Web开发

静态WEB:Applet作为最早的客户端实现WEB技术，但是由于Applet不能访问资源，静态WEB基本上都是不能访问数据库、文件资源。这个时候所有的动态效果都是在客户端完成的，静态WEB基本上是靠普通的HTML网页实现的，即使有了Applet，也只是在HTML之中加入了若干代码而已，本身没有太大改变。

除了Applet，JavaScript进行客户端的动态效果实现。

动态WEB:最大的一个特点，具备了交互性，而且在整个动态WEB之中，客户端所做的工作非常简单，就是一个普通的浏览器，但是在服务器端却有了很大的改变，服务器端首先通过一个WEB服务器插件，是用来区分是动态请求还是静态请求。

如果是静态请求，那么先将内同交给WEB服务器，并调用文件系统。

如果是动态请求，则将进入到一个WEB容器然后进行代码的拼凑的工作。

在整个操作流程，实际上动态WEB和静态WEB最大的区别：动态WEB可以进行数据库的连接，而静态WEB无法连接数据，现有的大部分程序都是围绕数据库进行的。

动态WEB实现方式：

CGI:Common Gateway Interface，可以使用各种语言实现，使用多进程形式进行处理，所以传统的CGI程序实际上是一种多进程的处理语言，所以性能很低。

PHP:本身在Linux下发展起来，由于没有大公司的支持，现在只作为个人开发，或者小型项目使用。技术含量不高。PHP比较简单。Apache + MySQL。

ASP:比较流行，但是基本用的很少，本身性能很差，和PHP出于统一层次。

ASP.NET:新的开发平台，性能够高，但是受到windows平台的限制。

JSP:不受平台限制，可以支持多种操作系统，开发平台，数据库，而且最主要的是Java是多线程的，所以JSP性能很高。

企业开发：

操作系统（Linux,Windows,UNIX）🡪数据库（DB2,Oracle,MySQL）🡪中间件（Tomcat,JBoss,WebLogic）🡪开发平台（Java EE）

htm和html区别在于，dos系统只能识别后缀字母为奇数个，htm为了兼容dos系统。

HTML学习笔记

<!DOCTYPE html>

里面可以dtd用来说明这个html版本。

<meta http-equiv=”keyword” content = “DELL”>

用来指定关键字

html基本结构

<元素 属性=”属性值”>内容（可以无内容）</元素>

如果元素没有内容，可以简化写成如下所示：

<元素 属性=”属性值” />

文件的指定可以只用文件夹/文件名来指定，文件的上溯（找到本文件的上层文件）可以使用../文件名（上溯多层文件夹使用../../文件名来完成）。注意html中使用/而非\表示文件目录，需区别于windows操作系统的文件目录表示方式。

元素分类：

分类方式1：

块元素：单独一块区域，例如<h1></h1><blockquote></ blockquote >等。

内联元素：只显示在段落的内容中，例如：<a></a><q></q><img>，内联元素必须包含在块元素里面。

分类方式2：

一般元素：<a></a>

空元素：<br><imag>，不必出现起始。

常用属性列表：

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名称 | 属性涵义 | 使用方法 | 备注 |
| <li></li> | 列表 | 通常外围必须与<ol></ol>（表示有序）或者<ul></ul>（无序列表）配套使用，且<ol></ol>和<ul></ul>可以嵌套使用。 |  |
| <a></a> | 链接 | <a href="subpage.html">this is my first html file</a> | 后面可以跟 target（是否需要新打开网页）href,title（鼠标放置在超链接上面的名字）等属性，或者#id名称（定位到网页特定位置） |
| <img> | 图片 | 后面可以跟上width height(限制图片的宽高比)，alt(图片不显示时，可以使用文字替代图片) | 需要注意的是img是内联元素不是块元素 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[www.baidu.com是网站名，而baidu.com](http://www.baidu.com是网站名，而baidu.com)是域名，拥有baidu.com之后即可创建多个基于baidu.com的网站。

html中的图片常见为两种格式，一种为jpg（1600万种颜色，有损压缩，背景图片不可以透明）另外一种为gif（256种颜色，无损压缩，背景图片可以透明）

最常用的字符实体:

|  |  |  |  |
| --- | --- | --- | --- |
| 显示结果 | 描述 | 实体名 | 实体号 |
|  | 不可拆分的空格 | &nbsp; | &#160; |
| < | 小于 | &lt; | &#60; |
| > | 大于 | &gt; | &#62; |
| & | and符号 | &amp; | &#38; |
| " | 引号 | &quot; | &#34; |
| ' | 单引号 |  | &#39; |

其他一些常用的字符实体

|  |  |  |  |
| --- | --- | --- | --- |
| 显示结果 | 描述 | 实体名 | 实体号 |
| ¢ | 分 | &cent; | &#162; |
| £ | 英镑 | &pound; | &#163; |
| ¥ | 人民币元 | &yen; | &#165; |
| § | 章节 | &sect; | &#167; |
| © | 版权 | &copy; | &#169; |
| ® | 注册 | &reg; | &#174; |
| × | 乘号 | &times; | &#215; |
| ÷ | 除号 | &divide; | &#247; |

<a> 标签定义超链接，它用于从一个页面连接到另一个页面。

常用的target两个属性，一个是\_blank，新打开一个页面，\_self在原页面跳转

<a href="http://www.baidu.com" target="\_blank">百度</a>

<!—跳转的页面可以是本地，也可以是网上其它地址-->

<a href="mailto:sc0303@126.com">发送邮件</a> <!—发送邮件-->

<img src="logo\_white.png" width="300"/><!—显示图片-->

<!—表格-->

<table border="1" width="300px" bgcolor="aqua" align="center">

<tr><td align="center">1</td><td>2</td><td>3</td></tr>

</table>

**html表单**

html表单元素主要是让用户输入元素，并提交给服务器。

基本语法：

<form action=”url”method=”提交方法（get， post），默认是get”

表单元素的格式，例如：

<input type = text(文本框) name = username/>

<input type = password(密码框)/>

<input type = hidden(隐藏域)/>

<input type = checkbox(复选框)/>

<input type = radio(单选框)/>

<input type = submit(提交按钮)/>

<input type = reset(重置按钮)/>

<input type = image(图片按钮)/>

</form>

<h1>登陆页面</h1>

<form action="ok.html" method="post">

<!—post和get的一个区别在于post属于传输的时候url不显示用户名，密码-->

用户名：<input type="text" name = "username"/>

<br/>

密&nbsp;&nbsp;码：<input type="password" name = "username"/>

<br/>

<input type="submit" value = "登陆"/>

<input type="reset" value = "重新填写"/>

</form>

div+css

div用来为HTML文档内大块内容提供结构和元素，div用来存放内容（图片、文字）的容器。div和span最大区别为，一个是块元素，一个是内联元素。.0

css：Cascading Style Sheets（层叠样式表单），用来表示HTML或者XML文件样式的语言，用来指定div容器中的图片、文字的样式（大小，颜色，背景，位置）。

和传统是用table方式相比，div+css简化网页代码，内容和样式分离。div放内容，css指定样式，内容和样式完全分离。

<!—不同style之间使用;分开-->

<div class="span12" style="margin-top:15px;font-size:30px">

css通常有两种形式，一种为内部css，一种为外部css：

<style type = "text/css">

.style1{

font-size: 20px;

color:red;

font-weight: bold;

}

</style>

上述的css写入在<head></head>中，为内部css中。

ccs可以继承，如果在body里面设置了字体，则所有包括在body里面的<a></a><q></q>都会继承该字体。但是子元素也可以定义自己的字体颜色，从而覆盖继承来的字体颜色。

ccs中使用/\*作为注释\*/，html需要使用<!—作为注释-->

css中的三种选择器

|  |  |  |
| --- | --- | --- |
| 选择器名称 | 定义语法 | 引用语法 |
| 类选择器 | .style1{}  可以通过p.style1{}来指定style属于哪个元素。 | <span class="style1"> </span> |
| id选择器 | #style1{}可以通过p#style1{}来指定style属于哪个元素。 | <span id="style2"> </span> |
| html选择器 | body（button,etc）{}  或者直接写成body{} /\*不指定body中哪部分内容\*/ | 只要在body内的内同都会改变样式 |
| 伪类 | a:link{}//未被访问的链接  a:visited{}//访问过的链接  a:hover{}//鼠标悬停链接 |  |
| 三种类型选择器的优先级为id选择器>类选择器>html选择器，注意可以同时使用多个选择器，所以会存在一个优先级的问题。**一个元素最多有一个id选择器，但是类选择器可以有多个。**<span class ="style1 style2" id="style3"> </span>，**如果多个class选择器发生冲突，以在css文件中后定义的class为主。所以在开发的过程中需要首先选择类选择器，这是因为如果后期需要更改样式的话，可以选择高优先级的id选择器来进行更改。**  **一个元素不能有多个id选择器，且同一个页面中不能有一个以上的元素用同一个id，而一个元素可以使用多个类选择器，且类选择器在同一个页面中出现多次。**  **同时如果有多个选择器同时定义了颜色属性，通常定义规则更具体的会胜出，如果没有明显的胜出者，即多个选择器定义具体程度相同，则选择CCS文件中更靠后的选择器（详见head firstP323）,且在html文件中引用的css文件，越靠后，优先级越大（详见head firstP421）** | | |

/\*使用通配符选择器设置页边距，margin后面的顺序是以上开始的顺时针旋转，即上、右、下、左，如果提供少于三个参数、需要查询文档来确定分别对应哪个位置\*/

\*{

margin: 0px 0px 0px 0px;

}

#style2 span{

font-style: italic;

}

<span id="style2">hello<span>hello world</span> world1</span>

**span为父子选择器，即span相当于style2的孩子，继承了style2的特点，另外附件了一些特点，而且可以多重继承，即#style2 span span,父子选择器适应于id选择器和类选择器。**

**对于多个类选择器来说，如果它们之间有相同点，则共同部分可以拿出来作为写成一个选择器如下：**

.style1, .style2, style{

}

**不同的类选择器之间使用逗号隔开。**

a:link{

}

a:hover{

}

a:visited{

}

行内元素（inline element）:特点是只占内容的宽度，默认不会换行<span></span>

块元素（block element）：特点是不管内容多少，占满整个，自动换行。<div></div> <p></p>

行内元素不可以容纳块元素，块元素可以容纳行内元素。

<span style="display:block" class = style1></span>

span按照块元素显示，通常style="display:block"写在css文件中。同样块元素也可以按照行内元素显示，style="display:inline"

**serif 以及sans serif**：

serif 是有衬线字体，意思是在字的笔画开始、结束的地方有额外的装饰，而且笔画的粗细会有所不同。相反的，sans serif 就没有这些额外的装饰，而且笔画的粗细差不多。

serif适合做正文，sans serif适合做标题。

对于中文来说，宋体属于serif 字体，而黑体、幼圆、雅黑则属于sans serif 字体。

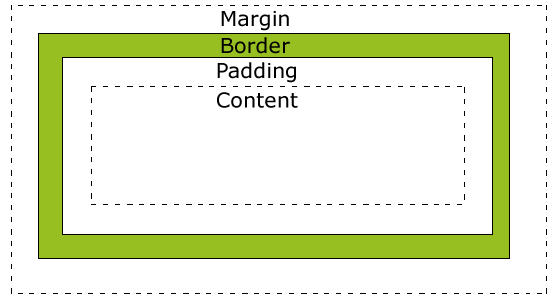
而英文，常用的 serif 字体为Times New Roman，常用的 sans serif 字体为Arial，Tahoma。

CCS中可以设置多个字体，浏览器会依据先后顺序插叙用户操作系统中是否有该字体。（head first P347）

**标准流：**在网页布局中，写在前面的元素，出现在前面，写在后面的元素，出现在后面，这是默认的布局，称为标准流。

**非标准流：**脱离元素本身的位置。

**盒子模型：**



每一个html元素都可以看做成一个盒子，两个元素之间的距离称之为margin，如果div1里面放置个div2，对于div2来说，两者之间的距离叫margin，对于div1来说，两者之间的距离叫做padding（尽量避免使用padding，padding会撑大盒子）。

**浮动:**浮动分左浮动、右浮动、清除浮动。浮动主要解决div定位的问题。如果一行宽度不够排列所有的div，则自动换行。

**语法：**float：right;//让元素尽量右移动，直到移动到上层盒子的最右侧。

clean:right; //清除浮动

**定位：**定位分为四种定位，static（默认，不认left和right参数）, relative, absolute, fixed。

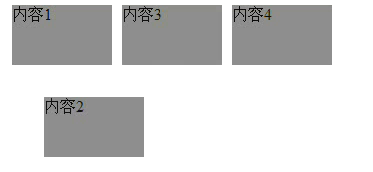
**语法：**position:relative;

left:40px;//相对于原来的位置进行移动，这就是relative定位。

top:100px;//注意区分上下左右。



相对定位，原来的位置还保留



绝对定位，让出原来的位置

相对定位的上下左右是相对于原先在的位置，而绝对定位是相对于上一层**非标准流**盒子的位置。fixed定位和absolute定位类似，区别在于fixed一直是相对于body定位的。

inherit定位：继承父亲的定位。

javascript

每一条JS语句建议以;结尾。

**数据类型：**

**Number**

JavaScript不区分整数和浮点数，统一用Number表示。

**字符串**

字符串是以单引号'或双引号"括起来的任意文本，比如'abc'，"xyz"等等。

**布尔值**

布尔值和布尔代数的表示完全一致，一个布尔值只有true、false两种值

&&与运算，||或运算，！非运算

**比较运算**

**第一种是==比较，它会自动转换数据类型再比较，很多时候，会得到非常诡异的结果；**

**第二种是===比较，它不会自动转换数据类型，如果数据类型不一致，返回false，如果一致，再比较。**

**由于JavaScript这个设计缺陷，不要使用==比较，始终坚持使用===比较。**

**另一个例外是NaN这个特殊的Number与所有其他值都不相等，包括它自己：**

**NaN === NaN; // false**

唯一能判断NaN的方法是通过isNaN()函数：

**isNaN(NaN); // true**

最后要注意浮点数的相等比较：

1 / 3 === (1 - 2 / 3); // false

**这不是JavaScript的设计缺陷。浮点数在运算过程中会产生误差，因为计算机无法精确表示无限循环小数。要比较两个浮点数是否相等，只能计算它们之差的绝对值，看是否小于某个阈值：**

Math.abs(1 / 3 - (1 - 2 / 3)) < 0.0000001; // true

JavaScript的设计者希望用null表示一个空的值，而undefined表示值未定义。事实证明，这并没有什么卵用，区分两者的意义不大。大多数情况下，我们都应该用null。undefined仅仅在判断函数参数是否传递的情况下有用。

**数组（两种表示方法）**

[1, 2, 3.14, 'Hello', null, true];

new Array(1, 2, 3); // 创建了数组[1, 2, 3]

对象：JavaScript的对象是由一组由键-值组成的**无需集合：**

var person = {

name: 'Bob',

age: 20,

tags: ['js', 'web', 'mobile'],

city: 'Beijing',

zipcode: null

};

通过person.name, person.zipcode获取对象的属性.

**变量：通过var进行声明。**

**在JavaScript中，使用等号=对变量进行赋值，同一个变量可以反复赋值，而且可以是不同类型的变量，但是var声明只能声明一次。**

JavaScript在设计之初，为了方便初学者学习，并不强制要求用var申明变量。这个设计错误带来了严重的后果：如果一个变量没有通过var申明就被使用，那么该变量就自动被申明为全局变量：

为了修补JavaScript这一严重设计缺陷，ECMA在后续规范中推出了strict模式，在strict模式下运行的JavaScript代码，强制通过var申明变量，未使用var申明变量就使用的，将导致运行错误。

启用strict模式的方法是在JavaScript代码的第一行写上：

**'use strict';**

**JavaScript中数据的奇葩用法（数组第一个元组下标为0）：**

var arr = [1, 2, 3.14, 'Hello', null, true];

arr.length = 6;

**arr; // arr变为[1, 2, 3, undefined, undefined, undefined]，给数组长度赋值会改变数据的长度。**

**arr[5] = 'x';**

**arr; // arr变为[1, 2, 3, undefined, undefined, 'x']，可以给超出边界的数组赋值。**

大多数其他编程语言不允许直接改变数组的大小，越界访问索引会报错。然而，JavaScript的Array却不会有任何错误。在编写代码时，不建议直接修改Array的大小，访问索引时要确保索引不会越界。

var arr = ['A', 'B', 'C', 'D', 'E', 'F', 'G'];

var aCopy = arr.slice();//slice数组的切片，也可以实现数组的复制。

var xiaohong = {

name: '小红',

'middle-school': 'No.1 Middle School'

};

xiaohong的属性名middle-school不是一个有效的变量，就需要用''括起来。访问这个属性也无法使用.操作符，必须用['xxx']来访问：

xiaohong['middle-school']; // 'No.1 Middle School'

xiaohong['name']; // '小红'

xiaohong.name; // '小红'

也可以用xiaohong['name']来访问xiaohong的name属性，不过xiaohong.name的写法更简洁。我们在编写JavaScript代码的时候，属性名尽量使用标准的变量名，这样就可以直接通过object.prop的形式访问一个属性了。

由于JavaScript的对象是动态类型，你可以自由地给一个对象添加或删除属性

var xiaoming = {

name: '小明'

};

xiaoming.age; // undefined

xiaoming.age = 18; // 新增一个age属性

xiaoming.age; // 18

delete xiaoming.age; // 删除age属性

xiaoming.age; // undefined

delete xiaoming['name']; // 删除name属性

xiaoming.name; // undefined

delete xiaoming.school; // 删除一个不存在的school属性也不会报错

如果我们要检测xiaoming是否拥有某一属性，可以用in操作符：

var xiaoming = {

name: '小明',

birth: 1990,

school: 'No.1 Middle School',

height: 1.70,

weight: 65,

score: null

};

'name' in xiaoming; // true

'grade' in xiaoming; // false

不过要小心，如果in判断一个属性存在，这个属性不一定是xiaoming的，它可能是xiaoming继承得到的：

'toString' in xiaoming; // true

因为toString定义在object对象中，而所有对象最终都会在原型链上指向object，所以xiaoming也拥有toString属性。

要判断一个属性是否是xiaoming自身拥有的，而不是继承得到的，可以用hasOwnProperty()方法：

var xiaoming = {

name: '小明'

};

xiaoming.hasOwnProperty('name'); // true

xiaoming.hasOwnProperty('toString'); // false

JavaScript把null、undefined、0、NaN和空字符串''视为false，其他值一概视为true。

**Map：**

var m = new Map([['Michael', 95], ['Bob', 75], ['Tracy', 85]]);

var m = new Map(); // 空Map

m.set('Adam', 67); // 添加新的key-value

m.set('Bob', 59);

m.has('Adam'); // 是否存在key 'Adam': true

m.get('Adam'); // 67

m.delete('Adam'); // 删除key 'Adam'

m.get('Adam'); // undefined

**Set：**一组key的集合，但不存储value。由于key不能重复，所以，在Set中，没有重复的key。

var s1 = new Set(); // 空Set

var s2 = new Set([1, 2, 3]); // 含1, 2, 3

s1.add(4)

s1.delete(3);

**遍历Array可以采用下标循环，遍历Map和Set就无法使用下标。为了统一集合类型，ES6标准引入了新的iterable类型，Array、Map和Set都属于iterable类型。**

'use strict';

var a = [1, 2, 3];

for (var x of a) {

}

alert('你的浏览器支持for ... of');

你可能会有疑问，for ... of循环和for ... in循环有何区别？

for ... in循环由于历史遗留问题，它遍历的实际上是对象的属性名称。一个Array数组实际上也是一个对象，它的每个元素的索引被视为一个属性。

**Django**

**ORM(对象关系映射)：**可以把数据库里面的数据映射面向对象语言里面的代码对象。

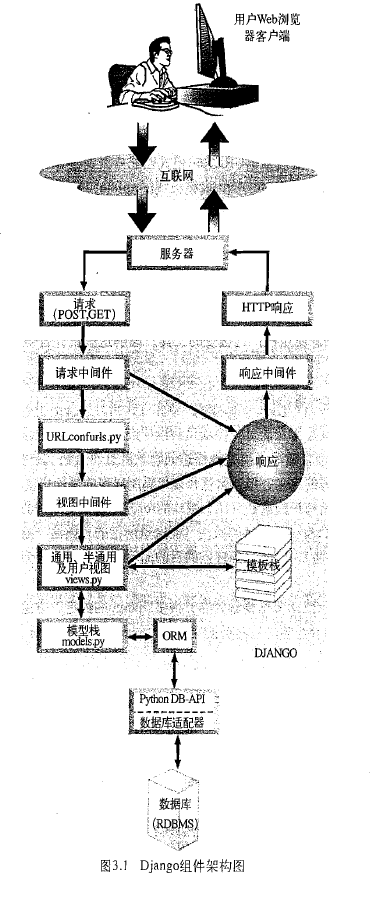
Django提供了一个强大的ORM机制，Python里面的类代表了数据库中的表，对象代表了其中每一行，而对象的属性则代表了列。

MVC（Model-View-Controller）：模型（控制数据），视图（定义显示的方法），控制器（在Model和View之间斡旋，并且让用户可以请求和操作数据）。

**Django的模型只负责把数据传入传出数据库。**

Django把表示层一分为二，视图方法定义了要显示模型里面的什么数据，而模板则定义了最终信息的显示方式。

框架自己则担当了控制器的角色-它提供了决定什么视图和什么模板 一起相应给定请求的机制。



**Django希望尽量Pythonic且DRY（Do not Repeat Yourself）**

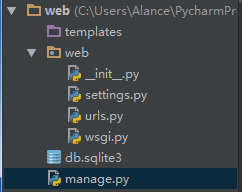
**安装Django：**

可以在官网中下载Django的ta文件后，使用tar –zxvf 命令，解压，然后使用sudo python setup.py install命令进行安装，安装完成之后可以使用import django

django.VERSION（必须大写）命令来查看django是否正确安装。

同样，如果是在windows环境下安装Django的话，把压缩文件解压后，放在Python的安装目录后，在cmd命令框中敲入python setup.py install，一会就安装好了。

一个Django工程包含的文件如下所示：



manage.py: 一个实用的命令行工具，可让你以各种方式与该 Django 项目进行交互。

内层 web/ 目录是你项目中的实际 Python 包。该目录名就是 Python 包名，通过它你可以导入它里面的任何东西。 (e.g. import mysite.settings).

web/\_\_init\_\_.py: 一个空文件，告诉 Python 该目录是一个 Python 包。(如果你是 Python 新手，请查看官方文档了解 关于包的更多内容 。)

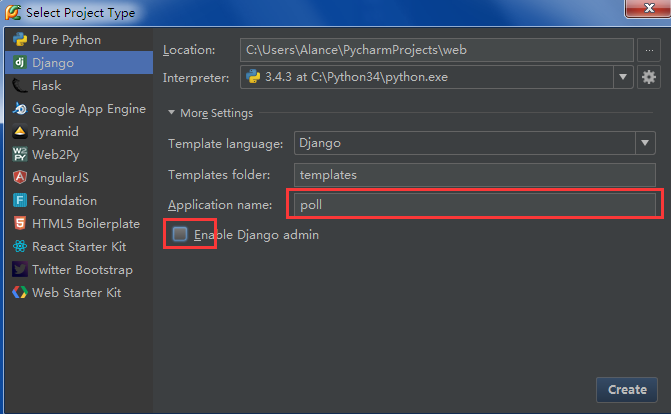
web/settings.py: 该 Django 项目的设置/配置。请查看 Django settings 将会告诉你如何设置。

web/urls.py: 该 Django 项目的 URL 声明; 一份由 Django 驱动的网站“目录”。请查看 URL dispatcher 可以获取更多有关 URL 的信息。

web/wsgi.py: 一个 WSGI 兼容的 Web 服务器的入口，以便运行你的项目。

**manage.py的常见命令：**

python manage.py startapp polls //新建一个app应用，当然如果只是新建一个app命令，可以在新建Django项目的时候在Pycharm中新建。

  
 需要注意的是务必将上图中的Enable Django admin前面的勾取消

python manage.py runserver //启动项目

python manage.py startapp poll //新建poll应用

python manage.py makemigrations poll//如果此步不执行，会报no such table: polls\_poll的错误。

python manage.py syncdb //syncdb 命令会给在 INSTALLED\_APPS 中有但数据库中没有对应表的应用执行 sqlall 操作。 该操作会为你上一次执行 syncdb 命令以来在项目中添加的任何应用创建对应的表、初始化数据和创建索引。 syncdb 命令只要你喜欢就可以任意调用，并且它仅会创建不存在的表。

python manage.py shell //进入python的shell界面

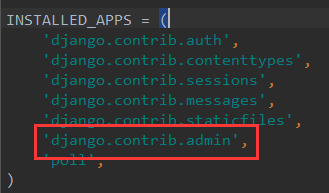
python manage.py createsuperuser //创建admin用户名和密码

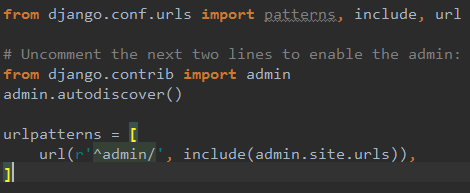
创建了一个测试账号sunchuan密码为：416998943

为你的员工或客户生成添加、修改和删除内容的管理性网站是个单调乏味的工作。 出于这个原因，Django 根据模型完全自动化创建管理界面。管理界面不是让网站访问者使用的，它是为了网站管理员准备的。

启动Django管理网站需要做三件事情：

* 在 INSTALLED\_APPS 设置中添加 "django.contrib.admin"



* 运行 python manage.py syncdb 命令。既然你添加了新应用到 INSTALLED\_APPS 中，数据库表就需要更新。
* 编辑你的 mysite/urls.py 文件并且将有关管理的行取消注释 – 共有三行取消了注释。该文件是 URLconf ；我们将在下一个教程中深入探讨 URLconfs 。现在，你需要知道的是它将 URL 映射到应用。最后你拥有的 urls.py 文件看起来像这样: 

为了让poll的数据在管理网页中可见，只需要做一件事：我们需要告诉管理网站 Poll 对象要有一个管理界面。为此，我们在你的 polls 目录下创建一个名为 admin.py 的文件，并添加如下内容：:

from django.contrib import admin

from polls.models import Poll

admin.site.register(Poll)

**你需要重启开发服务器才能看到变化。通常情况下，你每次修改过一个文件后开发服务器都会自动载入，但是创建一个新文件却不会触发自动载入的逻辑。**

**Django Model中的变量类型：**

CharField 和 TextField:两者区别为CharField为定长，而TextField长度为无限长。

EmailField，URLField和IPAddressField：这三个变量就是CharField中加上了一点额外的验证，可以用来验证它们的值分别是有效的Email地址、URL和IP地址。也可以轻松地在模型变量上加入验证来穿凿自己的“变量模型”

BooleanField：存储True或者Flase，有的时候还需要存储null，就出现了NullBooleanField。

TextField：TextField旨在数据库中保存了一个文件的路径，和它近亲FilePathField相近，但是它允许用户从浏览器上传一个文件，并将它过保存在服务器上。

外键：models.ForeignKey(“Author”)

manage.py函数：

syncdb：创建所有应用程序所需要的数据表

sql：显示create table调用

sqlall：从sql文件中初始化数据载入语句

sqlindexs：显示对主键创建索引的调用

sqlclear：显示drop table调用

sqlreset：sqlclear和sql的组合（drop加create）

sqlcustom：显示指定sql文件里面的自定义sql语句

loaddata：载入初始数据

dumpdata：把现有数据库里的数据输出为json，xml等格式。

和syscdb不同的是，这些sql系列的命令并不更新数据库，只是把对应的sql语句打印出来，让程序员有机会验证或者把它们保存到单独的sql脚本文件中。

**URL、HTTP机制和视图**

url(r'^$', index)：表示需要匹配的是网站的根URL。（**注意:**正则表达式都是缺少了大头的斜杠，因为默认都是以斜杠开头，故可以省略）

urlpatterns = [：打头的是前缀字符串，可以为空，如果非空代表所有下面的目录都是前缀字符串的子目录。（参见P106）

Django中（甚至Web框架中）用的最多的功能大概就是使用预定义的代码来编写组成大多数Web应用的CRDU操作，CRDU全称是Create、Read、Update、Delete。

Django模板语言有两个约定，变量的输出使用双大括号{{variable}}，其它则用标签{%commmand%}

Django提供了各种有用的模板过滤器来封装Web开发中常见的文本处理工作，例如转椅斜杠符号、大写首字、格式化日期等等。模板过滤器通过管道符号|进行转换，例如：

<li>{{ string | lower }}</li>

<td>{{ person.is\_available | yesno:"yes, no" }}</td>

模板的继承是通过{% extend %}和{% block %}两个模板标签来实现的，

**表单：**

表单创建之后可以使用form.save()方法，将他们的信息保存为数据库的一条记录，然后返回Model的对象结果。

如果需要对表单进行存储，可以通过以下语句：

newperson = form.save(commit=False)

newperson.middle = ‘Danger’

neeperson.save()

应用场景：网页游戏，地图搜索，web聊天，js是ajax、jquery等框架的基础。

javascript：用于外部开发的脚本语言（客户端执行），通常和jsp/php/asp/asp.net（服务器端执行）配合使用。

js的代码位置可以任意放置，如果js放置在html文件内部，可以使用如下的语法进行嵌套：

<script language = "javascript">

window.alert("hello world")

</script>

js代码也是顺序执行的。

**js变量的定义：**var num =123;//关键字var甚至可以去掉。

js中非零的值都为真。除了0, "",null, undefined,NaN,其它均表示false。

js中比较特殊的关系运算符：

||返回第一个不为false的值,如果你所有都为false,则返回最后一个值,||返回的结果不一定是布尔值

var a = 2;

var b = 0;

var c = b||a;

windows.alert(c); //输出结果为2

**函数语法：**

function test（num1， num2， num3）//参数切记，不需要加var。

{

}

js中还可以对函数取个别名，例如：

**var test2 =test;//给test起了个别名，如果括号右面是test（），而不是test，则将test（）函数的值赋给了test，而不是给函数test起了个别名。**

**test2(1,2,3);**

js作为脚本语言，支持可变参数。

js支持面向对象编程，访问对象方式：

1. 对象名.属性
2. 对象名[“属性”]//**这种访问方式的优势在于属性名可以使用拼接。**

js中比较奇葩的一点是，如果在函数内部定义的变量，或者函数没有使用this定义，则，这些变量都是内部变量或者函数，只能在内部使用。

function test()

{

}

this.show = function()

{

window.alert(“hello world”)

}

test.prototype.show2 = function()

{

window.alert(“test”)

}

使用

var 定义的属性都是内部属性。

**this不能放在类的外部使用，否则调用者变成了window（所有类的祖先）**

**JavaScript事件:**

**所有的时间都是以on开头，javascript中使用正则表达式最好加上开头^和结尾$**

**如果是radio是用数据选择，如果是checkbox使用**

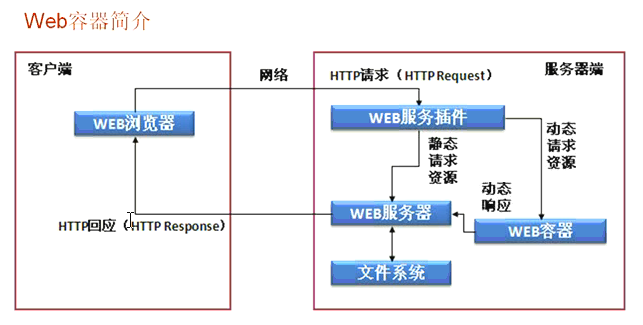
**XML与HTML区别：**

**HTML不具有扩展性**

**HTML侧重于如何显示信息，XML侧重于信息描述**

**XML两种解析接口：DOM、SAX，Javascript中以DOM为主**

**Tomcat**



用户发来的请求如果是动态请求，需要使用web容器进行处理，WEB容器本身专门用来处理代码拼凑功能，拼凑之后通过WEB服务器发还给客户端，WEB容器是可以连接数据库的。

WEB容器：名气比较大的收费：WebLogic，WebShere免费的：TOMCAT（使用人群较多）

虚拟目录所在的文件夹下面，必然包含WEB-INF/web.xml(必须放在第第一级目录中)

在server.xml文件中配置虚拟目录的映射，添加如下的配置信息：

<Context path="/bestpay" docBase="C:\DataVisualization"/>

注意bestpay前面必须带有/

HTTP状态码：

4XX表示客户端错误

5XX表示服务器错误

index.htm,index.html,index.jsp为服务器的默认首页，一旦输入虚拟目录之后，就可以直接调用此页面进行访问，默认主页的配置在web.xml配置文件中，配置代码如下：

<welcome-file-list>

<welcome-file>index.html</welcome-file>

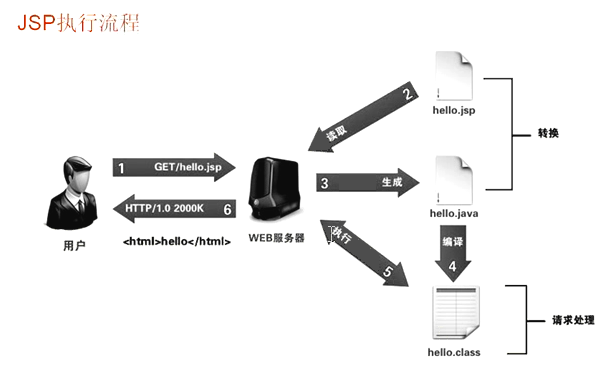
<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

更改上述代码中的index文件即可更改默认主页。

JSP执行流程



JSP程序依然是通过class文件运行的，第一访问的时候需要生成class文件，之后则不需要编辑，直接使用，每当改变jsp文件时都会重新生成\*.java文件和\*.class文件。