1. 无状态的HTTP 协议

*无状态服务器是指一种把每个请求作为**与之前任何请求都无关的独立的事务的服务器***

HTTP 协议采用 `连接 - 请求 - 应答 - 关闭连接` 模式

对于交互式的 Web 应用，保持状态很重要，有状态的协议可帮助多个请求和响应之间实现复杂的业务逻辑

2. 会话跟踪技术

*一种在**客户端与服务器之间保持 HTTP 状态的解决方案**，从开发角度考虑，是使上一次请求所传递的数据能够维持状态到下一次请求，并且辨认出是否相同的客户端发送出来的*

2.1. Cookie 技术

*Cookie 技术是一种**在客户端保持会话跟踪的解决方案***

- Cookie 是**指某些网站为了辨别用户身份而存储在用户终端上的文本信息**
- 在用户第一次访问服务器时，由服务器通过**响应头的方式发送给客户端浏览器**；当用户再次向服务器发送请求时会附带上这些信息

首次和非首次请求的区别

- 服务器对于第一次客户端请求的响应含有 `set - Cookie`
- 非第一次请求时，请求头直接带有 `Cookie`

作用

- 通过 `Cookie` ，服务器接收到客户端的浏览器请求时，通过请求头获取客户端特有信息，生成对应的内容

Cookie 的使用

- 创建： `Cookie ck = new Cookie('属性名' , '属性值')`
- 添加： `response.addCookie(ck);`
- 获取Cookie 集合： `Cookie [] cks = request.getCookies();`
- 设定访问路径： `ck.setPath("path");`
    - `"/"` 时，应用下的所有都能访问
    - 默认情况下只能被创建它的应用获取，通过setPath方法重新指定访问路径
- 设置存活时间： `ck.setMaxAge(second)`

Cookie 的响应

- `HttpServletResponse` 对象通过 `addCookie()` 方法，以增加 `set-Cookie` 响应头的方式将其响应给客户端浏览器，存储在客户端机器上
- `response.addCookie(Cookie ck)`

获取遍历客户端 Cookie

- `request.getCookies()`

Cookie 的访问路径

- 默认只能呗创建它的应用获取，`setPath()` 方法可以重新指定
- `ck.setPath("/")`：在服务器下所有应用的访问路径

Cookie 的存活时间

- 默认情况下保存在浏览器内存中，在浏览器关闭时失效，临时 Cookie（会话Cookie）
- 通过 `setMaxAge()` 方法设置 Cookie 的存活时间

缺点

- 1）`Cookie` 可能被浏览器禁用
- 2）`Cookie` 是与浏览器相关的，这意味着即使访问同一个页面，不同浏览器保存的Cookie 也是不能互相访问的
- 3）`Cookie` 可能被删除，因为每个Cookie 都是硬盘上的文件
- 4）`Cookie` 的大小和个数有限
- 5）`Cookie` 的安全性不够高

2.2. Session 技术

实现原理

- 使用 `HttpSession` 对象实现会话跟踪的技术，在服务器端保持会话跟踪的解决方案
- 是服务器在无状态 `HTTP` 协议下的用来**识别和维护具体某个用户**的主要方式
- `HttpSession` 对象会在用户第一次访问服务器时由容器创建（只有 `JSP`、`Servlet` 等程序，只访问 `HTML`、`IMAGE` 等静态资源不会创建）
- 当用户调用失效方法 `invalidate()` 或超过最大存活时间会失效，**此期间用户与服务器之间的多次请求都属于同一个会话**
- 创建时候服务器分配唯一的会话标识 `SessionId`，以 `JSESSIONID` 的属性名保存在客户端 `Cookie`，服务器通过读取 `JSESSIONID` 来识别不同的用户

创建获取

- `request.getSession()`：获取 `Session` 对象
- `request.getSession(boolean create)`：获取若没有 `Session` 关联，当参数为真时，被创建，为假返回空值

存取会话域属性的方法

- `void setAttribute(String key, Object value)`
- `Object getAttribute(String key)`
- `void removeAttribute(String key)`

管理生命周期的方法

- `session.getMaxInactiveInterval()`：获取最大存活时间，单位为秒
- `session.setMaxInactiveInterval(600)`：设置最大存活时间

- `void invalidate()`：会话立即失效
    - 清除会话对象及其所有会话属性，同时响应客户端浏览器清除 `Cookie` 中的 `JSESSIONID`

2.3. URL 重写技术

原理

- 服务器程序对接受的 URL 请求进行截取并重新写成网站可以处理的另一个 `URL` 的过程。
- 在不知道客户端浏览器是否支持 Cookie 的情况下，使用 URL 重写技术可以对请求的 URL 地址追加会话标识来实现会话跟踪
- 例如对于

```
http://localhost:8080/charpter04/EncodeURLServlet
  重写为：
http://localhost:8080/charpter04/EncodeURLServlet;jsessionid=xxxxxxxxxx
```

- `jsessiionid`：即为追加的会话标识，服务器即通过它来识别跟踪某个用户的访问

方法

- `encodeURL()`：对任意请求的 URL 进行重写
- `encodeRedirectURL()`：对 `sendRedirect()` 方法的 URL 进行重写
    - 根据请求信息中是否包含 `Set-Cookie` 请求头决定是否进行 URL 重写，若包含，将其原样输出。不包含将标识重写到 URL 中

注意事项

- 如果需要重写，那么必须对应用的所有请求都重写，将 `jsessionid` 维持下来
- 由于浏览器对 URL 地址长度的限制，要注意总长度
- 静态页面不能进行会话标识的传递

2.4. 隐藏表单域

实现原理

- 利用 Form 表单的隐藏表单域，可以在完全脱离浏览器对 Cookie 的使用限制，并且在用户无法从页面显示看到隐藏标识的情况下，将标识随请求一起传送给服务器处理，来实现会话跟踪

在 Form 表单中定义隐藏域

```
<form action = "xx" method = "post">
  <input type = "hidden" name = "userid"> <!-- 一个隐藏的表单域，在前台不显示-->
  <input type = "submit" value = "提交">
</form>
```

在服务器通过 `request` 对象获取隐藏域的值

- `String f = request.getParamter("userID")`