Dynamic Web Development

Lecture 10 – Passing Variables

Passing Variables Through a URL

This is what is happening when you see something like this:

`http://www.mydomain.com/story.php?id=12345`

or

`http://www.mydomain.com/story.php?id=12345&lang=en`

The browser will be using an HTTP GET request to retrieve the page from the server - but it is also sending some variable values TO the server, encoded in the URL.

These can be accessed by some code on the page.

Value hard-coded into link

```
<html>
  <head>
    <title>Find my Favourite Movie!</title>
  </head>
<body>
  <a href="moviesite.php?favmovie=Stripes">
    Click here to see information about my favourite movie.
  </a>
</body>
</html>
```

Passing Variables From One Page to Another

You may want someone to type in a user name on one page, and display it on another.

This means that you need a set of variables that are global to all of the pages in your website.

If the php configuration setting **`register_globals`** is set to ‘on’:

You can:

- Set up a variable on one page,
- Pass its value through a URL
- Use this value to 'poison' a variable on a different webpage.

Since PHP version 4.2, **`register_globals`** is ‘off’ by default, as it is considered a security risk.

Example: Misuse with register_globals = on

auth.php

```
<?php
// define $authorized = true only if user is authenticated

if ( authenticated_user() )
  {
  $authorized = 1;
  }

// Because we didn't first initialize $authorized as false, this might be
// defined through register_globals, like from GET auth.php?authorized=1
// So, anyone can be seen as authenticated!

if ( $authorized == 1 )
  {
  include "/highly/sensitive/data.php";
  }
?>
```


PHP: Using Register Globals - Manual - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://php.net/manual/en/security.globals.php
Go Links
php
downloads | documentation | faq | getting help | mailing lists | licenses | wiki | reporting bugs | php.net sites | links | conferences | my php.net
search for in the function list
PHP Manual
Security
Introduction
General considerations
Installed as CGI binary
Installed as an Apache module
Filesystem Security
Database Security
Error Reporting
Using Register Globals
User Submitted Data
Magic Quotes
Hiding PHP
Keeping Current
«Error Reporting
User Submitted Data»
view this page in Bulgarian
Last updated: Fri, 06 Nov 2009
Using Register Globals
Warning
This feature has been DEPRECATED as of PHP 5.3.0 and REMOVED as of PHP 6.0.0. Relying on this feature is highly discouraged.
Perhaps the most controversial change in PHP is when the default value for the PHP directive register_globals went from ON to OFF in PHP » 4.2.0. Reliance on this directive was quite common and many people didn't even know it existed and assumed it's just how PHP works. This page will explain how one can write insecure code with this directive but keep in mind that the directive itself isn't insecure but rather it's the misuse of it.
When on, register_globals will inject your scripts with all sorts of variables, like request variables from HTML forms. This coupled with the fact that PHP doesn't require variable initialization means writing insecure code is that much easier. It was a difficult decision, but the PHP community decided to disable this directive by default. When on, people use variables yet really don't know for sure where they come from and can only assume. Internal variables that are defined in the script itself get mixed up with request data sent by users and disabling register_globals changes this. Let's demonstrate with an example misuse of register_globals:
Example #1 Example misuse with register_globals = on
Internet
Start
3 Internet Ex...
HOTKEY KEYBO...
2 Windows Ex...
Microsoft Power...
13:03

Superglobal Arrays

PHP provides built-in arrays that are always available in all scopes.

Each array contains a set of variables that can be accessed from anywhere in your PHP code.

$_SERVER	information such as headers, paths, and script locations.
$_ENV	information about the current execution environment
$_GET	information passed via the GET method
$_POST	information passed via the POST method
$_FILES	information about a recently uploaded file
$_COOKIE	information about a cookie
$_SESSION	information about the current session

Which one you use depends on the method you are using to pass values between webpages.

Value Retrieved by using a Superglobal

```
<?php
  $filmvar = $_GET['favmovie'];
?>
<html>
  <head>
    <title>My Movie Site</title>
  </head>
  <body>
    <p>My favourite movie is:</p>
<?php
    echo $filmvar;
?>
    <br />
  </body>
</html>
```

It is always better to explicitly use a superglobal array to pass values between pages than just rely on **`register_globals`** being on.

urlencode( )

If the value that you are trying to pass through the URL contains spaces, ampersands or other similar characters, use this function:

```
$favmovie = urlencode("Life of Brian");
```

Instead of

```
$favmovie = "Stripes";
```

Value encoded using urlencode( )

```
<html>
  <head>
    <title>Find my Favourite Movie!</title>
  </head>
  <body>
<?php
    $myfavmovie = urlencode("Life of Brian");

    echo "<a href='moviesite.php?favmovie=$myfavmovie'>";
    echo "Click here to see information about my favourite movie!";
    echo "</a>";
?>
  </body>
</html>
```

Note the use of single quotes to enclose the href URL, instead of double quotes.

Disadvantages

Everyone can see the values of the variables – not very secure.

The user can change the variable value in the URL, which means that they could access something that they shouldn't.

A saved URL may have older variables embedded in it which are no longer valid.

Passing Variables Through Forms

One of the main uses for variables is to collect information from the user and use it to control what should appear on other webpages.

Forms consist of:

1: Opening tag <form>

- This must include two attributes - an action and a method.
- The action gives a URL to another page which will receive the data included in the form.
- The method (GET or POST) tells the form how the data will be carried. POST is more secure.

Forms 2

2: Content of the form, including input fields.

An input field must include two attributes - a type and a name.

Common types:

- Text
- Checkbox
- Radio button
- Select (drop down box)
- Password

The name of the input field will be the variable name which your PHP program can use to retrieve the value.

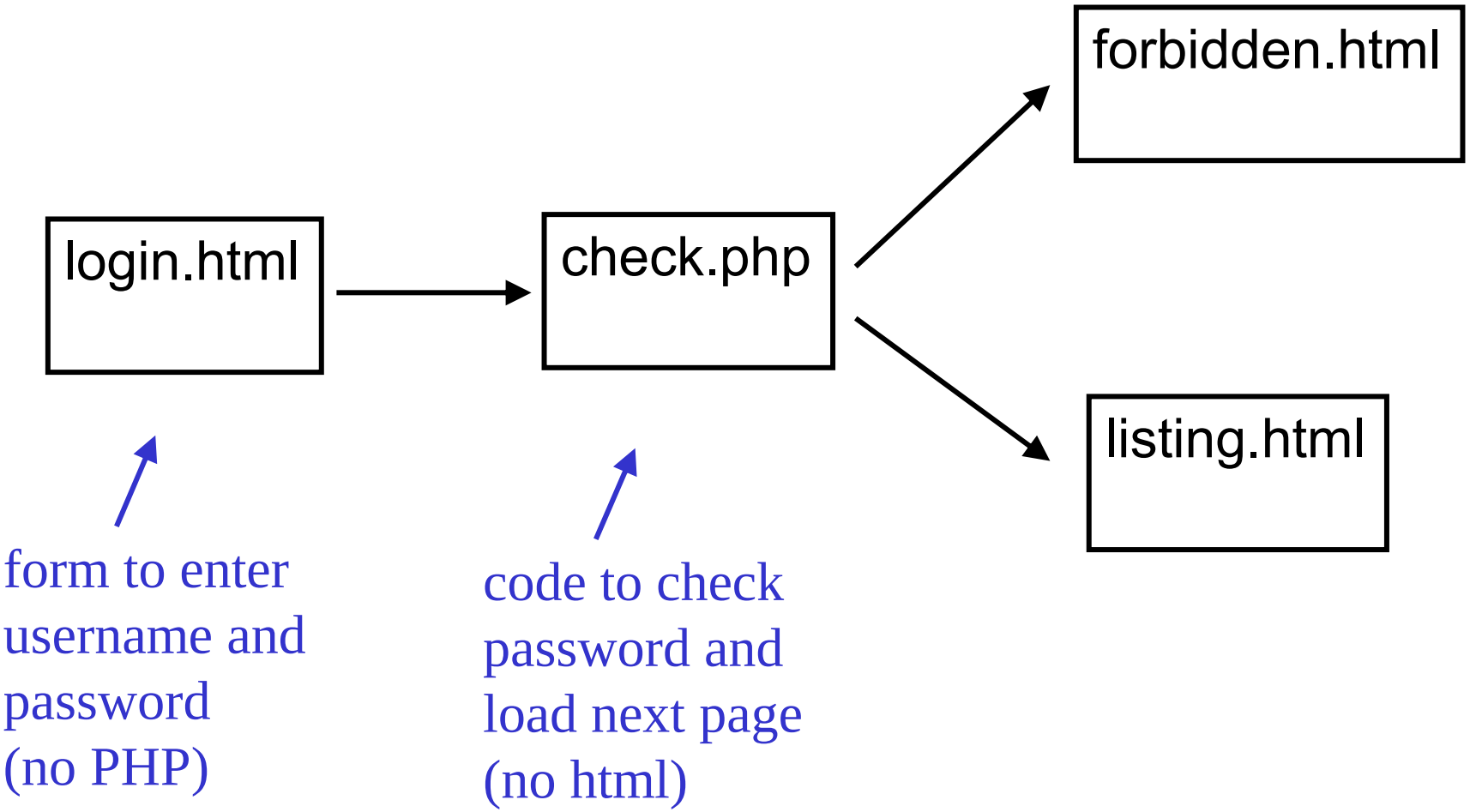
Forms 3

3: Action Buttons

Usually ‘Submit’ or ‘Reset’. It is possible to have user defined ones as well.

4: Closing tag </form>

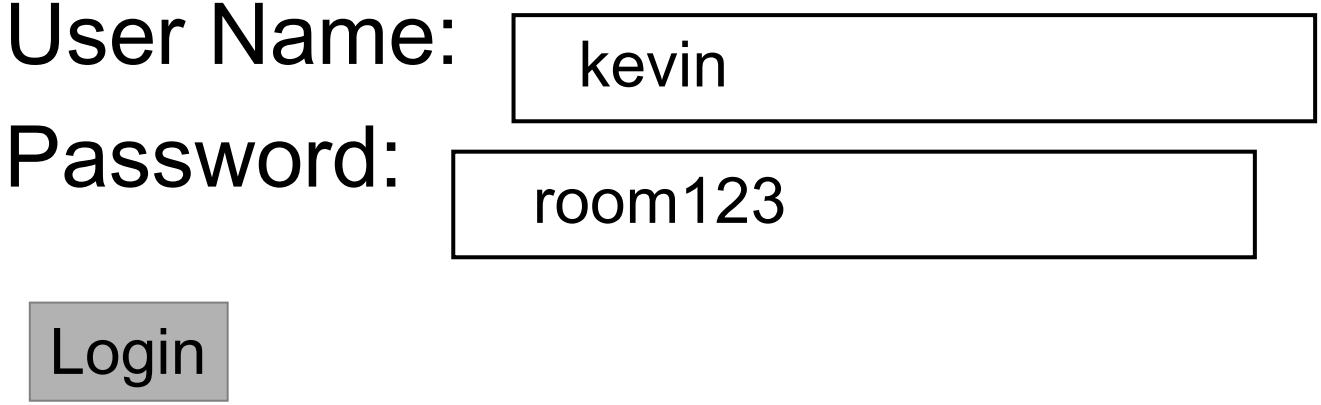
Page Map


forbidden.html
login.html
check.php
listing.html
form to enter username and password (no PHP)
code to check password and load next page (no html)

login.html

```
<html>
  <head></head>
  <body>
    <!-- Form with two fields and a submit button -->
    <form name="logscreen" action="check.php" method="get">
      User Name:
      <input type="text" name="user"></input>
      <br />
      Password:
      <input type="text" name="pass"></input>
      <br />
      <input type="submit" value="Login"></input>
    </form>
  </body>
</html>
```

This will generate an HTML form



The submit button will load the following page into the browser, and send the contents of the two text boxes:

check.php? user=kevin & pass=room123

check.php

```php
<?php

// Get the values from the URL and copy them into
variables
$uname = $_GET['user'];
$pword = $_GET['pass'];

// Compare the variables with the valid name and
password and redirect
// user to the appropriate page

if (($uname == "kevin") && ($pword == "room123"))
    header( "Location: listing.html");
else
    header( "Location: forbidden.html");

?>
```

Use of POST

The use of GET is not very secure.

The POST method works in the same way, except that the variables are not encoded in the URL.

They are transmitted to the server in a different part of the GET request packet which is hidden from the user.

They can be accessed by using the $_POST superglobal array.

login.html

```
<html>
  <head></head>
  <body>
    <!-- Form with two fields and a submit button -->
    <form name="logscreen" action="check.php" method="post">
      User Name:
      <input type="text" name="user"></input>
      <br />
      Password:
      <input type="text" name="pass"></input>
      <br />
      <input type="submit" value="Login"></input>
    </form>
  </body>
</html>
```

check.php

```
<?php

// Get the values from the request packet and copy
them into variables
$uname = $_POST[ 'user' ];
$pword = $_POST[ 'pass' ];

// Compare the variables with the valid name and
password and redirect
// user to the appropriate page

if (($uname == "kevin") && ($pword == "room123"))
    header( "Location: listing.html");
else
    header( "Location: forbidden.html");

?>
```