会话技术

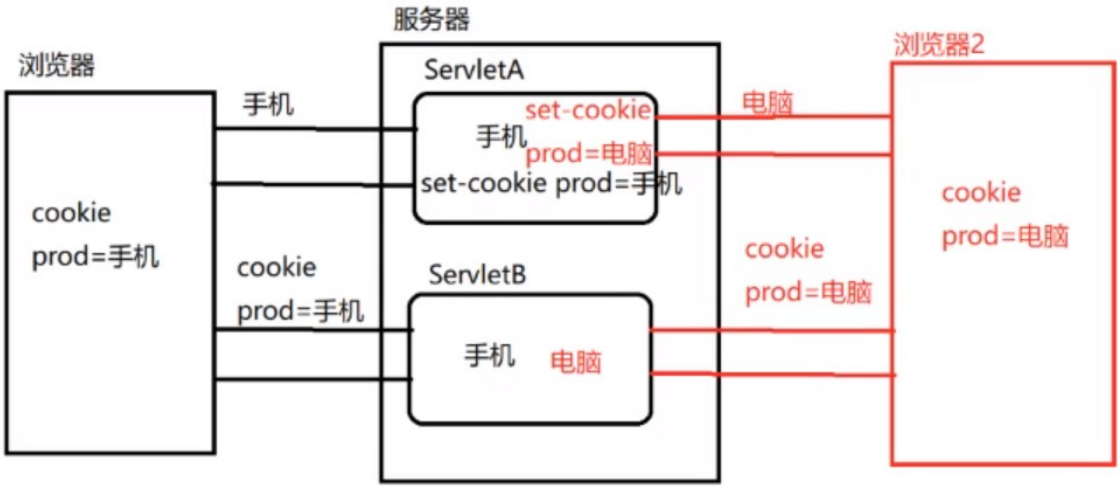
一　什么是会话

为了实现某一功能，浏览器和服务器之间可能会产生多次交互(多次请求和响应)，从浏览器访问服务器开始，到最后访问结束为止，期间产生的多次请求和响应加在一起，就可以称之为一次会话。

二　Cookie

1　原理

将会话中产生的数据保存在客户端



浏览器向服务器发送请求，请求中包含需要保存的数据，服务器获取数据，并通过 set-cookie 响应头(服务端设置请求头)将数据响应给浏览器，让浏览器保存

浏览器再次访问服务器时，会在请求中通过 cookie 请求头携带上次保存的数据(由浏览器自动携带)，服务器可以根据 cookie 请求头获取数据，通过这种形式来保存会话中产生的数据

2 实现 cookie

Cookie 是基于 set-cookie 响应头和 cookie 请求头实现
Sun 公司为了简化 cookie 操作，提供一套 cookie 的 API
Cookie cookie=new Cookie(String name,String value)
Cookie.getName
Cookie.geValue

2.A 将 cookie 添加到响应中

```
Cookie cookie=new Cookie("name","舒新胜");
response.addCookie(cookie);
```

2.B 从请求中获取

```
//获取，如果请求中没有 cookie 返回 null
Cookie[] cookies=request.getCookies();
cookies[0].getName();
cookies[0].getValue();
```

3 设置 cookie 的存活时间

```
//设置 cookie 存活时间，cookie 默认是会话级别，是保存在浏览器的内存中，浏览器一旦关闭，内存随之释放，cookie 也随之消失
//如果设置了 setMaxAge，则 cookie 保存在浏览器对应的磁盘目录中
cookie.setMaxAge(1000);
```

4 设置 cookie 路径

setPath(String path)
设置路径后，浏览器会在访问当前设置的路径是才会携带 cookie
通常设置为当前 web 应用的根路径，否则默认为当前响应的 Servlet 的路径

5 删除 cookie

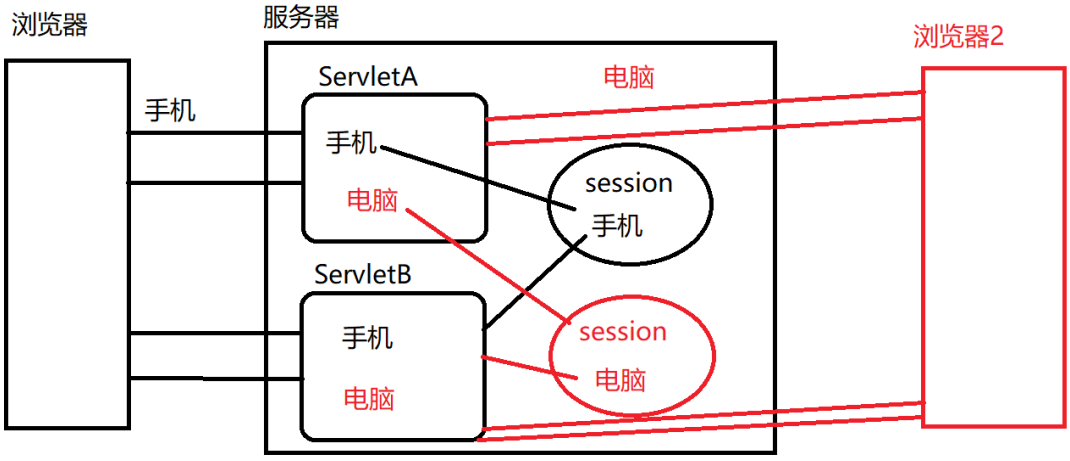
没有删除 cookie 的 API
如果想删除，则可以向浏览器发送一个同名，同 path 的 cookie，并设置最大生存时间为 0
浏览器通过 cookie 的名字和 path 判断是否为同一个 cookie

三 SESSION

1 SESSION 原理

1.A 4.1.session 原理

将会话中产生的数据保存在服务器端



浏览器向服务器发送请求，服务器接收请求参数，然后在服务器中检查是否有为当前浏览器服务的 session，如果有则直接拿来使用，如果没有则创建一个 session，并将数据保存到 session 中
当浏览器再次访问服务器时，服务器可以找到为当前浏览器服务的 session，从中取出数据
通过这种方式也可以保存会话中产生的数据

每个 session 对应一个 id，此 id 通过 cookie（临时）保存在浏览器中，访问服务器时携带 cookie，服务器根据这个 id 判断是哪个 session

2 Ssession 域

2.A 生命周期

2.A.1 创建

第一次调用 request.getSession()方法时创建 session 对象

```
HttpSession session = request.getSession();
```

如果服务器中已经有 session，会直接拿来使用，如果没有才创建

2.A.2 销毁

2.A.2.1 超时死亡：

默认 30 分钟不使用会超时销毁，及时关闭浏览器未超时也不会销毁，Cookie 关闭浏览器时会销毁(如果没有设置超时时间)

2.A.2.2 主动杀死：

可以调用 session.invalidate() 方法，立刻杀死 session

```
session.invalidate();
```

2.A.2.3 意外身亡：

当服务器意外关闭，session 也会销毁

如果服务器正常关闭，session 会被钝化，当服务器启动时再活化

2.B Session 数据的钝化与活化：

由于 session 中保存大量访问网站相关的重要信息，因此过多的 session 数据就会服务器性能的下降，占用过多的内存。因此类似数据库对象的持久化，web 容器也会把不常使用的 session 数据持久化到本地文件或者数据中。这些都是有 web 容器自己完成，不需要用户设定。

不用的 session 数据序列化到本地文件中的过程，就是钝化；

当再次访问需要到该 session 的内容时，就会读取本地文件，再次放入内存中，这个过程就是活化。

2.C 作用范围

整个会话

2.D 作用

整个会话范围内共享数据

2.E 使用

request.getSession(true)：若存在会话则返回该会话，否则新建一个会话。

request.getSession(false)：若存在会话则返回该会话，否则返回 NULL

request.getSession()：默认为 true