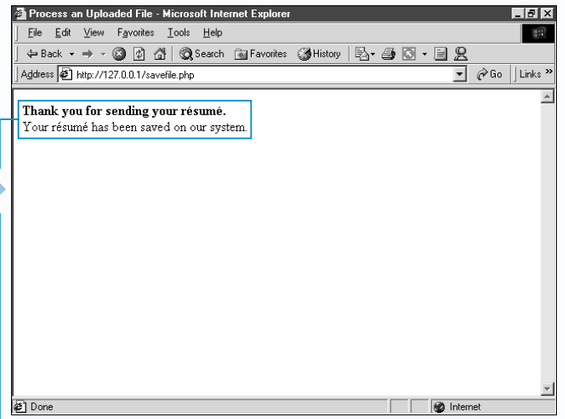
5 Between the parentheses, repeat steps 1 to 3, except type the **tmp_name** index name, enclosed in single quotation marks, in the second set of brackets to access the temporary name of the file.



6 To use the original filename to store the copy of the uploaded file, type a comma and then repeat steps 1 to 3, except type the **name** index name, enclosed in single quotation marks, in the second set of brackets.

7 Type the code that uses the accessed information about the uploaded file.

8 Save the page with the .php extension.



■ After a user uploads a file to the Web server using a form, the PHP page will be able to process the file and display the result of accessing information about the uploaded file.

■ The file is also permanently saved on the Web server.