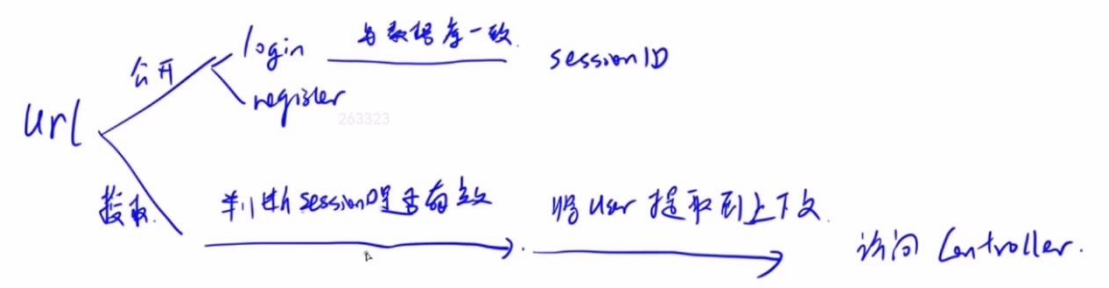
SessionID 模式

首先公开页面是只要 url 一来我们就直接可以访问这个页面的逻辑，也就是访问这个页面的 controller。我们在登录页面是如何去授权的呢?
在后端 service 里面有一个 UserDetailsServiceImpl 类，这个类会从数据库里读取用户信息，然后将用户传过来的用户名密码和数据库里存的用户名密码作一个比对。
如果跟数据库中的信息是一致的话，就会给用户发一个 SessionID，用户收到这个 SessionID 会将它存到本地，也就是存到 cookie 里面。
未来每次访问这个网站的时候，它都会自动将浏览器里存储的 SessionID 发送给服务器。
服务器在发给有用户 SessionID 后同时会自己存一份，可能会存到内存里也可能存到数据库里。
用户一旦登录成功就会得到一个 SessionID，相当于一个令牌。未来用户再去访问一些授权页面的时候，用户就会将这个 SessionID 带上。
服务器在接收到用户访问某个授权页面的请求时，它会从用户访问的请求里面将 SessionID 提取出来，判断 SessionID 是否有效。
这个很容易判断，因为服务器是把 SessionID 存下来的。所以用户请求来了之后服务器可以从数据库或内存里查一下它存不存在。
如果存在的话，不仅会存 SessionID，还会存它对应的用户是哪个用户，因为服务器会存一个映射。
如果服务器判断用户的 SessionID 有效，服务器端就会将 SessionID 对应的用户信息提取到上下文中。
每个 URL 都会对应一个 controller，所谓的“提取到上下文”，指的是在每个 controller 中都可以通过某种接口将用户信息（用户名、密码、头像等）提取出来了。
下一步就可以执行正常的访问逻辑（访问 controller）了。



Jwt-token 模式

上面是传统模式，这节课的模式不用 SessionID 来判断。
因为现在很多应用都是跨域的，而且很多应用不止一个端，既有 web 端也会有 app 端，所以这种情况下 SessionID 的模式就不是很方便。
一旦跨域，SessionID 就比较难用了。而且如果有多个服务器端，如果用户要访问所有页面就需要服务器将 SessionID 复制很多份，放到多台服务器上。
要实现跨域，用 Jwt 验证方式比较方便。而且 Jwt 验证无需在服务器端存储，这样用户只要获取一个令牌就可以登录多个服务器。
Jwt 验证的原理是用户登录成功后给用户一个 jwt-token，这个 token 会存到用户浏览器（本地）里。这个 jwt-token 里面存有用户信息。

具体来说，jwt-token 是这么生成的：一个字符串第一段存储用户信息，第二段存储密钥，这个密钥只有服务器端有，其他地方看不到。用不可逆的加密函数，将这个字符串映射成另一个字符串。这个加密函数能够实现一一映射。
如果说用户的登录信息是正确的，可以将用户信息配上密钥根据某种加密算法得到加密之后的字符串。
接下来将用户信息加上加密后的信息拼接到一块返回给用户。返回给用户的这个东西叫做 jwt-token。

服务器根据 jwt-token 中的第一段，配上自己的密钥，根据固定的加密方法生成加密后的字符串。判断这个加密后的字符串是否与 jwt-token 第二段一样即可。
为了防止其他人窃取用户的 jwt-token，可以这么来优化：
设置一个 access-token 和 refresh-token，前者有效时间比较短，比如 5min，后者有效时间比较长，比如 14 天。用户向服务器端发送请求的时候不要带 refresh-token，而要带 access-token。
每次 access-token 过期的时候，用户向服务器端发送一个 post 类型的请求，这个请求不是明文的（不像 get 请求），比较安全。这个 post 请求会根据 refresh-token 得到一个新的 access-token。
本项目比较简单，只做一个 token，有效期 14 天。