PDO

pdo的基本使用

使用PDO访问MySQL数据库时，真正的real prepared statements 默认情况下是不使用的。为了解决这个问题，你必须禁用prepared statements的仿真效果。下面是使用PDO创建链接的例子：

复制代码 代码如下:

```
$dbh = new PDO('mysql:dbname=dbtest;host=127.0.0.1;charset=utf8', 'user', 'pass');
$dbh->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
```

setAttribute（）这一行是强制性的，它会告诉 PDO 禁用模拟预处理语句，并使用 real parepared statements 。这可以确保SQL语句和相应的值在传递到mysql服务器之前是不会被PHP解析的（禁止了所有可能的恶意SQL注入攻击）。虽然你可以配置文件中设置字符集的属性(charset=utf8)，但是需要格外注意的是，老版本的 PHP（ < 5.3.6）在DSN中是忽略字符参数的。

我们来看一段完整的代码使用实例：

复制代码 代码如下:

```
$dbh = new PDO("mysql:host=localhost; dbname=demo", "user", "pass");
$dbh->setAttribute(PDO::ATTR_EMULATE_PREPARES, false); //禁用prepared statements的仿真效果
$dbh->exec("set names 'utf8'");
$sql="select * from test where name = ? and password = ?";
$stmt = $dbh->prepare($sql);
$exeres = $stmt->execute(array($testname, $pass));
if ($exeres) {
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
print_r($row);
}
}
$dbh = null;
```

上面这段代码就可以防范sql注入。为什么呢？

当调用 prepare() 时，查询语句已经发送给了数据库服务器，此时只有占位符 ？发送过去，没有用户提交的数据；当调用到execute()时，用户提交过来的值才会传送给数据库，他们是分开传送的，两者独立的，SQL攻击者没有一点机会。

但是我们需要注意的是以下几种情况，PDO并不能帮助你防范SQL注入

1、你不能让占位符 ？代替一组值，如：

复制代码 代码如下:

```
SELECT * FROM blog WHERE userid IN ( ? );
```

2、你不能让占位符代替数据表名或列名，如：

复制代码 代码如下:

```
SELECT * FROM blog ORDER BY ?;
```

3、你不能让占位符 ？代替任何其他SQL语法，如：

复制代码 代码如下:

```
SELECT EXTRACT( ? FROM datetime_column) AS variable_datetime_element FROM blog;
```