在web开发中，服务器可以为每一个用户浏览器创建一个会话对象（session对象）
注意：一个浏览器独占一个session对象（默认情况下）
因此，在需要保存用户数据时，服务器程序可以把用户数据写到浏览器独占的session对象中，当用户使用浏览器访问其他程序的时候，其他程序可以从用户的session中取得该用户的数据，为用户服务

ipconfig查看自己的IP地址

会话需要在同一个会话，session才管用

session的作用域比request作用域大，但是比servletcontext作用域小

Ctrl+t表示自动创建返回值类型

多个用户多个会话可以访问servletcontext中的共享数据
当前用户的多个servlet应用可以访问session中的共享数据（以用户为单位）
一次请求之内的servlet可以共享request中的共享数据

session也是一个域对象，session　servletcontext request
同一个会话下，可以使一个应用的多个资源共享数据
cookie客户端技术，只能存储字符串，httpsession服务端技术，可以存储对象
cookie只能存储少量信息，而且不安全，敏感信息不适合存储在cookie中

重点：

session依赖于cookie；session默认三十分钟有效

session常用对象
1，把数据存储在httpsession对象中，该对象也是一个域对象
void setAttribute（string name，object value）在session中存储对象
object getAttribute（string name）通过name从session中取得session对象
void　removeAttribute（string name）通过name移除session中的对象，session还在，但是其中的内

容被删除

httpsession getid（）获得
setMaxInactiveInterval（int interval）设置session的存活时间，单位是秒

invalidate（）使得此会话无效，销毁session

2，httpsession request.getsession（）原理

①获取名称为JSESSIONID的cookie值

②没有这样的cookie，创建一个新的httpsession对象，分配一个唯一的sessionid，并且向客户端写入

一个名为JSESSIONID=sessionid的cookie

③有这样的cookie，获取cookie的值，（即httpsession对象的值），从服务器的内存中根据id找到那

个httpsession对象，找到了，取出继续服务，没有找到返回第二步

httpsession request.getsession（Boolean create）

参数：true和getsession（）功能一样，

false根据客户端JSESSION的cookie的值，找对应的httpsession对象，找不到返回null（不会创建新的，只是查询）

session的状态

创建：当浏览器第一次访问服务器状态资源就创建request.getsession（）

服务器：应用运行时，活着

死了：　session invalidate（）强制销毁

超时，默认三十分钟

setmaxinactiveinterval（int）单位秒

在xml中session-config配置<session-timeout>1</session-timeout>参数1表示一分钟

当内存溢出或者服务器重启的时候，会执行以下的操作（实现序列化和反序列化接口）：

持久化（钝化或搁置）：存盘：内存中的一个全景图，io实现序列化，实现可序列化接口

反序列化（活化或激活）：从内存中将序列化的数据反序列化出来，成为可视的数据

序列化：把内存中的数据写入磁盘，以二进制方式存储的二进制用户名和密码

session完成用户登录，实现验证码验证

JSP代码：

```
<script type="text/javascript">
```

```
        function changecode(){
                var img = document.getElementsByTagName("img")[0];
                img.src = "/newS/servlet/testyanzhengma?time="+new
Date().getTime();
        }
</script>
  </head>

  <body>
  <%
  String  msg = (String)request.getAttribute("msg");
  if(msg!=null)
  {
  out.print(msg);
  }
   %>
   <form action="/newS/servlet/doyanzheng" method = "GET">
用户名<input type = "text"name = "use"/><br/>
密  码<input type = "password"name = "pass"/><br/>
验证码<input type = "text"name = "yzm"/>
<img src="/newS/servlet/testyanzhengma" onclick="changecode()"/>
<a href = "javascript:changecode()">看不清，换一张</a>
<br/>
<input type = "submit" value = "提交"/><br/>
</form>
  </body>
</html>
```

验证码实现代码servlet：

```
        ValidateCode vc = new ValidateCode(110,25,4,9);
                //向session中保存验证码
                request.getSession().setAttribute("scode", vc.getCode());
                vc.write(response.getOutputStream());
```

检验验证码的错误与否servlet代码：

```
                response.setContentType("text/html;charset=utf-8");
                request.setCharacterEncoding("utf-8");
                PrintWriter out = response.getWriter();
                //获取表单数据
                String  name = request.getParameter("use");
                String  pwd = request.getParameter("pass");
                String  code = request.getParameter("yzm");
                String scode =
(String)request.getSession().getAttribute("scode");
                //处理业务逻辑
                if ("tom".equals(name)&&"123".equals(pwd)) {
                        if (!code.equals(scode)) {
                                String msg = "验证码错误！";
                                request.setAttribute("msg", msg);

request.getRequestDispatcher("/index.jsp").forward(request, response);
                        }
                        out.print("登录成功！");
                }
```