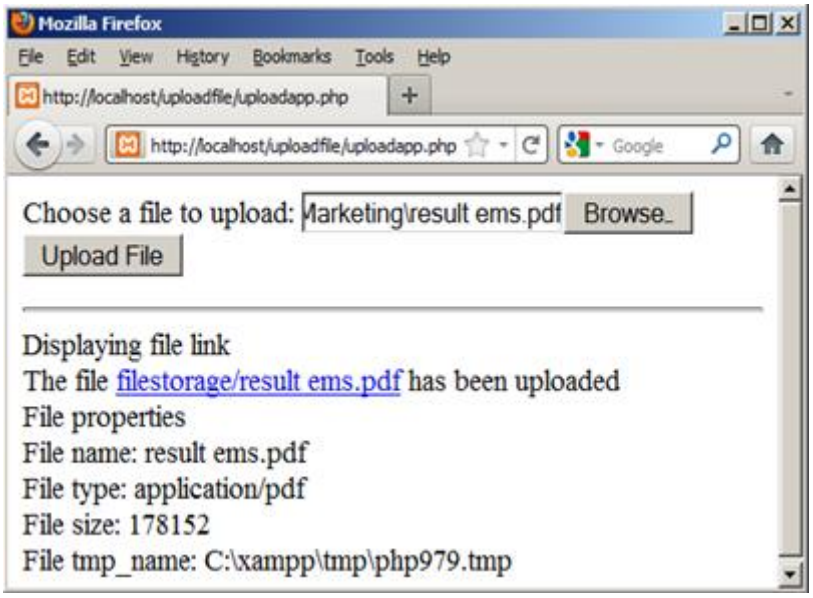
File Upload

Upload file to the server
A very useful aspect of PHP is its ability to manage file uploads to your server. Allowing users to upload a file to your server opens a whole can of worms, so please be careful when enabling file uploads.

EXERCISE 1: UPLOADING A FILE WITHOUT RESTRICTION

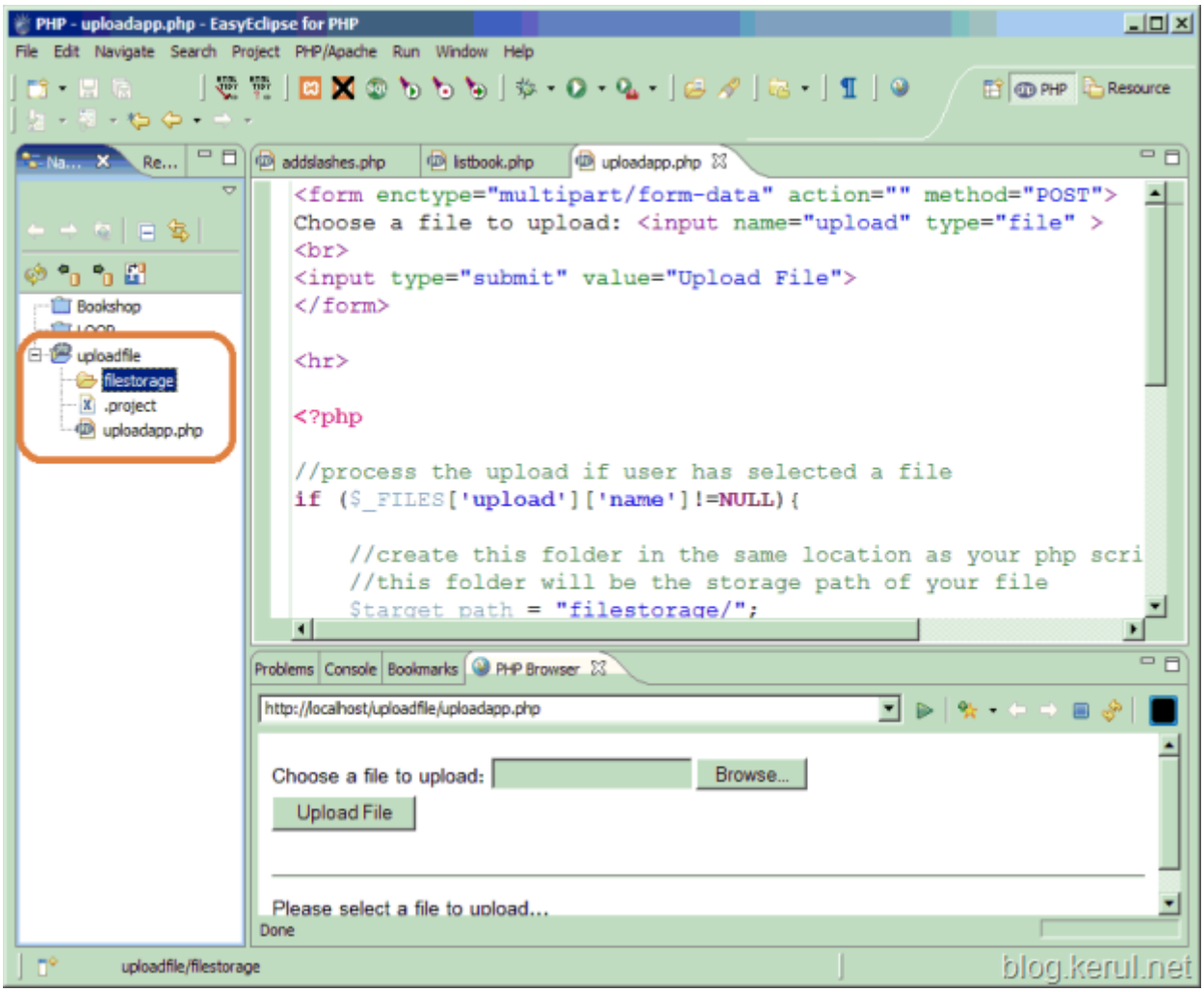
WARNING: This is a very simple file uploading exercise, intentionally created for training purposes to expose students the concept of file uploading processes. DO NOT use this in production, because it has no security feature…

What it does: This application will have the capability to upload any selected file into the web server.
Output: This application will look like this in Firefox;



Upload selected file into the web server

For the purpose of this exercise, you need to create another folder in the project folder. This folder will serve as the storage for whatever files the user uploaded.



Create new folder in the project folder

The PHP code: It is a single file PHP script.

```
<form enctype="multipart/form-data" action="" method="POST">
Choose a file to upload: <input name="upload" type="file" >
<br>
<input type="submit" value="Upload File">
</form>
<hr>

<?php
//process the upload if user has selected a file
if ($_FILES['upload']['name']!=NULL){

    //create this folder in the same location as your php script
    //this folder will be the storage path of your file
    $target_path = "filestorage/";
    $target_path = $target_path . $_FILES['upload']['name'];

    if(move_uploaded_file($_FILES['upload']['tmp_name'], $target_path)) {
        //Display file link
        echo "Displaying file link<br>";
        echo "The file <a href='$target_path'>$target_path</a> " .
             "has been uploaded<br>";
        echo "File properties <br>";
        echo "File name: ". $_FILES['upload']['name']."<br>";
        echo "File type: ". $_FILES['upload']['type']."<br>";
        echo "File size: ". $_FILES['upload']['size']."<br>";
        echo "File tmp_name: ". $_FILES['upload']['tmp_name']."<br>";
    }
    else{
        //uploading error
        echo "There was an error uploading the file, " .
             "please try again!<br>";
        echo "File error code: ". $_FILES['upload']['error'];
    }
}
else{//no file selected
    echo "Please select a file to upload...";
}

?>
```

Explanation: elaboration of what's happening in the PHP code.

The form elements;

- **`enctype="multipart/form-data`**" - Necessary for our to-be-created PHP file to function properly (read more). It is used for submitting forms that contain files, non-ASCII data, and binary data.
- **`method="POST"`** - Informs the browser that we want to send information to the server using POST. It won't work if you set it to GET.
- **`input name="upload"`** - upload is how we will access the file in our PHP script.

The server processes;

When the PHP script is executed, the uploaded file exists in a temporary storage area on the server. If the file is not moved to a different location it will be destroyed! To save the file we need to make use of the **$_FILES** associative array.
The **$_FILES** array is where PHP stores all the information about files. There are two elements of this array that we will need to understand for this example.

- **upload** - *upload* is the reference we assigned in our HTML form. We will need this to tell the **$_FILES** array which file we want to play around with.

- **$_FILES['upload']['name']** - name contains the original path of the user uploaded file.

- **$_FILES['upload']['tmp_name']** - *tmp_name* contains the path to the temporary file that resides on the server. The file should exist on the server in a temporary directory with a temporary name.

- **if ($_FILES['upload']['name']!=NULL)** – means if there is no file selected by the user.

- **$target_path** – is the full path of the file saved in the server. By right it is supposed to be in *filestorage/your_file.ext*

- **move_uploaded_file($_FILES['upload']['tmp_name'],$target_path)** – a PHP function to ensure that the file designated by filename **($_FILES['upload']['tmp_name'])** is a valid upload file (meaning that it was uploaded via PHP's HTTP POST upload mechanism). If the file is valid, it will be moved to the filename given by destination (in **$target_path**). (read more –> http://php.net/move_uploaded_file)

**Credit to TiZag.com. Modified from the tutorial in http://www.tizag.com/phpT/fileupload.php
.

EXERCISE 2: UPLOADING A FILE WITH FILE TYPE RESTRICTIONS

This exercise will impose file type restrictions. It will only receive image files with *PNG, GIF or JPG format*. Before uploading the file to the server, check the file type. Display the image in the output.

```
<form enctype="multipart/form-data" action="" method="POST">
Choose an image to upload (PNG, JPG, GIF): <input name="upload" type="file"
>
<br>
<input type="submit" value="Upload Image">
</form>
<hr>

<?php
//process the upload if user has selected a file
if ($_FILES['upload']['name']!=NULL){

    //create this folder in the same location as your php script
    //this folder will be the storage path of your file
    $target_path = "filestorage/";
    $target_path = $target_path . $_FILES['upload']['name'];
    if ($_FILES['upload']['type']== 'image/x-png'
        || $_FILES['upload']['type']== 'image/png'
        || $_FILES['upload']['type']== 'image/jpeg'
        || $_FILES['upload']['type']== 'image/pjpeg'
        || $_FILES['upload']['type']== 'image/gif'){
    if(move_uploaded_file($_FILES['upload']['tmp_name'], $target_path)) {
        //Display file link
        echo "Displaying file link<br>";
        echo "The file <a href='$target_path'>$target_path</a> " .
             "has been uploaded<br>";
        echo "File properties <br>";
        echo "File name: ". $_FILES['upload']['name']."<br>";
        echo "File type: ". $_FILES['upload']['type']."<br>";
        echo "File size: ". $_FILES['upload']['size']."<br>";
        echo "File tmp_name: ". $_FILES['upload']['tmp_name']."<br>";
    }
    else{
        //uploading error
        echo "There was an error uploading the file, " .
             "please try again!<br>";
        echo "File error code: ". $_FILES['upload']['error'];
    }
    }
}
else{//no file selected
    echo "Please select a file to upload...";
}

?>
```

EXERCISE 3: UPLOADING A FILE WITH FILE TYPE AND SIZE RESTRICTIONS

Create a file uploading application that only receives a *PDF file* with *file size smaller* than *100KB*. Display the link of the transferred file.

```
<form enctype="multipart/form-data" action="" method="POST">
Choose a PDF file less than 100KB: <input name="upload" type="file" >
<br>
<input type="submit" value="Upload PDF">
</form>

<hr>

<?php

//process the upload if user has selected a file
if ($_FILES['upload']['name']!=NULL){

    //create this folder in the same location as your php script
    //this folder will be the storage path of your file
    $target_path = "filestorage/";

    if ($_FILES["upload"]["type"] == "application/pdf"
            && $_FILES["upload"]["size"] <=102400){
        $target_path = $target_path . $_FILES['upload']['name'];
        if(move_uploaded_file($_FILES['upload']['tmp_name'], $target_path))
{
            //Display file link
            echo "Displaying file link<br>";
            echo "The file <a href='$target_path'>$target_path</a> has been
uploaded";
        }
        else{
            //uploading error
            echo "There was an error uploading the file, please try
again!<br>";
            echo "File type: ".$_FILES["upload"]["type"]."<br>";
            echo "File size: ".$_FILES["upload"]["size"]."<br>";
        }
    }
    else {
        echo "Only PDF file <100KB is allowed. Try again<br>";
        echo "File type: ".$_FILES["upload"]["type"]."<br>";
        echo "File size: ".$_FILES["upload"]["size"]."<br>";
    }
}
else{//no file selected
    echo "Please select an image file to upload...";
}
?>
```