

# 第十章 WEB安全基础

知识点一：WEB基础

知识点二：WEB编程环境

知识点三：JavaScript实践

知识点四：PHP语言

知识点五：PHP连接数据库

知识点六：HTTP会话管理

知识点七：HTTP请求

知识点八：Cookie实战

知识点九：十大WEB安全威胁

# 知识点一：WEB基础

**超文本传输协议 (HTTP, HyperText Transfer Protocol)**是互联网上应用最为广泛的一种网络协议。所有的WWW文件都必须遵守这个标准。设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法。通过HTTP或者HTTPS协议请求的资源由**统一资源标示符** (Uniform Resource Identifiers) (或者, 更准确一些, **URLs**) 来标识。



关于HTTP协议的详细内容请参考RFC2616。HTTP协议采用了**请求/响应**模型。客户端向服务器发送一个请求，请求头包含请求的方法、URL、协议版本、以及包含请求修饰符、客户信息和内容的类似于MIME的消息结构。服务器以一个状态行作为响应，响应的内容包括消息协议的版本，成功或者错误编码加上包含服务器信息、实体元信息以及可能的实体内容。





“超文本”就是指页面内可以包含图片、链接，甚至音乐、程序等非文字元素。**超文本标记语言Html**的结构包括“**头**”部分（英语：Head）、和“**主体**”部分（英语：Body），其中“头”部提供关于网页的信息，“主体”部分提供网页的具体内容。

一个网站对应多个HTML文件，超文本标记语言文件以.htm（磁盘操作系统DOS限制的外语缩写）为扩展名或.html（外语缩写）为扩展名。可以使用任何能够生成TXT类型源文件的文本编辑器来产生超文本标记语言文件，只用修改文件后缀即可。

## 标准的超文本标记语言文件都具有一个基本的整体结构



标记一般都是成对出现（部分标记除外例如：<br/>），即超文本标记语言文件的开头与结尾标志和超文本标记语言的头部与实体两大部分。有三个双标记符用于页面整体结构的确认。



标记符<html>，说明该文件是用超文本标记语言（本标签的中文全称）来描述的，它是文件的开头;而</html>，则表示该文件的结尾，它们是超文本标记语言文件的开始标记和结尾标记。

<head></head>

<head></head>

这2个标记符分别表示头部信息的开始和结尾。**头部中包含的标记是页面的标题、序言、说明等内容，它本身不作为内容来显示，但影响网页显示的效果。**头部中最常用的标记符是标题标记符和meta标记符，其中**标题标记符用于定义网页的标题，它的内容显示在网页窗口的标题栏中**，网页标题可被浏览器用作书签和收藏清单。

<body></body>

网页中显示的实际内容均包含在这2个正文标记符之间。正文标记符又称为实体标记。



**JavaScript一种直译式脚本语言，它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在HTML（标准通用标记语言下的一个应用）网页上使用，用来给HTML网页增加动态功能。**

JavaScript是一种属于网络的脚本语言，已经被广泛用于Web应用开发，常用来为网页添加各式各样的动态功能,为用户提供更流畅美观的浏览效果。通常JavaScript脚本是通过嵌入在HTML中来实现自身的功能的。





Javascript脚本语言同其他语言一样



有它自身的**基本数据类型**，**表达式和算术运算符及程序的基本程序框架**。Javascript提供了四种基本的数据类型和两种特殊数据类型用来处理数据和文字。而变量提供存放信息的地方，表达式则可以完成较复杂的信息处理。

日常  
用途

嵌入动态文本于HTML页面、对浏览器事件做出响应、读写HTML元素、在数据被提交到服务器之前验证数据、检测访客的浏览器信息、控制cookies，包括创建和修改等。



示例如下

```
<html>

<head>
  <title>Javascript简单示例</title>
</head>
<body>
  <script language="javascript">
    alert("第一个javascript");
  </script>
</body>
</html>
```



## 知识点二：WEB编程环境

## WEB编程语言



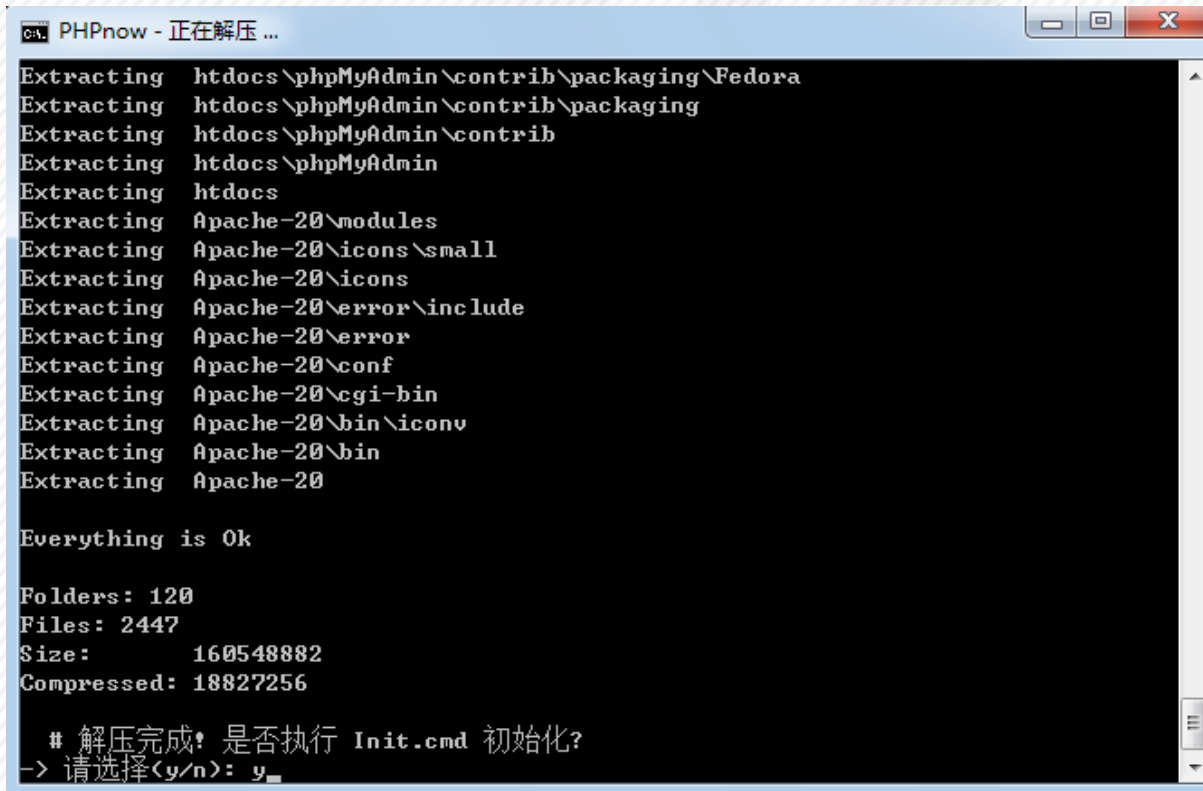
分为**WEB静态语言**和**WEB动态语言**，WEB静态语言就是通常所见到的超文本标记语言（标准通用标记语言下的一个应用），WEB动态语言主要是ASP、PHP、JAVASCRIPT、JAVA、CGI等计算机脚本语言编写出来的执行灵活的互联网网页程序。

- PHPnow是Win32下绿色免费的Apache + PHP + MySQL 环境套件包。简易安装、快速搭建支持虚拟主机的 PHP 环境。附带 PnCp.cmd控制面板，帮助你快速配置你的套件，使用非常方便。
- PHPnow 是绿色的，解压后执行 Setup.cmd 初始化，即可得到一个 PHP + MySQL 环境。然后就可以直接安装 Discuz!, PHPWind, DeDe, WordPress 等程序。



## PHPnow 是绿色的

解压后执行 Setup.cmd 初始化，即可得到一个 PHP + MySQL 环境。  
在 windows7 及以上版本安装的时候，会遭遇管理员权限、路径包含非英文字符的问题，安装所注意的细节如下：



```
C:\> PHPnow - 正在解压 ...
Extracting htdocs\phpMyAdmin\contrib\packaging\Fedora
Extracting htdocs\phpMyAdmin\contrib\packaging
Extracting htdocs\phpMyAdmin\contrib
Extracting htdocs\phpMyAdmin
Extracting htdocs
Extracting Apache-20\modules
Extracting Apache-20\icons\small
Extracting Apache-20\icons
Extracting Apache-20\error\include
Extracting Apache-20\error
Extracting Apache-20\conf
Extracting Apache-20\cgi-bin
Extracting Apache-20\bin\iconv
Extracting Apache-20\bin
Extracting Apache-20
Everything is Ok

Folders: 120
Files: 2447
Size: 160548882
Compressed: 18827256

# 解压完成! 是否执行 Init.cmd 初始化?
-> 请选择(y/n): y
```

(1) 解压phpnow后（切忌路径不能出现中文，有可能导致以后服务无法正常启动），点setup.cmd即可进入安装界面，之后，选择init.cmd：



然而，很可能无法安装成功



提示apache\_cn服务安装失败。这个是因为需要管理员权限方可完成服务的安装。



(2) 解决管理员权限问题的做法是：**到c:\windows\system32中找到cmd.exe，右键，以管理员方式运行；**启动后，通过dos对话框切入phpnow的安装路径：



管理员: 初始化 PHPnow 1.5.6-1 - Apache 2.0 + PHP + MySQL 5.0

90 个目录 2,757,373,952 可用字节

C:\Windows\system32>e:

E:\>dir

驱动器 E 中的卷是 新加卷  
卷的序列号是 0EE5-7314

E:\ 的目录

2016/09/11	09:19	<DIR>	phpnow
		0 个文件	0 字节
		1 个目录	1,915,523,072 可用字节

E:\>cd phpnow

然后运行init.cmd

















```
管理员: 初始化 PHPnow 1.5.6-1 - Apache 2.0 + PHP + MySQL 5.0

: 正在安装 Apache ...
:
: 正在启动 Apache ...
:
: 启动 Apache 完成;
:
:
: 正在启动 MySQL 5.0 ...
:
Service successfully installed.
MySQL5_pn 服务正在启动:
MySQL5_pn 服务已经启动成功。
:
: 启动 MySQL 5.0 完成;
:
:
: 现在为 MySQL 的 root 用户设置密码. 重要! 请切记!
:
-> 设置 root 用户密码: 123456
```

安装完毕，将看到phpnow的默认界面。  
此时，安装完后目录如下：

## 安装完后目录如下

名称	修改日期	类型	大小
 Apache-20	2015-05-14 15:00	文件夹	
 htdocs	2015-05-24 10:23	文件夹	
 MySQL-5.0.90	2015-05-14 15:00	文件夹	
 php-5.2.14-Win32	2015-05-14 15:00	文件夹	
 Pn	2015-05-14 15:01	文件夹	
 PnCmds	2010-09-22 2:51	文件夹	
 ZendOptimizer	2010-09-25 22:20	文件夹	
 7z.exe	2010-09-08 17:27	应用程序	159 KB
 Init.cm_	2010-09-25 22:33	CM_ 文件	10 KB
 PnCp.cmd	2010-09-22 3:14	Windows 命令脚本	15 KB
 Readme.txt	2010-09-25 22:34	文本文档	2 KB
 更新日志.txt	2010-09-25 22:26	文本文档	3 KB
 关于静态.txt	2009-04-14 23:11	文本文档	1 KB
 升级方法.txt	2009-02-04 9:05	文本文档	1 KB

点击PnCp.cmd, 如下



# 选择序号20，即可启动PHPnow

打开网页，访问http://127.0.0.1如下：

我的主页

为何只能本地访问？  
获取外网 IP 失败！

127.0.0.1

# Let's PHP now !

Server Information	
SERVER_NAME	127.0.0.1
SERVER_ADDR:PORT	127.0.0.1:80
SERVER_SOFTWARE	Apache/2.0.63 (Win32) PHP/5.2.14
PHP_SAPI	apache2handler
php.ini	D:\PHPnow\php-5.2.14-Win32\php-apache2handler.ini
网站主目录	D:/PHPnow/htdocs
Server Date / Time	2016-05-02 16:44:25 (+08:00)
Other Links	phpinfo()   phpMyAdmin

PHP 组件支持	
Zend Optimizer	Yes / 3.3.3
MySQL 支持	Yes / client lib version 5.0.90
GD library	Yes / bundled (2.0.34 compatible)
eAccelerator	No

MySQL 连接测试			
MySQL 服务器	<input type="text" value="localhost"/>	MySQL 数据库名	<input type="text" value="test"/>
MySQL 用户名	<input type="text" value="root"/>	MySQL 用户密码	<input type="password"/>
			<input type="button" value="连接"/>

## 点击phpMyAdmin

将进入数据库管理界面，可以自行创建数据库、表以及录入数据等：

phpMyAdmin

localhost ▶ mydb

结构 SQL 搜索 查询 导出 导入 操作 权限 删除

	表 ▲	操作	记录数 1	类型	整理	大小	多余
<input type="checkbox"/>	inttable	     	2	MyISAM	latin1_swedish_ci	2.0 KB	-
<input type="checkbox"/>	t1	     	0	MyISAM	latin1_swedish_ci	1.0 KB	-
<input type="checkbox"/>	userinfo	     	2	MyISAM	latin1_swedish_ci	2.0 KB	-
	3 个表	总计	4	MyISAM	latin1_swedish_ci	5.1 KB	0 字节

↑ 全选 / 全不选 选中项: ▼

打印预览 数据字典

✱ 在数据库 mydb 中新建一个数据表

名字:  字段数:

执行

 <sup>1</sup> 可能接近。参见 [FAQ 3.11](#)

# 知识点三：JavaScript实践



使用工具Dreamweaver，来编辑产生一个静态网页，该网页命名为js.htm，存储到PHPnow\htdocs下。

在网页js.htm中，进行如下四个代码的编辑和运行，可以看到Javascript对浏览器中网页内各元素的读写功能。



## document.write ()函数

```
<html>
  <head>
    <title>Javascript简单示例</title>
    <script language="javascript">
      for(i= 1; i <= 100; i++){
        num= Math.floor(Math.random() * 100); //0-99
        document.write(num,"");
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

之间的随机数

## 单击按钮后调用函数

```
<html>
  <head>
    <title>Javascript简单示例</title>
    <script language="javascript">
      function func1(){
        alert("按钮单击后调用的函数1！ ");
      }
      function func2(){
        alert("按钮单击后调用的函数2！ ");
      }
    </script>
  </head>
  <body>
    <!--单击后调用两个函数用","隔开 -->
    <input type="button" value="单击我" onClick="func1(), func2()" />
  </body>
</html>
```

## 使用对象：同样使用function。

```
<html>
  <head>
    <title>Javascript简单示例</title>
  </head>
  <body>
    <script language="javascript">
      function Student(name, school, grade){
        this.name= name;//注意这里要用this
        this.school=school;
        this.grade= grade;
      }
      hui= new Student("noting_gonna", "XX学校", "小学二年级");
      //使用with可以省略对象名
      with(hui){
        document.write(name+ ": " + school + "," + grade + "<br />");
      }
      if(window.hui){
        document.write("hui这个对象存在");
      }
      else
        document.write("hui这个对象不存在");
    </script>
  </body>
</html>
```

## 获取input (text) 中的内容 :name.value

```
<html>
  <head>
    <title>Javascript简单示例</title>
  </head>
  <body>
    <script language="javascript">
      function getLoginMsg(){
        //以下也达到了省略对象名称的作用
        loginMsg= document.loginForm;
        alert("账号： " +loginMsg.userID.value + "\n" + "密码： " +loginMsg.password.value);
      }
      function setLoginMsg(Object){
        alert(Object.id);
      }
    </script>
    <form name="loginForm">
      账号： <input type="text" name="userID" /><br />
      密码： <input type="text" name="password" /><br />
      <input type="button" value="登录" onclick="getLoginMsg()" />
      记住密码<input type="checkbox" id="这是checkbox的id" onclick="setLoginMsg(this)" />
    </form>
  </body>
</html>
```

# 知识点四：PHP语言





## 四种标记

标记	解释	示例
<code>&lt;?php ?&gt;</code>	标准php标记	<code>&lt;?php echo \$variable; ?&gt;</code>
<code>&lt;script language="php"&gt; &lt;/script&gt;</code>	长标记	<code>&lt;script language="php"&gt; echo \$vaiable; &lt;/script&gt;</code>
<code>&lt;? ?&gt;</code>	短标记	<code>&lt;? echo \$variable; ?&gt;</code>
<code>&lt;% %&gt;</code>	仿ASP	<code>&lt;%= \$variable; %&gt;</code>



PHP需要在每个语句后用分号结束指令

在一个PHP代码段中的最后一行可以不用分号结束

## 注释

使用注释可以增加语言的可读性，PHP支持三种C/C++、perl、unix-shell风格的注释：

- # ——Perl式的单行注释
- // ——C++式的单行注释
- /\* \*/ ——C/C++式多行注释

## 变量解析

**变量解析**当遇到符号（\$）时产生，解析器会尽可能多地取得后面的字符以组成一个合法的变量名，然后将变量值替换他们，如果\$后面没有有效的变量名，则输出"\$"。如果想明确的变量名可以用花括号把变量名括起来。

```
<?php
$username = "liuzheli";
$SQLStr = "SELECT * FROM userinfo where username='$username'";
echo $SQLStr ;
?>
```

在对上述php段进行解析时，第一个\$标识的username将被解析为一个变量，因为第一次定义，将分配内存空间被赋以初值liuzheli。在第二个标识的变量SQLStr的初值中，因为\$username已经被解析为变量，所以，最终显示的结果将是：SELECT \* FROM userinfo where username=' liuzheli '。

同样也可以解析数组索引或者对象属性。对于数组索引，右方括号 (]) 标志着索引的结束。对象属性则和简单变量适用同样的规则。

# 知识点五：HTTP会话管理

## HTTP会话管理



在计算机术语中，**会话是指一个终端用户与交互系统进行通讯的过程**，比如从输入账户密码进入操作系统到退出操作系统就是一个会话过程。



会话较多用于网络上，TCP的三次握手就创建了一个会话，TCP关闭连接就是关闭会话。用平述的语言可以解释为：你拨打你女友的电话号码，你女友接听，然后一翻“亲爱的”，直到任何一方挂掉电话，这个过程就是一个会话。你挑逗一只小狗，它跟你互动，也是会话；它不鸟你，那就不形成会话。

## HTTP协议属于无状态的通信协议

无状态是指，当浏览器发送请求给服务器的时候，服务器响应，但是当同一个浏览器再发送请求给服务器的时候，他不知道你就是刚才那个浏览器。简单地说，就是服务器不会去记得你，所以是无状态协议。



本质是，**HTTP是短连接的**，请求响应后，断开了TCP连接，下一次连接与上一次无关。

为了识别不同的请求是否来自同一客户，需要引用HTTP会话机制




即：**多次HTTP连接间维护用户与同一用户发出的不同请求之间关联的情况称为维护一个会话（session）**。通过会话管理对会话进行创建、信息存储、关闭等。



## HTTP会话的实现机制



Cookie与session是与HTTP会话相关的两个内容，其中Cookie存在在浏览器， session存储在服务器中。



Cookies是服务器在本地机器上存储的小段文本并随每一个请求发送至同一个服务器。网络服务器用HTTP头向客户端发送cookies，在客户终端，浏览器解析这些cookies并将它们保存为一个本地文件，它会自动将同一服务器的任何请求缚上这些cookies。



具体来说，cookie机制采用的是在客户端保持状态的方案

**它是在用户端的会话状态的存贮机制，需要用户打开客户端的cookie支持。**

cookie的作用是为了解决HTTP协议无状态的缺陷所作的努力。

- ✓ **正统的cookie分发是通过扩展HTTP协议来实现的，服务器通过在HTTP的响应头中加上一行特殊的指示以提示浏览器按照指示生成相应的cookie。然而纯粹的客户端脚本如JavaScript也可以生成cookie。而cookie的使用是由浏览器按照一定的原则在后台自动发送给服务器的。**
- ✓ **浏览器检查所有存储的cookie，如果某个cookie所声明的作用范围大于等于将要请求的资源所在的位置，则把该cookie附在请求资源的HTTP请求头上发送给服务器。**

## cookie的内容主要包括

名字，值，过期时间，路径和域。路径与域一起构成cookie的作用范围。若不设置过期时间，则表示这个cookie的生命期为浏览器会话期间，关闭浏览器窗口，cookie就消失。**这种生命期为浏览器会话期的cookie被称为会话cookie。**

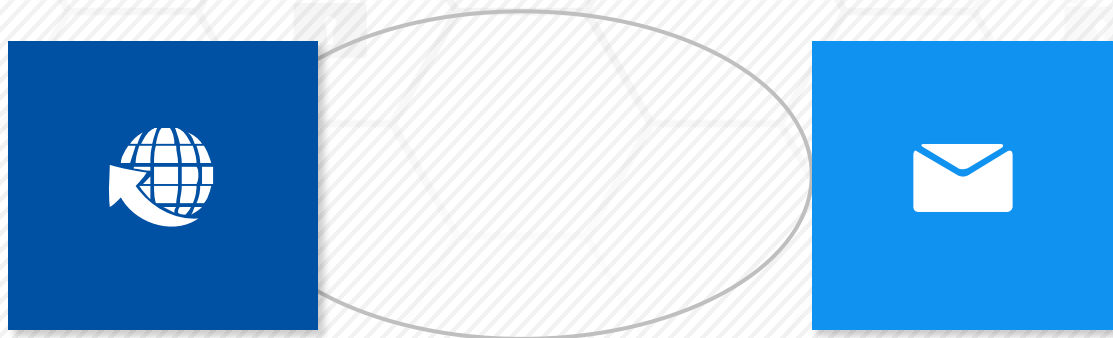
会话cookie一般不存储在硬盘上而是保存在内存里，当然这种行为并不是规范规定的。

- ✓ **若设置了过期时间，浏览器就会把cookie保存到硬盘上，关闭后再次打开浏览器，这些cookie仍然有效直到超过设定的过期时间。**
- ✓ **存储在硬盘上的cookie可以在不同的浏览器进程间共享，比如两个IE窗口。而对于保存在内存里的cookie，不同的浏览器有不同的处理方式。**

## Session机制是一种服务器端的机制

服务器使用一种类似于散列表的结构（也可能就是使用散列表）来保存信息。

- ✓ 当程序需要为某个客户端的请求创建一个session时，服务器首先检查这个客户端的请求里是否已包含了一个session标识（称为session id），如果已包含则说明以前已经为此客户端创建过session，服务器就按照session id把这个session检索出来使用（检索不到，会新建一个），如果客户端请求不包含session id，则为此客户端创建一个session并且生成一个与此session相关联的session id。
- ✓ session id的值应该是一个既不会重复，又不容易被找到规律以伪造的字符串，这个session id将被在本次响应中返回给客户端保存。



保存这个session id的方式可以采用cookie

这样在交互过程中浏览器可以**自动的**按照规则把这个标识发挥给服务器。一般这个cookie的名字都是类似于SEESIONID。

所以，一种常见的HTTP会话管理就是，服务器端通过session来维护客户端的会话状态，而在客户端通过cookie来存储当前会话的session id。

还有一种技术叫做**表单隐藏字段**。就是服务器会自动修改表单，添加一个隐藏字段，以便在表单提交时能够把session id传递回服务器。



**URL重写**: 但cookie可以被人为的禁止，必须有其他机制以便在cookie被禁止时仍然能够把session id传递回服务器。经常被使用的一种技术叫做**URL重写**，就是把session id直接附加在URL路径的后面。

## 知识点六：HTTP请求



## 使用工具Dreamweaver

来编辑产生一个静态网页，该网页命名为login.htm，存储到PHPnow\htdocs下。

**注：htdocs是PHPnow的web应用的根目录。**



所编辑的login.htm代码如下:

```
<form id="form1" name="form1" method="post" action="loginok.php">
  <table width="900" border="0" cellspacing="0" cellpadding="0">
    <tr>
      <td height="20">姓名</td>
      <td height="20"><label>
        <input name="username" type="text" id="username" />
      </label></td>
    </tr>
```





所编辑的login.htm代码如下:

```
<tr>
  <td height="20">口令</td>
  <td height="20"><label>
    <input name="pwd" type="password" id="pwd" />
  </label></td>
</tr>
<tr>
  <td height="20">&nbsp;</td>
  <td height="20"><label>
    <input type="submit" name="Submit" value="提交" />
  </label></td>
</tr>
</table>
</form>
```



## Form 表单

**表单是一个包含表单元素的区域。**表单区域里包含了两个文本框（<input>）、一个确认按钮(submit)。确认按钮的作用是当用户单击确认按钮时，表单的内容会被传送到另一个文件。

## 表单属 性

□ **表单的动作属性(action)定义了目的文件的文件名。**由动作属性定义的这个文件通常会对接收到的输入数据进行相关的处理。在上面的表单中定义了接受表单输入的处理文件为“loginok.php”。

□ **method属性指定了与服务器进行信息交互的方法为POST。**



Http定义了与服务器交互的不同方法，最基本的方法有4种



URL全称是资源描述符，我们可以这样认为：一个URL地址，它用于描述一个网络上的资源，而HTTP中的GET、POST、PUT、DELETE就对应着对这个资源的查，改，增，删4个操作。

**GET**一般用于获取/查询资源信息，**而POST**一般用于更新资源信息，早期的系统由于不支持DELETE，因此PUT和DELETE用的较少。

处理提交输入的第一个php文件代码如下：

```
<?php
$username = $_POST['username'];
$password = $_POST['pwd'];
$sqlStr = "SELECT * FROM userinfo
where username='$username' and
pwd='$password'";
echo $sqlStr ;
?>
```



搭建环境，将表单的输入改为GET，php的程序也改为GET，看看变化在哪里？

具体的，POST和GET的区别如下：

### GET请求的数据会附在URL之后

（就是把数据放置在HTTP协议头中），以?分割URL和传输数据，参数之间以&相连，如：login.action?name=sean&password=123。

### POST把提交的数据则放置在是HTTP包的包体中

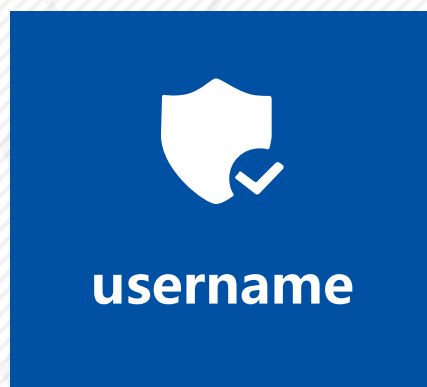
**POST的安全性要比GET的安全性高：**

- (1) GET模式下，通过URL就可以作数据修改**
- (2) GET模式下，用户名和密码将明文出现在URL上，**因为登录页面有可能被浏览器缓存、其他人查看浏览器的**历史纪录**，那么别人就可以拿到你的账号和密码了
- (3) GET模式下，提交数据还可能会造成跨站请求伪造攻击**

# 知识点七：PHP连接数据库

## 建表

在myDB库中，设有一个表userinfo，包含两个字段，即



假设在上述loginok.php中实现对输入的用户名和密码进行认证，代码如下：



```
<?php
$conn=mysql_connect("localhost", "root", "123456"); //连接数据库
$username = $_POST['username'];
$password = $_POST['pwd'];
$sqlStr = "SELECT * FROM userinfo where username='$username' and
pwd='$password'"; echo $sqlStr ;
$result=mysql_db_query("MyDB", $sqlStr, $conn); //执行数据库SQL语句
if ($row=mysql_fetch_array($result))//读取数据内容
    echo "<br>OK<br>";

else
    echo "<br>false<br>";
mysql_free_result($result); // 释放资源
mysql_close($conn); // 关闭连接
?>
```



如果登陆成功，则显示OK，否则，显示false。



## 数据库应用开发三步骤

### 1. 连接数据库:

```
$conn=mysql_connect("local  
host", "root", "123456");
```



数据库的连接  
分为几步:

### 2. 执行SQL操作:

```
$result=mysql_db_query("MyDB",  
$SQLStr, $conn);
```

### 3. 关闭连接:

```
mysql_free_result($result);  
mysql_close($conn);
```

## 查询数据后显示数据

数据库的操作主要依赖于SQL语句，查询数据并显示的一个例子如示例

```
<?php
// 定位到第一条记录
mysql_data_seek($result, 0);
// 循环取出记录
while ($row=mysql_fetch_row($result))
{
    for ($i=0; $i<mysql_num_fields($result); $i++ )
    {
        echo $row[$i];
        echo " | ";
    }
    echo "<br>";
}
?>
```



```
<?php
$conn=mysql_connect("localhost", "root", "123456"); //连接数据库
$sqlStr = "SELECT * FROM userinfo "; //echo $sqlStr ;
$result=mysql_db_query("MyDB", $sqlStr, $conn); //执行数据库SQL语句
// 定位到第一条记录
mysql_data_seek($result, 0);
// 循环取出记录
while ($row=mysql_fetch_row($result))
{
    for ($i=0; $i<mysql_num_fields($result); $i++ )
    {
        echo $row[$i];
        echo " | ";
    }
    echo "<br>";
}
mysql_free_result($result); // 释放资源
mysql_close($conn); // 关闭连接
?>
```



创建show.php来演示验证

# 知识点八：Cookie实战

## Cookie

### Cookie 和 session

**共性：**都可以暂时保存在多个页面中使用的变量。

**区别：**cookie存放在客户端浏览器，session保存在服务器。它们之间的联系是session ID一般保存在cookie中，来实现HTTP会话管理。

### Cookie 工作 原理

**生成Cookie：**当客户访问某网站时，PHP可以使用setcookie函数告诉浏览器生成一个cookie，并把这个cookie保存在c:\Documents and Settings\用户名\Cookies目录下。

**使用Cookie：**Cookie是HTTP标头的一部分，当客户再次访问该网站时，浏览器会自动把与该站点对应的cookie发送到服务器，服务器则把从客户端传来的cookie将自动地转化成一个PHP变量。

## SetCookie函数

对 cookie 进行赋值的函数为setcookie，赋值成功返回 true，否则返回 false。

函数原型如下：

`setcookie(name, value, expire, path, domain, secure)`

name 必需。规定 cookie 的名称。

value 必需。规定 cookie 的值。

expire 可选。规定 cookie 的有效期。

举例，`time()+3600*24*30` 将设置 cookie 的过期时间为 30 天。

secure 可选。规定是否通过安全的 HTTPS 连接来传输 cookie。

domain 可选。规定 cookie 的域名。

path 可选。规定 cookie 的路径。

## 如下实例设置COOKIE

```
<?php
if ($row=mysql_fetch_array($result))//读取数据内容
{
    setcookie("uname",$username);//创建一个索引为uname的cookie
    echo "<br>OK<br>";
} else {
    echo "<br>false<br>";
}
?>
```

在loginok.php中对应修改



如下示例获取Cookie

可以通过 `$HTTP_COOKIE_VARS["uname"]` 或 `$_COOKIE["uname"]` 来访问索引为name的 cookie 的值。

```
<?php
if ($_COOKIE["uname"]==null)
    echo "should cookie";
else
    echo "ok";
?>
```

在show.php中对应修改



## Cookie

由于示例所使用的是**内存COOKIE**，即没有设定COOKIE值的**expires参数**，也就是没有设置COOKIE的失效时间情况下，这个COOKIE在关闭浏览器后将失效，并且不会保存在本地。

**验证：**关闭浏览器后，重新打开浏览器直接访问useCookie.php，则发现，提示should cookie。

通过show.php进行验证



## 设置Cookie有效期

修改示例中的COOKIE类型为设置expires参数

即COOKIE的值指定了失效时间，比如

```
setcookie("uname",$username, time()+3600*24*30),
```

那么这个COOKIE 会保存在本地，关闭浏览器后再访问网站useCookie.php，则发现，提示ok。

可见：在COOKIE有效时间内所有的请求都会带上这个本地保存COOKIE。

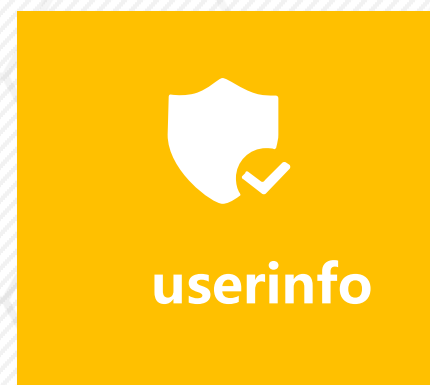


## 结论

如果一个页面依赖于某个cookie，而cookie的值或者文件被泄露后，即使没有登录，也可能利用该cookie来访问该页面，这就是cookie在客户端不安全引发的后果。



## 进一步WEB开发实战



**添加新闻：**输入新闻标题、新闻内容，将数据添加到news表中

**查看新闻列表：**显示新闻的标题

**查看新闻：**点某个新闻标题，查看该新闻的详细内容

**页面跳转：**登录成功后，跳转到添加新闻页面。

**请同学们根据教材和视频自行实践**

# 知识点九：十大WEB安全威胁

## Web应用十大安全威胁排名

根据2010年OWASP（开放Web软件安全项目—Open Web Application Security Project）发布的Web应用十大安全威胁排名，排在前十位的安全风险依次为：**注入**、**跨站脚本**、**遭破坏的身份认证和会话管理**、**不安全的直接对象引用**、**伪造跨站请求**、安全配置错误、不安全的加密存储、没有限制的URL访问、传输层保护不足和未验证的重定向和转发。



注入

跨站  
脚本

跨站请  
求伪造

注入、跨站脚本和跨站请求伪造将在第十一章介绍

## (1) 基本概念

- “遭破坏的认证和会话管理”是指**攻击者窃听了用户访问HTTP时的用户名和密码，或者是用户的会话，从而得到sessionID或用户身份信息，进而冒充用户进行HTTP访问的过程。**
- 由于HTTP本身是无状态的，HTTP的每次访问请求都要带有个人凭证，SessionID是用户访问请求的凭证。sessionID本身很容易在网络上被嗅探到，所以攻击者往往通过监听sessionID来实现进一步的攻击，这就是这种安全风险居高不下的的重要原因，但这种形式的攻击主要针对身份认证和会话。



## (2) 密码的安全性

- 密码（**口令**）是最常见的一种认证手段，持有正确密码的人被认为是可信的。使用密码进行认证的优点是成本低，认证过程实现简单
  - 缺点是密码认证是一种比较弱的安全手段，因而存在被猜解的可能。
- 
- 目前黑客们常用的破解密码手段，不是暴力破解，而是使用一些弱口令去尝试进行字典攻击破解，比如123456，admin等，同时猜解用户名，直到发现使用这些弱口令的账户为止。由于用户名往往是公开的信息，攻击者可以收集一份用户名的字典，这种攻击成本很低，然而效果却很好。密码的保存也有一些需要注意的地方，例如，密码必须使用不可逆的加密算法或者是单向散列函数算法进行加密后存储到数据库中。这可以最大程度地保证密码的私密性。



### (3) 用户的认证必须通过加密信道进行传输

- 在用户登录时，在用户输入用户名和密码后一般通过POST的方法进行传输，认证信息可通过不安全的HTTP传递，也可通过加密的HTTPS传递。  
**有些网站在登录页面显示的是HTTPS，而事实上却是用HTTP。**
- **检测是否使用HTTPS的最简单方法就是使用网络嗅探工具，如通过SnifferPro或Ethereal嗅探数据包来判断是否加密。**



#### (4) 会话劫持攻击

- SessionID一旦在生命周期内被窃取，就等于账户失窃。由于SessionID是用户登录持有的认证凭证，因此黑客不需要再想办法通过用户名和密码进行登录，而是直接使用窃取的SessionID与服务器进行交互。
- **会话劫持**就是一种窃取用户SessionID后，使用该SessionID登录进入目标账户的攻击方法，此时攻击者实际上是利用了目标账户的有效Session。**如果SessionID是被保存在Cookie中，则这种攻击被称为Cookie劫持。**



## (5) 会话保持攻击

- **Session是有生命周期的**，当用户长时间未活动后，或者用户点击退出后，服务器将销毁Session。
- **如果攻击者窃取了用户的Session，并一直保持一个有效的Session**（比如间隔性地刷新页面，以使服务器认为这个用户仍然在活动），而服务器对于活动的Session也一直不销毁，**攻击者就能通过此有效Session一直使用用户的账户，即成为一个永久的“后门”，这就是会话保持攻击。**



## (5) 会话保持攻击

- 下面一段代码，能保持Session始终有效。

```
<script>
var url="http://bbs.example.com/index.php?sid=1";
Window.setInterval("keepsid()",6000);//按照指定的周期（以毫秒计）
来调用函数或计算表达式
Fuction keepsid()
{
    Document.getElementById("a1").src=url+"&time="+Math.random();
}
</script>
<iframe id="a1" src=""></iframe>
```

IFrame内联框架被用来在当前 HTML 文档中嵌入另一个文档  
其原理就是不停地刷新页面，以保持Session不过期

### (1) 基本概念

- 不安全的直接对象引用在OWASP TOP 10排名中居于第四位，它可以被归于访问控制一类威胁。
- **直接对象引用**：是指WEB应用程序的开发人员将一些不应公开的对象引用直接暴露给用户，使得用户可以通过更改URL等操作直接引用对象。
- **不安全的直接对象引用**：是指一个用户通过更改URL等操作可以成功访问到未被授权的内容。比如一个网站上的用户通过更改URL可以访问到其他用户的私密信息和数据等。

## (2) 不安全的直接对象引用的原理

```
String query="SELECT * FROM accts WHERE account=?";  
PreparedStatement pstmt=connection.prepareStatement(query, ...);  
pstmt.setString(1,request.getParameter("acct"));  
ResultSet results=pstmt.executeQuery();
```

代码中通过一个**未被验证的用户账号**来获取相关数据，在这样的情况下，攻击者可以通过在浏览器中简单修改 “acct”参数的值发送到不同的用户账号来获取信息。





### 3 安全配置错误

举例：未能及时对软件进行更新，**默认的用户名密码没有及时修改**等等

### 4 不安全的加密存储

所谓不安全的加密存储指的是Web应用系统**没有对敏感性资料进行加密**，或者采用的加密算法复杂度不高可以被轻易破解，或者加密所使用的密钥非常容易检测出来。

举例：CSDN网站泄密、Facebook用户信息泄密等



当传输层没有进行安全保护时，会遇到下面的安全威胁

### 会话劫持

HTTP是无状态协议，客户端和服务端并没有建立长连接。服务器为了识别用户连接，服务器会发送给客户端SessionID。如果传输层保护不足，攻击就可以通过嗅探的方法获取传输内容，提取SessionID，冒充受害者发送请求。

### 中间人攻击

**中间人攻击(Man-in-the-middle attack)，即MITM。**HTTP连接的目标是Web服务器，如果传输层保护不足，攻击者可以担任中间人的角色，在用户和Web服务器之间截获数据并在两者之间进行转发，使用户和服务端之间的整个通信过程暴露在攻击者面前。

