1.java 基础知识

1 集合的排序

1. API 方法介绍

1. method

实现 Comparator 接口，重写 compare 方法

```
int compare(T  o1,
            T  o2)
```

参数：
o1 - 要比较的第一个对象。
o2 - 要比较的第二个对象。
返回：
根据第一个参数小于、等于或大于第二个参数分别返回负整数、零或正整数。
抛出：
ClassCastException - 如果参数的类型不允许此 Comparator 对它们进行比较。

2.method

在创建商品类时实现 Comparator 方法，重写 comparable 方法

int **compareTo**(T o)

参数：
o - 要比较的对象。
返回：
负整数、零或正整数，根据此对象是小于、等于还是大于指定对象。
抛出：
ClassCastException - 如果指定对象的类型不允许它与此对象进行比较。

2.示例

Method_1_example

```
package com.Tset;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import org.junit.Test;
public class GoodsComparator {
	class Goods {
		private String GoodsID;
		private String GoodsName;
		private double GoodsPrice;
		private int GoodsNum;
		public Goods() {
				}
		public Goods(String goodsID, String goodsName, double goodsPrice,
				int goodsNum) {
			super();
			GoodsID = goodsID;
			GoodsName = goodsName;
			GoodsPrice = goodsPrice;
			GoodsNum = goodsNum;
		}

		public String getGoodsID() {
			return GoodsID;
		}

		public void setGoodsID(String goodsID) {
			GoodsID = goodsID;
		}

		public String getGoodsName() {
			return GoodsName;
		}

		public void setGoodsName(String goodsName) {
			GoodsName = goodsName;
		}
```

```
    public double getGoodsPrice() {
        return GoodsPrice;
    }

    public void setGoodsPrice(double goodsPrice) {
        GoodsPrice = goodsPrice;
    }

    public int getGoodsNum() {
        return GoodsNum;
    }

    public void setGoodsNum(int goodsNum) {
        GoodsNum = goodsNum;
    }
}
@Test
public void showAsGoodsPriceUpTest() {
    ArrayList<Goods> goodList = new ArrayList<Goods>();
    Goods goods_1 = new Goods("s001", "小雪碧", 3, 1000);
    goodList.add(goods_1);
    Goods goods_2 = new Goods("s002", "可口可乐", 4, 2000);
    goodList.add(goods_2);
    Goods goods_3 = new Goods("s003", "王老吉", 5, 3000);
    goodList.add(goods_3);
    Goods goods_4 = new Goods("s004", "加多宝", 5, 8);
    goodList.add(goods_4);
    Goods goods_5 = new Goods("s005", "大雪碧", 7, 100);
    goodList.add(goods_5);
    class ComparatorGoods implements Comparator<Goods> {
        @Override
        public int compare(Goods goods1, Goods goods2) {
            return goods1.getGoodsNum() - goods2.getGoodsNum();
        }
    }
    ComparatorGoods comparator = new ComparatorGoods();
    Collections.sort(goodList, comparator);//传递一个集合，一个比较方法
    System.out.println("商品编号" + '\t' + "商品名称" + '\t' + "商品价格" + '\t'
            + "商品数量");
    for (Goods goods : goodList) {
        System.out.print(goods.getGoodsID() + "\t");
        System.out.print(goods.getGoodsName() + "\t");
        System.out.print(goods.getGoodsPrice() + "\t");
        System.out.println(goods.getGoodsNum());
```

```
        }

    }
}
```

Method_2_example

```
package com.Tset;

import java.util.ArrayList;
import java.util.Collections;
import org.junit.Test;

public class GoodsComparator {
    class Goods implements Comparable<Goods>{
        private String GoodsID;
        private String GoodsName;
        private double GoodsPrice;
        private int GoodsNum;
        public Goods() {
            // TODO Auto-generated constructor stub
        }
        public Goods(String goodsID, String goodsName, double goodsPrice,
                int goodsNum) {
            super();
            GoodsID = goodsID;
            GoodsName = goodsName;
            GoodsPrice = goodsPrice;
            GoodsNum = goodsNum;
        }

        public String getGoodsID() {
            return GoodsID;
        }

        public void setGoodsID(String goodsID) {
            GoodsID = goodsID;
        }

        public String getGoodsName() {
            return GoodsName;
```

```
    }

    public void setGoodsName(String goodsName) {
        GoodsName = goodsName;
    }

    public double getGoodsPrice() {
        return GoodsPrice;
    }

    public void setGoodsPrice(double goodsPrice) {
        GoodsPrice = goodsPrice;
    }

    public int getGoodsNum() {
        return GoodsNum;
    }

    public void setGoodsNum(int goodsNum) {
        GoodsNum = goodsNum;
    }
    @Override
    public int compareTo(Goods o) {
        // TODO Auto-generated method stub
        return this.GoodsNum-o.GoodsNum;
    }
}

@Test
public void showAsGoodsPriceUpTest() {
    ArrayList<Goods> goodList = new ArrayList<Goods>();
    Goods goods_1 = new Goods("s001", "小雪碧", 3, 1000);
    goodList.add(goods_1);
    Goods goods_2 = new Goods("s002", "可口可乐", 4, 2000);
    goodList.add(goods_2);
    Goods goods_3 = new Goods("s003", "王老吉", 5, 3000);
    goodList.add(goods_3);
    Goods goods_4 = new Goods("s004", "加多宝", 5, 8);
    goodList.add(goods_4);
    Goods goods_5 = new Goods("s005", "大雪碧", 7, 100);
    goodList.add(goods_5);
    Collections.sort(goodList);     //只传递一个集合对象
    System.out.println("商品编号" + '\t' + "商品名称" + '\t' + "商品价格" + '\t'
            + "商品数量");
```

```
            for (Goods goods : goodList) {
                System.out.print(goods.getGoodsID() + "\t");
                System.out.print(goods.getGoodsName() + "\t");
                System.out.print(goods.getGoodsPrice() + "\t");
                System.out.println(goods.getGoodsNum());
            }

        }
}
```

2.系统暂停

1.API 介绍

类 Thread

静态方法

static void	**sleep**(long millis) 在指定的毫秒数内让当前正在执行的线程休眠（暂停执行），此操作受到系统计时器和调度程序精度和准确性的影响。
static void	**sleep**(long millis, int nanos) 在指定的毫秒数加指定的纳秒数内让当前正在执行的线程休眠（暂停执行），此操作受到系统计时器和调度程序精度和准确性的影响。

2.example

```
    public static void sleep(int time) {
        try {
```

```
                Thread.sleep(time);
            } catch (InterruptedException e) {
            e.printStackTrace();                    //功能是输出当前异常的对象使用堆栈的轨迹
        }
}
```

3.throw 与 throws

1.throw 用来抛出自定义异常

```
throw new RuntimeException(e);
```

2.throws 用来向上层抛出异常，本方法不作处理

```
public void method() throws Exception{}
```

3.附加说明

public static native void sleep(long millis) throws InterruptedException;

throws 语气出现在函数头中，用来表明该成员函数可能抛出各种异常。调用该方法的 **java** 语句就必须包含在 **try-catch** 语句块中

throw 语句用来明确抛出异常

try{

}

catch(异常类型 异常对象){

}

Finally{

}

4.Java 正则表达式详解（重要）

1. 正则表达式介绍

一、正则表达式基础知识 （文档参见 http://www.jb51.net/article/16829.htm）

我们先从简单的开始。假设你要搜索一个包含字符“cat”的字符串，搜索用的正则表达式就是“cat”。如果搜索对大小写不敏感，单词“catalog”、“Catherine”、“sophisticated”都可以匹配。也就是说：

```
正则表达式：cat
匹配：cat，catalog，Catherine，sophisticated
```

1.1 句点符号

假设你在玩英文拼字游戏，想要找出三个字母的单词，而且这些单词必须以“t”字母开头，以“n”字母结束。另外，假设有一本英文字典，你可以用正则表达式搜索它的全部内容。要构造出这个正则表达式，你可以使用一个通配符——句点符号“.”。这样，完整的表达式就是“t.n”，它匹配“tan”、“ten”、“tin”和“ton”，还匹配“t#n”、“tpn”甚至“t n”，还有其他许多无意

义的组合。这是因为句点符号匹配所有字符，包括空格、Tab 字符甚至换行符：

```
正则表达式：t.n
匹配：tan，Ten，tin，ton，t n，t#n，tpn，等
```

1.2 方括号符号

为了解决句点符号匹配范围过于广泛这一问题，你可以在方括号（“[]”）里面指定看来有意义的字符。此时，只有方括号里面指定的字符才参与匹配。也就是说，正则表达式“t[aeio]n”只匹配“tan”、“Ten”、“tin”和“ton”。但“Toon”不匹配，因为在方括号之内你只能匹配单个字符：

```
正则表达式：t[aeio]n
匹配：tan，Ten，tin，ton
```

1.3 “或”符号

如果除了上面匹配的所有单词之外，你还想要匹配“toon”，那么，你可以使用“|”操作符。“|”操作符的基本意义就是“或”运算。要匹配“toon”，使用“t(a|e|i|o|oo)n”正则表达式。这里不能使用方扩号，

因为方括号只允许匹配单个字符；这里必须使用圆括号“()”。圆括号还可以用来分组，具体请参见后面介绍。

1.4 表示匹配次数的符号

表一显示了表示匹配次数的符号，这些符号用来确定紧靠该符号左边的符号出现的次数：

表一：表示次数的符号

符号	次数
*	0 次或者多次
+	1 次或者多次
?	0 次或者 1 次
{n}	恰好 n 次
{n,m}	从 n 次到 m 次

假设我们要在文本文件中搜索美国的社会安全号码。这个号码的格式是 999-99-9999。用来匹配它的正则表达式如图一所示。在正则表达式中，连字符（“-”）有着特殊的意义，它表示一个范围，比如从 0 到 9。因此，匹配社会安全号码中的连字符号时，它的前面要加上一个转义字符“\”。

图一：匹配所有 123-12-1234 形式的社会安全号码

假设进行搜索的时候，你希望连字符号可以出现，也可以不出现——即，999-99-9999 和 999999999 都属于正确的格式。这时，你可以在连字符号后面加上“？”数量限定符号，如图二所示：

图二：匹配所有 123-12-1234 和 123121234 形式的社会安全号码

下面我们再来看另外一个例子。美国汽车牌照的一种格式是四个数字加上二个字母。它的正则表达式前面是数字部分“[0-9]{4}”，再加上字母部分“[A-Z]{2}”。图三显示了完整的正则表达式。

图三：匹配典型的美国汽车牌照号码，如 8836KV

1.5 “否”符号

“ˆ”符号称为“否”符号。如果用在方括号内，“ˆ”表示不想要匹配的字符。例如，图四的正则表达式匹配所有单词，但以“X”字母开头的单词除外。

图四：匹配所有单词，但“X”开头的除外

1.6 圆括号和空白符号

假设要从格式为“June 26, 1951”的生日日期中提取出月份部分，用来匹配该日期的正则表达式可以如图五所示：

图五：匹配所有 Moth DD,YYYY 格式的日期

新出现的“\s”符号是空白符号，匹配所有的空白字符，包括 Tab 字符。如果字符串正确匹配，接下来如何提取出

月份部分呢？只需在月份周围加上一个圆括号创建一个组，然后用 ORO API（本文后面详细讨论）提取出它的值。修改后的正则表达式如图六所示：

图六：匹配所有 Month DD,YYYY 格式的日期，定义月份值为第一个组

1.7 其它符号

为简便起见，你可以使用一些为常见正则表达式创建的快捷符号。如表二所示：

表二：常用符号

表二：常用符号	
符号	等价的正则表达式
\d	[0-9]
\D	[^0-9]
\w	[A-Z0-9]
\W	[^A-Z0-9]
\s	[\t\n\r\f]
\S	[^\t\n\r\f]

例如，在前面社会安全号码的例子中，所有出现“[0-9]”的地方我们都可以使用“\d”。修改后的正则表达式如下

```
\d{3}\-\d{2}\-\d{4}
```

匹配所有 123-12-1234 格式的社会安全号码

2. API 介绍

Matcher

public boolean **find**(int start)

重置此匹配器，然后尝试查找匹配该模式、从指定索引开始的输入序列的下一个子序列。

如果匹配成功，则可通过 start、end 和 group 方法获取更多信息，而 find() 方法的后续调用将从此匹配操作未匹配的第一个字符开始。

返回：

当且仅当从给定索引开始的输入序列的子序列匹配此匹配器的模式时才返回 true。

抛出：

IndexOutOfBoundsException - 如果开始点小于零或大于输入序列的长度。

Pattren

public final class **Pattern**
extends Object
implements Serializable

正则表达式的编译表示形式。

指定为字符串的正则表达式必须首先被编译为此类的实例。然后，可将得到的模式用于创建 Matcher 对象，依照正则表达式，该对象可以与任意字符序列匹配。执行匹配所涉及的所有状态都驻留在匹配器中，所以多个匹配器可以共享同一模式。

因此，典型的调用顺序是

```
Pattern p = Pattern.compile("a*b");
Matcher m = p.matcher("aaaaab");
boolean b = m.matches();
```

在仅使用一次正则表达式时，可以方便地通过此类定义 matches 方法。此方法编译表达式并在单个调用中将输入序列与其匹配。语句

```
boolean b = Pattern.matches("a*b", "aaaaab");
```

等效于上面的三个语句，尽管对于重复的匹配而言它效率不高，因为它不允许重用已编译的模式。

此类的实例是不可变的，可供多个并发线程安全使用。Matcher 类的实例用于此目的则不安全。

方法摘要

static Pattern	**compile**(String regex) 将给定的正则表达式编译到模式中。
static Pattern	**compile**(String regex, int flags) 将给定的正则表达式编译到具有给定标志的模式中。
int	**flags**() 返回此模式的匹配标志。
Matcher	**matcher**(CharSequence input) 创建匹配给定输入与此模式的匹配器。
static boolean	**matches**(String regex, CharSequence input) 编译给定正则表达式并尝试将给定输入与其匹配。

```
Pattern p = Pattern.compile("a*b"); //将给定的正则表达式编译到模式中。
Matcher m = p.matcher("aaaaab"); //创建匹配给定输入与此模式的匹配器。
boolean b = m.matches(); //尝试将整个区域与模式匹配。
m.find(); /*尝试查找与该模式匹配的输入序列的下一个子序列。此方法从匹配器区域的开头开始，如
      果该方法的前一次调用成功了并且从那时开始匹配器没有被重置，则从以前匹配操作没有
*/匹配的第一个字符开始。
m.group();// 返回由以前匹配操作所匹配的输入子序列。
```

3. 总结

```
Pattern p = Pattern.compile(正则表达式);
Matcher m = p.matcher(待匹配项);
boolean b = m.matches();    //完全匹配返回真
```

```
Pattern p = Pattern.compile(正则表达式);
Matcher m = p.matcher(待匹配项);
boolean b = m.find();      //模糊查询，出现正则表达式即返回真
```

待匹配项.matches（正则表达式）

5.数据输入异常处理

需要输入一个整型数据。

Method_1：（利用字符串正则表达式）

```
int choose = 0;
boolean charge = true;
while (charge) {
   String str = scanner.nextLine();
   if(str.matches("[0-4]{1}")){
      charge = false;
      choose = Integer.valueOf(str);
   }else{
      System.out.println("您输入的指令有误，
请重新输入： ");
   }
}
```

Method_2:（利用异常处理）

第一次尝试失败：

```
    public static Scanner getScanner() {
        return new Scanner(System.in);
    }
```

```
int choose = 0;
boolean charge = true;
while (charge) {
    try {
        choose = scanner.nextInt();
        charge = false;
    } catch (Exception e) {
        System.out.println("输入错误，请重新输入:");
    }
}
```

输入不是整型数之后，系统一直捕捉异常，不断输出“输出错误，请重新输入：”

原因分析：

（参考文档：http://www.docin.com/p-653655159.html）

Scanner 是在循环体外初始化，第一次输入的数据没有清空，scanner 类就不会等待再输入新的数据，

正确用法

```
int choose = 0;
boolean charge = true;
while (charge) {
    try {
        choose = new Scanner(System.in).nextInt();
        charge = false;
    } catch (Exception e) {
        System.out.println("输入错误，请重新输入:");
    }
}
```

6.Java 反射

在 Class 中提供了 4 个相关的方法获得类型的属性：getField(String name):Field getFields():Field[] getDeclaredField(String name):Field getDeclaredFields():Field[] 其中 getField 用于返回一个指定名称的属性，但是这个属性必须是公有的，这个属性可以在父类中定义。如果是私有属性或者是保护属性，那么都会抛出异常提示找不到这个属性。getFields 则是返回类型中的所有公有属性，所有的私有属性和保护属性都找不到。getDeclaredField 获得在这个类型的声明中定义的指定名称的属性，这个属性必须是在这个类型的声明中定义，但可以使私有和保护的。getDeclaredFields 获得在这个类型的声明中定义的所有属性，包括私有和保护的属性都会被返回，但是所有父类的属性都不会被返回。

6.1 反射机制的作用：

1，反编译：.class-->.java

2,通过反射机制访问 java 对象的属性，方法，构造方法等；

6.2 三种方法获取类

class.forName(“类名”)

对象.getClass

类.class

2.web 开发

1. 基础知识

1 套接字

源 IP 地址和目的 IP 地址以及源端口号和目的端口号的组合称为套接字。其用于标识客户端请求的服务器和服务。

2.Cookie 详解

1.四大参数

1.key 和 value:键值映射 k-v

2.age：有效时间

两种存放方式：（1）浏览器缓存（2）Cookie 文件

-1：默认值，浏览器方式保存的 Cookie 对象，关闭浏览器，对象消失

0：删除 ， key,path,domain 必须全部相同才认为是一个对象

正数：单位秒。

3.path:有效路径。

一般设置站点根目录

path="/";当前服务器下所有站点都可用

4.domain:有效域名

一般一级域名和二级域名是两个 web 工程，为使 Cookie 在这两个工程中都能运行，例如（www.baidu.com，map.baidu.com）就要将域名设置为“.baidu.com”

2.js 中创建和使用 Cookie

```
var date = new Date(2015,9,20,0,0,0);
document.cookie="addr=beijing;expires="+date.toUTCString();
document.cookie="phone=137232972;expires="+date.toUTCString();
```

```
var cookies = document.cookie.split("; ");
for(var i=0;i<cookies.length; i++){
    var cname = cookies[i].split("=")[0];
    alert(cname);
    var cvalue = cookies[i].split("=")[1];
    if(cname == "name"){
        document.cookieform.name.value=cvalue;
    }
}
```

3.Cookie 中不能直接使用中文

抛出异常

```
java.lang.IllegalArgumentException: Control character in cookie value or attribute.
        org.apache.tomcat.util.http.CookieSupport.isHttpSeparator(CookieSupport.jav
```

解决方案：编解码

```java
import java.io.UnsupportedEncodingException;
import java.net.URLDecoder;
import java.net.URLEncoder;


public class Test {

    public static void main(String[] args) throws UnsupportedEncodingException {
        // TODO Auto-generated method stub
        String name = "王武";
        String en_name = URLEncoder.encode(name, "utf-8");
        System.out.println(en_name);
        String de_name = URLDecoder.decode(en_name, "utf-8");
        System.out.println(de_name);
    }
```

3.域名详解

一级域名与二级域名

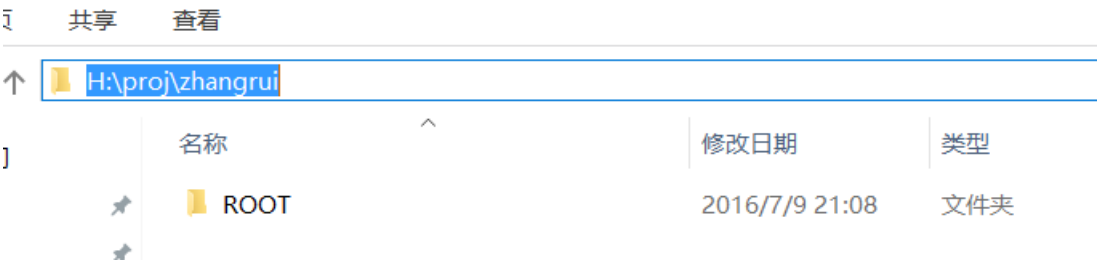
一级域名：www.baidu.com

二级域名：map.baidu.com

4.虚拟域名

步骤 1.创建目录，虚拟站点。

将部署工程移到此目录下



步骤 2：在 Tomcat 服务器上创建虚拟主机

Tomcat/conf/server.xml 文件中创建 host

<host appBase=”步骤 1 目录” autoDeploy=”true” name=”www.baidu.com” unpackWAR=”true”>
</host>

注意使用“/”

步骤 3. 更改本地域名解析文件

C:\Windows\System32\drivers\etc\下更改 hosts 文件

增加 127.0.0.1　　虚拟域名（如 127.0.0.1　www.zhangrui.com）

步骤 4：将工程文件改名为 ROOT（每个站点下默认访问 ROOT 工程）

步骤 5：更改 Tomcat sever at local host 更改为默认端口号 80

5.get 请求中的中文处理。

request.setCharacterEncoding(“utf-8”);这个方法只能针对 post 请求方式处理中文

get 方式请求通过 URLEncoder 编解码处理

http 协议规定在地址栏传递的文本都是按“iso-8859-1”处理的

```
jsp，获取get请求中的中文的方式
<%=new String( request.getParameter("name").getBytes("iso-8859-1") , "utf-8" ) %>
```

2. 工程配置

1. Tomcat 存储路径问题

路径名中不能存在 jar 文件夹，否则 eclipse 部署报错

2.Jboss 插件的配置

1. 在 Eclipse 目录下新建文件夹取名：links

2. 在 links 目录下新建文本文件，后缀名取为**.link
3. 把下载好的插件解压出来，新建文件夹取名 eclipse，把解压出来的 features 和 plugins 的文件夹复制到 eclipse 中
4. 用文本编辑器打开 link 文件，里面写上：path=步骤 3 中的文件所在目录

3.strusts2 框架

1. Struts 中 Action 中存 Cookie

```
CookieUser Name=
newCookie("username",user.getUserName());
            UserName.setMaxAge(0);
            UserName.setPath("/");
   ServletActionContext.getResponse().addCookie(UserName);
UserName.setPath("/");    //服务器下所有应用共享
UserName.setPath("/WebProjName/");    //当前应用可用
```

正常的cookie只能在一个应用中共享，即一个cookie只能由创建它的应用获得。
1.可在同一应用服务器内共享方法：设置cookie.setPath("/");
本机tomcat/webapp下面有两个应用：cas和webapp_b，
1）原来在cas下面设置的cookie，在webapp_b下面获取不到，path默认是产生cookie的应用的路径。
2）若在cas下面设置cookie的时候，增加一条cookie.setPath("/");或者cookie.setPath("/webapp_b/");就可以在webapp_b下面获取到cas设置的cookie了。
3）此处的参数，是相对于应用服务器存放应用的文件夹的根目录而言的(比如tomcat下面的webapp)，因此cookie.setPath("/");之后，可以在webapp文件夹下的所有应用共享cookie，而cookie.setPath("/webapp_b/");是指cas应用设置的cookie只能在webapp_b应用下的获得，即便是产生这个cookie的cas应用也不可以。
4）设置cookie.setPath("/webapp_b/jsp")或者cookie.setPath("/webapp_b/jsp/")的时候，只有在webapp_b/jsp下可以获得cookie，在webapp_b下面但是在jsp文件夹外的都不能获得cookie。
5）设置cookie.setPath("/webapp_b");，是指在webapp_b下面才可以使用cookie，这样就不可以在产生cookie的应用cas下面获取cookie了
6）有多条cookie.setPath("XXX");语句的时候，起作用的以最后一条为准。
6）设置多个path的方法？？？
2.跨域共享cookie的方法：设置cookie.setDomain(".jszx.com");
A机所在的域：home.langchao.com,A有应用cas
B机所在的域：jszx.com，B有应用webapp_b
1）在cas下面设置cookie的时候，增加cookie.setDomain(".jszx.com");，这样在webapp_b下面就可以取到cookie。
2）这个参数必须以“.”开始。
3）输入url访问webapp_b的时候，必须输入域名才能解析。比如说在A机器输入： http://lc-bsp.jszx.com:8080/webapp_b,可以获取cas在客户端设置的cookie，而B机器访问本机的应用，输入： http://localhost:8080/webapp_b则不可以获得cookie。
4）设置了cookie.setDomain(".jszx.com");，还可以在默认的home.langchao.com下面共享。
5）设置多个域的方法？？？

2.耦合与解耦合获得内置对象

1.耦合

ServletActionContext.getRequest()

2.解耦合

使用 Map 集合，类实现接口

4.html 页面设计

1 表单中确定重置按钮居中解决办法

//用 P 标签包含，设置 P 标签中文本格式居中

```
<p style=" margin:0 auto; text-align:center">
    <input type="submit"    value="确定"> <input type="reset" value="重
    置">
    </p>
```

2.常用特殊字符

HTML 原代码	显示结果	描述
&lt;	<	小于号或显示标记
&gt;	>	大于号或显示标记
&amp;	&	可用于显示其它特殊字符
&quot;	"	引号
&reg;	®	已注册
&copy;	©	版权
&trade;	™	商标
		半个空白位
		一个空白位
		不断行的空白

5.Hibernate 框架

1.获取 session 对象

```
Configuration configuration = new Configuration().configure();
```

```
ServiceRegistry serviceRegistry = new ServiceRegistryBuilder()
        .applySettings(configuration.getProperties())
        .buildServiceRegistry();
SessionFactory sessionFactory = configuration
        .buildSessionFactory(serviceRegistry);
Session session = sessionFactory.getCurrentSession();
```

2.<property name="hbm2ddl.auto"> </property>标签

(1)<property name="hibernate.hbm2ddl.auto"> create-drop </property>

create-drop:表示在 hebarinate 初始化时创建表格,程序运行结束的时候会删除相应的表格，在实际项目中不用

(2)<property name="hibernate.hbm2ddl.auto">create</property>

在 hibernate 初始化时会创建表格，在运行结束之后不删除表格，而是在下一次运行的时候如果有旧的删掉，没有旧的，重新建表格

(3)<property name="hibernate.hbm2ddl.auto">update</property>

只是根据映射文件去和数据库中的表对应起来，如果不一致，就更新表的结构

(4)<property name="hibernate.hbm2ddl.auto">validate</property>

校验映射文件和数据库中的表是不是能对应起来，不能对应报错，实际中常用

3.主键序列化

1.rom 映射文件配置

```
<id name="id" type="java.lang.Integer" column="ID">
        <generator class="sequence">
            <param name="sequnece">SEQ_Table</param>
        </generator>
</id>
```

2.数据库添加序列

```
create sequence seq_table
    minvalue 1
    maxvalue 99999
    start with 2
increment by 1
```

4.Hibernate 和 Struts 存在 jar 包冲突

apache-tomcat-8.0.36\webapps\myHibernate\WEB-INF\lib 路径下删除相同 jar 包，只留一个

5. hbm 映射实体类

主键设置

<generator class="？ " />

6.使用 c3p0

```
<property name="dialect">org.hibernate.dialect.MySQLDialect</property>
<property name="connection.url">jdbc:mysql://localhost:3306/hibernatedb</property>
<property name="connection.username">root</property>
<property name="connection.password">root</property>
<property name="connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="myeclipse.connection.profile">hibernatedb</property>
<property name="connection.useUnicode">true</property>
<property name="connection.characterEncoding">UTF-8</property>
<!-- 最大连接数 -->
<property name="hibernate.c3p0.max_size">20</property>
<!-- 最小连接数 -->
<property name="hibernate.c3p0.min_size">5</property>
<!-- 获得连接的超时时间,如果超过这个时间,会抛出异常，单位毫秒 -->
<property name="hibernate.c3p0.timeout">120</property>
<!-- 最大的PreparedStatement的数量 -->
<property name="hibernate.c3p0.max_statements">100</property>
<!-- 每隔120秒检查连接池里的空闲连接 ，单位是秒 -->
<property name="hibernate.c3p0.idle_test_period">120</property>
<!-- 当连接池里面的连接用完的时候，C3P0一下获取的新的连接数 -->
<property name="hibernate.c3p0.acquire_increment">2</property>
<!-- 每次都验证连接是否可用 -->
<property name="hibernate.c3p0.validate">true</property>
```

7.Hibernate 注解的方式

Hibernate.cfg.xml 中使用<mapping class=””/>

@Entity 当前类是一个持久化的类，与数据库中的一张表对应

@Table(name=”数据库表名”)

@ID 主键

@GeneratedValue(strategy = “主键生成策略”)

@column 映射到数据库中的列

@ManyToOne(fetch=FetchType.LAZY)多对一映射，延迟加载

@ManyToOne(fetch=FetchType.EAGER)非延迟加载

@JoinColumn(name=””)引用列

@OneToMany(cascade = CascadeType.ALL,fetch=FetchType.LAZY,mappedBy=”字段”)

级联策略，延迟加载，放弃控制权，参考字段描述

6. Spring 框架

1. Spring 体系结构

（1）Spring core:工厂，创建对象。维护对象之间的依赖关系。（IoC）

（2）Spring AOP:面向切面编程。代理

（3）Spring DAO:实现数据库支持。可以支持 JDBC

（4）Spring ORM:对 ORM 类的框架进行支持。整合 Hibernate

（5）Spring Context:Spring 上下文模块。主要针对 Spring 配置文件，ApplicationContext

（6）Spring WEB:整合针对 WEB 应用程序，整合 Struts2

（7）Spring MVC:针对 MVC 模式的实现

2.利用 Spring 获取对象

```
ApplicationContext applicationContext = new ClassPathXmlApplicationContext(
                "/applicationContext.xml");
        Object object  = (Object) applicationContext.getBean("object");
```

7. jsp

1. Forward 与 include 的区别

<jsp:include>标签用于把另外一个资源的输出内容插入进当前 JSP 页面的输出内容之中，这种在 JSP 页面执行时的引入方式称之为动

态引入。

<jsp:forward>从一个 JSP 文件传递 request 信息到另外一个 JSP 文件，<jsp:forward>后面的部分将不会被执行。

可以使用 <jsp:param> 传递参数。

<jsp:include>将包含的文件放在 JSP 中和其他一起执行。

<jsp:forward>标签用于把请求转发给另外一个资源。

3.数据库

1.数据库分页查询

```
select * from
  select A.*, rownum rn from(
(select * from Table_name) A
WHERE ROWNUM<max
)
WHERE RN >=min
```

必须将小于最大值条件写在内层作为外层查询表，否则得不到数据