day45-部署项目和爬虫

部署项目

一,概述

在企业中，一般都采用linux系统作为Web应用服务器，所以我们需要在linux系统搭建项目运行环境。在linux系统上搭建运行环境需要安装jdk、myql、tomcat相关软件。

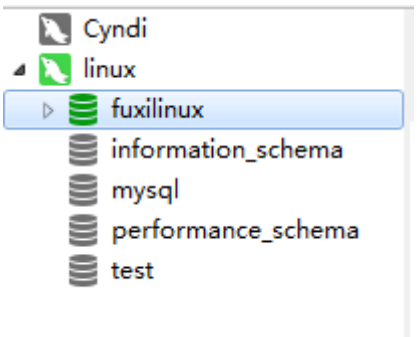
二,发布项目到服务器

1.linux系统myql导入数据库

- 首先将tomcat和mysql开启

```
[root@cyndi bin]# ps -ef|grep mysql
root      1374     1  0 23:02 ?        00:00:00 /bin/sh /usr/bin/mysqld_safe --datadir=/var/lib/mysql --pid-file=/var/lib/mysq
mysql     1469  1374  0 23:02 ?        00:00:00 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib/
ar/lib/mysql/cyndi.err --pid-file=/var/lib/mysql/cyndi.pid
root      1850  1727  0 23:18 pts/0    00:00:00 grep mysql
[root@cyndi bin]# ps -ef|grep tomcat
root      1802     1  5 23:16 pts/0    00:00:07 /usr/local/java/jdk1.8.0_181/bin/java -Djava.util.logging.config.file=/usr/loca
ng.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Djdk.tls.ephemeralDHKeySize=2048 -Djava.protoc
esources -Dorg.apache.catalina.security.SecurityListener.UMASK=0027 -Dignore.endorsed.dirs= -classpath /usr/local/tomcat/apache
cal/tomcat/apache-tomcat-8.5.32/bin/tomcat-juli.jar -Dcatalina.base=/usr/local/tomcat/apache-tomcat-8.5.32 -Dcatalina.home=/usr
a.io.tmpdir=/usr/local/tomcat/apache-tomcat-8.5.32/temp org.apache.catalina.startup.Bootstrap start
root      1852  1727  0 23:18 pts/0    00:00:00 grep tomcat
[root@cyndi bin]#
```

- 本地数据库导出，使用Navicat远程数据库操作



- linux系统mysql导入数据库文件
- 查看导入的表与数据

2. 导出war包部署到tomcat上

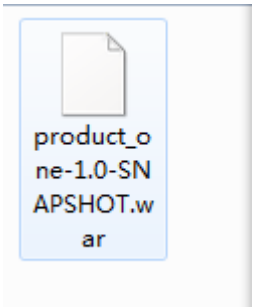
- 上传war文件 注意**c3p0-config.xml内的地址及密码是否正确**
- 如何修改mysql密码

```
方法2：用mysqladmin
格式：mysqladmin -u用户名 -p旧密码 password 新密码
例子：mysqladmin -uroot -p123456 password 123
```

- pwd查看绝对路径 移动war包到webapps目录后会自动解压

```
[root@cyndi ~]# mv product_one-1.0-SNAPSHOT.war /usr/local/tomcat/apache-tomcat-8.5.32/webapps
[root@cyndi ~]# cd /usr/local/tomcat/apache-tomcat-8.5.32/webapps
[root@cyndi webapps]# ll
总用量 78404
drwxr-x---. 14 root root     4096 8月   22 07:23 docs
drwxr-x---.  6 root root     4096 8月   22 07:23 examples
drwxr-x---.  5 root root     4096 8月   22 07:23 host-manager
drwxr-x---.  5 root root     4096 8月   22 07:23 manager
drwxr-x---. 10 root root     4096 9月    3 23:42 product_one-1.0-SNAPSHOT
-rw-r--r--.  1 root root 80257691 9月    3 15:32 product_one-1.0-SNAPSHOT.war
drwxr-x---.  3 root root     4096 8月   22 07:23 ROOT
```

- 打开浏览器浏览网站，注意war包名就是项目名 同时主要不要手残把war包删了




```
http://192.168.195.128:8080/product_one-1.0-SNAPSHOT
```

> 如果有乱码, 在数据库的url上添加编码:

```
<property name="jdbcUrl">jdbc:mysql://localhost:3306/travel-43?characterEncoding=utf-8</property>
```

3.解决第一次启动慢

- hostname #查看主机名
- vi /etc/hosts #修改本地dns配置, 在127.0.0.1 最后加上 " 自己的主机名"

```
"/etc/hosts" 2L, 158C
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4 itheima09
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
~
~
~
~
```

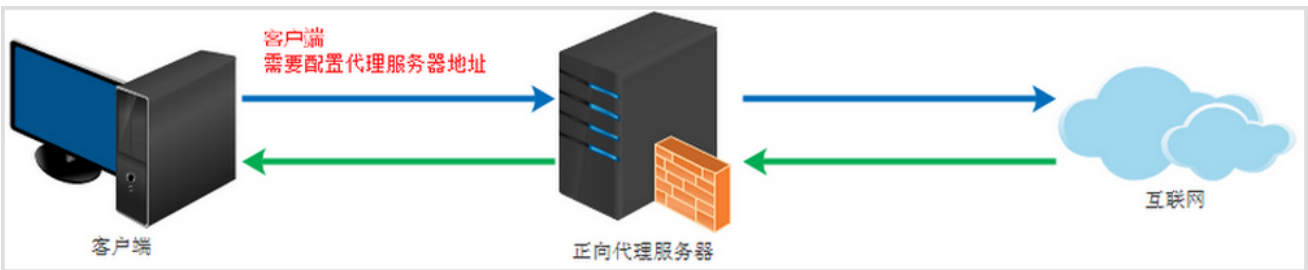
写上自己上面查询出来的

三,nginx反向代理

1 ,正向代理

1.1正向代理介绍

正向代理类似一个跳板机，代理访问外部资源。

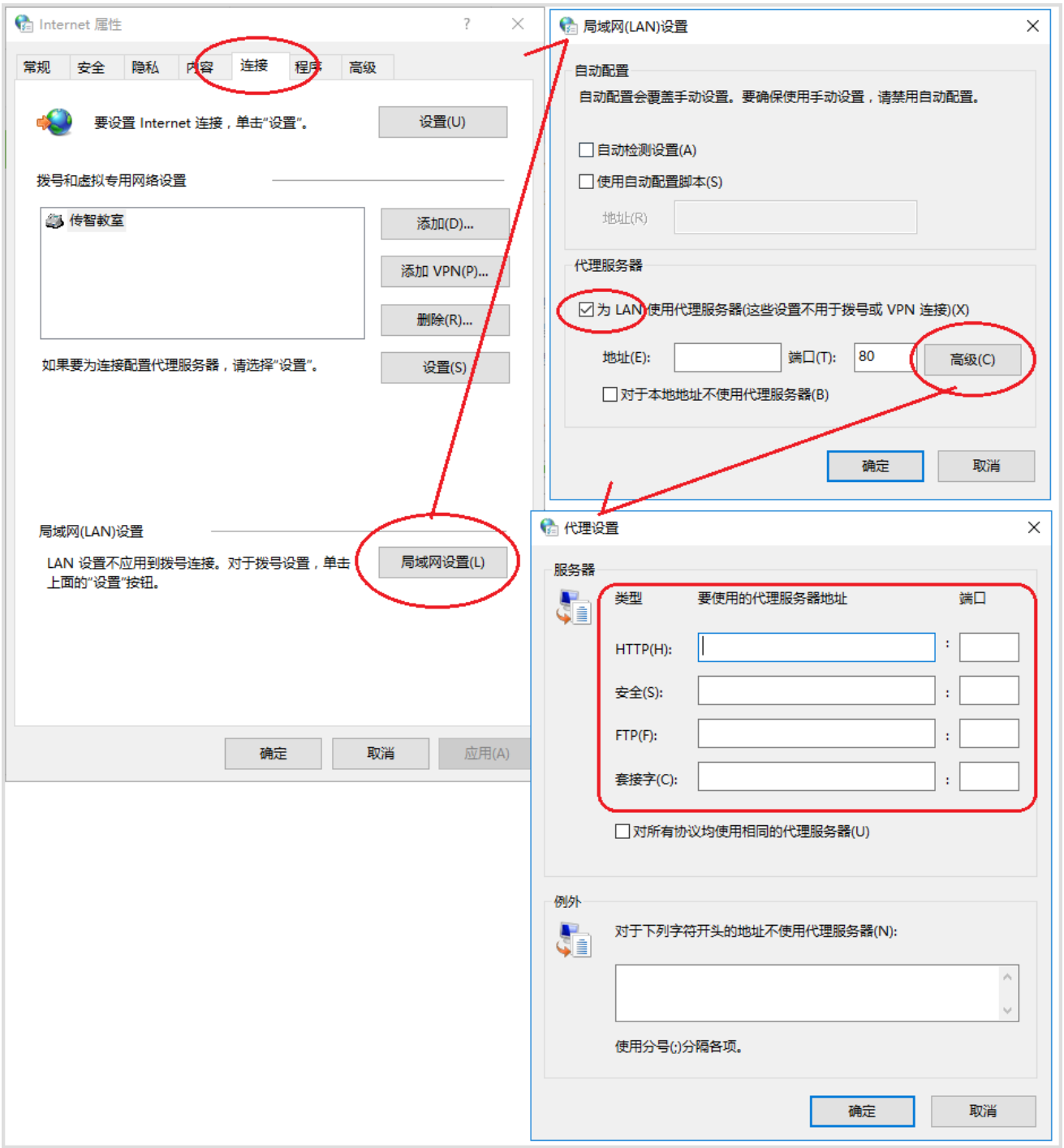


举个例子：

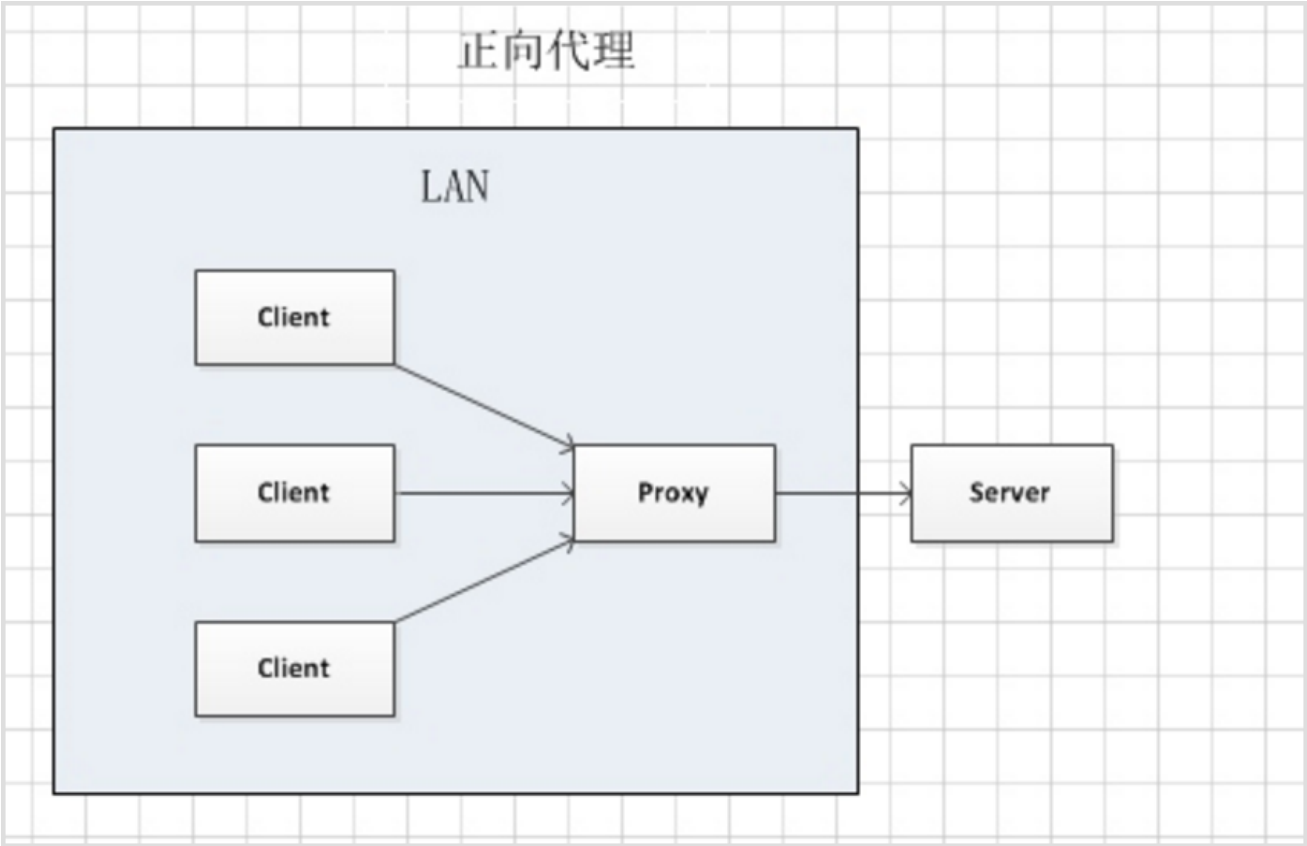
我是一个用户，我访问不了某网站，但是我能访问一个代理服务器，这个代理服务器呢,它能访问那个我不能访问的网站，于是我先连上代理服务器,告诉他我需要那个无法访问网站的内容，代理服务器去取回来,然后返回给我。从那个网站的角度来看，只在代理服务器来取内容的时候有一次记录，有时候并不知道是用户的请求，也隐藏了用户的资料，这取决于代理告不告诉网站。

注意：客户端必须设置正向代理服务器，当然前提是要知道正向代理服务器的IP地址，还有代理程序的端口。

本地客户端设置代理服务器操作如下：



1.2正向代理原理



总结来说：正向代理 是一个位于客户端和原始服务器(origin server)之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。客户端必须要进行一些特别的设置才能使用正向代理。

1.3正向代理的用途

（1）访问原来无法访问的资源，如google

（2）可以做缓存，加速访问资源

（3）对客户端访问授权，上网进行认证

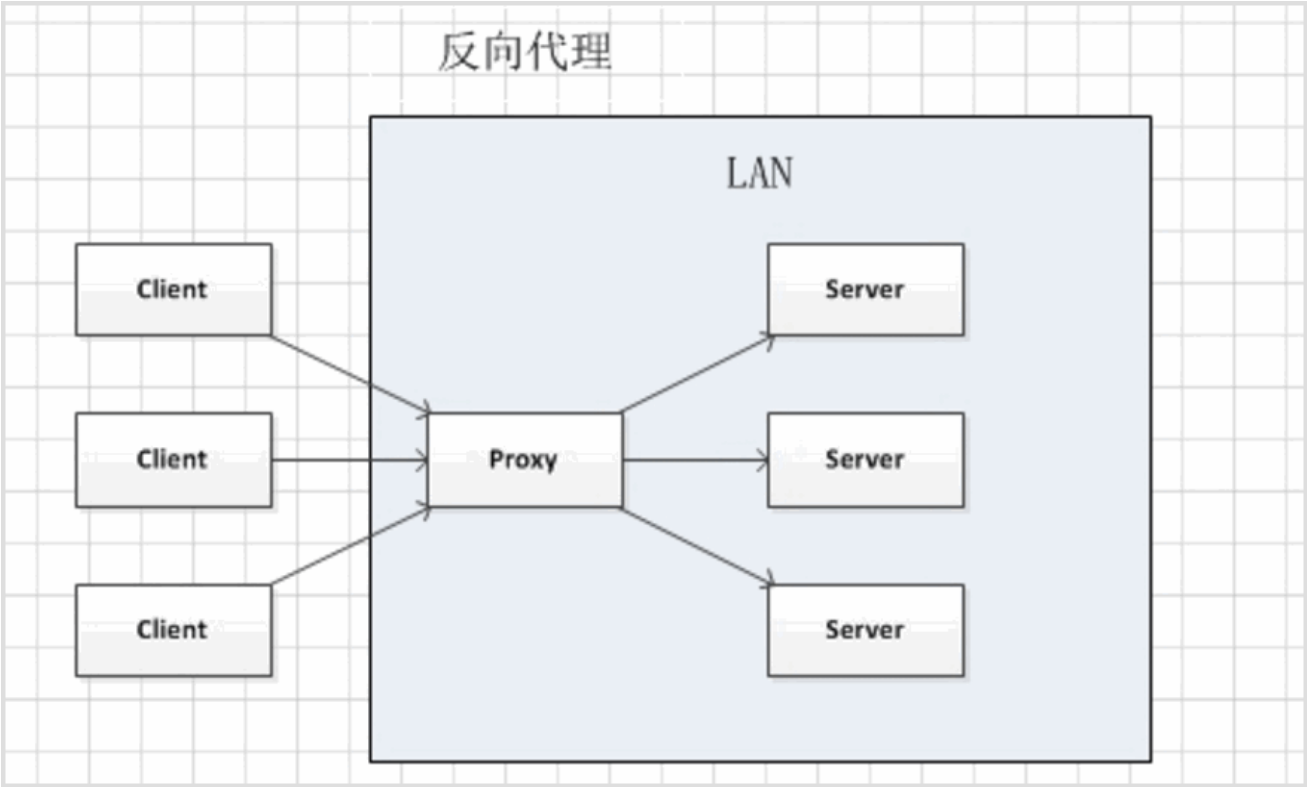
（4）代理可以记录用户访问记录（上网行为管理），对外隐藏用户信息

2.反向代理

2.1反向代理介绍

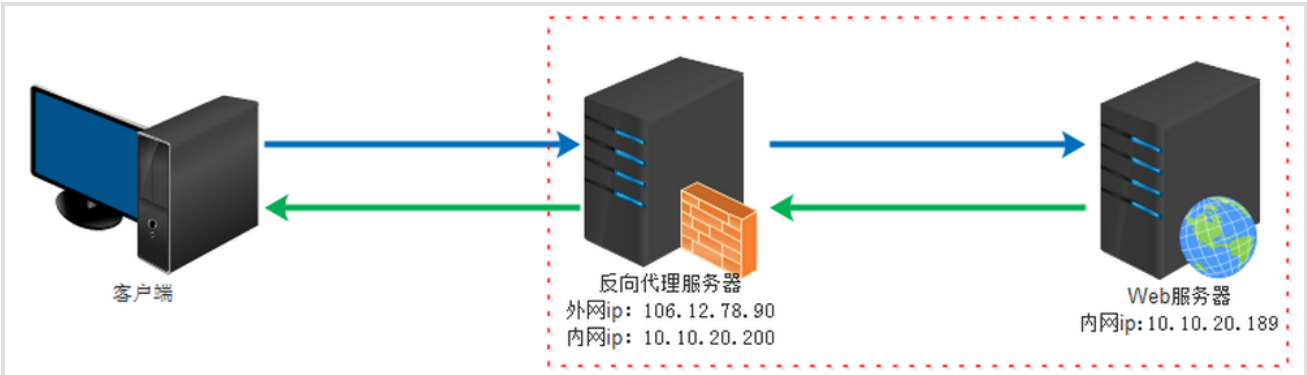
反向代理（Reverse Proxy）方式是指以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器。

2.2反向代理原理

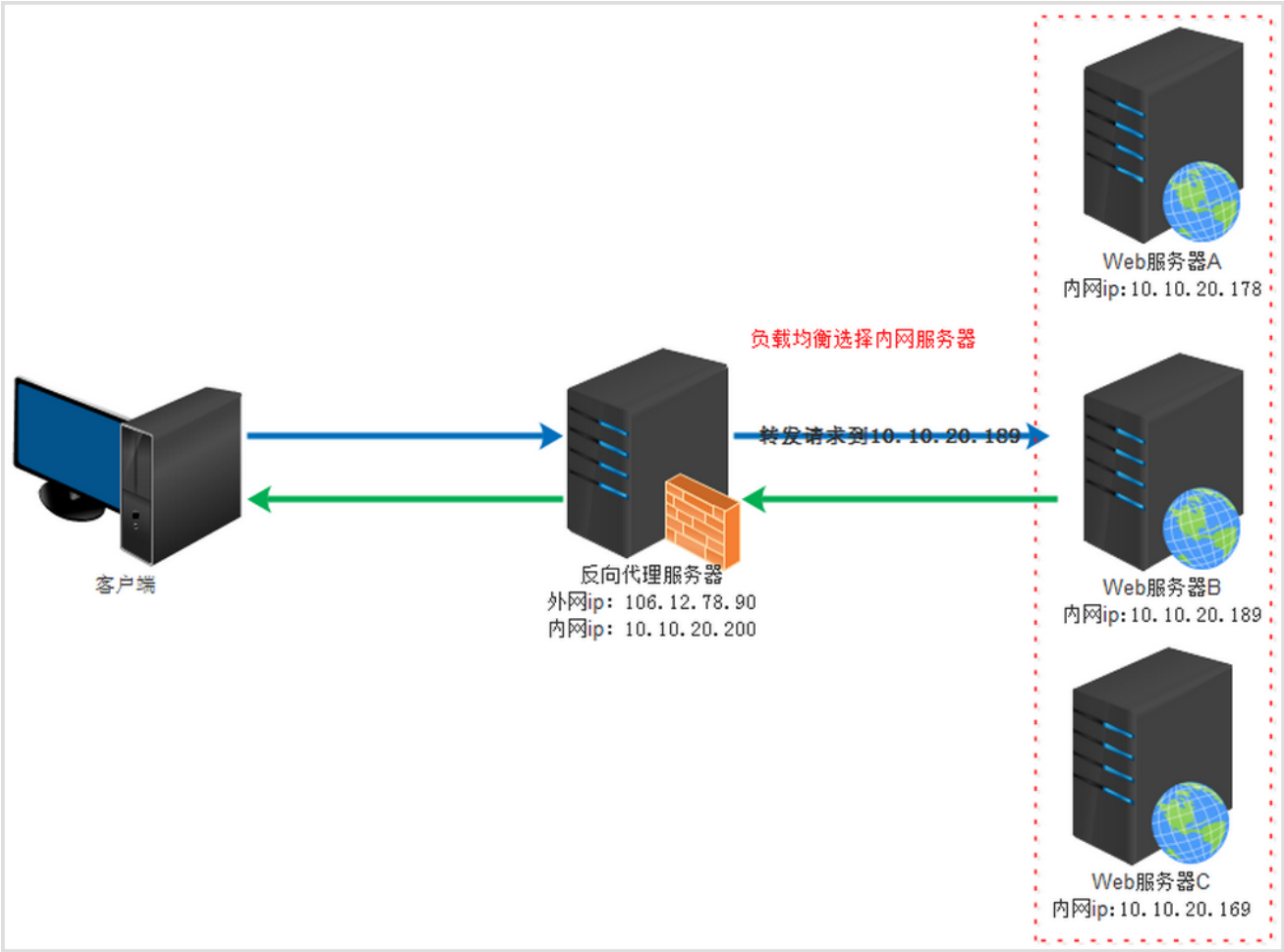


2.3反向代理的作用

（1）保证内网的安全，可以使用反向代理提供防火墙（WAF）功能，阻止web攻击。大型网站，通常将反向代理作为公网访问地址，Web服务器是内网。



（2）负载均衡，通过反向代理服务器来优化网站的负载。当大量客户端请求代理服务器时，反向代理可以集中分发不同用户请求给不同的web服务器进行处理请求。（本阶段不实现，负载均衡技术项目阶段讲解）



3. 反向代理实现原理与正向代理区别

正向代理中，proxy和client同属一个局域网（Local Area Network，LAN），对server透明；

反向代理中，proxy和server同属一个局域网（Local Area Network，LAN），对client透明；

实际上proxy在两种代理中做的事都是代为收发请求和响应，不过从结构上来看正好左右互换了下，所以把后出现的那种代理方式叫反向代理。

4.配置反向代理

- 步骤1：编辑配置文件

```
cd /usr/local/nginx/conf
vi nginx.conf
```

- 步骤2:修改配置文件nginx.conf　增加upstream test{server localhost:8080;}并修改server内容 #为注释

```
#增加反向代理tomcat
upstream test{server localhost:8080;}
server {
        listen 80;
        server_name localhost;

        location / {
            # root   html;
            # index  index.html index.htm;
            # 访问tomcat(使用反向代理)
            proxy_pass http://test;
         }
}
```

- 步骤3: 重启nginx

```
cd /usr/local/nginx/sbin
./nginx -s reload
```

- 步骤4：打开浏览器，远程浏览

```
http://192.168.147.160/travel-43/
```

5.nginx常见命令

```
#查看运行进程状态：
ps aux | grep nginx

#停止nginx:
./nginx -s stop

#重启nginx(配置文件变动后需要重启才能生效):
./nginx -s reload

#检查配置文件是否正确:
./nginx -t

#查看nginx版本
$ ./nginx -v
```

自动登录

一,思路分析

在login.html中增加复选框自动登录，登录后携带 auto字段到服务器

```
<label>
           <input type="checkbox" name="auto"/>自动登陆
</label>
```

在服务器中对auto做非空判断，然后新建Cookie将当前User对象的属性以字符串形式存到Cookie中，然后到过滤器中将字符串切割重新新建User对象，然后新建Session

```
public void loginuser(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        ResultMsg resultMsg=null;
        Map map=new HashMap();
        try {
            String auto = request.getParameter("auto");
            Map<String, String[]> parameterMap = request.getParameterMap();
            User user=new User();
            BeanUtils.populate(user,parameterMap);
            if(user==null||"".equals(user)){
                resultMsg=new ResultMsg(false,null,"检查OK");
            }else {
                User logeduser = resService.loguser(user);
                if (logeduser != null) {
                    if(auto!=null&&!"".equals(auto)){
                        Cookie autoCookie = new Cookie("autolog", user.getUsername() + "-" +
user.getPassword());
                        autoCookie.setMaxAge(3600);
                        autoCookie.setPath(request.getContextPath());
                        response.addCookie(autoCookie);
                    }
                    request.getSession().setAttribute("user", logeduser);
                    resultMsg = new ResultMsg(true, null, "登录成功");
                }else{
                    resultMsg = new ResultMsg(false, null, "登录失败");
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
            resultMsg=new ResultMsg(false,null,"登录失败");
        }  finally {
            String s = objectMapper.writeValueAsString(resultMsg);
            response.getWriter().print(s);
        }
    }
```

自动登录的灵魂是过滤器，浏览器请求服务器设置过滤器过滤所有路径，则浏览器在结果中可以得到Session值，实现自动登录功能

```
@WebFilter("/*")
public class LoginFilter implements Filter {
    public void destroy() {

    }
```

```
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws
ServletException, IOException {
        HttpServletRequest request= (HttpServletRequest) req;
        HttpServletResponse response= (HttpServletResponse) resp;
        User existsuser = (User) request.getSession().getAttribute("user");
        if(existsuser==null){
        Cookie[] cookies =  request.getCookies();
        Cookie cookie = CookieUtils.getCookie(cookies, "autolog");
        if(cookie!=null) {
            String cookieValue = cookie.getValue();
            String user = cookieValue.split("-")[0];
            String psw = cookieValue.split("-")[1];
            User loguser = new User(user, psw);
            ResService resService = new ResService();
            User returnuser = resService.loguser(loguser);
            if (returnuser != null) {
                request.getSession().setAttribute("user", returnuser);
            }
        }
        }
        chain.doFilter(request, response);
    }

    public void init(FilterConfig config) throws ServletException {
    }
}
```

CookieUtils工具类

```
public class CookieUtils {
    public static Cookie getCookie(Cookie[]cookies,String key){
     if(cookies==null||key==null||"".equals(key)){
         return null;
     }
        for (Cookie cookie : cookies) {
         if(key.equals(cookie.getName())){
             return cookie;
         }
        }
        return null;
    }
}
```

爬虫案例

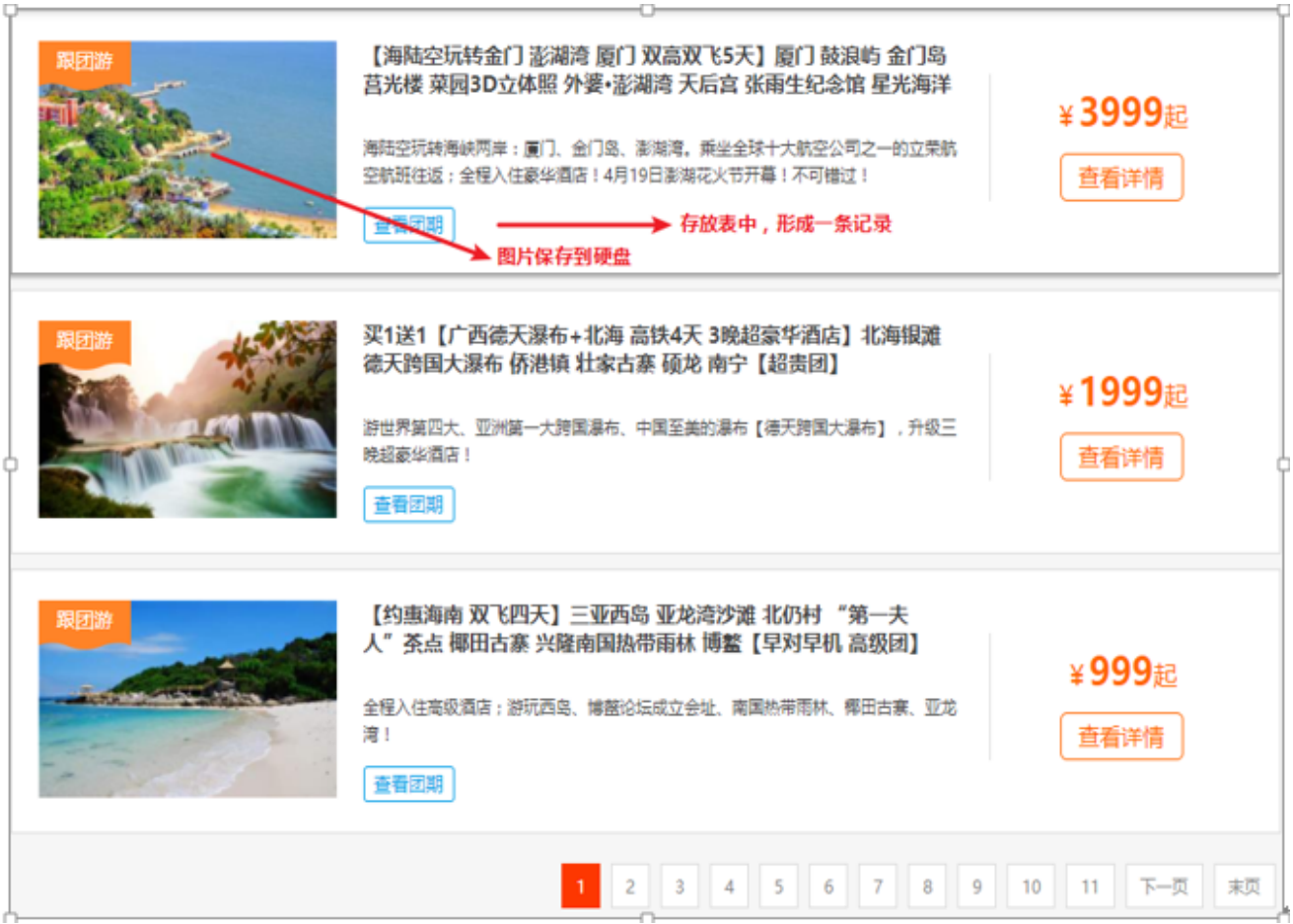
一,案例需求

在金马国旅首页搜索框搜索“国内”，

网址 `http://www.jinmalvyou.com/search/index/view_type/1/keyword/%E5%9B%BD%E5%86%85`

将页面数据收集到javaweb数据库tab_route表中，

其中将商品缩略图图片收集到本地硬盘上G:\img目录下。



二.技术分析

1.爬虫概述

1.1什么是爬虫

当我们有什么技术问题不会时，经常百度一下。那么百度是如何根据我们搜索的问题再浩瀚的互联网中查找答案的呢？

难道是点击百度搜索立刻去互联网所有资源中去搜索数据？

当用户点击百度搜索引擎查询时，百度不是立刻去互联网中搜索资源，而是搜索百度自己服务器内部数据库数据，但是百度有专门的服务器利用爬虫技术在浩瀚的互联网资源中收集数据（url、title、description、其他等信息）到内部数据库中。

这个收集互联网数据的技术就是网络爬虫技术

1.2网络爬虫技术的好处

- 可以实现搜索引擎
- 大数据时代，可以让我们获取更多的数据源
- 可以更好地进行搜索引擎优化
- 有利于就业

2.JSOP概述

2.1什么是JSOP

Jsoup是一款基于Java 的HTML（html也是XML文档的一种）解析器，可直接解析某个URL地址、HTML文本内容。

它提供了一套非常省力的API，可通过DOM，CSS以及类似于jQuery的操作方法来取出和操作数据。

2.1 Jsoup的应用场景

1. 解析XML文档并操作和管理XML元素、属性、元素体等数据。
2. 解析HTML页面并进行数据收集，例如：爬虫

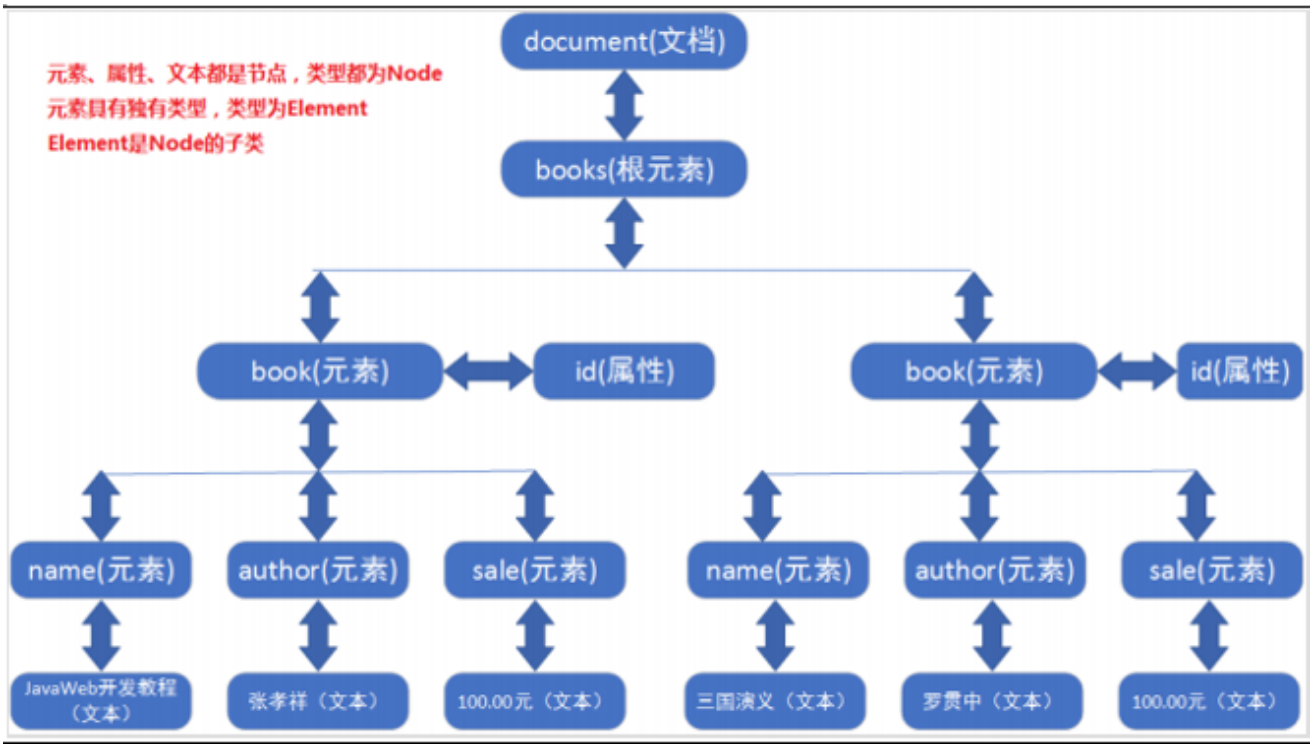
2.3 JSOP解析原理

XML DOM 和 HTML DOM一样，XML DOM 将整个文档(html,xml)加载到内存，生成一个DOM树，并获得一个Document对象，通过Document对象就可以对DOM进行操作。

- 以下面books.xml文档为例：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<books>
    <book id="it0001">
        <name>JavaWeb开发教程</name>
        <author>张孝祥</author>
        <sale>100.00元</sale>
    </book>
    <book id="it0002">
        <name>三国演义</name>
        <author>罗贯中</author>
        <sale>100.00元</sale>
    </book>
</books>
```

- 生成对应的结构模型



3.JSOP的基本使用

3.1使用步骤

1. 添加坐标依赖

```xml
   <dependency>
      <groupId>org.jsoup</groupId>
      <artifactId>jsoup</artifactId>
      <version>1.11.2</version>
    </dependency>
```

2. 获得document对象

3.2获得document对象

- 方式一:根据XML文档字符串获取Document

```
String strXML = "<?xml version='1.0' encoding='UTF-8'?><book><name>JavaWeb入门</name></book>";
Document document = Jsoup.parse(strXML);
```

- 方式二:根据文件获取Document

```
String path = JsoupTest01.class.getClassLoader().getResource("index.html").getPath();
File file = new File(path);
Document document = Jsoup.parse(file, "utf-8");
```

- 方式三:根据url获取Document

```
String url = "http://www.baidu.com";
Document document = Jsoup.connect(url).get();
```

3.3常用的解析方法

方法类型	返回类型	方法	说明
获取元素方法	org.jsoup.select.Elements	getElementsByTag(String tagName)	根据标签名获取一组标签名称一样的元素列表
获取元素方法	org.jsoup.select.Elements	getElementsByClass(String className)	根据元素中属性class的值为className元素的列表（该方法是HTML文档解析的特有方法）
获取元素方法	org.jsoup.nodes.Element	getElementById(String id)	根据指定id获取一个元素对象
获取元素方法	org.jsoup.select.Elements	children()	获取当前元素的子元素列表
获取元素方法	org.jsoup.nodes.Node	parent()	获取当前元素的父节点对象
获取数据方法	java.lang.String	text()	获取元素体文本数据
获取数据方法	java.lang.String	html()	获取元素体里面所有所有数据（包含子元素和文本数据，该方法也是HTML文档解析的特有方法）
获得属性	org.jsoup.nodes.Attributes	attributes()	获得当前元素的所有属性
获得属性值	java.lang.String	attr()	获得指定属性的属性值

- 练习

```
/**
 * Document对象常用方法
 * 需求1：获取index.html中所有h3标签名称的元素列表,获得h3标签里面的文本
 * 需求2：获取index.html中所有元素含有class属性值为item,并且获得标签里面的html内容
 * 需求3：获取index.html中元素属性id="footer"的一个元素,并且获得其所有的属性
 * 需求4：获取index.html中元素属性id="footer"的元素的父元素
 * 需求5：获取index.html中元素属性id="footer"的元素的所有子元素
 */
```

```java
public class JsoupTest02 {
    @Test
    //需求1：获取index.html中所有h3标签名称的元素列表,获得h3标签里面的文本
    public void fun01() throws Exception {
        String path = JsoupTest01.class.getClassLoader().getResource("index.html").getPath();
        File file = new File(path);
        Document document = Jsoup.parse(file, "utf-8");

        Elements h3Eles = document.getElementsByTag("h3");
        for (Element h3Ele : h3Eles) {
            System.out.println(h3Ele.text());
        }
    }

    @Test
    //需求2：获取index.html中所有元素含有class属性值为item,并且获得标签里面的html内容
    public void fun02() throws Exception {
        String path = JsoupTest01.class.getClassLoader().getResource("index.html").getPath();
        File file = new File(path);
        Document document = Jsoup.parse(file, "utf-8");

        Elements elements = document.getElementsByClass("item");
        for (Element element : elements) {
            System.out.println(element.html());

        }
    }

    @Test
    //需求3：获取index.html中元素属性id="footer"的一个元素,并且获得其所有的属性

    public void fun03() throws Exception {
        String path = JsoupTest01.class.getClassLoader().getResource("index.html").getPath();
        File file = new File(path);
        Document document = Jsoup.parse(file, "utf-8");

        Element footerEle = document.getElementById("footer");

        Attributes attributes = footerEle.attributes();
        for (Attribute attribute : attributes) {
            System.out.println(attribute.getKey() + ":" + attribute.getValue());
        }
    }

    @Test
    // 需求4：获取index.html中元素属性id="footer"的元素的父元素
    public void fun04() throws Exception {
        String path = JsoupTest01.class.getClassLoader().getResource("index.html").getPath();
        File file = new File(path);
        Document document = Jsoup.parse(file, "utf-8");

        Element footerEle = document.getElementById("footer");

        Element parent = footerEle.parent();
```

```
        System.out.println(parent.nodeName());
    }

    @Test
    // 需求5：获取index.html中元素属性id="footer"的元素的所有子元素
    public void fun05() throws Exception {
        String path = JsoupTest01.class.getClassLoader().getResource("index.html").getPath();
        File file = new File(path);
        Document document = Jsoup.parse(file, "utf-8");

        Element footerEle = document.getElementById("footer");
        Elements childrens = footerEle.children();
        for (Element children : childrens) {
            System.out.println(children.nodeName());
        }
    }

}
```

4,Jsoup选择器解析

4.1概述

选择器大家并不陌生，Jsoup提供的API方法使得我们可以类似jQuery、CSS选择器一样快速查找出想要的元素内容，接下来就要我们一起学习Jsoup选择器解析XML/HTML文档技术。

4.2选择器解析语法

4.2.1获得Element语法

- Document对执行选择器方法API介绍

方法	返回类型	说明
document.select(String query)	Elements	query是选择器，select方法是根据选择器查询元素列表

- Elements对象API方法介绍

方法	返回类型	说明
firist()	org.jsoup.select.Element	获取元素列表中的第一个元素对象
last()	org.jsoup.select.Element	获取元素列表中的最后一个元素对象
eq(int index)	org.jsoup.select.Element	获取元素列表中的指定索引的元素对象

4.2.2选择器语法

- 基本选择器

选择器类型	选择器代码	说明
ID选择器	#id	根据id获取元素，id前面使用#符号
class选择器	.class	根据元素class属性获取元素
标签选择器	div	根据元素名称获取元素
属性选择器	[attr]	获取含有该属性名为attr的所有元素
属性选择器	[^attr]	利用属性名attr开头的前缀来查找元素
属性选择器	[attr=value]	利用属性值来查找元素
属性选择器	[attr^=value], [attr$=value], [attr*=value]	利用匹配属性值开头、结尾或包含属性值来查找元素

- 组合选择器

选择器代码	说明
el.class	定位class为指定值的元素 例如 div.head -> xxxx
el[arrt]	定义所有包含某属性的元素, 比如： a[href]
parent child	找某个元素下的所有子元素，选择器之间使用空格隔开
parent > child	找某个元素下的所有直接子元素
siblingA + siblingB	查找在A元素后面第一个同级元素B，比如：div.head + div
siblingA ~ siblingX	查找A元素后面所有同级X元素，比如：h1 ~ p

- 伪类选择器

选择器代码	说明
:gt(n)	查找大于索引找元素
:eq(n)	找到索引值等于 指定索引值的元素
:has(selector)	查找子元素匹配选择器
:not(selector)	查找选择器不匹配的元素，或者排除指定的选择器元素

三.案例分析

一个url资源地址可以使用Jsoup解析获的一个Document对象，国内搜索结果列表页面地址http://www.jinmalvyou.com/search/index/view_type/1/keyword/%E5%9B%BD%E5%86%85，使用Jsoup根据

url获得一个Document对象进行解析，但问题是这里只能获得第1页数据，如果获取其他页的旅游线路商品列表数据呢？

点击分页数字观察发现，如下所示：

- 第1页url：http://www.jinmalvyou.com/search/index/view_type/1/keyword/%E5%9B%BD%E5%86%85
- 第2页url：http://www.jinmalvyou.com/search/index/view_type/1/keyword/%E5%9B%BD%E5%86%85/p/2.html
- 第3页url：http://www.jinmalvyou.com/search/index/view_type/1/keyword/%E5%9B%BD%E5%86%85/p/3.html
- 第30页url：http://www.jinmalvyou.com/search/index/view_type/1/keyword/%E5%9B%BD%E5%86%85/p/30.html

  根据上面的分页数据列表地址，可以分析出分页数字1~32页。每一页的旅游线路列表数据都是一个独立的地址，所以我们需要遍历1~32所有页，获得每一页Document对象进行解析抓取旅游线路列表数据。

四.代码实现

1.环境的准备

- 添加坐标依赖

```
<dependencies>
        <!--单元测试-->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
        <!--JSOUP-->
        <dependency>
            <groupId>org.jsoup</groupId>
            <artifactId>jsoup</artifactId>
            <version>1.11.2</version>
        </dependency>
        <!--mysql驱动-->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.26</version>
            <scope>compile</scope>
        </dependency>
        <!--c3p0连接池-->
        <dependency>
            <groupId>c3p0</groupId>
```

```
            <artifactId>c3p0</artifactId>
            <version>0.9.1.2</version>
        </dependency>
        <!--jdbcTemplate-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>4.1.2.RELEASE</version>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>4.1.2.RELEASE</version>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>4.1.2.RELEASE</version>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>4.1.2.RELEASE</version>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
            <version>1.1.1</version>
            <scope>compile</scope>
        </dependency>
        <!--HttpClient-->
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpclient</artifactId>
            <version>4.5.3</version>
        </dependency>
        <!--IO流-->
        <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
            <version>2.5</version>
        </dependency>
    </dependencies>
```

- 数据库

```
/*旅游线路商品表*/

CREATE TABLE tab_route(
```

```
    rid INT NOT NULL AUTO_INCREMENT,
    rname VARCHAR(100) NOT NULL,
    price DOUBLE NOT NULL,
    routeIntroduce VARCHAR(200),
    rflag CHAR(1) NOT NULL,
    rdate VARCHAR(19),
    isThemeTour CHAR(1) NOT NULL,
    count INT DEFAULT 0,
    cid INT NOT NULL,
    rimage VARCHAR(200),
    sid INT,
    sourceId VARCHAR(50),
    PRIMARY KEY (rid),
    UNIQUE KEY AK_nq_sourceId (sourceId)
);
```

- Route.java

```
/**
 * 旅游线程
 */
public class Route implements java.io.Serializable{

    private int rid;//线路id
    private String rname;//线路名称
    private double price;//价格
    private String routeIntroduce;//线路介绍
    private String rflag; //是否上架，0代表没有上架，1代表是上架
    private String rdate; //上架时间
    private String isThemeTour;//是否主题旅游，0代表不是，1代表是
    private String sourceId;//数据的来源id
    private int cid;//所属分类
    private int sid;//所属商家
    private int count;//收藏数量
    private String rimage;//缩略图
    ...
}
```

2代码实现

- RounteService.java

```
package com.itheima.test;

import com.itheima.bean.Route;
import com.itheima.dao.RouteDao;
import com.itheima.utils.HttpClientUtils;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;

import org.jsoup.select.Elements;
```

```
import org.junit.Test;

import java.net.URLEncoder;
import java.text.SimpleDateFormat;
import java.util.Date;

public class CrawlerDemo {

    @Test
    public void fun01() throws Exception {

        RouteDao routeDao = new RouteDao();
        //1. 创建document对象
        String url="http://www.jinmalvyou.com/search/index/view_type/1/keyword/"+
URLEncoder.encode("国内","utf-8");
        Document document = Jsoup.connect(url).get();
        //2. 获得li元素对象集合
        Elements liElements = document.select(".rl-b-li");
        //3. 遍历li元素对象集合
        for (Element liElement : liElements) {
            //4. 没遍历一次 封装成一个Rounte对象
            Route route = new Route();

            //a 封装sourceId
            Element aEle = liElement.select("a").first();
            String hrefAttibute = aEle.attr("href");
            hrefAttibute = hrefAttibute.substring(hrefAttibute.indexOf("=")+1);
            route.setSourceId(hrefAttibute);

            //b.封装rname
            Element imgEle = aEle.select("img").first();
            String rname = imgEle.attr("title");
            route.setRname(rname);

            //c.封装价格
            String price = liElement.select("strong").first().text();
            route.setPrice(Double.parseDouble(price));

            //d. 封装线路介绍
            Elements pEle = liElement.select("p").eq(1);
            String introduce = pEle.attr("title");
            route.setRouteIntroduce(introduce);

            //e,封装缩略图路径
            String imgSrc = imgEle.attr("src");
            String rImage =  imgSrc.substring(imgSrc.lastIndexOf("/")+1);
            route.setRimage(rImage);

            //封装是否上架
            String rflag = "1";
            route.setRflag(rflag);


            //封装上架时间
```

```
            String rdate = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());
            route.setRdate(rdate);

            //封装是否主题旅游
            String isThemeTour = "1";
            route.setIsThemeTour(isThemeTour);

            //封装cid
            int cid = 1;
            route.setCid(cid);

            //封装sid
            int sid = 1;
            route.setSid(sid);

            //封装数量
            int count = 100;
            route.setCount(count);

            //图片下载
            String path = "G:/img/"+rImage;
            System.out.println("imgSrc="+imgSrc);
            imgSrc = "http:"+imgSrc; //文件下载时候,需要带上http: 不然会报错

            HttpClientUtils.downloadFile(imgSrc,path);

            routeDao.addRoute(route);
        }
    }
}
```

- RouteDao

```
public class RouteDao {

    private JdbcTemplate jdbcTemplate = new JdbcTemplate(C3P0Util.getDataSource());

    public void addRoute(Route route) throws SQLException {
        String sql = " insert into
tab_route(rname,price,routeIntroduce,rflag,rdate,isThemeTour,count,cid,rimage,sid,sourceId)" +
                     " values(?,?,?,?,?,?,?,?,?,?,?)";
        Object[] params = {
                route.getRname(),
                route.getPrice(),
                route.getRouteIntroduce(),
                route.getRflag(),
                route.getRdate(),
                route.getIsThemeTour(),
                route.getCount(),
                route.getCid(),
                route.getRimage(),

                route.getSid(),
```

```
                route.getSourceId(),
        };
        jdbcTemplate.update(sql,params);
    }
}
```