第一次作业

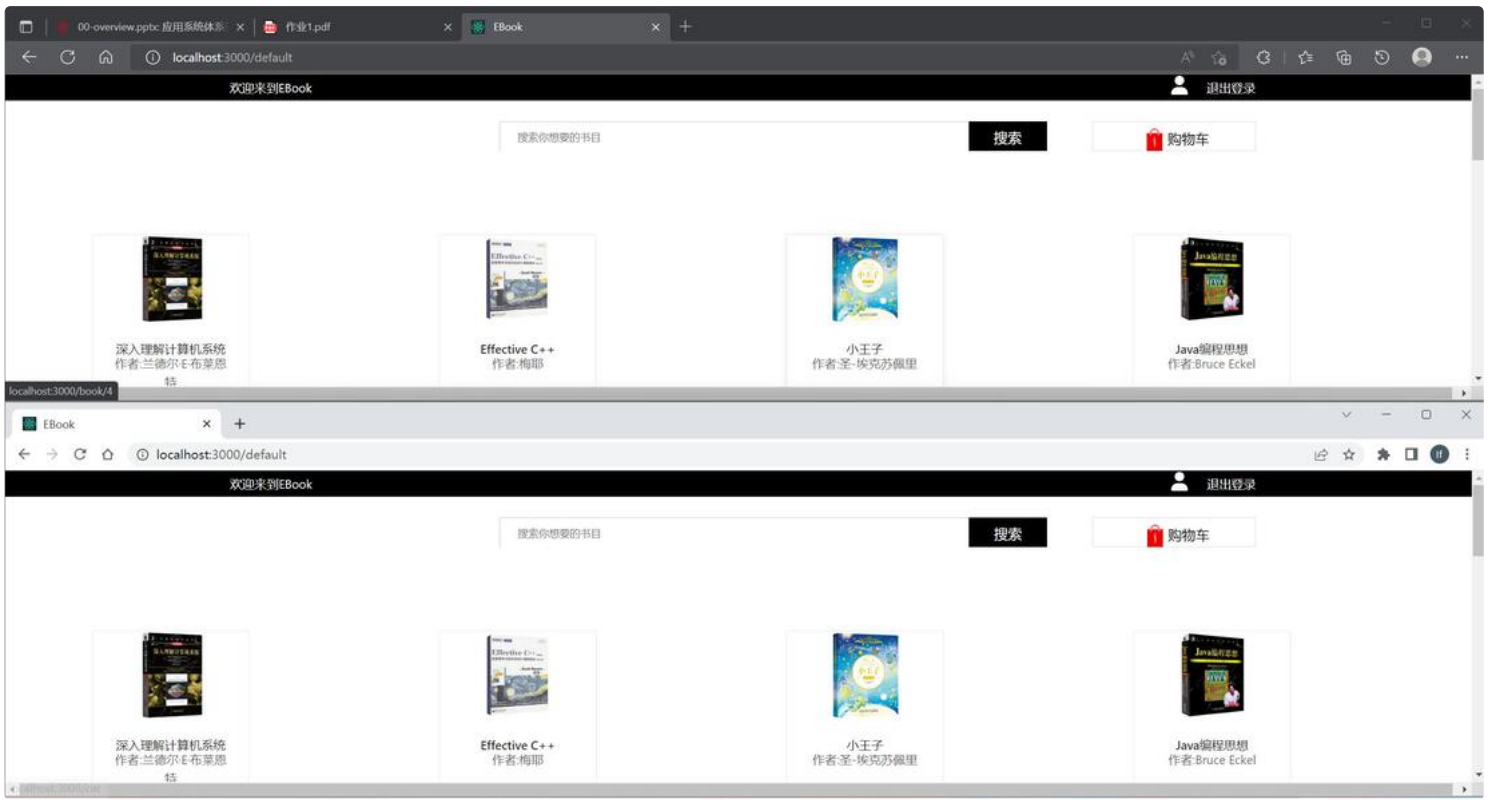
作业 A

Scope 设计：

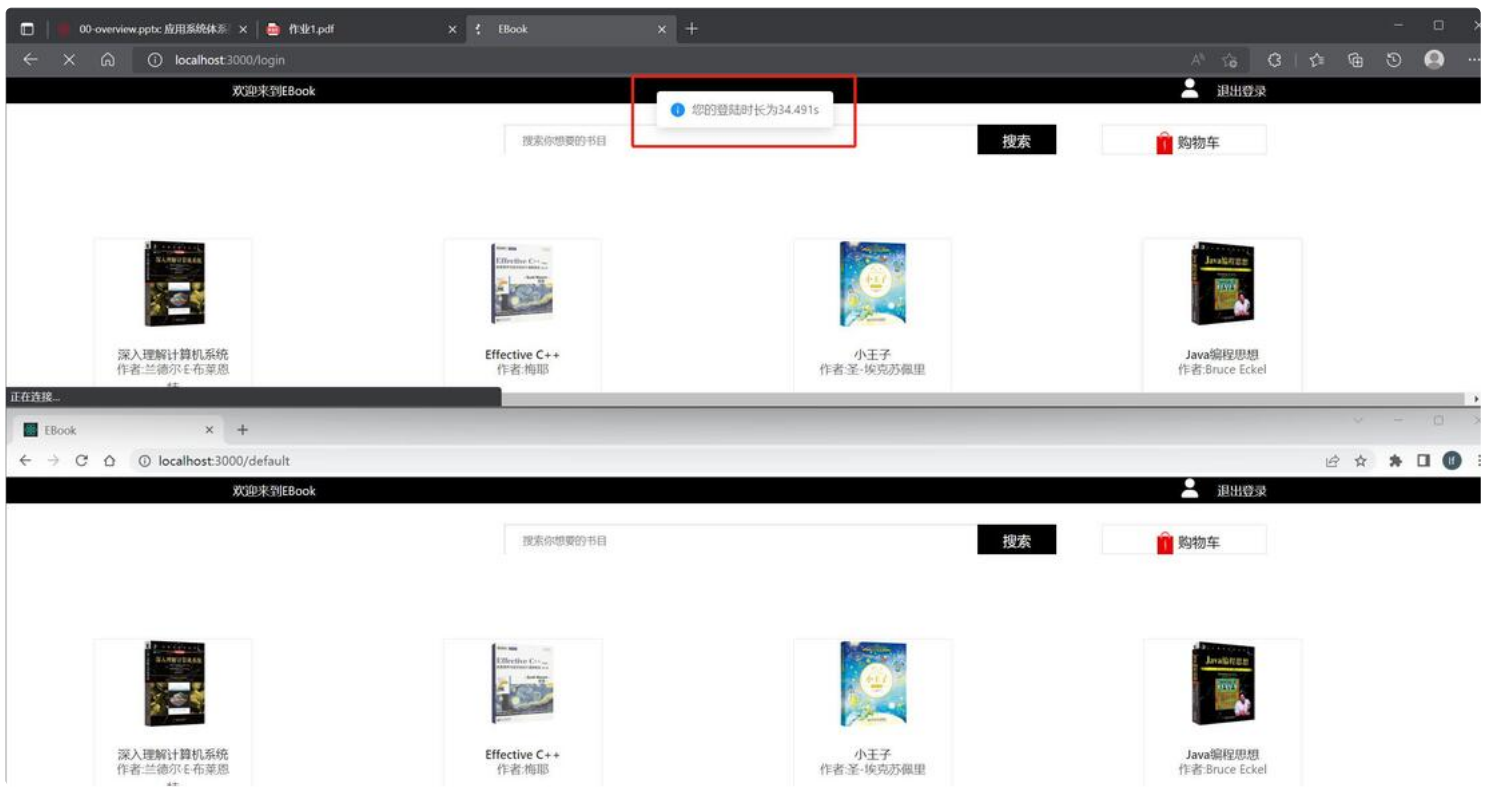
- 在 UserController 处使用 WebApplicationContext 注入 userService
- 对 UserService 使用 @SessionScope 声明

依据：

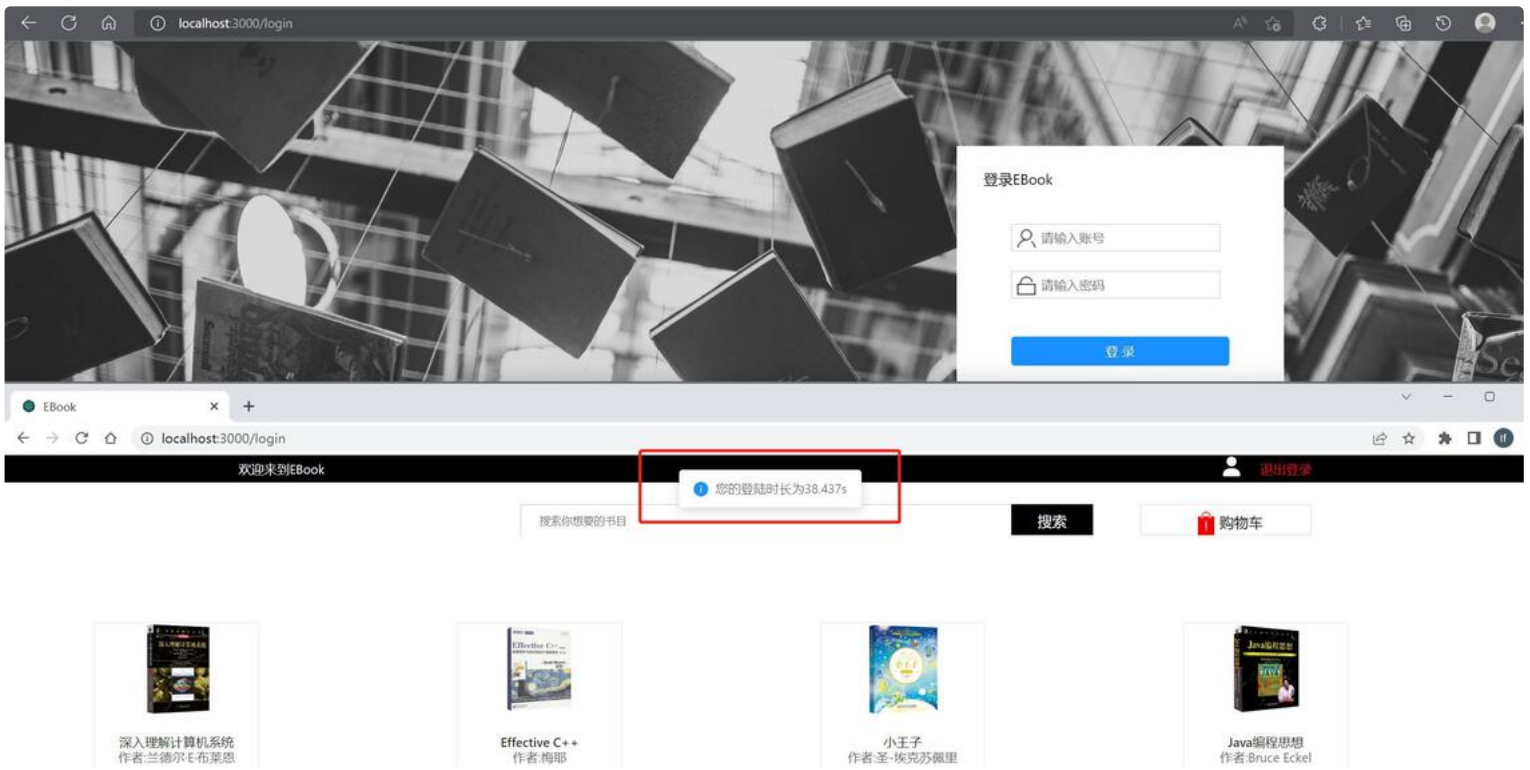
要实现为不同浏览器使用的用户记录时间，即需要为每个 Session 计时，因此需要为每个 session 维护一个 service 实例，因此采用 SessionScope，controller 示例数无影响。



两个浏览器登陆后（上下分屏）



浏览器 1 用户点击退出登录后



浏览器 2 用户点击退出登录后

作业 B

	(1) createOrder	(2) saveOrder (In OrderDao)	(3) saveOrderItems (In OrderItemDao)	result
1	正常	正常	正常	正常
2	result = 10/0	正常	正常	整个事务回滚
3	正常	result = 10/0	正常	整个事务回滚
4	正常	正常	result = 10/0	整个事务回滚
5	正常	正常	正常 REQUIRES_NEW	(1) (2)正常, (3) Lock wait timeout exceeded
6	result=10/0,在 saveOrderItems 之前	正常	正常 REQUIRES_NEW	整个事务回滚
7	result=10/0,在 saveOrderItems 之后	正常	正常 REQUIRES_NEW	(1)(2)回滚，(3) Lock wait timeout exceeded
8	正常	result=10/0	正常 REQUIRES_NEW	整个事务回滚
9	正常	正常	result=10/0 REQUIRES_NEW	(1)(2)成功，(3)Lock wait timeout exceeded(因为数据库操作在除零之前)

原因

1. 一切正常
2. (1)(2)(3)在同一事务， （1）出错则整个事务回滚
3. (1)(2)(3)在同一事务， （2）出错则整个事务回滚
4. (1)(2)(3)在同一事务， （3）出错则整个事务回滚
5. (1)(2)在同一事务（记作 A），未出错正常执行，（3）处于另一事务（记作 B），二者竞争统一资源，B 需要等待 A 提交才能操作表，A 需要等待 B 执行完才能提交，故陷入死锁
6. (1)(2)在同一事务，在（3）执行之前出错，(1)(2)事务回滚，（3）无法执行
7. (1)(2)在同一事务，在（3）之后出错，(1)(2)事务回滚，（3）处于另一事务，不会由于(1)(2)回滚，但由于与 5 相同的理由执行超时
8. (1)(2)在同一事务，在（3）执行之前出错，(1)(2)事务回滚，（3）无法执行
9. (1)(2)在同一事务，在（3）执行时出错，(1)(2)事务回滚，（3）由于与 5 相同的理由执行超时，由于除零错在数据库操作之后，因此不会触发除零错