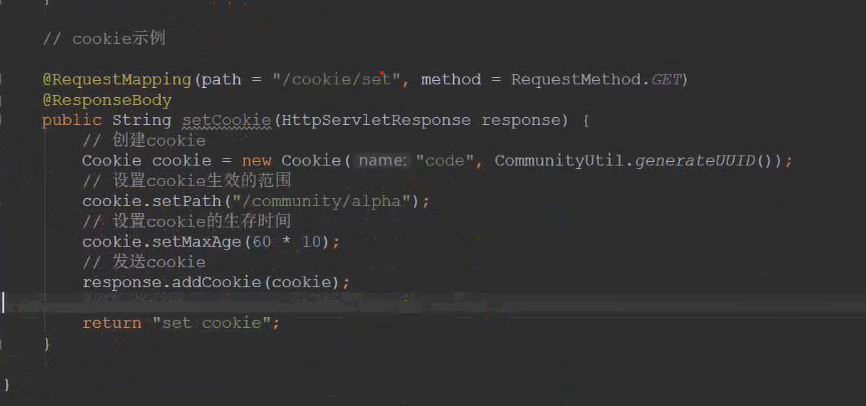
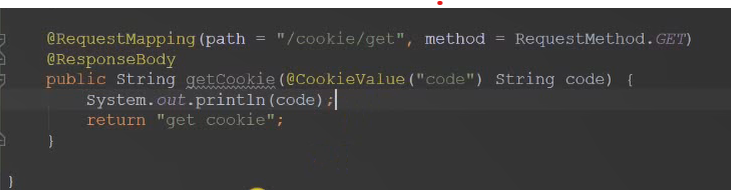
为什么要用cookie：Http的两个成功执行的请求之间是没有关系的(HTTP是**无状态的**)，需要通过cookie来解决问题，**使得请求能共享相同的上下文信息**，达成相同的状态

cookie是服务器发送到浏览器，并保存在浏览器端的一小块数据，浏览器下次访问该服务器时，会自动携带块该数据，将其发送给服务器，一个cookie只能保存一组key，value，上图的四步为通常的cookie设置，最后的返回值是返回给游览器的字符串，非网址，是乱写的

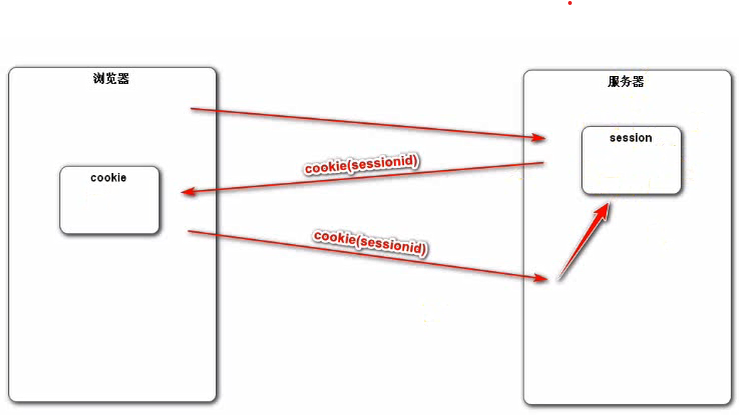
cookie在**浏览器关掉就消失**，但可以**设置生存时间**来防止



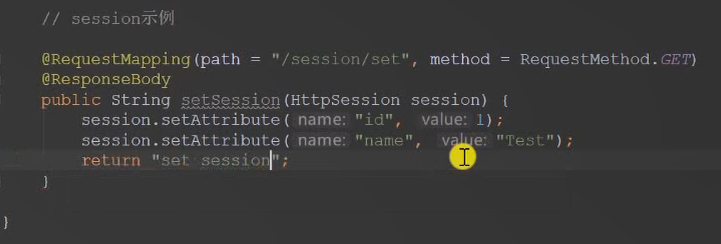


获得cookie，注意cookie在@RequestMapping中传入给了服务器，可以在响应体中通过@CookieValue（key值） String code来获得cookie中的某一值，全部获得cookie的话是个数组

Session - 是JavaEE的标准，用于**在服务端记录客户端信息**。 - 数据存放在服务端更加安全，但是也会增加服务端的**内存压力**。



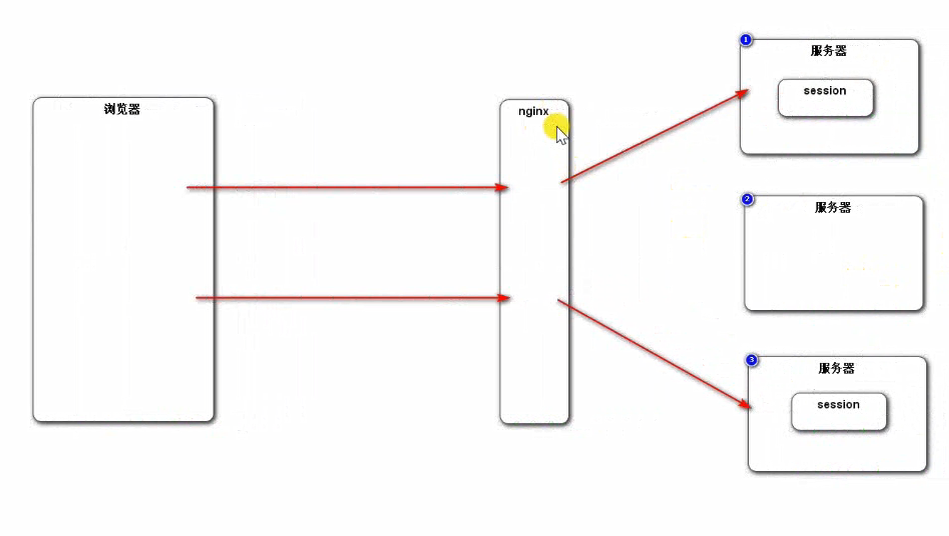
传cookie是为了能够查找到对应的session



Session是和model等类型一样，一经声明spring mvc自动注入

**分布式session问题**

**为何用cookie不用session**：



当浏览器发出请求，传给nginx代理，经过负载均衡，可能第一次的请求发给了服务器1，服务器1存储了一份session返给浏览器一个cookie（session id），之后第二次浏览器发出请求，经过反向代理可能给服务器3发送了这个cookie，但服务器3没有对应的session，会新建一个session，但这个session和第一次的请求数据创建的session不一样，会造成数据错误

**可以通过设置负载均衡的分配策略**：

1.用粘性session，只要是同一个浏览器ip就固定的分配给同一个服务器处理，但这样就会负载不均衡，性能不好；

2.第二种策略是同步session，当一个服务器创建了session后，会同步给每个服务器，首先一台服务器同步给其他很多服务器，会影响性能，而且服务器和服务器之间会产生耦合，产生关联，会影响部署；

3.第三种方案是共享session，单独有一个服务器，专门存放session，当需要获得session时，都去找此服务器，但这样如果这个服务器挂掉了，其他服务器没法工作，违背了分布式的初衷：解决性能瓶颈，如果将此服务器搞一个集群，那和方案二没有区别，仍然需要共享；

目前主流方案时能将数据存放到**cookie**里就放到cookie里，敏感数据比如密码之类的，就放到**nosql数据库**里（关系型数据库会把数据存放到硬盘里，速度慢）

**用Redis解决分布式session问题**

**也就是用redis来放登录凭证，因为不能放服务器**

