Lab10

姓名：陆勇雍

学号：19302010034

Exercise 7：

首先，config.php 文件通过 define 定义了常量，分别为数据库主机名‘DBHOST’，数据库“DBNAME”，数据库连接的用户名“DBUSER”，密码“DBPASS”，数据库名称以及字符编码“DBCONNSTRING”。这样做的好处是如果数据库或者用户密码需要更换可以直接在 config.php 文件里更改这些常量，其他文件只需要调用这个 config.php，不用修改。

Lab10-exercise7-pdo.php 文件中：

```
$pdo = new PDO(DBCONNSTRING,DBUSER,DBPASS);
```

通过创建 PDO 的一个实例来连接数据库，

```
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

设置了抛出 exceptions 异常。

```
$sql = “select * from Artists order by LastName”;
```

这是一条 sql 语句，意思是从 artists 表中选取所有数据并按照 LastName，以字母顺序排列。

```
$result = $pdo->query($sql);
```

执行这条 sql 语句，并把结果赋值给$result 变量。

```
While ($row = $result->fetch()) {
echo $row[‘ArtistID’] . “ – “ . $row[‘LastName’] . “<br/>”;
}
```

把得到的结果数据一条条的以 ArtistId-LastName 的形式输出

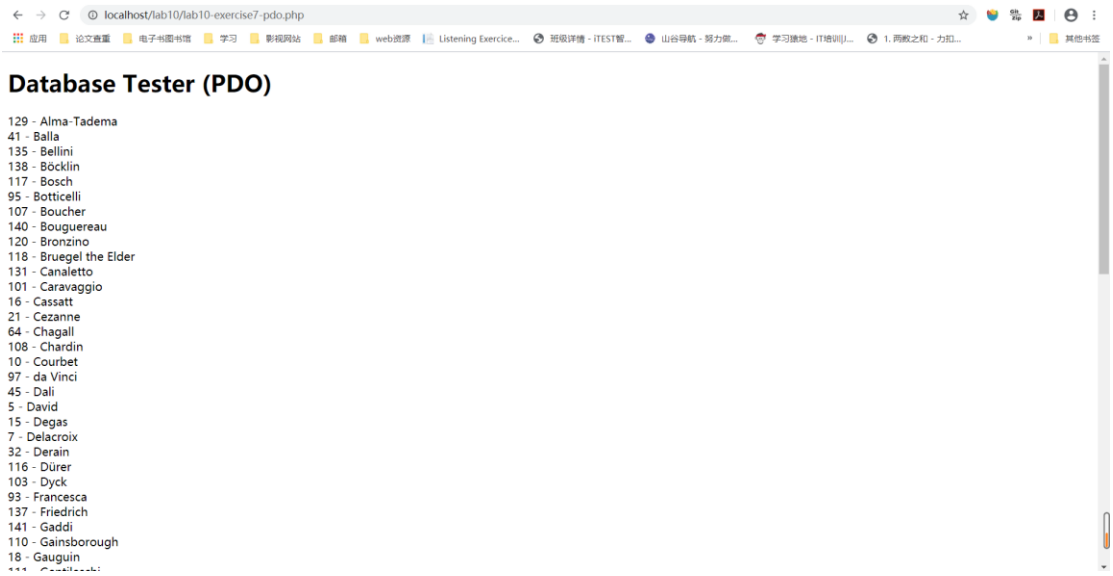
```
$pdo = null;
```

通过赋值 null 给$pdo 变量来关闭连接。

```
try{
}catch (PDOException $e) {
die( $e->getMessage() );
}
```

捕获异常。

截图如下：



lab10-exercise7-mysqli.php 文件中：

```
$connection = mysqli_connect(DBHOST, DBUSER, DBPASS, DBNAME);
```

通过 mysqli_connect()函数来连接数据库。参数分别为主机名，用户名，密码，使用的数据库。并返回一个代表到 MySQL 服务器的连接的对象。

```
if ( mysqli_connect_errno() ) {
  die( mysqli_connect_error() );
}
```

如果连接出现错误则返回错误描述的字符串，并且退出。

```
$sql = "select * from Genres order by GenreName";
```

这是一条 sql 语句，意思是从 genres 表中选取所有数据并按照 GenreName，以字母顺序排列。

```
if ($result = mysqli_query($connection, $sql)) {
        while($row = mysqli_fetch_assoc($result)) {
                echo '<option value="' . $row['GenreID'] . '">';
                echo $row['GenreName'];
                echo "</option>";
        }
        mysqli_free_result($result);
}
```

执行这条 sql 语句，并把查询到的结果集对象赋值给$result,
通过 mysqli_fetch_assoc 函数，循环遍历$result 中的数据，并把每一条分别对应输出为一条 html 语句：

```
<option value='xx'> Name</option>
```

即把这些数据综合生成了一个下拉列表。

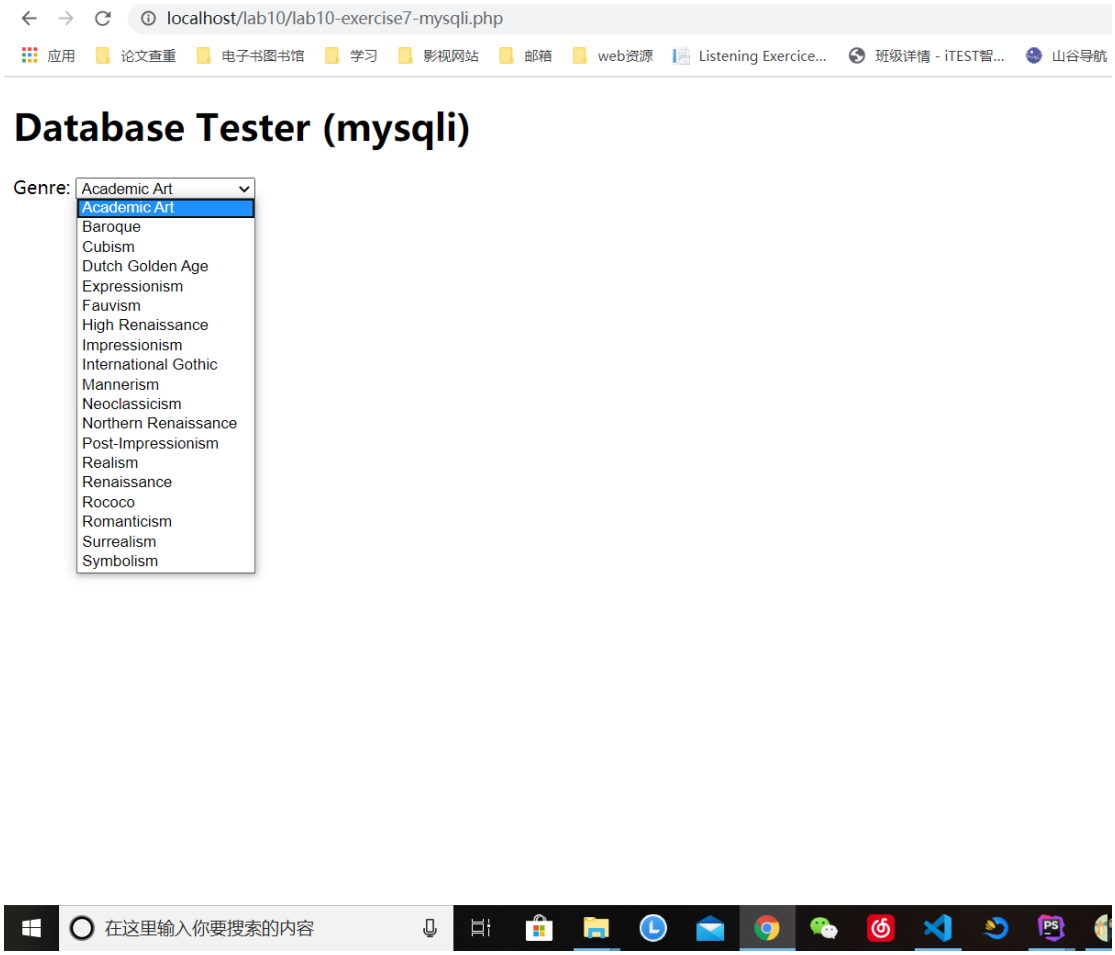
```
mysqli_free_result($result);
```

释放结果内存

```
mysqli_close($connection);
```

关闭数据库连接。

截图如下：



Exercise 8：

outputArtists()：

创建 PDO 实例对象连接 art 数据库，并通过

```
$sql = "select * from Artists order by LastName limit 0,30";
$result = $pdo->query($sql);
```

从 Artists 表中选取所有数据并按照 LastName，以字母顺序排列，并从索引值 0 开始，查询 30 条，将结果对象集赋值给$result，通过循环遍历每一条数据，为每一条数据对应生成一条 html 语句：

```
<a href="lab10-exercise8.php?id=ArtistID"  class="(active) item">LastName</a>
```

即为每条数据生成了一个连接,其中，类名 class 会进行判断，如果当前网址中有 id（如 http://localhost/lab10/lab10-exercise8.php?id=129，可以通过$_GET 变量得到这个 id）并且这个 id 和这条数据的 ArtistID 相等，则会添加 class 类，最后的效果相当于点击了这个链接，跳转后，该链接会高亮和其他未点击的链接区分开来。

OutputPaintings():

这个函数实现了链接跳转后根据 url 的 id 值呈现出相对应的图片。

先进行判断有无 id 值，即 http://localhost/lab10/lab10-exercise8.php 和 http://localhost/lab10/lab10-exercise8.php?id=129 的区别。通过创建 PDO 实例对象连接数据库，并从 Paintings 表中查找所有 ArtistId 等于 url 的 id 的数据，并通过循环遍历，为每一条数据调用 outputSinglePainting 函数，也即生成一串 html 语句来呈现图片。

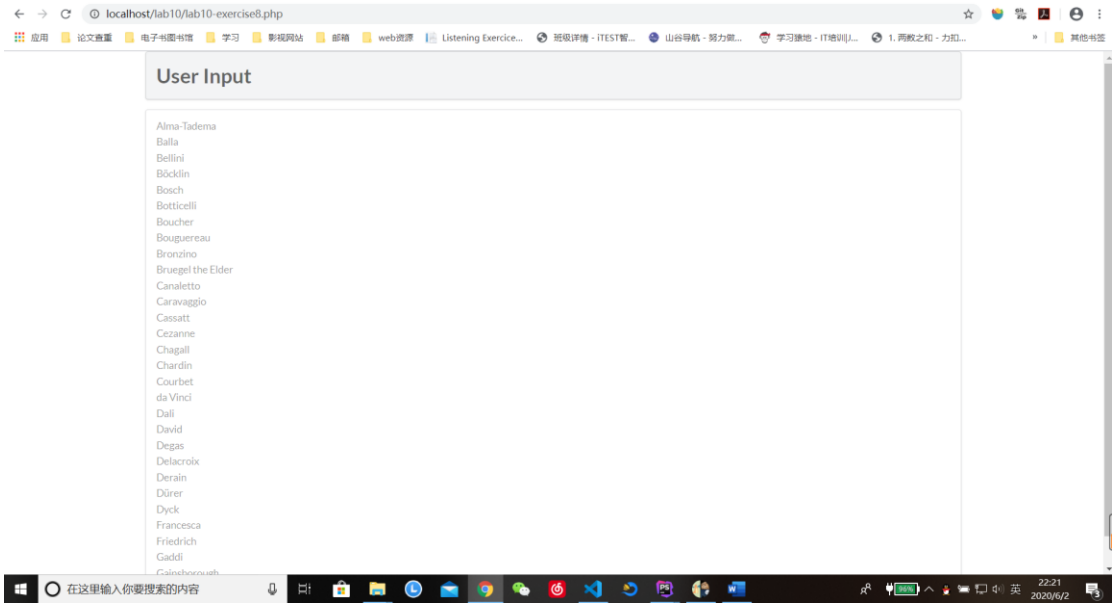
outputSinglePainting($row):

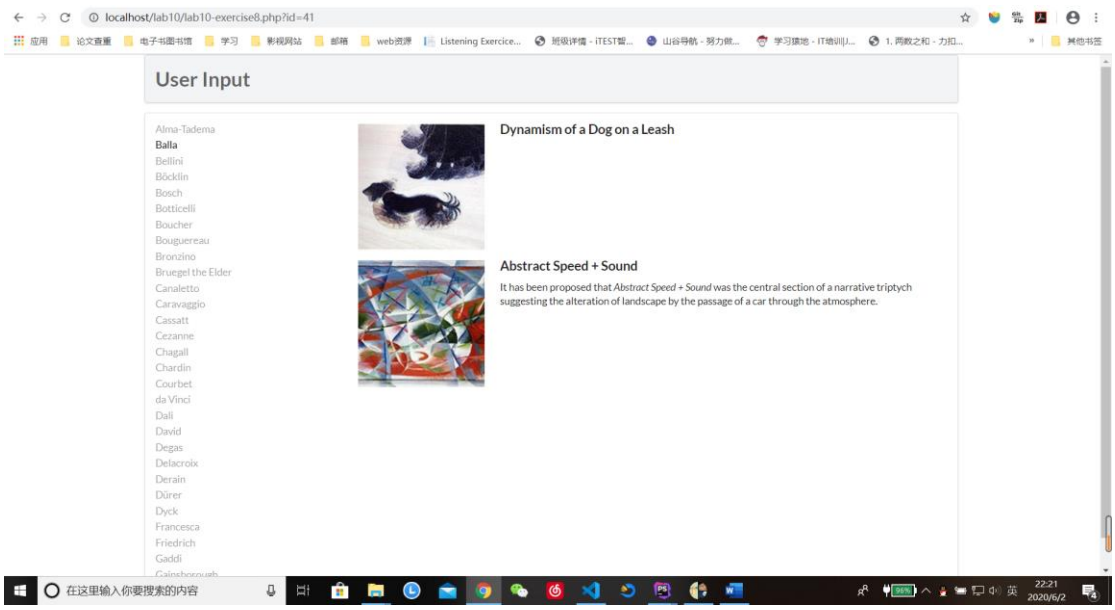
这个函数接受一个数组，生成一串 html 语句：

```
<div class='item'>
    <div class='image'>
        <img src=images/art/works/square-medium/xxx.jpg>
    </div>
    <div class='content'>
        <h4 class='header'> xx  </h4>
        <p class='description'> xx  </p>
    </div>
</div>
```

即生成一个图片，标题，描述。

截图如下：





Exercise 9：

执行 sql 语句的方式：

mysqli_query()函数
PDO 中执行 sql 语句：

1. int PDO::exec(string statement)

exec()方法返回执行 SQL 语句后受影响的行数，参数 satatement 是要执行的 SQL 语句，该方法返回执行 SQL 语句时受影响的行数，通常用于 INSERT，DELETE 和 UPDATE 语句中。

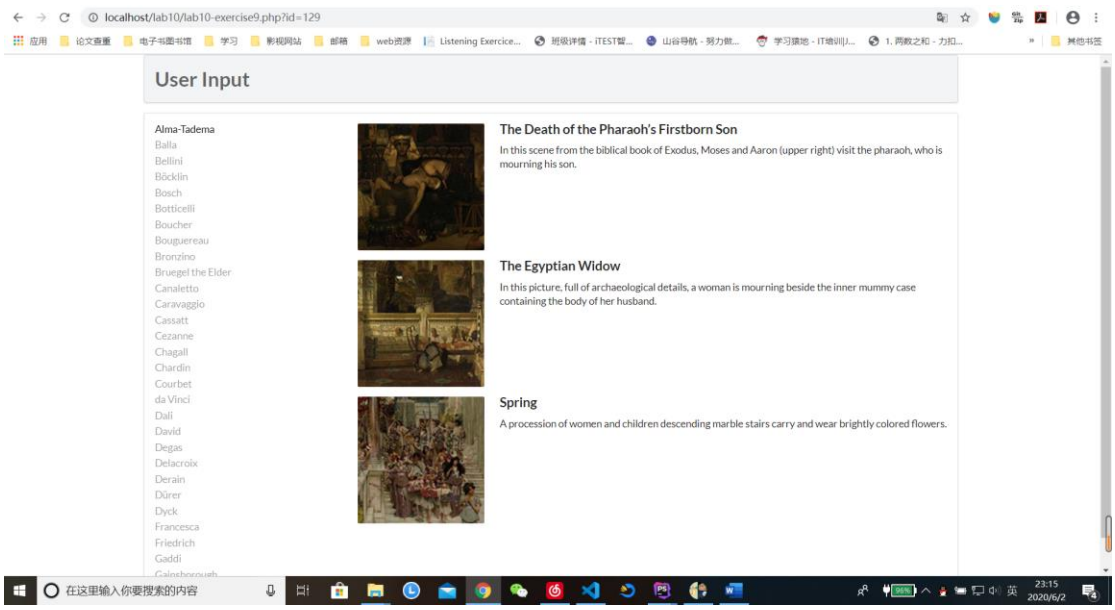
2. PDOStatement PDO::query(string statement)

query()方法用于返回执行查询后的结果集，参数 satatement 是要执行的 SQL 语句，它返回的是一个 PODStatement 对象

3. PDOStatement PDO::prepare(string statement[,array driver_options])
   bool PDOStatement::execute([array input_parameters])

预处理语句包括 prepare()和 execute()两种方法。首先，通过 prepare()方法做查询准备工作，然后通过 execute()方法执行查询，并且还可以通过 bindParam()方法来绑定参数给 execute()方法

PreparedStatement 优点：
1.PreparedStatement 预编译 SQL 语句，性能更好
2.PreparedStatement 无须拼接 SQL 语句，编程更简单
3.PreparedStatement 可以放置 SQL 注入，安全性更好
截图如下：



Exercise 10:

outputGenres():

创新 PDO 实例对象连接数据库，并在 Genres 表中选择 GenreId，GenreName，Description 的列的内容，并且以 GenreID 排序。循环遍历每个数据，并调用 outputSingleGenre（$row）函数。即生成一张张的图片。

outputSingleGenre（$row）：

根据传进的参数生成一组 html 语句：

```
<div class='ui fluid card'>
  <div class='ui fluid image'>
      <a href='genre.php?id=id'>
          <img src='images/art/genres/square-medium/id.jpg'>
      </a>
  </div>
  <div class='extra'>
      <h4>
          <a href='genre.php?id=id'>Genrename</a>
      </h4>
   </div>
</div>
```

即生成了一个带有链接的图片和字段。

constructGenreLink($id,$label):

根据传入的参数生成一个 a 标签

xx

综合起来实现了生成一个图片动态展示的页面，并且可以通过这些图片跳转到对应的详情页面。

截图如下：