Introduction to Advanced Server Side Issues - PHP

Advanced Server Side Issues

Form validation

How do we validate form data? The way least we should do is pass all variables through PHP's html special chars() function. This function will replace HTML class like < and > to their HTML version alt, &gt, etc.

Example

```
<?php
$name = htmlspecialchars($_POST['name']);
$email = htmlspecialchars($_POST['email']);
?>
```

This is much safer now and prevents possible attackers from exploiting out code by injecting HTML or JavaScript code.

If we know exactly what kind of data to expect, we can make further steps to ensure the user has entered what we want. Let's do two more things.

1. Strip unnecessary characters from the data.
2. If quotes are escaped with a slash (\), remove that.

Instead of writing the same code over and over again, we can create a function that will do all the checking for us.

```
<?php
$name = check_input($_POST['name']);
$email = check_input($_POST['email']);
?>
<html?
</html>
<?php
function check_input($data){
$data = trim($data);
$data = striplashes($data);
$data = htmlspecialchars($data);
Return $data;
}
?>
```

Note the check_input function at bottom. What it does is takes the data passed to the function, strips, unwanted characters.

Required and Optional Field

In the previous examples, the scripts worked fine if you didn't enter any data. However, many times you want to make input fields required.

Let's edit **check_input** function from previous.

```
<?php
function check_input($data, $problem = "error"){
$data = trim($data)
$data = stripslashes($data)
$data = htmlspecialcharacters($data);
if($problem. && strlen($data) == 0){
die($problem)
}
Return $data
}
```

Validate Email Address

There is no way to be 100% sure an email is actually working unless we send an email there. What we usually do is check if the email syntax is valid. Here is a simple way to check if data entered into field named "email" is an email address without any unnecessary complication and fancy regular expressions.

```
$email = htmlspecialchars(%_POST[‘email’]);
if(!preg_match(“/([\w\-] + \@[\w\-] +)/”, $email)
{
die(“Email address not valid”);
}
```

Validate URL address

If we have an input field named "website", we can check for a valid URL like,

```
$url = htmlspecialchars($_POST[‘website’]);
if(!preg_match(“/^(http?:W + ]\w\-] + \ [\w\-] +) /; $(m))
{
die(“URL address not valid”);
}
```

Digit 0-9 only

```
if(preg_match(“/\D/”, $age))
{
die(“Please enter numbers only for age.”);
}
```

Letters a-z and A-Z only

```
if(preg_match(“/[^„a-z A-Z]/”, $text))
{
die(“Please enter letters only!”];
}
```

Anything but whitespace

```
if(Preg_match(“/\S/”, $text))
{
die(“don’t enter any space”);
}
```

Things to remember

1. Form data is validated by passing all variables through PHP's htmlspecial chars() function. This function will replace HTML class like < and > to their HTML version alt, &gt, etc.
2. Input fields to enter data might be required.

Database Connectivity in PHP

Database connectivity in PHP

In php, we use database to store our details.

When we create a new database, we must specify the first arguments to the mysqli object(servername, Username and password).

Create Connection

```
<?php
$servername = “localhost”;
$username = “username”;
$password = “password”;
// create connection
$con = mysql_connect($servername, $username, “”);
// check connection
if($con)
{
die(“connection failed:” .mysql_error());
}
?>
```

Create Database

```
$sql = “CREATE DATABASE myDB”;
if(mysql_query($con, $sql)){
Echo “Database Created”;
}
else{
Echo “Error”.mysql_error($con);
}
```

Create Table

The CREATE TABLE statement is used to create a table in mysql.

We will create a table named “MyGuests” with 5 columns: “id”, “firstname”, “lastname”. “email” nd “reg_date”.

```
CREATE TABLE MyGuests(
id INT(6) UNSIGNED AUTO_INCREMENT.PRIMARY KEY,
firstname VARCHAR(20) NOT NULL,
lasrtname VARCHAR(20) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)
```

In PHP:

```
// create connection first. Then,
// sql to create table
$sql = “CREATE TABLE MyGuests(
id INT(6) UNSIGNED AUTO_INCREMENT.PRIMARY KEY,
firstname VARCHAR(20) NOT NULL,
lastname VARCHAR(20) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP)”;
if(mysql.query($con,sql))
{
```

```
echo “Table Created”;
}
else
echo “Error”;
```

Insert Data into Database

After a database and a table have been created. We can start adding data in them.

Rules:

- The sql query must be quoted in php.
- String values inside the sql query must be quoted.
- Numeric values must not be quoted.

Example

//after database connection

```
$sql = “INSERT INTO MyGuests(firstname, lastname, email)
VALUES(‘John’, ‘Doe’, ‘john@ex.com’);
if(mysql.query($con, $sql){
echo “New record Created”;
}
else{
echo “Error”.mysql_error($con);
}
Mysql_close($con);
We can insert multiple records in mysql like,
$sql = “INSERT INTO MyGuests(firstname, lastname, email)
VALUES(‘John’, ‘Doe’, ‘john@ex.com’);
$sql = “INSERT INTO MyGuests(firstname, lastname, email)
VALUES(‘Mary’, ‘Moe’, ‘mary@ex.com’);
if(mysql_multi_query($con, $sql)){
Echo “New records added”;
}
else{
echo “Error”;
}
Mysql_close($con);
```

Select Data from Database

The SELECT statement is used to select data from one or more tables.

```
SELECT column_name(s) FROM table_name
```

Or we can use the * character to select ALL columns from a table:

SELECT * FROM table_name

Example:

```
<?php
//create connection
$con = mysql_connect(‘localhost”, “user”, “”);
if($con){
die(“connection failed”;
}
$sql = “SELECT id, firstname FROM MyGuests;
$result = mysql_query($con, $sql);
```

```
if(mysql_num_rows($result)>0){
while($row = mysql_fetch_assoc($result)){
echo “id:”.$row[“id”].”_name:”.$rwo[“firstname”].””.”<br>”;
}
}
else{
echo “0 results”;
}
mysql_close($con);
?>
```

Delete Data from Database

The DELETE statement is used to delete records from a table

DELETE FROM table_name WHERE some_column = some_value

The WHERE clause specifies which record or records that should be deleted.

Example:

```
//sql to delete a record
$sql = “DELETE FROM MyGuests WHERE id = 3;
if(mysql_query($con,$sql)){
echo “Record deleted successfully”;
}
else{
echo “Error”;
}
mysql_close($con);
?>
```

Update Data in Database

The UPDATE statement is used to update existing records in a table:

UPDATE table_name

SET column1 = value1, column2 = value2

WHERE some_column = some_value

Note: the WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated.

Example:

```
<?php
//create connection
$con = mysql_connect(“localhost”, “user”, “”);
//check connection
if($con){
die(“connection failed”);
}
$sql = UPDATE MyGuests SET lastname = ‘Doe’ WHERE id = “2”;
if(mysql_query($con, $sql)){
echo “Record updated successfully”;
}
else{
echo “Error”;
}
```

```
    mysql_close($con);
    ?>
```

Authentication in Database

We can authenticate the user in database using GRANT query.

Syntax:

Grant all privileges on *.* → all tables

to username@localhost identified by “password” with grant option

example:

grant create, insert, update, select

on Mydb.*

to ram@localhost

identified by “abc”;

with grant option;

To remove the authentication from the user, we use REVOKE query

Syntax:

revoke all privileges

on *.*

from username@localhost;

Example:

revoke create, delete

on Mydb.*

from ram@localhost

Note: RESTRIC – if authorization is given to only one user otherwise CASCADE.

Authentication by IP address

In some rare instances, we may wish to limit access to a certain page or pages to certain IP addresses. It may be because we have an internal network that has a publicly viewable website. We may wish to have certain pages be viewable only by certain machines on internal network.

Or it may be that we have our own remote website, and we wish to restrict access to a certain page or pages so that only we, from a static IP address, can access those pages.

These methods work by determining the IP addresses of the user trying to view the pages and checking it against a set value. If the IP address matches, then the page can be viewed.

This method could be fooled by IP spoofing or other hacks, and should not be used to protect sensitive information, such as credit card numbers, proprietary information, or the like/ it is simple method that can protect pages from the majority of users surfing the net.

When using this script, we should be aware that our computer may have more than one IP address and that the one your browser is reporting to the script may not be the same one we are attempting to verify against. This is especially true when the browser and web server resides on the same machine, as we will probably encounter when testing the script.

For example, computers have a default local IP 127.0.0.1, otherwise known as “localhost”. When we start up the web server and type in “http://localhost”, our browser is telling the script that we are at IP address of 127.0.0.1. If we type in the IP address as 192.168.0.1, our browser reports that we are at 192.168.0.1, but the script only accepts the IP of 127.0.0.1. If we run into problems, try echoing the value of $REMOTE_ADDR to screen to see which IP.

Example: If authorization PHP

```
<?php
$accept = array(“127”, “0”, “0”, “1”);
$remote = explode(“.” $REMOTE_ADDR);
$match = 1;
for($i=0; $i<sizeof($accept); $i++){
if($remote[$i] != $accept[$i]){
$match = 0;
}}
if($match){
echo”<h2>Access Granted</h2>”;
}
else{
echo”<h2>Access Forbidden</h2>”;
}
?>
```

Things to remember

1. The function to connect to MySQL is called mysql_connect.
2. The function used to perform queries is named - mysql_query().
3. We don't actually need to close the connection, PHP will automatically close the connection when the script ends.
4. We can authenticate the user in database using GRANT query.

XML Parsers

XML Parsers

It is software that reads and parses XML. It passes data to the invoking application. The application does something useful with the data. There are two types of parser:

1. SAX
2. DOM

SAX (Simple API for XML)

- SAX is an ad-hoc (but very popular) standard.
- SAS was developed by David Megginson and it is open source.
- SAX is used when document is large and device is memory constrained.
- When should we use something else? If we need to modify the document. SAX does not remember previous events unless we write explicit code to do so.
- Which languages are supported? Java, Perl, C++, Python
- SAX works through callbacks: we call the parser, it calls methods that we supply.

DOM (Document Object Model)

The DOM is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of a document. The DOM is a W3C standard.

The DOM is separated into 3 different parts:

- Core DOM
- XML DOM
- HTML DOM

The DOM defines the objects and properties of all document elements and the method to access them.

HTML DOM

The HTML DOM is a standard object model for HTML. It is a standard programming interface to HTML. HTML DOM defines the objects and properties of all HTML elements and the method to access them.

In other words, the HTML DOM is a standard for how to get, change, add or delete HTML.

DOM Nodes

The DOM says,

- The entire document is a document node.
- Every HTML element is an element node.
- Text in HTML elements are test nodes.
- HTML attributes is an attribute node.
- Comments are comment node.

Example:

```
<html>
<head><title> DOM </title></head>
<body>
<h1> lesson 1 </h1>
```

<p> Hello World </P>

</body>

</html>

The root node in the HTML above is <html>. All other nodes in the document are contained within <html>. The <html> node has two child nodes: <head> and <body>. The <head> node holds a <title> node. The <body> node holds a <h1> and <p> node. Text of an element node is stored in a text node.

A common error in DOM processing is to expect an element node to contain text. However, the test of an element node is stored in a text node.

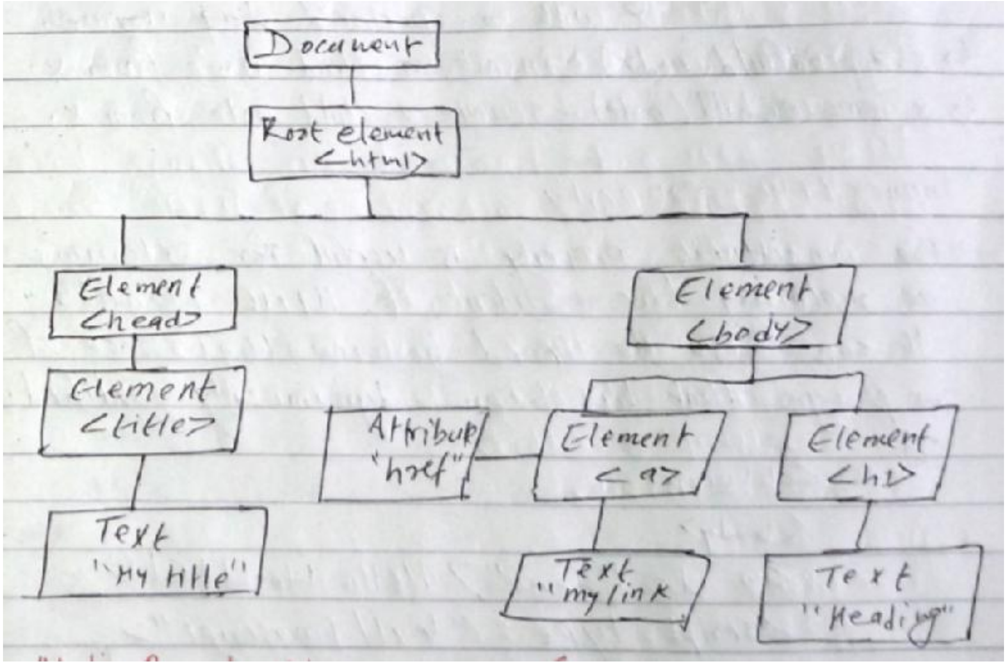
In this example: <title> DOM tutorial </title>, the element node <title> holds a text mode with the value "DOM tutorial".

"DOM tutorial" is not the value of the <title> element.

However, in the HTML DOM, the value of the test node can be accessed by the inner HTML property.

HTML DOM node Tree

The html DOM views an HTML document as a tree structure, the tree structure is called a node tree. All nodes can be accessed through the tree. Their contents can be modified or deleted and new element can be created. The node tree below shows the set of nodes and the connections between them. The tree starts at a root node and braches out to the text nodes at the lowest level of the tree.



Node Parents, Children and Siblings

- The top node is called the root.
- Every node has exactly one parent node (except root).
- A node can have any number of children.
- A leaf is a node with no children.
- Siblings are nodes with the same parent.

HTML DOM properties

- innerHTML – the text value of X.
- nodeName – the name of x.

- nodeValue – the value of x.
- parentNode – the parent nodes of X.
- childNodes – the child nodes of X.
- attributes – the attributes nodes of X.

HTML DOM Methods

- getElementById(id) – get element with special id.
- getElementByTagName(name) – get all elements with a specified tag name.
- appendChild(node) – insert a child node from X
- removeChild(node) – remove a child node from X

InnerHTML Property

The innerHTML property is useful for returning or replacing the content of HTML elements. It can also be used to view the source of a page that has been dynamically modified.

example:

```
<html>
<body>
<p id ="demo">Hello World!</p>
<script  type = "text/JavaScript">
document.getElementById("demo").innerHTML = "Paragraph changed!";
</script>
</body>
</html>
```

Example:

```
<html>

<body>

< p id = “P1”> Hello World! </p>

< script type = “text/JavaScript”>

(xt = document.getElemenById(“P1”).childNodes[0].nodeValue;

</script>

</body>

</html>
```

Accessing Nodes

We can access a node in three ways

1. Using getElementById() method

node.getElementById(“id”);

2. Using getElementByTagName() method

node.getElementByTagName(“tagname”);

3. Navigating the Node tree, using node relationships

```
<html>
<body>
<p>Example list:</p>
```

```
<ul id="myList"><li>Coffee</li><li>Tea</li></ul>

<button onclick="myFunction()">Try it</button>

<p><strong>Note:</strong> Whitespace inside elements is considered
as text, and text is considered as nodes.</p>

<p>If you add whitespace before the first LI element, the result
will be "undefined".</p>
<p id="demo"></p>
<script>
function myFunction() {
    var list =
document.getElementById("myList").firstChild.innerHTML;
    document.getElementById("demo").innerHTML = list;
}
</script>
</body>
</html>



<html>
<body>
<p>Click the button to change the text of this paragraph.</p>
<p>This is also a paragraph.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction() {
    document.getElementsByTagName("P")[0].innerHTML = "Hello
World!";
}
</script>
</body>
</html>
```

Changes an HTML element

HTML DOM and JavaScript can change their inner element and attributes of HTML elements.

Example:

```
<html>
<body>
<script type = "text/JavaScript">
 document.body.style.backgroundColor = "red";
 //document.body.bgcolor="red";
</script>
</body>
</html>
```

Example:

```
<html>
<body>
<p id="P1"> Hello! </p>
<script type = "text/JavaScript">
document.getElementById("P1").innerHTML= "World!";
</script>
```

```
</script>
</body>
</html>


<html>
<body>
<h1 id="id01">My First Page</h1>
<p id="id02"></p>

<script>
document.getElementById("id02").innerHTML =
document.getElementById("id01").innerHTML;
</script>

</body>
</html>
```

XML DOM

The XML DOM is a standard mode for XML and standard programming interface for XML. It is a platform and language independent. The XML DOM defines the objects and properties of all XML elements and the methods is access them.

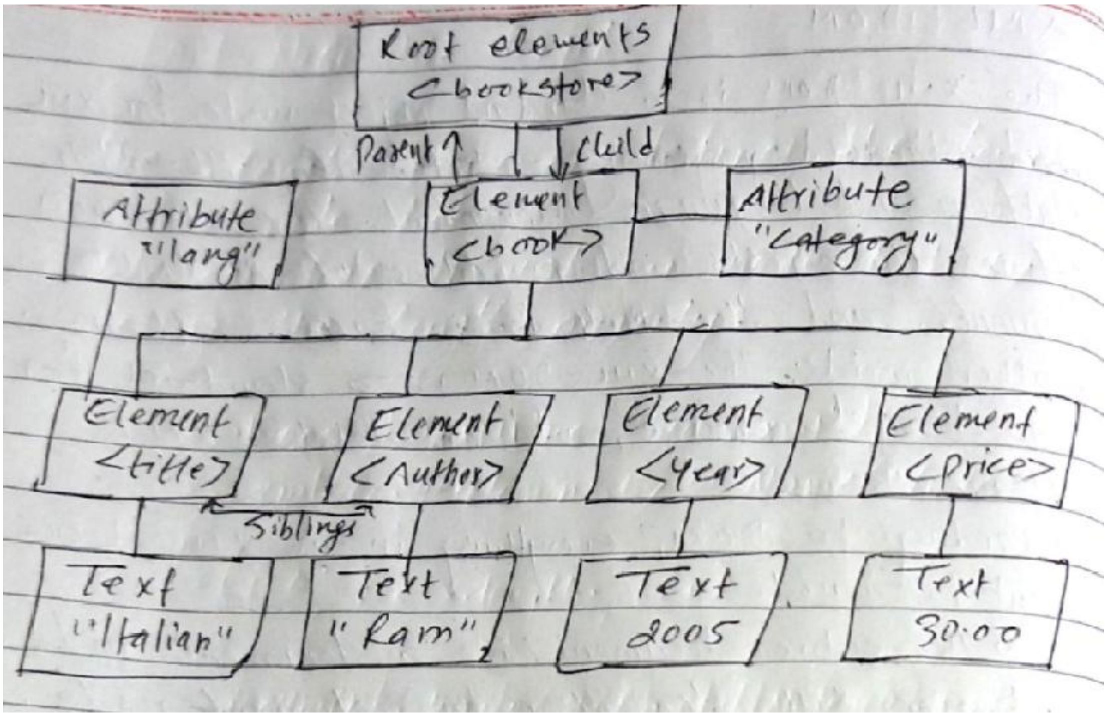
In other words, the XML DOM is a standard for how to get, change, add or delete XML elements.

According to DOM, everything in an XML document is a node.

- The entire document is a document node.
- Every HTML element is an element node.
- Text in HTML elements are test nodes.
- Every attribute is an attribute node.
- Comments are comment nodes.

XML DOM Node Tree

The XML DOM views an HTML document as a tree structure, the tree structure is called a node tree. All nodes can be accessed through the tree. Their contents can be modified or deleted and new element can be created. The node tree below shows the set of nodes and the connections between them. The tree starts at a root node and branches out to the text nodes at the lowest level of the tree.



HTML DOM properties

- nodeName – the name of x.
- nodeValue – the value of x.
- parentNode – the parent nodes of X.
- childNodes – the child nodes of X.
- attributes – the attributes nodes of X.

HTML DOM Methods

- getElementByTagName(name) – get all elements with a specified tag name.
- appendChild(node) – insert a child node from X
- removeChild(node) – remove a child node from X

In the list above, X is a node object.
getElementsByTagName() Method

getElementsByTagName() returns all elements with a specified tag name.

syntax:

node.getElementsByTagName("tagname");

example:

The following example returns all <title> elemets under the X element.

X.getElementsByTagName("title");

The example only returns <title> elements under the X node. To remove all <title> elements,

xmlDoc.getElementsByTagName("title");

DOM Node List

The getElementsByTagName() method returns a node list. A node list is an array of nodes. The following code loads "books.xml" into XMLDoc using loadXMLDoc() and stores a list of <title> nodes in variable X.

xmlDoc = loadXMLDoc("books.xml");

X = xmlDoc.getElementsByTagName("title");

The <title> elements in X can be accessed by index number. To access a third <title>, we can write

T = X[2]

DOM Node list Length

The length property defines the length of a node list. We can loop through a node list by using the length property.

```
xmlDoc=loadXMLDoc("books.xml");
x=xmlDoc.getElementsByTagName("title");
for (i=0;i<x.length;i++)
  {
  document.write(x[i].childNodes[0].nodeValue);
  document.write("<br />");
  }
```

Node Properties

In the XML DOM, each node is an object. Objects hace methods and properties that can be accessed and manipulated by JavaScript.

Three important node properties are:

- nodeName
- nodeValue
- nodeType

Get the value of an Element

The following code retrives the text node value of the first <title> element.

xmlDoc = loadXMLDoc("books.xml");

X = xmlDoc.getElementsByTagName("title") [0]. childNodes[0];

Txt = X.nodeValue;

Then,

Result:

Txt = "Italian"

Example Explained:

1. load "books.xml" into xmlDoc using LoadXMLDoc()
2. get text node od the first <title> element node
3. set the txt variable to be the value of the text node.

Things to remember

1. SAX, Simple API for XML, is an ad-hoc standard and works through callbacks: we call the parser, it calls methods that we supply.
2. The DOM is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of a document. The DOM is a W3C standard.
3. The DOM says, entire document, every HTML element, text in HTML element, HTML attributes, comments all are nodes.
4. We can access a node in three ways: 1. Using getElementById() method 2. Using getElementByTagName() method 3.Navigating the Node tree, using node relationships
5. The XML DOM is a standard mode for XML and standard programming interface for XML that defines the objects and properties of all XML elements and the methods is access them.