1+X Web前端开发（中级）知识点

1.响应式开发技术（ Bootstrap框架）

Bootstrap 是由 Twiter 公司(全球最大的微博)的两名技术工程师研发的一个基于HTML、CSS、JavScript 的开源框架。该框架代码简洁、视觉优美，可用于快速、简单地构建基于 PC 及移动端设备的 Web 页面需求。

1.1 响应式网站开发

响应式网站设计是一种网络页面设计布局，其理念是：集中创建页面的图片排版大小，可以智能地根据用户行为以及使用的设备环境进行相对应的布局。

1.1.1 技术手段

- 一切弹性化
- 响应式图片

1.1.2 响应式布局与自适应布局的区别

响应式布局等于流动网格布局，而自适应布局等于使用固定分割点来进行布局。

1.1.3 响应式设计的步骤

1）设置 Meta 标签

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

2）通过媒体查询来设置样式

```
@media screen and (max-width: 980px) {
  #head { … }
  #content { … }
  #footer { … }
}
```

媒体查询的关键词有：max-width、min-width、not、only和and。

```
/* 当屏幕在大于等于600px小于等于900px，body的背景颜色设置为green。 */
@media screen and (min-width:600px) and (max-width:900px){
  body {background-color:green;}
}
/* "iphone.css"样式适用于最大设备宽度为480px */
<link rel="stylesheet" media="screen and (max-device-width:480px)"
href="iphone.css" />

/* 样式代码使用在除打印设备和设备宽度小于1200px下所有设备中。 */
@media not print and (max-width: 1200px){样式代码}

/* 显示屏幕且设备宽度为240px的设备才会引用android240.css文件 */
<link rel="stylesheet" media="only screen and (max-device-width:240px)"
href="android240.css" />
```

3）设置多种视图宽度

```
/ iPad /
@media only screen and (min-width: 768px) and (max-width: 1024px) {}
/ iPhone /
@media only screen and (min-width: 320px) and (max-width: 767px) {}
```

1.2 Bootstrap特点

- 跨设备、跨浏览器
- 响应式布局
- 提供的全面的组件
- 内置 jQuery 插件
- 支持 HTML5、CSS3
- 支持 LESS 动态样式

1.3 中文官方网站及下载和CDN

```
中文官方网站：
    V3.X版本：https://v3.bootcss.com/
    V4.X版本：https://v4.bootcss.com/
    V5.X版本：https://v5.bootcss.com/
CDN：
    https://www.bootcdn.cn/twitter-Bootstrap/
```

1.4 栅格布局

Bootstrap 内置了一套响应式、移动设备优先的流式栅格系统，随着屏幕设备或视口（viewport）尺寸的增加，系统会自动分为最多 12列。

“行（row）”必须包含在 .container （固定宽度）或 .container-fluid （100%宽度）中，以便为其赋予合适的排列（aligment）和内补（padding）。

栅格系统中的列是通过指定 1 到 12 的值来表示其跨越的范围。如果一“行（row）”中包含了的列大于 12，多余的列所在的元素将被作为一个整体另起一行排列。

相应设备类型：超小屏(<768px)、小屏(>=768px)、中屏(>=992px)和大屏>=1200px)。

内层.container 容器的自适应宽度为：自动、750px、970px 和 1170px。自动的意思为，如果你是手机屏幕，则全面独占一行显示。

w-100表示width:100%；h-100表示height:100%;

	超小屏幕 手机 (<576px)	**小屏幕 平板 (≥576px)**	**中等屏幕 桌面显示器 (≥768px)**	**大屏幕 大桌面显示器 (≥992px)**	**超大桌面显示器 (≥1200px)**
`.container` 最大宽度	None （自动）	540px	720px	960px	1140px
类前缀	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-lg-`
列数	12	12	12	12	12
最大列宽	自动	~62px	~81px	~97px	
间隙宽度（槽宽）	30px （每列左右均有 15px）				
可嵌套	是				
列排序	是				

eg:

- 移动设备优先

```
<meta name="viewport" content="width=device-width, initial-scale=1,maximum-scale=1, user-scalable=no" />
<!-- 布局容器
Bootstrap 需要为页面内容和栅格系统包裹一个.container 容器。由于 padding 等属性的原因，这两种容器类不能相互嵌套。-->
<!-- 固定宽度 -->
<div class="container">
...
</div>
<!-- 100%宽度 -->
<div class="container-fluid">
...
</div>
```

- 基本用法

```
<div class="container">
    <div class="row">
        <div class="col">1 of 2</div>
        <div class="col">2 of 2</div>
    </div>
    <div class="row">
        <div class="col">1 of 3</div>
        <div class="col">2 of 3</div>
        <div class="col">3 of 3</div>
    </div>
</div>
```

- **多屏响应式列**

```
<div class="container">
  <div class="row">
    <div class="col-md-8">.col-md-8</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  </div>
  <div class="row">
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  </div>
  <div class="row">
    <div class="col-6">.col-6</div>
    <div class="col-6">.col-6</div>
  </div>
</div>
```

- 使用row-cols-*类名将每行做等分处理

```
<div class="container">
  <div class="row row-cols-3">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

- 偏移列

偏移列通过 **offset-*-*** 类来设置。第一个星号(*)可以是 **sm、md、lg、xl**，表示屏幕设备类型，第二个星号(*)可以是 **1** 到 **11** 的数字。为了在大屏幕显示器上使用偏移，请使用 **.offset-md-*** 类。这些类会把一个列的左外边距（margin）增加 ***** 列，其中 ***** 范围是从 **1** 到 **11**。

例如：.offset-md-4 是把.col-md-4 往右移了四列格。

```
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4 offset-md-4">.col-md-4 .offset-md-4</div>
</div>
<div class="row">
  <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
  <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
</div>
<div class="row">
  <div class="col-md-6 offset-md-3">.col-md-6 .offset-md-3</div>
</div>
```

1.5 全局CSS样式

** 1.5.1 排版样式 **

Bootstrap 将全局 font-size 设置为14px，line-height 行高设置为 1.428(即 20px)；<p>（段落）元素还被设置了等于 1/2 行高（即 10px）的底部外边距（margin）。

1.6 表格

基本表格：.table

条纹状表格：.table-striped

带边框的表格：.table-bordered

鼠标悬停：.table-hover

黑色背景表格：.table-dark

无边框表格：.table-borderless

紧缩表格：table-condensed

指定意义的颜色类：.table-primary、.table-success、.table-info、.table-warning、.table-danger .table-active .table-secondary .table-light

较小的表格：.table-sm

响应式表格：.table-responsive

1.7 表单

- 基本格式

  .mb-2 类设置底部边距， **.mr-sm-2** 类来设置右边距

```
<form>
    <div class="form-group">
        <label for="exampleInputEmail1">电子邮件</label>
        <input type="email" class="form-control" id="exampleInputEmail1"
placeholder="请输入您的电子邮件">
    </div>
    <div class="form-group">
        <label for="exampleInputPassword">密码</label>
        <input type="password" id="exampleInputPassword1" class="form-control"
placeholder="请输入您的密码">
    </div>
    <button type="submit" class="btn btn-primary">提交</button>
</form>
```

- 内联表单

  让表单左对齐浮动，并表现为 inline-block 内联块结构。

```
<form class="form-inline"></form>
```

注意：在屏幕宽度小于 576px 时为垂直堆叠，如果屏幕宽度大于等于576px时表单元素才会显示在同一个水平线上。

- 表格网格

  使用我们的网格类可以构建更复杂的表单。将这些用于需要多列、不同宽度和其他对齐。

  也可以换成`.row`标准`.form-row`网格行的变体，它覆盖了默认的列间距，以获得更紧凑。

  ```
  <form>
    <div class="row">
      <div class="col">
        <input type="text" class="form-control" placeholder="First name">
      </div>
      <div class="col">
        <input type="text" class="form-control" placeholder="Last name">
      </div>
    </div>
  </form>
  ```

- 设置水平滚动范围 .form-control-range

```
<form>
  <div class="form-group">
    <label for="formControlRange">Example Range input</label>
    <input type="range" class="form-control-range" id="formControlRange">
  </div>
</form
```

- 单选框

```
<div class="form-check">
  <input class="form-check-input" type="radio" name="exampleRadios"
id="exampleRadios1" value="option1" checked>
```

```
  <label class="form-check-label" for="exampleRadios1">
    Default radio
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="exampleRadios"
id="exampleRadios2" value="option2">
  <label class="form-check-label" for="exampleRadios2">
    Second default radio
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="exampleRadios"
id="exampleRadios3" value="option3" disabled>
  <label class="form-check-label" for="exampleRadios3">
    Disabled radio
  </label>
</div>
```

- 复选框

  使用 `.form-check-inline` 类可以让选项显示在同一行上：

  ```
  <div class="form-check">
    <label class="form-check-label">
      <input type="checkbox" class="form-check-input" value="">Option 1
    </label>
  </div>
  <div class="form-check">
    <label class="form-check-label">
      <input type="checkbox" class="form-check-input" value="">Option 2
    </label>
  </div>
  <div class="form-check disabled">
    <label class="form-check-label">
      <input type="checkbox" class="form-check-input" value=""
  disabled>Option 3
    </label>
  </div>
  ```

- 开关

  开关具有自定义复选框的标记，但使用 `.custom-switch` 该类来呈现切换开关，开关也支持该 `disabled` 属性。

  ```
  <div class="custom-control custom-switch">
    <input type="checkbox" class="custom-control-input" id="customSwitch1">
    <label class="custom-control-label" for="customSwitch1">Toggle this
  switch element</label>
  </div>
  ```

- 下拉选择框

  自定义 `<select>` 菜单只需要一个自定义类，`.custom-select` 即可触发自定义样式。

```
<select class="custom-select">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

1.8 按钮

- 基本按钮

Bootstrap 提供了很多丰富按钮供开发者使用。可作为按钮使用的标签或元素转化成普通按钮。

```
<a href="###" class="btn btn-default">Link</a>
<button class="btn btn-default">Button</button>
<input type="button" class="btn btn-default" value="input">
```

- 预定义样式

```
.btn-default  默认样式
.btn-success  成功样式
.btn-info  一般信息样式
.btn-warning 警告样式
.btn-danger  危险样式
.btn-primary 首选项样式
.btn-link  链接样式
.btn-light 浅灰色样式
.btn-dark 黑色样式
.btn-link 链接样式
```

- 尺寸大小

```
<button class="btn btn-lg">Button</button>
<button class="btn">Button</button>

<button class="btn btn-sm">Button</button>
<button class="btn btn-xs">Button</button>
```

- 块级按钮

```
<button class="btn btn-block">Button</button>
```

- 激活状态

```
<button class="btn active">Button</button>
```

- 禁用状态

```
<button class="btn disabled">Button</button>
```

- 带轮廓线的按钮

  默认的修饰符类替换为 `.btn-outline-*` 系列类，已去除按钮上的所有背景图片和颜色。

```
<button type="button" class="btn btn-outline-primary">Primary</button>
<button type="button" class="btn btn-outline-secondary">Secondary</button>
<button type="button" class="btn btn-outline-success">Success</button>
<button type="button" class="btn btn-outline-danger">Danger</button>
<button type="button" class="btn btn-outline-warning">Warning</button>
```

1.9 图片

- 三种形状

```
.img-rounded
.img-circle
.img-thumbnail
```

  注意：IE8及以下版本不支持 CSS3 中的圆角属性。

- 响应式图片

```
<img src="img/pic.png" alt="图片" class="img-responsive">
```

1.10 组件

- 徽章（Badge）

```
<button type="button" class="btn btn-primary">
  Notifications <span class="badge badge-light">4</span>
</button>
<!-- 改变徽章的外观 -->
<span class="badge badge-primary">Primary</span>
<span class="badge badge-success">Success</span>
<span class="badge badge-warning">Warning</span>
```

  拓展：.badge-pill可以让徽章的边角变得更圆滑，也可以在<a>标签上使用 `.badge-*` 类。

- 面包屑导航（Breadcrumb）

```
<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item"><a href="#">Library</a></li>
    <li class="breadcrumb-item active" aria-current="page">Data</li>
  </ol>
</nav>
```

- 下拉菜单

```
<div class="dropdown">
    <button class="btn btn-secondary dropdown-toggle" type="button" data-toggle="dropdown" aria-expanded="false">
        下拉按钮
    </button>
    <div class="dropdown-menu">
        <a class="dropdown-item" href="#">首页</a>
        <a class="dropdown-item" href="#">资讯</a>
        <a class="dropdown-item" href="#">关于我们</a>
    </div>
</div>
```

- 按钮组

```
<div class="btn-toolbar">
    <div class="btn-group">
        <button type="button" class="btn btn-secondary">左</button>
        <button type="button" class="btn btn-secondary">中</button>
        <button type="button" class="btn btn-secondary">右</button>
    </div>
    <div class="btn-group">
        <button type="button" class="btn btn-primary">1</button>
        <button type="button" class="btn btn-primary">2</button>
        <button type="button" class="btn btn-primary">3</button>
    </div>
</div>
```

- 列表组

```
<ul class="list-group">
  <li class="list-group-item">An item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
  <li class="list-group-item">A fourth item</li>
  <li class="list-group-item">And a fifth one</li>
</ul>
```

- 卡片（Card）

  父级元素的类名为`.card`;使用`.card-header`创建头部样式;使用`.card-body`创建卡片内容;使用`.card-footer`创建卡片的底部样式。

```
<div class="card">
  <div class="card-header">头部</div>
  <div class="card-body">内容</div>
  <div class="card-footer">底部</div>
</div>
```

  注意：可以在元素上使用`.card-title`类来设置卡片的标题 。`.card-text`类用于设置卡片正文的内容。`.card-link`类用于给链接设置颜色。

- 导航

```
<ul class="nav">
```

```
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled">Disabled</a>
  </li>
</ul>
```

- 导航栏

  使用 `.navbar` 类来创建一个标准的导航栏，后面紧跟: `.navbar-expand-xl|lg|md|sm` 类来创建响应式的导航栏.

  创建不同颜色导航栏 `.bg-primary`, `.bg-success` `.bg-info`, `.bg-warning`, `.bg-danger`, `.bg-secondary`, `.bg-dark` , `.bg-light`

  .navbar-dark:深色背景，加此样式可以突显文字。

  .navbar-light:浅色背景，加此样式可以突显文字。

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)
</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-expanded="false">
          Dropdown
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item" href="#">Action</a>
          <a class="dropdown-item" href="#">Another action</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Something else here</a>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled">Disabled</a>
      </li>
```

```
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search"
placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>
    </form>
  </div>
</nav>
```

- 巨幕（Jumbotron）

  创建一个大的灰色背景框，里面可以设置一些特殊的内容和信息。主要是展示网站的关键性区域。

```
<ol class="breadcrumb">
    <li><a href="#">Home</a></li>
    <li><a href="#">Library</a></li>
    <li class="active">Data</li>
</ol>
```

- 分页

```
<nav aria-label="Page navigation example">
  <ul class="pagination">
    <li class="page-item"><a class="page-link" href="#">Previous</a></li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item"><a class="page-link" href="#">Next</a></li>
  </ul>
</nav>
```

- 警告框

```
<div class="alert alert-success" role="alert">...</div>
```

- 进度条

```
<!-- 动画效果 -->
<div class="progress">
  <div class="progress-bar progress-bar-success progress-bar-striped
active" style="min-width:20px;width:60%">60%</div>
</div>
```

- 媒体对象

  在容器元素上添加 `.media` 类，然后将多媒体内容放到子容器上，子容器需要添加 `.media-body` 类，然后添加外边距，内边距等效果:

```
<div class="media">
    <img src="./img/p3.jpg" width="200" class="mr-3" alt="...">
    <div class="media-body">
        <h5 class="mt-0">Media heading</h5>
        <p>Will you do the same for me? It's time to face the music I'm
no longer your muse. Heard it's
            beautiful, be the judge and my girls gonna take a vote. I can
feel a phoenix inside of me.
            Heaven is jealous of our love, angels are crying from up
above. Yeah, you take me to utopia.</p>
    </div>
</div>
```

1.11 JS插件

```
（注：下面内容可以结合官网进行讲解）
    过渡效果
    模态框
    下拉菜单
    滚动监听
    标签页
    工具提示
    弹出框
    警告框
    按钮
    Collapse
    Carousel
```

1.12 案例实操

1-分类信息

项目需求：

酒类商品的分类信息页面，项目采用Bootstrap框架构建，分类信息界面使用导航和栅格系统布局，PC端和移动端能够自适应显示，内容分为三部分。

顶部：包括导航栏、折叠导航栏、下拉列表框，导航栏显示在PC端界面当中，折叠导航栏显示在移动端界面当中，下拉列表框当中显示“葡萄酒”的类别。

信息栏：信息栏左侧显示产地（中国、法国、美国和加拿大），信息栏右侧显示酒的详细信息（酒的名称、描述、价格、是否包邮、容量、酒精度数）。

底部：显示版权信息。

【效果图】

（1） category.html在PC端效果，如图所示。

图 酒类商品的分类信息页面（PC端）

（2）category.html在移动端效果，如图所示。

图 酒类商品的分类信息页面（移动端）

2-注册页面

项目需求：

模拟门户网站的注册页面，页面顶部是页头导航栏，中间是form表单，底部是页脚。用Bootstrap的导航栏组件来制作页头，用Bootstrap的表单制作注册表单，注册表单内使用Bootstrap输入框组、按钮组件来制作表单项，页面效果如图所示。

图 注册界面显示效果

2.MySQL数据库基础与应用

2.1 安装并启动服务

1）安装MySQL（直接安装XAMPP集成环境即可，也可单独安装MySQL）

```
XAMPP = Apache+MySQL+PHP+Perl（X是系统的平台代号）
```

a）下载

下载地址：https://www.apachefriends.org/zh_cn/download.html

图 下载XAMPP

b）安装

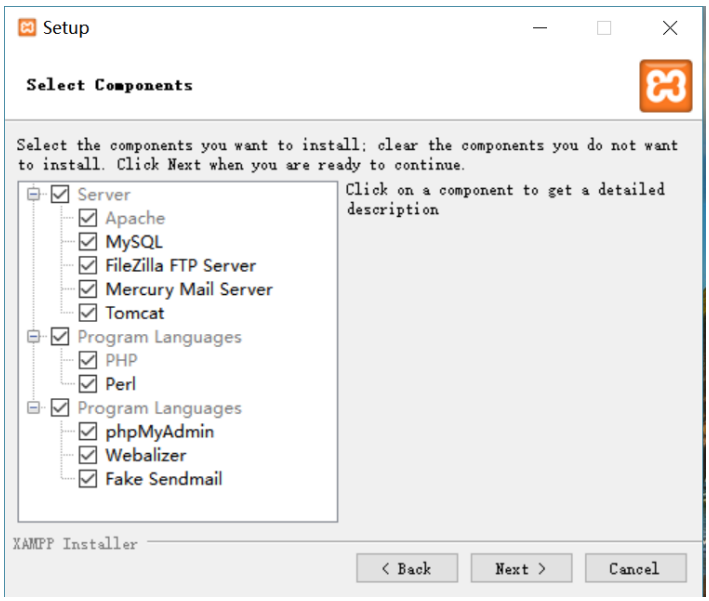


图 安装XAMPP

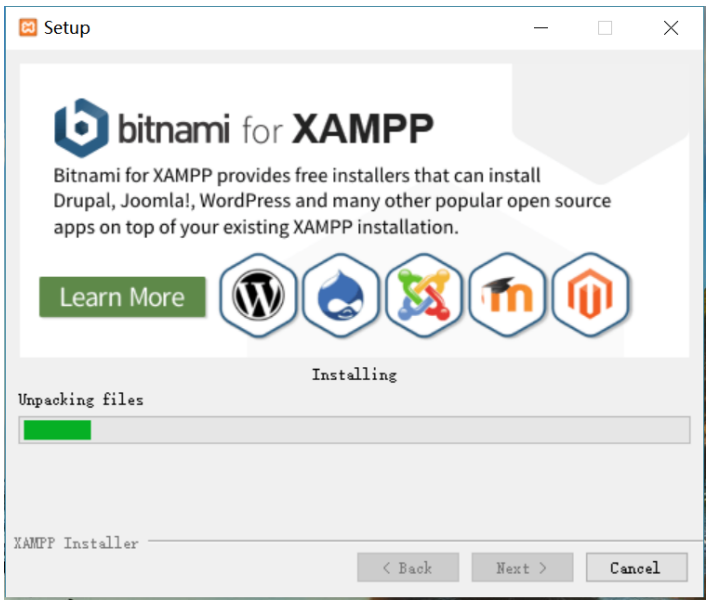


图 XAMPP安装过程中

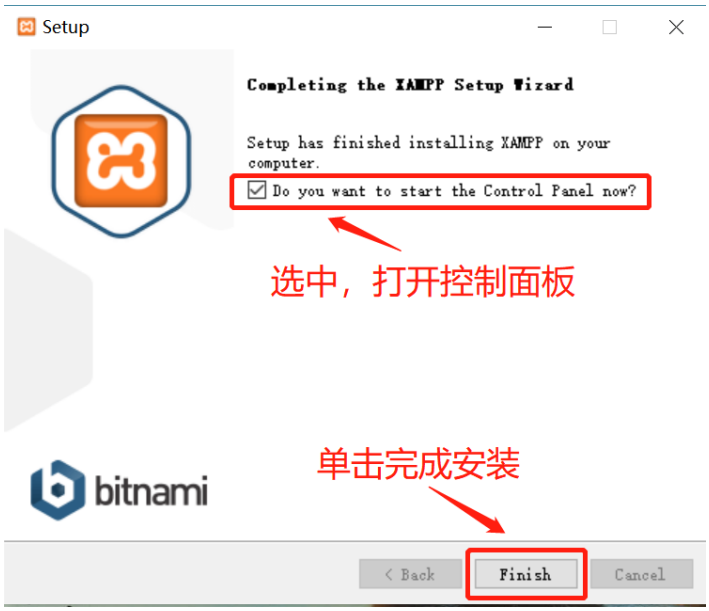


图 XAMPP安装完成

c）启动服务

图 启动MySQL服务

2）进入MySQL操作后台（如果显示下图所示结果，表示MySQL启动成功）

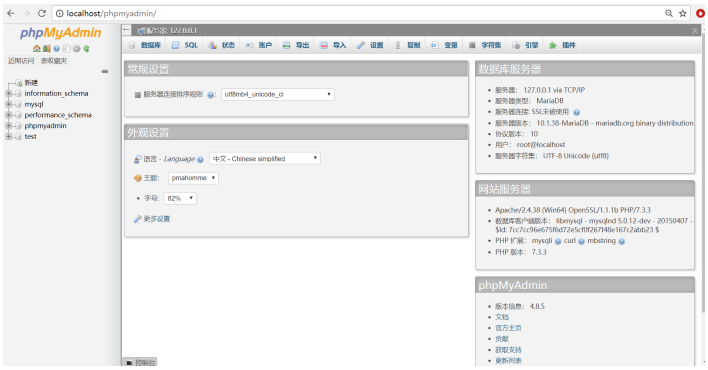


图 MySQL操作界面

3）如果为了方便操作，可以选择安装MySQL可视化操作软件Navicat软件（注：考试时不预装该软件）。

2.2 基础知识

MySQL属于关系数据库。

SQL(Structured Query Language)是用于访问和处理数据库的 标准计算机语言。使用SQL 访问和处理数据系统中的数据，这类数据库包括：Oracle,mysql,Sybase, SQLServer, DB2, Access 等等。

SQL语言由DDL、DML、DQL、DCL等几部分语言组成。

SQL 对大小写不敏感，一般数据库名称、表名称、字段名称全部小写。

MySQL要求在每条 SQL 命令的末端使用分号。

2.3 数据库/表基本操作

2.3.1 登录与退出

- DOS中登录

```
mysql -u用户名 -p密码 -h主机 -P端口
```

eg:

```
mysql -uroot -p
-- 或
mysql -P3306 -h127.0.0.1 -uroot -p -- 注意：-P3306中的P为大写
```

```
-- 如果有密码则输入，没有就直接回车

-- 修改密码
-- 1.进入MySQL环境下，输入：
set password for '帐号'@'主机IP'=password('新密码');
-- 2.在DOS环境下
mysqladmin -u帐号 -p password 新密码

-- 删除密码
use mysql;
update user set password='' where user='root';
```

如图所示：

```
C:\xampp\mysql\bin>mysql -uroot -p
Enter password: ****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.20 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

- **DOS中退出**

```
exit或\q或按Ctrl+C组合键
```

3.3.2 数据库操作

- 创建数据库

```
create database 库名;
```

eg:

```
create database stuManager;
-- 创建数据库要使用下面这种格式，否则表中如果出现汉字时可能会出现乱码。
create database stuManager default character set utf8 collate utf8_general_ci; -
- 创建数据库同时设置编码格式
```

查看数据库：

```
show databases;
```

删除数据库：

```
drop database 库名;
```

打开数据库：

```
use 库名;
```

显示当前打开的数据库：

```
select database();
```

2.3.3 数据表

- 注释

```
单行注释：# 注释内容
多行注释：/* 注释内容 */
行末注释：-- 注释内容
```

- 数据类型（常用）

```
-- 字符型
char              -- 定长字符串
varchar           -- 变长字符串
text              -- 文本
-- 整型
tinyint           -- 极小整型
smallint          -- 小整型
int或integer      -- 整型
bigint            -- 大整型
-- 浮点型
float             -- 单精度
double            -- 双精度
decimal           -- 定点型
-- 日期型
date              -- 日期型
time              -- 时间型
year              -- 年份
datetime          -- 日期时间型
timestamp         -- 时间戳
```

- 约束

```
not null/null    -- 非空/空
default          -- 默认值
primary key      -- 主键
foreign key      -- 外键
unique           -- 唯一
```

注意：外键约束是为了保持数据一致性，完整性，实现一对一或一对多关系。子表（具有外键列的表）和 父表（子表所参照的表），存储引擎只能为innoDB。

- 创建数据表

```
create table 表名(列1 类型 [宽度] 约束，列2 类型 [宽度] 约束，……);
```

eg：

```
use stuManager;

# 创建学生表
```

```
drop table if exists student;
create table student (
    sno varchar(3) not null primary key comment '学号', -- 主键约束（值不能为空，且唯一）
    sname varchar(4) not null comment '姓名', -- 非空约束
    ssex char(2) default "男" comment '性别', -- 默认值约束
    sbirthday varchar(10) default null comment '出生日期',
    class varchar(5) not null comment '班级'
) engine=innodb default charset=utf8; -- 设置编码

# 创建教师表
drop table if exists teacher;
create table teacher (
    tno varchar(3) primary key,
    tname varchar(4) not null,
    tsex varchar(2) not null,
    tbirthday datetime not null,
    prof varchar(6) default null,
    depart varchar(10) not null
) engine=innodb default charset=utf8;

# 创建课程表
drop table if exists course;
create table course (
  cno varchar(5) primary key,
  cname varchar(10) not null,
  credit int not null check (credit>0 and credit<10), -- 检查约束
  tno varchar(10) not null,
  foreign key(tno) references teacher(tno)
) engine=innodb default charset=utf8;

# 创建成绩表
drop table if exists score;
create table score(
    sno varchar(3) not null,
    cno varchar(5) not null,
    grade int,
    primary key(sno, cno),
    foreign key(sno) references student(sno),-- 外键约束
    foreign key(cno) references course(cno)
)engine=innodb default charset=utf8;

# 创建登录用户表
drop table if exists admin;
create table admin(
    id int primary key auto_increment, -- 自动编号约束
    username varchar(20) not null unique, -- 唯一性约束
    password varchar(10) not null,
    createtime datetime
)engine=innodb default charset=utf8;
```

- 查看表结构

```
desc 表名;
```

- 修改表结构

```
alter table 表名 [add|change|modify|drop] 列名 [类型];
```

eg:

```
-- 添加一列
alter table student add department varchar(20);
desc student;
-- 添加多列
alter table student add age int, add addr varchar(20);
-- 修改列
alter table student modify sbirthday date;
alter table student change sbirthday sdate year;
alter table student change department department varchar(15);
## change可以更改列名和列类型 (每次都要把新列名和旧列名写上, 即使两个列名没有更改,只是改了类型)
## modify只能更改列属性。只需要写一次列名, 比change省事点
-- 删除列
alter table student drop department; -- 删除一列
alter table student drop age,drop addr; -- 删除多列
```

- 添加数据

```
insert into 表名[(列名表)] value/values(值列表或select语句) [value/values(值列表或select语句) ...];
```

eg:

```
# 添加一条数据
insert into student values('001','王小二','男',1998,'20030'); -- 按列依次添加
insert into student(sno,sname,class) values('101','小乔','20031');
# 添加多条数据
insert into student value('002','张小莉','女',1999,'20030'),('003','张军','男',1997,'20030');
```

给三张表分别添加多条数据，方便后续操作。

注意：必须先给主键所在的表添加数据，再给外键所在的表添加数据，因为它们之间有依赖关系存在。

```
-- 帐号管理表（admin）
insert into admin(username,password) values('hsping', '123456');
-- 学生表（student）
insert into student value('102', '李军', '男', '1999', '20033'),
('103', '陆君', '男', 1999, '20031'),
('105', '匡明', '男', 1998, '20031'),
('107', '王丽', '女', 1997, '20033'),
('108', '曾华', '男', 2000, '20033'),
('109', '王芳', '女', 2001, '20031');
-- 教师表（teacher）
insert into teacher
value('804', '李诚', '男', '1958-12-02', '副教授', '计算机系'),
    ('825', '王萍', '女', '1972-05-05', '助教', '计算机系'),
    ('831', '刘冰', '女', '1977-08-14', '助教', '电子工程系'),
    ('856', '张旭', '男', '1969-03-12', '讲师', '电子工程系');
set foreign_key_checks=1; # 启动外键约束
```

```
-- 课程表（course）
insert into course value ('6-166', '数据电路',4, '856'),
('3-105', '计算机导论',3,'825'),
('3-245', '操作系统',3,'804'),
('9-888', '高等数学',5,'831');
-- 成绩表（score）
insert into score value('103', '3-245', 86.0),
('105', '3-245', 75.0),
('109', '3-245', 68.0),
('103', '3-105', 92.0),
('105', '3-105', 88.0),
('109', '3-105', 76.0),
('102', '3-105', 64.0),
('107', '3-105', 91.0),
('108', '3-105', 78.0),
('109', '6-166', 85.0),
('108', '6-166', 81.0);
```

- 修改数据

```
update 表名 set 列1 = 表达式1,列2 = 表达式2，…… [where <条件>];
```

eg:

```
-- 将所有选修3-105课程的学生都加上5分。
update score set grade = grade - 5 where cno = '3-105';
```

- 删除数据

```
delete from 表名 [where <条件>];
truncate table <表名>;
```

eg:

```
-- 复制表student为stu和stu_bak
create table stu select * from student;
create table stu_bak select * from stu;

show tables;
select * from student;
-- 删除stu_bak表
drop table stu_bak;
```

2.4 基本查询（单表查询）

```
-- 基本语法结构
select [distinct] *|列名表 [as <别名>]
from <表名列表>
where <条件>
group by 列名1[,列名2，...]
having <分组条件>
order by  列名1[asc|desc][,列名2[asc|desc]，...]
limit [skip],[count]
```

```
/*
条件
    1）关系运算符： > >= < <= != = ==
    2）模糊比较：like (_、%)

    注意：模糊比较只能对字符串操作

    3）介于...之间：[not] between...and...
    4）逻辑运算符：not and or
    5）in： 相当于逻辑或
    6）exsits： 是否存在
    7）[=/>/<]any/some/all

集合函数
    sum:对数值型求和
    avg:对数值求平均值
    max:求最大值
    min:求最小值
    count:计数(跟列名没有任何列没有关系，可以写成：count(列名)或count(*))
*/
```

- 去重

```
-- 有哪些学生参加了考试?
select distinct 表 from 字段;
```

- 输出列重命名（别名）

```
-- 查询每位学生的学号和姓名。
```

- 分组

```
-- 查询每位学生的考试总分。
```

- 分组条件

```
-- 查询选修两门及两门以上课程的学生。
```

- 排序

```
-- 对学生的成绩进行降序输出。
```

- 分页

```
-- 查询前三位学生的情况。
```

2.5 关系运算

关系运算包括四种：投影（select）、选择（where/having）、连接（join）、除（没有直接的对应语句，需要根据需求书写复杂的SQL，通常都会包含子查询）。

2.6 多表查询

```
-- 查询每一位学生每一门课的考试成绩。
-- 提示：三张表(a作为student别名；b作为course别名；c作为score别名)
```

2.7 视图

视图是虚拟表，是从基本表中派生出来的，可以像基本表一样对其进行操作。

```
创建视图：
    create view 视图名 as select语句;
删除视图：
    drop view 视图名;
```

eg:

```
-- 创建视图呈现每一位学生每一门课的考试成绩。
create view v_stu_course_score as (
    --- 逻辑sql语句
);

show tables;
-- 删除上面创建的视图。
drop view v_stu_course_score;
```

2.8 索引

索引是面向数据库本身的，用于查询优化等操作。

- 索引类型

```
普通索引：仅加速查询
唯一索引：加速查询 + 列值唯一（可以有null）
主键索引：加速查询 + 列值唯一（不可以有null）+ 表中只有一个
组合索引：多列值组成一个索引，专门用于组合搜索，其效率大于索引合并
全文索引：对文本的内容进行分词，进行搜索
```

```
-- 创建普通索引
create index 索引名 on 表名(列名);
-- 创建唯一索引
create unique index 索引名 on 表名(列名);
-- 创建普通组合索引
create index 索引名 on 表名(列名1,列名2,...);
-- 创建唯一组合索引
create unique index 索引名 on 表名(列名1,列名2,...);
-- 查看索引
show index from 表名 [from <数据库名>];
```

2.9 触发器

触发器（trigger），也叫触发程序，是与表有关的命名数据库对象，是 MySQL中提供给程序员来保证数据完整性的一种方法，它是与表事件 INSET、UPDATE、DELETE相关的一种特殊的存储过程，它的执行是由事件来触发，比如当对一个表进行 INSET、UPDATE、DELETE 事件时就会激活它执行。因此，删除、新增或者修改操作可能都会激活触发器，所以不要编写过于复杂的触发器，也不要增加过多的触发器，这样会对数据的插入、修改或者删除带来比较严重的影响，同时也会带来可移植性差的后果，所以在设计触发器的时候一定要有所考虑。

特征：

1. 有begin...end体，begin...end之间的语句可以写的简单或者复杂；
2. 什么条件会触发：I、D、U；
3. 什么时候触发：在增删改前或者后；
4. 触发频率：针对每一行执行；
5. 触发器定义在表上，附着在表上。

语法格式：

```
create trigger 触发器名 触发时机 触发事件 on 表名 for each row 触发器程序体
# 触发时机取值为before或after。
# 触发事件可以是insert、update或delete。
# for each row：表示任何一条记录上的操作满足触发事件都会触发该触发器。
# 触发器程序体可以是一条语句，也可以是由begin...end包裹的多条语句。
```

eg：

```
-- 创建触发器，如向学生表中添加数据时，将添加时的时间加到admin表中
```

2.10 存储过程

存储过程（Stored Procedure）是一种在数据库中存储复杂程序，以便外部程序调用的一种数据库对象。

存储过程是为了完成特定功能的 SQL语句集，经编译创建并保存在数据库中，用户可通过指定存储过程的名字并给定参数(需要时)来调用执行。

- 语法格式

```
create procedure 存储过程名称(in|out|inout 参数名称 参数类型,.....)
begin
    过程体;
end

说明：
    in 输入参数：表示调用者向过程传入值（传入值可以是字面量或变量）
    out 输出参数：表示过程向调用者传出值(可以返回多个值)（传出值只能是变量）
    inout 输入输出参数：既表示调用者向过程传入值，又表示过程向调用者传出值（值只能是变量）
```

eg：

```
-- 创建存储过程，用于统计20030班学生人数。
```

- 查询

```
-- 查看所有存储过程状态。
show procedure status;
-- 查看对应数据库下所有存储过程状态。
show procedure status where db='数据库名';
-- 查看名称包含 student 的存储过程状态。
show procedure status where name like '%student%';
-- 查询存储过程详细代码。
show create procedure 过程名;
```

- 调用存储过程

  语法格式：

```
call 存储过程名 ([过程参数[,...]])
```

  eg：

```
call getStudentCount();
```

2.11 权限管理

- 添加权限（和已有权限合并，不会覆盖已有权限）

```
grant 权限 on `数据库名`.* to `用户名`@`主机名`;

# 权限可以是create、alert、drop、update、delete、select、grant option、all等，但不能是revoke。
```

  eg：

```
grant select,create,drop,update,alter on *.* to 'root'@'localhost' identified by 'root' with grant option;
```

- 删除权限

```
revoke 权限 ON `数据库名`.* to `用户名`@`主机名`;
```

- 查看用户权限

```
show grants for 'root'@'localhost';
```

2.12 事务

MySQL 事务主要用于处理操作量大，复杂度高的数据。

一般来说，事务是必须满足4个条件（ACID）：

```
原子性（Atomicity，或称不可分割性）：事务中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback）到事务开始前的状态。
一致性（Consistency）：事务要求所有的DML语句操作的时候，必须保证同时成功或者同时失败。
隔离性（Isolation，又称独立性）：事务A和事务B之间具有隔离性。
持久性（Durability）：事务处理结束后，对数据的修改就是永久的，即便系统故障也不会丢失。
```

- 事件操作

```
开启事务：begin或start transaction
事务结束：commit/rollback之后
提交事务：commit
回滚事务：rollback
```

eg:

```
-- 对学号为105做退学处理。
# 开始
begin;
    delete from student where sno='105';
    delete from score where sno='105';
# 提交
commit;
# 回滚
rollback;
```

- 事务开启的标志和事务结束的标志

```
开启标志：
    任何一条DML语句(insert、update、delete)执行，标志事务的开启。
结束标志(提交或者回滚)：
    提交：成功的结束，将所有的DML语句操作历史记录和底层硬盘数据来一次同步。
    回滚：失败的结束，将所有的DML语句操作历史记录全部清空。
```

2.13 数据库备份与恢复

- 数据库备份

```
mysqldump -u用户名 -p密码 数据库名 > 文件名.sql
```

注意：这里备份的文件的扩展名不一定是.sql!

eg:

```
mysqldump -uroot -p mydb>d:\db.txt
```

- 数据库恢复

```
mysql -u用户名 -p密码 新数据库名 < 文件名.sql
```

eg:

```
-- 先创建数据student_manager（在MySQL环境中）。
create database student_manager DEFAULT CHARACTER SET utf8 COLLATE
utf8_general_ci;;
-- 恢复数据到上面创建的数据中（在DOS命令环境中）。
mysql -uroot -p student_manager < d:\db.txt
```

3.PHP技术与应用

3.1 简介

PHP（原名 Personal Home Page的缩写，已经正式更名为 "PHP: Hypertext Preprocessor"，中文名：“超文本预处理器”）是一种通用开源脚本语言。语法吸收了 C 语言、Java和 Perl 的特点，利于学习，使用广泛，主要适用于 Web 开发领域。PHP 独特的语法混合了 C、Java、Perl 以及 PHP 自创的语法。用 PHP 做出的动态页面与其他的编程语言相比，PHP 是将程序嵌入到 HTML文档中，执行效率很高。

3.2 环境配置

要想使用这门语言，需要搭配相应的开发环境，主要包括：

- Apache web 服务器
- MySQL 数据库
- PHP 语言引擎

安装XAMPP集成环境（前面已安装），下面完成配置。

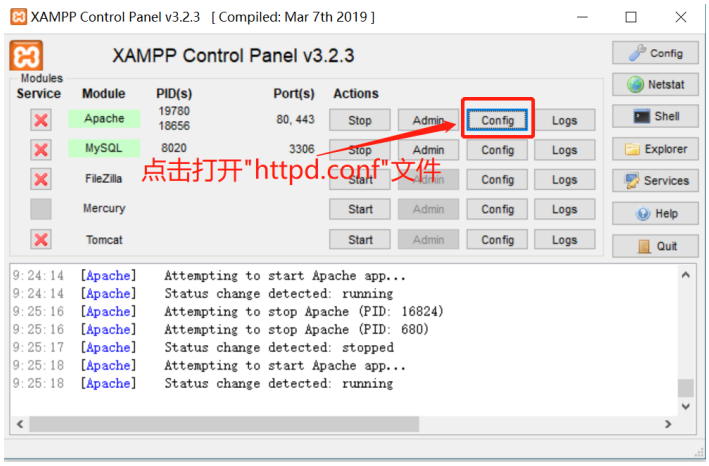


图 控制台PHP配置

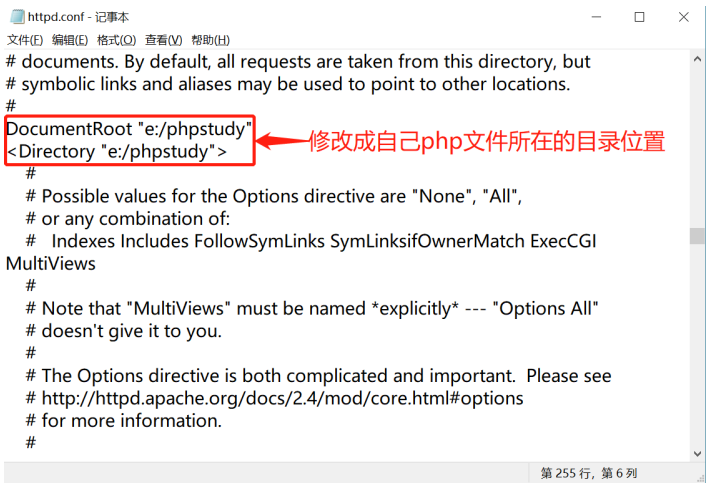


图 PHP文件路径映射设置

注意：配置文件修改后，必须重启服务。

3.3 语言基础

3.3.1 注释符与结束符

```
// 单行注释
/* 多行注释 */
// 结束符使用英文分号 “;”
```

3.3.2 常用命令和系统函数

- echo

  echo输出 : 只能输出字符串、数字、布尔（true:1 false:空）类型的数据。

- var_dump()

此函数显示一个或多个表达式的结构信息，包括表达式的类型与值、长度。数组将递归展开值，通过缩进显示其结构。常用来调试。

- print()

  print() 和 echo() 用法一样，但是echo速度会比print快一点点。

- print_r()

print_r 可以输出stirng、int、float、array、object等，输出array时会用结构表示， print_r 输出成功时返回true。

print_r 可以通过 print_r($str,true) 来使print_r不输出而返回 print_r 处理后的值。

```
echo 、 print 的区别：
    echo,print 的区别在于 echo 可以输出多个变量值,而print只有一个变量，做为一个字符串输出.另一点区别在于 echo 没有返回值， print 有返回值1； print 不能输出数组和对象。
    一般来说，PHP中动态输出HTML内容，是通过 print 和 echo 语句来实现的，在实际使用中，print 和 echo 两者的功能几乎是完全一样。可以这么说，凡是有一个可以使用的地方，另一个也可以使用。但是两者之间也还是一个非常重要的区别：在 echo 函数中，可以同时输出多个字符串，而在 print 函数中则只可以同时输出一个字符串。同时， echo 函数并不需要圆括号，所以 echo 函数更像是语句而不像是函数。
    echo 和 print 都不是函数，而是语言结构，所以圆括号都不是必需的。
```

- require 与 include

  1）require

  常用于引入重要文件，若引入失败会直接影响到当前整个脚本，引入失败报Error错误。

```
require("myfile.php")
```

  2）include

  常用于引入普通 文件，若引入失败不会对当前脚本有较大的影响，引入失败报Warning错误。

```
include("myfile.php")
```

  3）require_once

  避免重复引入，其他规则同 require。

4）include_once

避免重复引入，其他规则同 include。

> require() 语句的性能与 include() 相类似，都是包括并运行指定文件。不同之处在于：对 include() 语句来说，在执行文件时每次都要进行读取和评估；而对于 require() 来说，文件只处理一次（实际上，文件内容替换 require() 语句）。这就意味着如果可能执行多次的代码，则使用 require() 效率比较高。另外一方面，如果每次执行代码时是读取不同的文件，或者有通过一组文件迭代的循环，就使用 include() 语句。

- header

  向客户端发送原始的 HTTP 报头。

  i）跳转页面

```
header('Location:'.$url);  //Location和":"之间无空格。
```

  ii）声明content-type，设置编码

```
header('content-type:text/html;charset=utf-8');
```

  iii）返回response状态码

```
header('HTTP/1.1 404 Not Found');
```

  iv）在某个时间后执行跳转

```
header('Refresh: 10; url=http://www.baidu.com/');  //10s后跳转。
```

  v）执行http验证

```
header('HTTP/1.1 401 Unauthorized');
header('WWW-Authenticate: Basic realm="Top Secret"');
```

  vi）执行下载操作

```
header('Content-Type: application/octet-stream'); //设置内容类型
header('Content-Disposition: attachment; filename="example.zip"'); //设置MIME用户作为附件
header('Content-Transfer-Encoding: binary'); //设置传输方式
header('Content-Length: '.filesize('example.zip')); //设置内容长度
```

4.3.3 变量

声明变量：$

```
$变量名 = 表达式;
```

命名规则：字母数字下划线、首字符不能为数字、严格区分大小写、且不能使用关键字！推荐驼峰命名法。

4.3.4 常量

使用 define关键字定义常量，常量命名要全部大写，常量的数据类型不能是资源、对象。

```
define('常量名',表达式);
const 常量名 = 表达式;
```

```
可以用 define() 函数来定义常量。在php5.3.0以后，可以使用 const 关键字在类定义的外部定义常量，先前版本 const 关键字只能在类(class)中使用。一个常量一旦被定义，就不能再改变或取消定义。
```

eg:

```
// 定义常量
define("FRONT_END","前端开发");
```

```
常量和变量的区别：
    1、常量前面没有美元符号($)；
    2、常量只能通过 define() 函数定义，而不能通过赋值语句；
    3、常量可以不用理会作用域，可以在任何地方定义和访问；
    4、常量一旦定义就不能重新定义或取消定义；
    5、常量的值只能是标量。
```

4.3.5 数据类型

- 四种标量类型

```
布尔型（boolean）:true或false。
整　型（integer）范围：2^32 或 2^64（超出自动转换为浮点型）。
浮点型（float）范围：双精度。
字符串（string）：单引号（不支持解析变量）或双引号（支持解析变量，不支持表达式）引起来的内容。
```

- 两种复合（引用）类型

```
数组型（array）
对象型（object）
```

- 两种特殊类型

```
资源型（resource）：变量可以是文件夹、一个文件、从数据库中得到的结果集等。
空 型（null）：表示一种状态，变量没有任何值。
```

4.3.6 数组

- 数组类型

```
关联式数组：下标为有意义的字符串。
索引式数组：下标为默认从 0 开始的数值。
```

- 定义数组

1）直接赋值方式定义

```php
$a[] = 10;
$a[] = 20;
$a['name'] = '张三';
$a['sex'] = '男';
```

2）使用 array()函数

```php
$b = array(10,20,30);
$b = array('name'=>'张三','sex'=>'男','age'=>28);
```

3）快捷赋值

```php
$c = [10,20,30];
$c = ['name'=>'张三', 'sex'=>'男','age'=>28];
```

- 二维数组与多维数组
- 数组的遍历

  for 遍历索引数组；foreach遍历关联数组；

  拓展：list （只用于索引数组）
- 数组的输出

```php
var_dump();
```

- 复制数组

4.3.7 运算符

```
算术运算符
    + - * / %
赋值运算符
    = += -= *= /= %= .=
比较运算符
    > >= < <= != !== == ===
逻辑运算符
    && || not and or
字符串运算符
    .
三元运算符
    ==? :
错误抑制符
    @
提升优先级符号
    ()
```

3.4 流程控制

- 分支结构

```php
if (条件){
    语句;
}else if (条件){
    语句;
```

```
} else {
    语句;
}

// 或:
if (条件):
    语句;
elseif (条件):
    语句;
else:
    语句;
endif;
```

- 情况语句（多分支）

```
switch (表达式) {
    case 值1: 语句;break;
    case 值2: 语句;break;
        ...
    default:
        语句;
}
```

- 循环结构

1）while循环

```
while(条件表达式) {
    // 若条件成立，则执行这里的循环体
    // 改变初值的条件 ;
}
```

2）do...while循环

```
do {
    // 循环体
    // 改变初值的条件 ;
} while(条件表达式);
```

3）for循环

```
for(初值;范围;递增) {
    // 循环体
}
```

3.5 函数

3.5.1 库函数

库函数分为数学函数、字符串函数、数组函数、目录函数、文件系统函数、时间函数和错误处理函数等几类。

- 数学函数

函数名	语法	说明
rand	rand(min,max)	返回一个随机整数
abs	abs(mixe)	返回数字的绝对值
min	min(array) min(mixed,mixed[,...])	返回最小值
max	max(array) max(mixed,mixed[,...])	返回最大值
round	round(float[,int[,int]])	四舍五入
floor	floor(float)	舍去小数部分取整
ceil	ceil(float)	小数部分非零返回整数部分就加1
pow	pow(mixed,mixed)	返回base的exp次方的值
sqrt	sqrt(float)	返回平方根
sin	sin(float)	返回正弦值
cos	cos(float)	返回余弦值
tan	tan(float)	返回正切值
deg2rad	deg2rad(float)	将角度转换为弧度
exp	float exp(floatarg)	返回e的arg幂次值，e为自然对数的底数，值为.718282
pi	float pi(void)	返回圆周率的值

- 字符串函数

函数	功能
trim()	去除字符串首尾空白等特殊符号或指定字符序列
strtolower()	转换为小写
strtoupper()	转换为大写
htmlspecialchars()	格式化字串中的html标签
strip_tags()	函数剥去HTML、XML以及PHP的标签
md5()	计算字符串的MD5散列值
strlen()	字节长度utf8编码西文：1字节，汉字：3字节
mb_strlen()	字符个数
substr()	字符串截取函数（中文乱码）
mb_substr()	字符串截取函数（中文不乱码）
strstr()	查找字符串的首次出现，开始向后截取全部，并返回
strpos()、stripos()	查找字符串在另一字符串中第一次、最后一次出现的位置（不区分大小写）
strrpos()、strripos()	同上（区分大小写）
str_replace()	字符串替换函数
str_repeat（）	重复一个字符串
ltrim()、trim()、rtrim()	去除左侧、两侧、右侧多余字符(默认删除空格)
explode(符号,字符串)	字符串→数组
implode(符号,数组)	数组→字符串
md5（）、sha1（）	字符串加密

- 数组函数

函数	功能
array_values()	数组中所有的值放入一个新的索引数组，并返回
array_keys()	数组中所有的键放入一个新的索引数组，并返回
array_reverse()	颠倒数组顺序
in_array(元素，数组)	检查数组中是否存在某个值
count()	计算数组中的单元数目或对象中的属性个数
array_unshift()	在数组开头插入一个或多个元素
array_push()	在数组结尾插入一个或多个元素
array_unique()	移除数组中重复的值
array_pop()	删除数组最后一个元素
array_shift()	删除数组开头的元素
array_splice(数组，索引，数量)	删除指定元素（ 删除元素后，会重排数组索引，用unse删除不会重排数组索引）
sort()	排序(升序)
rsort()	排序（降序）
array_merge()	合并一个或多个数组

数组转 JSON

函数	说明
json_encode(数组)	JSON格式编码（参数必须是utf-8编码，否则会得到空字符或者null）
json_decode(字符串，参数)	对JSON格式的字符串进行解码，参数: true:返回数组 false:返回对象

3.5.2 自定义函数

```
function 函数名称(形参列表){
    函数体;
}
```

3.6 变量的作用域

- 局部变量

  在函数内部定义，只作用于函数内部。

- 全局变量

  在函数外部定义，作用于当前整个脚本，在函数内部使用需要使用 global 关键字声明。

注意：global 只能用于引入全局变量，子函数引入父函数变量，不能使用 global 。

- 静态变量

在函数内部定义， 作用于函数内部 ，使用 static关键字声明。

3.7 正则表达式

- 元字符

元字符	说明
a-z	英文小写字母
A-Z	英文大写字母
0-9	数字
\r \n \t	非打印字符
\d	数字，相当于0-9
\D	\d取反
\w	字母数字下划线
\W	\w取反
\s	空白字符
\S	非空白字符
[]	任意匹配[]中单个字符
.	匹配任意字符（换行符除外）
{n}	匹配n次
{n,}	匹配至少n次
{n,m}	至少n次，最多m次
*	匹配0个或多个，相当于{0,}
+	匹配1个或多个，相当于{1,}
?	匹配0个或1个，相当于{0,1}
^	1、匹配正则开头 2、放在[^]，内容取反
$	匹配正则结尾
|	匹配|两侧任选其一
()	1.分组 2.子存储

注意：正则需要转义字符： ( ) { } ? * + . [ ] \ / ^ $ | 。

- 正则表达式使用格式

```
/foo bar/
#^[^0-9]$#
+php+
%[a-zA-Z0-9_-]%
```

3.8 文件处理系统

windows操作系统中常见文件类型：文件、目录、未知文件。

- 目录操作

```
mkdir(完整路径目录)：创建一个目录
rmdir(完整路径目录)：删除一个目录

opendir(完整路径目录)：打开目录
readdir(资源)：读取目录

closedir(资源)：关闭目录,释放内存
is_dir(完整路径目录)：判断是否为一个有效目录
scandir(完整路径目录)：扫描目录，返回文件数组
```

3.9 文件操作

- 文件属性函数

```
filesize(完整路径文件名)：取得一个文件的大小(字节)。
filectime(完整路径文件名)：获取文件的创建时间(create)。
filemtime(完整路径文件名)：获取文件的修改时间(modify)。
fileatime(完整路径文件名)：获取文件的访问时间(access)。
```

- 文件操作函数

```
fopen(完整路径文件名 ， 参数)：打开文件(参数：r只读 w写入 x 创建写入)。
fread(资源,长度)：读取文件 长度：字节。
fwrite(资源,内容)：写入的内容。
fclose(资源)：关闭文件(资源)，释放内存。
filegetcontents(完整路径文件名)：将整个文件读入一个字符串，相当于fopen、fread、fclose组合动作。
fileputcontents(完整路径文件名，要写入的数据，[FILE_APPEND])：将字符串写入文件，相当于fopen、fwrite、fclose组合动作FILE_APPEND：追加写入。
readfile()：将内容读入内存缓冲区。
copy(源文件 ， 目标文件)：拷贝文件(如果目标文件已存在，将会被覆盖)。
unlink(完整路径文件名)：删除文件。
```

3.10 PHP 与 Web 页面交互

3.10.1 数据提交

Web 表单提交数据有两种方式：GET方法和 POST 方法。

使用方法：

```
$_GET['变量名'];
$_POST['变量名'];
```

3.10.2 会话控制

会话控制是一种跟踪用户的通信方式。

- cookie

```
cookie是在服务器端创建，并写回到客户端浏览器。
cookie内容的存储是“键/值”对的方式，键和值都只能是字符串。
cookie一般用于记录用户的信息。
cookie不允许跨域访问。
cookie大小限制在4K以内。
```

语法格式：

```php
// 设置cookie
setcookie(key, value, 有效期);

// 获取cookie
var_dump($_COOKIE);
```

- session

session 变量存储单一用户的信息，并且对于应用程序中的所有页面都是可用的。

session 与 cookie相似，只是原来将信息存在客户端，现在保存到服务端，一般存在文件系统中。

注意：session_start()函数要写在语句之前。

```php
// 开启session
session_start();
// 注意：session_start()函数之前不能有任何输出。

// 存储数据
$_SESSION['键名'] = 值;

// 获取session信息
$_SESSION['键名'];

// 销毁session中的信息
unset($_SESSION['键名']); // unset() 函数用于释放指定的session变量

// 销毁session文件
session_destroy(); // session_destroy()函数彻底销毁session
```

- cookie 与 与 session 的区别

```
存放位置：前者存放在客户端，后者存放在服务器端；
安全性：前者不够安全，后者安全；
资源占用：前者存放在客户端，不占服务器资源，后者占服务器资源；
生命周期：前者固定时长，后者每次访问，重新计算时长；
文件大小：前者4k，后者不限制。
```

建议将登录信息等重要信息存放为 session ，其他信息如果需要保留，可以放在cookie。

3.11 面向对象

3.11.1 定义类与对象实例化

```
// 使用class关键字定义一个类，类名的首字母要求大写
class School{
    // 成员属性
    // 成员方法
}
// 使用new运算符，实例化一个类的对象
$zh = new School();
// 使用instanceof判断$hxsd是否是School的实例
$bool=$zh instanceof School;
var_dump($bool);
```

3.11.2 修饰符

访问修饰符：public、protected和private。

```
public 表示全局，类内部外部子类都可以访问；
private 表示私有的，只有本类内部可以使用；
protected 表示受保护的，只有本类或子类或父类中可以访问。

public: 公有类型
在子类中可以通过 self::属性名(或方法名) 调用public方法或属性,parent::方法名 调用父类方法
在实例中可以能过 $obj->属性名(或方法名) 来调用  public类型的方法或属性。

protected: 受保护类型
在子类中可以通过 self::属性名(或方法名) 调用protected方法或属性,parent::属性名(或方法名) 调用父类方法。
在实例中不能通过 $obj->属性名(或方法名) 来调用protected类型的方法或属性。

private: 私有类型
该类型的属性或方法只能在该类中使用
在该类的实例、子类中、子类的实例中都不能调用私有类型的属性和方法。
```

额外修饰符：

```
static：静态。可以不new实例，直接通过 类::方法 类::属性的方式调用。
final：最终。修饰的类不能被继承，修饰的方法不能被重写。
abstract：抽象。定义抽象类。
```

注意：

1）类中只能包含成员属性、成员方法、类常量；

2）类中成员属性和成员方法的位置可以互换，建议成员属性在前，成员方法在后。

3.11.3 面向对象高阶操作

```
// 自行扩展下面这部分内容
继承，多态，重载，抽象性，封装，接口，构造函数，析构函数
```

3.11.4 __autoload()

__autoload()方法并不是一个魔术方法，但是这个方法非常有用，当实例化或继承一个不存在的类，会自动调用__autoload()，同时将该类的类名作为参数。

3.12 PHP操作MySQL

3.12.1 访问数据库

```
<?php
    $server = "localhost";
    $username = "root";
    $password = "root";
    $dbname = "mydb";
    // 快速写法
    $mysqli = new mysqli($server, $username, $password, $dbname);
    // 兼容写法
    $mysqli = new mysqli($server, $username, $password);
    mysqli_select_db($mysqli, $dbname);
    // 对象写法
    $mysqli = new mysqli();
    $mysqli -> connect($server, $username, $password);
    mysqli_select_db($mysqli, $dbname);
?>
```

3.12.2 断开 MySQL 服务器

使用 mysqli_close()函数来关闭与 MySQL服务器的连接。

3.12.3 执行 SQL 语句

语法格式：

```
mysqli_query (数据库连接对象, string query[, int resultmode]);

// query: 是要执行的SQL语句
// resultmode: 是可选参数，它的默认值是 MYSQLISTORERESULT，如果需要查询的数据量很大，需要使用MYSQLIUSERESULT
```

3.12.4 解析结果集

```
free() close() free_result()：释放与结果集占用的内存
fetch_row()：以索引数组方式返回一行数据
fetch_assoc()：以关联数组方式返回一行数据
fetch_array()：以数组的方式返回一行数据
fetch_object()：以对象的方式返回一行数据
data_seek()：移动结果集中的指针到任意行
num_rows：获取结果集中行的数量
```

3.13 Laravel框架

Laravel是一款简洁、优雅的PHP Web 开发框架，它可以让你从面条一样杂乱的代码中解脱出来；它可以帮你构建一个完美的网络 APP，而且每行代码都可以简洁、富于表达力。

官网：https://www.phpcomposer.com/。

3.13.1 安装并创建项目环境

1）下载安装composer管理器

下载（下载地址：https://getcomposer.org/download/）并安装composer。

也可以(确认安装了PHP)：

```
a)php -r "copy('https://install.phpcomposer.com/installer', 'composer-setup.php');" // 下载安装脚本composer-setup.php到当前目录
b)php composer-setup.php // 执行安装过程
c)php -r "unlink('composer-setup.php');" // 删除安装脚本
```

2）安装laravel并生成项目

```
laravel 有两种安装方式：
    第一种是直接使用composer create-project。
        composer create-project --prefer-dist laravel/laravel 项目名

    第二种方式是使用laravel的安装器。
        composer require "laravel/installer"
    以后就可以各种new项目了。
        laravel new 项目名
```

项目环境搭建时，可能会出现搭建不成功的情况，可以参照下面策略进行处理：

a.http_proxy 问题

```
C:\php_project>composer create-project --prefer-dist laravel/laravel larprj
Creating a "laravel/laravel" project at "./larprj"

 [Composer\Downloader\TransportException]
 Unable to use a proxy: malformed http_proxy url
```

解决办法：使用 cmd 将代理设置为零。

```
 set http_proxy=0
```

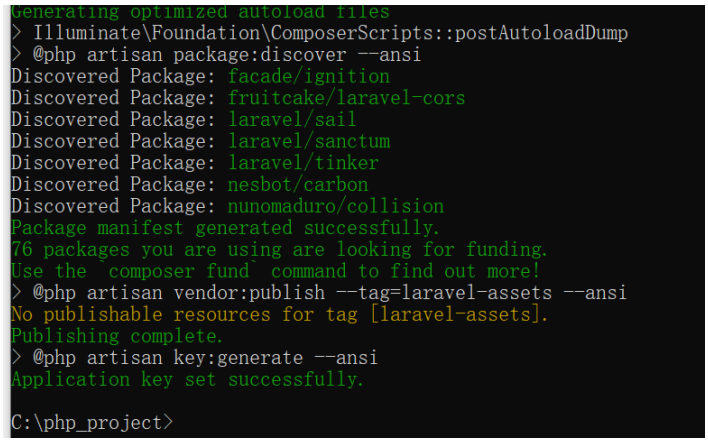
```
C:\php_project>composer create-project --prefer-dist laravel/laravel larprj
Creating a "laravel/laravel" project at "./larprj"

  [InvalidArgumentException]
  Could not find package laravel/laravel with stability stable.
```

解决办法：改为国内镜像

```
composer config -g repo.packagist composer https://packagist.org
```

b.重新搭建项目环境

```
composer create-project --prefer-dist laravel/laravel myproj
```



3）目录结构

```
  blog
    |----app/ *             // 包含应用程序的核心代码（Controller 、 Middleware 、
Model)
    |----bootstrap/         // 包含引导框架并配置自动加载的文件
    |----config/ *          // 包含应用程序所有的配置文件（app.php 、 database.php）
    |----database/ *        // 包含数据填充和迁移文件
    |----public/ *          // 包含了入口文件 index.php 、资源文件（Js/Css/Img）
    |----resoure/ *         // 包含了视图和未编译的资源文件（LESS/SASS）
    |----routes/ *          // 包含了应用的所有路由定义（web.php/api.php）
    |----storage/           // 包含编译的Blade模板
    |----tests/             // 包含自动化测试文件
    |----vendor/            // 包含了你的Composer依赖包
    |----.env*              // 环境配置文件
    |----artisan*           // Artisan是Laravel中自带的命令行工具的名称
    |----composer.json      // Composer用于安装依赖包的配置文件
    |----composer.lock      // 优先读取的依赖版本文件，可确保使用者使用相同版本依赖包
    |----package.json       // 依赖包详细描述文件
    |----phpunit.xml        // 一个面向程序员的PHP测试框架
    |----readme.md          // 使用说明书
    |----server.php         // 模拟了web server的rewrite功能，主要用于测试
    |----webpack.mix.js     // 资源文件打包配置文件
```

4）配置

在使用前，要确保以下几个内容的配置是正确的，配置有问题会影响我们的项目开发。

- 虚拟域名

框架安装完毕后要给项目定义一个虚拟域名，具体步骤如下：

i）在Laravel框架中，没有index.php文件，而是用server.php文件来替代的，所以我们需要修改Apache的http.conf文件，在<IfModule dir_module>下面添加server.php即可。具体修改如下图所示：

```
..
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
<IfModule dir_module>
    DirectoryIndex index.php index.pl index.cgi index.asp index.shtml index.html index.htm \
                   default.php default.pl default.cgi default.asp default.shtml default.html default.htm \
                   home.php home.pl home.cgi home.asp home.shtml home.html home.htm server.php
</IfModule>

#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<Files ".ht*">
    Require all denied
</Files>
```

手动添加

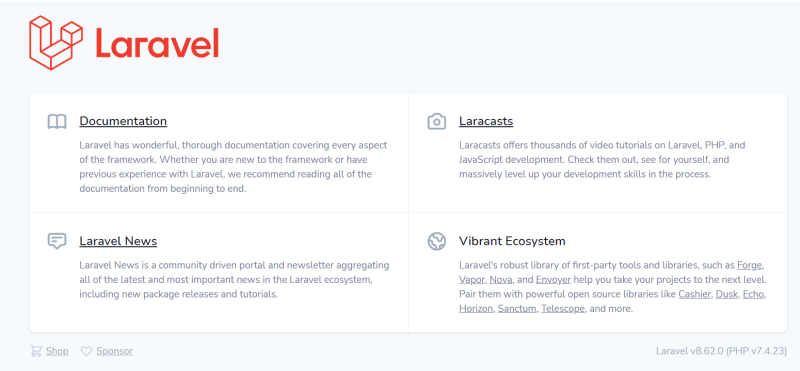
ii）打开C:\xampp\apache\conf\extra目录下的httpd-vhosts.conf文件，修改内容如下：

```
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost container.
# The first VirtualHost section is used for all requests that do not
# match a ##ServerName or ##ServerAlias in any <VirtualHost> block.
#
##<VirtualHost *:80>
    ##ServerAdmin webmaster@dummy-host.example.com
    ##DocumentRoot "C:/xampp/htdocs/dummy-host.example.com"
    ##ServerName dummy-host.example.com
    ##ServerAlias www.dummy-host.example.com
    ##ErrorLog "logs/dummy-host.example.com-error.log"
    ##CustomLog "logs/dummy-host.example.com-access.log" common
##</VirtualHost>

<VirtualHost *:80>
    ServerAdmin webmaster@dummy-host2.example.com
    DocumentRoot "e:/phpstudy/blog"
    ServerName www.lovehsp.com
    ErrorLog "logs/dummy-host2.example.com-error.log"
    CustomLog "logs/dummy-host2.example.com-access.log" common
</VirtualHost>
```

iii）打开C:\Windows\System32\drivers\etc目录下的hosts文件，添加自己的域名。如下图所示：

```
# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
127.0.0.1  localhost
127.0.0.1  bandicam.com
127.0.0.1  cert.bandicam.com
127.0.0.1  bandisoft.com
127.0.0.1  ssl.bandisoft.com
127.0.0.1  www.bandicam.com
127.0.0.1  www.lovehsp.com
# XMind Mind-Mapper
0.0.0.0 xmind.net
0.0.0.0 www.xmind.net
```

iv）配置成功后，重启服务，直接访问http://www.lovehsp.com/，即可出现以下界面，则安装成功！



3.13.2 路由

- 简单路由

  路由文件在 Routes/web.php。

```
// 这是一个测试路由，当我们访问www.lovehsp.com/public/test时，会在页面输出Hello World
Route::get('/test', function () {
    return 'Hello World';
});
```

- 请求方式

  路由指定了请求方式，必须由该方式进行访问，否则页面会提示未找到。

```
Route::get($uri, $callback);    // GET方式请求(常用)
Route::post($uri, $callback);   // POST方式请求(常用)
Route::put($uri, $callback);    // 伪造PUT方法请求
Route::patch($uri, $callback);  // 伪造PATCH方法请求
Route::delete($uri, $callback); // 伪造DELETE方法请求
```

- 路由参数

单个参数

路由中的参数，可以在闭包程序中直接使用，也可以在控制器中直接获取。

```
// 当我们请求www.lovehsp.com/public/user/666时，会将666的参数传递到当前路由
Route::get('user/{id}', function ($id) {
    return '当前用户的id是：'.$id;
});
```

多个参数

参数与参数之间的符号分割没有明确的限定，通常为/或-。

```
// 当我们请求www.lovehsp.com/public/user/5-info-1时，可以访问到如下路由的内容
Route::get('user/{id}-{service}-{page}', function ($id, $service, $page) {
    // 输出参数信息
    return '您正在查看用户id为：'.$id.'的'.$service.'服务的第'.$page.'页的内容！';
});
```

可选参数

某些情况下，有的参数是可有可无的，我们可以使用?问好来标注改参数是可有可无的。

```
// 当我们请求www.lovehsp.com/public/user/TeacherHou或www.lovehsp.com/public/user时
Route::get('user/{name?}', function ($name = null) {
    if($name != null){
        return "这是TeacherHou的信息!";
    }else{
        return "这里是用户的列表信息!";
    }
});
```

- 命名路由

某些路由的可能会非常非常长，例如 users/list/info/{id}-{service}-{page}...，若想要在跳转到或获取该路由时，可能不太方便，因此，Laravel 给我们准备了命名路由。

```
// 直接在路由后方追加name方法，并定义当前路由的别名
Route::get('/article', function () {
    return view('article');
})->name('art.php');
```

这样一来，若我们想要进行跳转，我们可以这样写：

```
Route::get('/jump', function () {
    $url = route('art.php');// 生成URL
    return redirect()->route('art.php');    // 生成重定向
});
```

- 路由重定向

eg1:

```
Route::get('/test', function () {
    return redirect('/user/007');
});
```

3.14.3 CSRF跨域请求伪造

跨站请求伪造是一种恶意的攻击，它凭借已通过身份验证的用户身份来运行未经过授权的命令。我们要做的就是防止有人恶意攻击我们的网站。因此需要借助Laravel 帮我们生成的这张 CSRF「令牌」。

- 基本使用

```
<!-- 通过表单发送post请求时，需要添加令牌验证 -->
<form method="POST" action="/profile">
    {{ csrf_field() }}
    ...
</form>
```

- JS 的 的ajax请求

需要添加的内容有两个地方，一个在 head 头部，一个在 post 请求内部。

```
<!-- head头添加token请求 -->
<meta name="csrf-token" content="{{ csrf_token() }}">
<!-- 定义发送post请求的按钮 -->
```

```
<button id="btn">单击发送ajax的post请求</button>
<!-- 定义发送post请求的js代码(使用jQuery) -->
<script>
    $('#btn').click(function(){
        // 单击事件中添加如下代码
        $.ajaxSetup({
            headers: {
                'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
            }
        });
        // 发送 post 请求给指定路由
        $.post('/testAjax',{},function(data){
            alert(data);
        });
    });
</script>
```

3.13.4 控制器

我们之前的都是将所有的请求处理逻辑放在路由文件的闭包函数中，显得很臃肿，其实我们可以使用控制器类组织管理相对复杂的业务逻辑处理。控制器用于将相关的 HTTP 请求封装到一个类中进行处理，这些控制器类存放在 `app/Http/Controllers` 目录下。

- 定义控制器
    - 命令行创建控制器

      ```
      php artisan make:controller UserController(控制器名称)
      ```

    - 手动创建

      在app/Http/Controllers文件夹下创建控制器.php文件即可。

      注意： 所有的 Laravel 控制器应该继承自 Laravel 自带的控制器基类 `App\Http\Controllers\Controller`

- 在控制器中创建方法

  例如我们创建一个test()函数和profile函数

  ```
  <?php
      namespace App\Http\Controllers;
      use Illuminate\Http\Request;//加载Request
      class UserController extends Controller{
          public function test(){
              return "我是一个测试控制器函数";
          }
          public function profile(Request $request,$id){
              echo "通过控制器访问的当前id是：".$id;
          }
      }
  ```

- 在路由中访问

  如果一个请求匹配上面的路由 URI，`UserController` 的 `test` 和 `profile` 方法就会被执行，当然，路由参数也会被传递给这个方法。

  routes/web.php文件定义路由。

```php
<?php
    use Illuminate\Support\Facades\Route;
    // 引入控制器
    use App\Http\Controllers\UserController;

    // v8以下使用控制器
      Route::get("/test","UserController@test");
      Route::get("/prof/{id}","UserController@prof");
    // v8以上使用控制器
    Route::get("/test",[UserController::class,'test']);
    Route::get("/prof/{id}",[UserController::class,'profile']);
```

3.13.4 获取请求

路由和控制器我们均已经经过了系统的学习，本节的课程主要练习 HTTP 的请求，以及在控制器当中如何进行获取，很重要的一个操作对象是 $request 请求对象。

```php
<?php
    // 当前控制器的命名空间
    namespace App\Http\Controllers;
    // 用于获取 HTTP 请求的处理类导入
    use Illuminate\Http\Request;
    class UsersController extends Controller
    {
        // 在指定的方法形参列表中，使用类型约束的方法，传入请求对象
        public function store(Request $request)
        {
            // 使用 request 请求对象进行数据的获取，而非 $_POST 的原生写法
            $name = $request -> input('name');
            // 你还可以使用另外一种方式获取
            $name = $request -> name;
        }
    }
```

- 基本应用

请求路径

```php
// 若请求的目标地址是：http://test.com/users/show，则path会获取users/show
$uri = $request -> path();
```

检测请求路径

```php
//  若请求的路径是：  http://test.com/admin/users ，则下面实例为真
if ($request -> is('admin/*')) {
    // 此处代码略
}
```

获取完整 URL

```
//  没有请求参数 ...
$url = $request -> url();
//  有请求参数 ...
$url = $request -> fullUrl();
```

获取请求方法

```
// 若此时请求到该方法的方式为 GET ，则输出 GET
$method = $request -> method();
// 判断当前方法是通过哪种请求方式请求的
if ($request -> isMethod('post')) {
    //
}
```

3.13.5 获取请求数据

获取所有数据

```
// 以数组形式存储
$res = $request -> all();
```

获取指定数据

```
// 会获取到 form 表单中表单项 name 的值
$name = $request -> input('name');
// 指定默认值 ( 若表单提交信息中没有 name 相对应的值，则采用默认值 )
$name = $request -> input('name', 'Sally');
```

3.13.6 Blade模板引擎

- 区块占位

首先需要定义一个公共模板，我们这里在resources->views->layouts目录中生成了master.blade.php 模板。

```
<!-- 文件保存于resources/views/layouts/master.blade.php -->
<html>
  <head>
    <title>标题-@yield('title')</title>
  </head>
  <body>
    @section('sidebar')
      这是父级模板的边栏。
    @show

    <div class="container">
      @yield('content')
    </div>
  </body>
</html>
```

@yield是不可扩展的， @section既可以被替代，又可以被扩展。

@section 定义一个片段。如果使用了@parent关键字，父模板中的内容会被保留，并且添加新的内容，输出的结果是“默认的内容 扩展的内容”。

@yield() 占位符 。不能被扩展，所以即使加上了@parent也不起作用，输出的内容只有“新的标题”，替换了“默认的标题”。

- 继承布局

公共模板定义好了，那意味着你的框架已经打好了，接下来就是让其他模块来继承公共模块了，然后将需要进行内容替换的位置进行替换即可。

```
<!-- 文件保存于 resources/views/child.blade.php -->
<!-- 继承公共模板内容 -->
@extends('layouts.master')
<!-- 替换单行占位的内容 -->
@section('title', '我的首页')
<!-- 替换区块占位的 sidebar 内容 -->
@section('sidebar')
  @parent   <!-- 这里的parent代表依然要延续使用公共框架的内容 -->
  <p>这是添加到父级模板的侧边栏。</p>
@endsection

@section('content')
  <p>这是子模板的页面。</p>
@endsection
```

在上面的例子作用 `sidebar` 挂件利用 `@parent` 指令来追加布局中的 sidebar 部分的内容，如果不使用则会覆盖掉布局中的这部分。`@parent` 指令会在视图被渲染时替换为布局中的内容。

Blade 视图可以像原生 PHP 视图一样使用全局帮助函数 `view` 来返回渲染后的内容：

```
Route::get('blade', function () {
  return view('child');
});
```

- 控制结构

if 语句

```
@if (count($records) === 1)
    我有一条记录!
@elseif (count($records) > 1)
    我有多条记录!
@else
    我没有任何记录!
@endif
```

switch 语句

```
@switch($i)
    @case(1)
        First case...
        @break
    @case(2)
        Second case...
        @break
    @default
        Default case...
@endswitch
```

循环结构

```
@for ($i = 0; $i < 10; $i++)
    目前的值为 {{ $i }}
@endfor

@foreach ($users as $user)
    <p>此用户为 {{ $user -> id }}</p>
@endforeach

@while (true)
    <p>死循环了。</p>
@endwhile
```

3.14.7 案例：在线答题系统

- 需求分析

  使用Laravel框架编写一个简单的在线答题系统。

  - 试题共4道数学题，每题3 个选项，都为单选题。每题25 分，共100 分。目内容、选项和答案如下。

    第1 题： 1 0 + 4 = ？ 选项： A.12 B.14 C.16 答案：B

    第2 题： 2 0 - 9 = ？ 选项： A.7 B.13 C.11 答案：C

    第3 题： 7 x 3 = ？ 选项： A.21 B.24 C.25 答案：A

    第4题： 8 / 2 = ？ 选项： A.10 B.2 C.4 答案：C

  - 每做完一题，单击“下一题” 按钮，提交当前题目答案，并显示下一题的内容，如图所示。

- 在出现最后一题时，按钮显示为“提交”,显示答对的题数和得分情况，总分数为100分，如图所示.

3.14 ThinkPHP框架

ThinkPHP是一个免费开源的，快速、简单的面向对象的轻量级PHP开发框架，是为了敏捷WEB应用开发和简化企业应用开发而诞生的。ThinkPHP从诞生以来一直秉承简洁实用的设计原则，在保持出色的性能和至简的代码的同时，也注重易用性。

3.14.1 项目环境搭建

1.Composer安装（方法同Laravel，如已安装，此步略过）。

2.项目环境搭建

```
composer create-project topthink/think=6.0.x-dev 项目名
```

注意：ThinkPHP 6.0需要PHP7.1.0以上的版本支持。

3.项目启动

（1）第一种在浏览器中输入地址：

```
http://localhost/public/
```

（2）第二种

通过cmd进入“项目名”目录中，运行命令：php think run

打开浏览器输入：127.0.0.1:8000或localhost:8000

如果浏览器输出如下图所示，表示环境搭建成功。

3.14.2 项目目录结构

```
www                     WEB部署目录（或者子目录）
├─app                   应用目录
│  ├─controller         控制器目录
│  ├─model              模型目录
│  ├─ ...               更多类库目录
│  │
│  ├─common.php         公共函数文件
│  └─event.php          事件定义文件
│
├─config                配置目录
│  ├─app.php            应用配置
│  ├─cache.php          缓存配置
│  ├─console.php        控制台配置
│  ├─cookie.php         Cookie配置
│  ├─database.php       数据库配置
│  ├─filesystem.php     文件磁盘配置
│  ├─lang.php           多语言配置
│  ├─log.php            日志配置
│  ├─middleware.php     中间件配置
│  ├─route.php          URL和路由配置
│  ├─session.php        Session配置
│  ├─trace.php          Trace配置
│  └─view.php           视图配置
│
├─view                  视图目录
├─route                 路由定义目录
│  ├─route.php          路由定义文件
│  └─ ...
│
├─public                WEB目录（对外访问目录）
│  ├─index.php          入口文件
│  ├─router.php         快速测试文件
│  └─.htaccess          用于apache的重写
│
├─extend                扩展类库目录
```

```
├─runtime                应用的运行时目录（可写，可定制）
├─vendor                 Composer类库目录
├─.example.env           环境变量示例文件
├─composer.json          composer 定义文件
├─LICENSE.txt            授权说明文件
├─README.md              README 文件
├─think                  命令行入口文件
```

3.14.3 配置路由和控制器

- 打开config/route.php文件，修改控制器后缀

```
'controller_suffix'   => true,
```

- 修改app/Index.php控制器

  将Index.php文件改名为IndexController.php

  将类名Index改为IndexController

  注意：删除原有方法，创建 index、info、getAllArt、getArtById四个方法

```
<?php
namespace app\controller;
use app\BaseController;
class IndexController extends BaseController
{
    public function index(){
        return '首页';
    }
    public function info(){
        return '详情页';
    }
    public function getAllArt(){
        return '文章列表数据';
    }
    public function getArtById($id) {
        return $id;
    }
}
```

- 打开route/app.php为控制器的每个方法配置路由

```
<?php
    use think\facade\Route;
    Route::get('/', '/Index/index');
    Route::get('/info', '/Index/info');
    ……
```

- 根据路由访问四个地址

```
http://127.0.0.1:8000
http://127.0.0.1:8000/info
……
```

- 或者使用第一种访问路径的方法：

```
http://localhost/defraudtest/public/
http://localhost/defraudtest/public/info
……
```

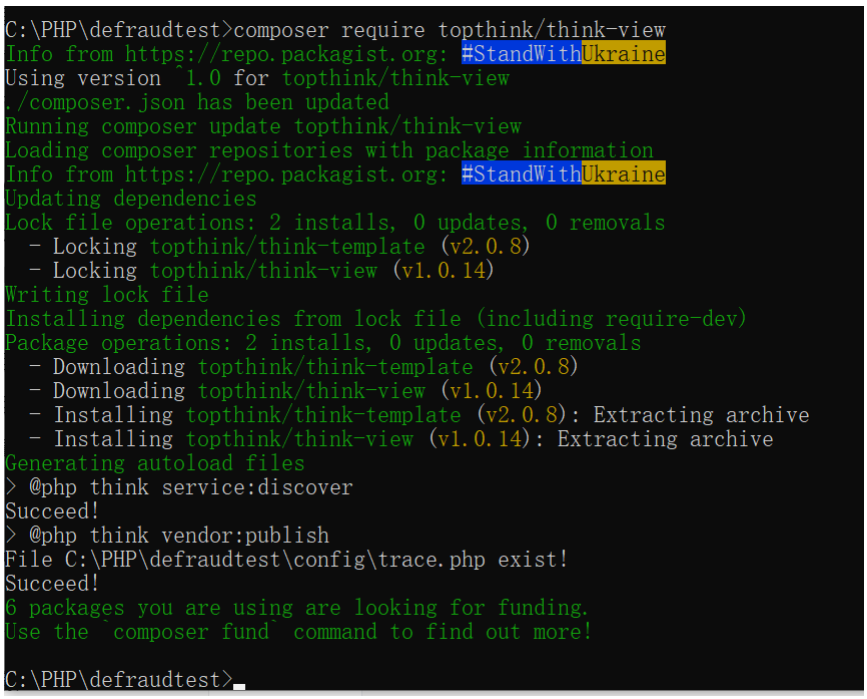
3.14.4 模板引擎和静态页面设计

（1）模板引擎

- 安装ThinkPHP自带的模板引擎，打开cmd进入工程根目录，执行安装模板引擎命令

```
composer require topthink/think-view
```

安装过程如图



- 安装完成后，在IndexController.php控制器中引入View类

```
<?php
    namespace app\controller;
    use app\BaseController;
    use think\facade\View;
    class IndexController extends BaseController {
        ……
    }
```

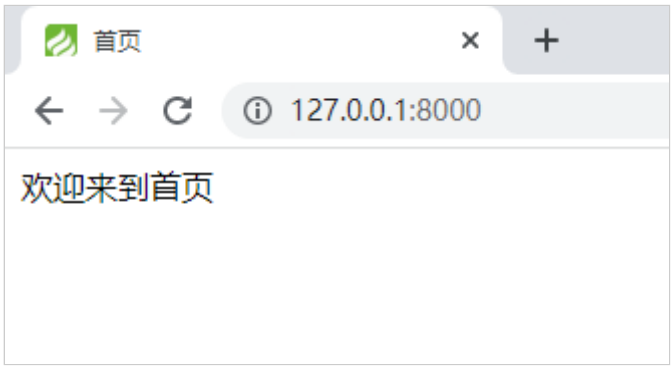
- 在view下创建Index目录，再在Index目录中创建index.html文件

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>首页</title>
    </head>
    <body>
        欢迎来到首页
    </body>
</html>
```

- 在controller/IndexController方法中渲染刚刚创建好的index.html文件

```php
<?php
    namespace app\controller;
    use app\BaseController;
    use think\facade\View;
    class IndexController extends BaseController {
        public function index() {
            // return '首页';
            return View::fetch();
        }
    }
```

使用谷歌浏览器访问:http://127.0.0.1:8000



(2) 页面设计

- 在public/static目录下创建css、js、images等目录（也可以引入第三方UI框架，如Bootstrap等）
- 在config/view.php文件中配置所需的路径

```php
    ......
    // 标签库标签结束标记
    'taglib_end'    => '}',
    // 添加内容
    // 配置静态资源路径
    'tpl_replace_string' => [
        '__STATIC__'  =>  '/static',
        '__JS__'      =>  '/static/js',
        '__CSS__'     =>  '/static/css',
        '__IMG__'    =>  '/static/images',
    ]
]
```

- 在index.html中引入

  如果有数据需要请求直接使用Ajax调用即可

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="__CSS__/……">
    <link rel="stylesheet" href="__STATIC__/……">
    <script src="__JS__/……"></script>
</head>
<body>
</body>
```

3.14.5 数据库配置和模型

(1) 配置数据库

- 修改.example.env文件为.env
- 打开.env文件配置数据库信息

```
[DATABASE]
TYPE = mysql
HOSTNAME = 127.0.0.1
DATABASE = test
USERNAME = username
PASSWORD = password
HOSTPORT = 3306
CHARSET = utf8
DEBUG = true
```

(2) 模型

- 模型会自动对应数据表，模型类的命名规则是除去表前缀的数据表名称，采用驼峰法命名，并且首字母大写，如：
    - tp_user => User
    - tp_user_type => UserType
- 通过建立`Model`获取数据的方法是在`app`下建立`model`文件夹，创建文件名与数据库真实表名具有一致或映射（比如前缀）关系的模型文件，如数据库中表名是**tp_article**可创建**Article.php**并使其继承`think\Model`，见代码如下：

```
<?php
    namespace app\model;
    use think\Model;
    class Article extends Model {
        // 明确指示本类 model 映射表名为 tp_article;
        // 也可以设置表名：protected $table = "tp_article";
        protected $name = 'tp_article';
        ……
    }
?>
```

注意：model 的数据字段和表字段是对应的，默认会自动获取全部，包括字段名和字段类型。自动获取会导致增加一次查询，因此在模型中进行主动配置字段信息会减少至少一次IO，通过设置`protected $schema`明确定义字段信息

- 修改Index控制器中的getArtById和getAllArt方法，调用Article模型，查询数据

```
<?php
    namespace app\controller;
    use app\BaseController;
    use think\facade\View;
    use app\model\ArticleModel;
    class IndexController extends BaseController {
        public function index() {
            // return '首页';
            return View::fetch();
        }

        public function info() {
            // return '详情页';
            return View::fetch();
        }
        public function getAllArt() {
            ……
        }
        public function getArtById($id) {
            ……
        }
    }

?>
```

在谷歌浏览器中输入http://127.0.0.1:8000/article路径，测试数据.

对表中操作：

- 新增：Db::table("表名")->insert(数据);
- 删除：$result = Db::table("表名")->delete(删除条件);

  无条件删除所有数据：Db::name("表名")->delete(true);
- 更新：Db::table("表名")->where(条件)->update(新数据);
- 按条件查询：Db::table("表名")->where(条件)->find();
- 查询某个字段的值可以用:Db::table("表名")->where(条件)->value(字段1，字段2...);
- 查询全部：Db::table("表名")->select();