使用PHP实现防止sql注入功能

# 开发环境

1、环境搭建：Windows 7+Apache 2.4.18+MySQL 5.7.11+PHP 7.1.0。

2、文本编辑器：Sublime 3。

# 二、主要技术

本案例主要在PHP中分别使用PDO的quote()方法和prepare()预处理语句来实现防止sql注入的功能。

所谓sql注入，就是通过把sql命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的sql命令。具体来说，它是利用现有应用程序，将恶意的sql命令注入到后台数据库引擎执行的能力，它可以通过在Web表单中输入恶意的sql语句得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行sql语句。

# 三、效果图展示

实现效果图，如图1所示。

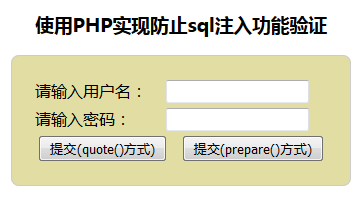


图1 效果图（操作界面）

# 四、具体步骤

## 4.1、数据库设计

### （1）创建数据表

创建一个名为“phpDemo”的数据库，并创建表“087”，表结构设计如下所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** |
| uname | varchar(50) | NO | PRI | *NULL* |
| upwd | varchar(50) | NO |  | *NULL* |

### （2）向表中添加数据

## 4.2、界面设计

### （1）表单设计

<h3><center>使用PHP实现防止sql注入功能验证</center></h3>

<form action="index.php" method="post">

<table>

<tr>

<td>请输入用户名：</td>

<td><input type="text" name="uname"></td>

</tr>

<tr>

<td>请输入密码：</td>

<td><input type="password" name="upwd"></td>

</tr>

<tr>

<td colspan="2"><center><input type="submit"

name="sub1" value="提交(quote()方式)">&nbsp;&nbsp;

<input type="submit" name="sub2" value="提交(prepare()方

式)"></center></td>

</tr>

</table>

</form>

### （2）样式设计（CSS样式）

<style>

body{

cellpadding:0;

cellspacing:0;

margin:0;

padding:0;

background-color: #ffffff;

}

table{

width:340px;

line-height:1.5em;

margin:auto;

padding:20px;

border:1px solid #CCCCCC;

border-radius:8px;

background-color: #E2DDA2;

}

#result{

width:340px;

line-height:2em;

margin:10px auto;

text-align:center;

background-color: #CCCCCC;

}

</style>

## 4.3、获取用户输入

$username=$\_POST['uname']; // 用户名

$userpwd=$\_POST['upwd']; // 密码

## 4.4、判断用户输入是否为空

if(empty($username)||empty($userpwd)){

exit();

}

## 4.5、quote()方式防止sql注入

if(isset($\_POST['sub1'])){

// 连接数据库

$dbms='mysql'; //数据库类型

$host='10.10.3.33'; //数据库主机名

$dbName='phpDemo'; //使用的数据库

$user='root'; //数据库连接用户名

$pass='51xueweb'; //对应的密码

$dsn="$dbms:host=$host;dbname=$dbName";

try {

$dbh = new PDO($dsn, $user, $pass); //初始化一个PDO

对象

$dbh->query('SET NAMES utf8'); // 设置字符集

} catch (PDOException $e) {

die ("Error!: " . $e->getMessage() . "<br/>");

}

$uname=$dbh->quote($username);

$upwd=$dbh->quote($userpwd);

$sql="select \* from `087` where uname=$uname and

upwd=$upwd";

$result=$dbh->query($sql);

if($result->rowCount()){

echo "<div id='result'>登录成功！</div>";

}else{

echo "<div id='result'>登录失败！</div>";

}

}

## 4.6、prepare()预处理语句防止sql注入

if(isset($\_POST['sub2'])){

try{

$pdo=new PDO('mysql:host=10.10.3.33;

dbname=phpDemo','root','51xueweb'); // 连接数据库

$sql="select \* from `087` where uname=:username

and upwd=:userpwd"; // sql查询语句

$stmt=$pdo->prepare($sql); // 预处理语句

$stmt->execute(array(":username"=>$username,

":userpwd"=>$userpwd));

// 判断是否登录成功

if($stmt->rowCount()>0){

echo "<div id='result'>登录成功！</div>";

}else{

echo "<div id='result'>登录失败！</div>";

}

}catch(PDOException $e){

echo $e->getMessage(); // 输出错误信息

}

}

# 五、网页测试

①开启Apache和MySQL服务器后，打开浏览器，在浏览器地址栏中输入文件地址，例如：“localhost:8888/PHParticle/087/Codes/ index.php”，可以看到程序运行结果如图2所示。

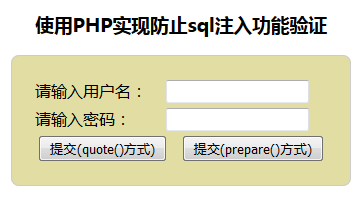


图2 案例结果图（操作界面）

②在两个文本框内分别输入用户名“demo”和密码“12345”，然后点击【提交(quote()方式)】按钮，可以看到程序运行结果如图3所示。

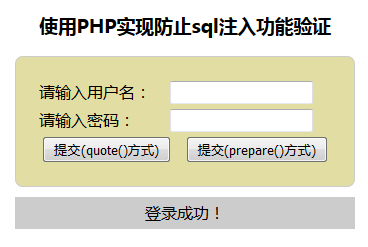


图3 案例结果图（quote()方式提交）

③模拟sql注入攻击：在两个文本框内分别输入用户名“’or 1=1 #”和密码“12345”，然后点击【提交(quote()方式)】按钮，可以看到程序运行结果如图4所示。

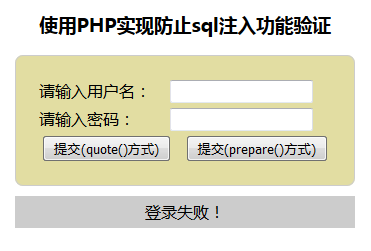


图4 案例结果图（quote()方式防止sql注入）

④在两个文本框内分别输入用户名“demo”和密码“12345”，然后点击【提交(prepare()方式)】按钮，可以看到程序运行结果如图5所示。

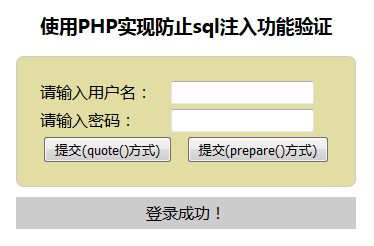


图5 案例结果图（prepare()方式提交）

⑤模拟sql注入攻击：在两个文本框内分别输入用户名“’or 1=1 #”和密码“12345”，然后点击【提交(prepare()方式)】按钮，可以看到程序运行结果如图6所示

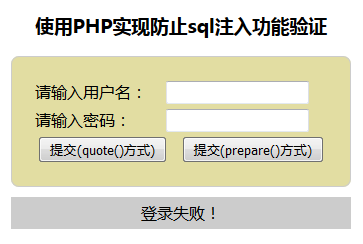


图6 案例结果图（prepare()方式防止sql注入）