第一章作业-2020211502-王小龙

1-思考题.

a.

图中Nginx的作用有：

作为负载均衡器，可以根据不同的策略将客户端的请求分配给多个后端服务器，提高系统的可用性和性能。

需要Nginx-S和Nginx-M的原因：

为了提高nginx的可用性和稳定性，可以使用keepalived（心跳检查）和虚拟IP来配置nginx高可用集群，当一个nginx服务器出现故障时，另一个nginx服务器可以接管服务。

b.

该架构可以如下方式扩容：

- 业务拆分
- 应用集群部署（分布式部署，集群部署和负载均衡）
- 多级缓存
- 单点登录（分布式Session）
- 数据库集群（读写分离，分库分表）
- 服务化
- 消息队列
- 其他技术，如CDN，反向代理，分布式文件系统，大数据处理等系统

c.

消息队列ActiveMQ的作用有以下几点：

- 消除高并发访问高峰：用户的请求数据先写入消息队列，再由消息队列分发给后端服务器处理，避免数据库压力过大和系统响应延迟。

- 实现业务解耦：不同的业务模块之间通过消息队列进行通信，降低了耦合度和依赖性，提高了系统的灵活性和可维护性。
- 实现数据同步：通过消息队列将数据变更事件发送给其他系统或组件，保证数据的一致性和完整性。
- 实现消息广播：通过消息队列将同一条消息发送给多个订阅者，实现发布/订阅模式。

d.

使用FastDFS这种分布式文件系统的原因有以下几点：

- 解决大容量存储问题：当文件数量和大小超过单台服务器的承载能力时，需要将文件分散存储在多台服务器上，而不是使用单点存储或者复制存储。
- 解决高并发访问问题：当文件的访问量非常高时，需要通过负载均衡和缓存等技术，提高文件的读写性能和可用性，而不是依赖单台服务器的带宽和IO。
- 解决数据一致性问题：当文件发生变更或删除时，需要保证所有服务器上的文件都能同步更新或清除，而不是出现脏数据或丢失数据。

e.

使用分布式Session的主要目的是解决在分布式系统中，同一个用户的多个请求可能被分发到不同的服务器上，导致Session不一致的问题。

图中分布式Session大概是存储在第三方数据库（如Redis、MongoDB等），每个服务器都从数据库中获取或更新Session。

2-思考题.

a.

1. 设计质量是指架构设计是否满足了业务需求和技术约束，是否具有良好的可维护性、可重用性和概念完整性。具体来说：

- 可维护性是指架构设计是否易于修改和扩展，是否能够适应变化的需求和环境，是否能够降低维护成本和风险。
- 可重用性是指架构设计是否能够复用已有的组件或模块，是否能够提高开发效率和质量，是否能够避免重复造轮子。
- 概念完整性是指架构设计是否符合一致的原则和规范，是否能够清晰地表达架构的意图和逻辑，是否能够减少歧义和误解。

2. 系统质量是指软件系统在质量方面的需求，例如可用性、可修改性、性能、安全性等。
- 可支持性是指软件系统能够适应不同的环境和用户需求，以及能够快速地修复错误和提供服务。
- 可测试性是指软件系统能够方便地进行测试，以验证其功能和质量，以及能够提供有效的测试反馈。

3. 运行时质量是指软件系统在运行过程中表现出来的质量属性，例如可用性、性能、安全性等。

4. 用户质量是指软件系统能够满足用户的需求和期望，以及能够提供良好的用户体验。

b.

分解设计思路如下：

- 按照功能模块分解，将系统划分为不同的子系统，例如用户管理子系统、订单管理子系统、票务管理子系统、支付管理子系统等，每个子系统负责一类相关的功能，实现高内聚低耦合。
- 按照层次结构分解，将每个子系统进一步划分为不同的层次，例如表示层、业务逻辑层、数据访问层等，每个层次负责一类相关的任务，实现清晰的职责划分和接口定义。
- 按照部署结构分解，将每个层次进一步划分为不同的部署单元，例如Web服务器、应用服务器、数据库服务器等，每个部署单元负责一类相关的资源管理和服务提供，实现可伸缩性和可靠性。

3-思考题.

a.

相对于单体架构的集群化部署，Serverless 架构有以下优势：

- 降低了启动成本和运营成本，因为不需要购买或维护服务器，只需按照实际使用量付费。
- 提高了研发交付速度，因为不需要关注底层的技术选型、配置、部署等细节，可以专注于业务逻辑和用户体验。
- 实现了弹性扩缩容，因为云服务提供商会根据负载变化自动调整资源分配和服务规模。
- 增强了可用性和冗余性，因为云服务提供商会保证单个机器故障不会导致服务中断，并且可以将副本分布在不同的地理位置以应对灾难情况。

b.

所谓的冷启动问题，指的是 Serverless 架构在弹性伸缩时可能会触发环境准备（初始化工作空间）、下载文件、配置环境、加载代码和配置、函数实例启动完整的实例启动流程，导致原本数毫秒或数十毫秒可以得到的请求响应需要在数百毫秒或数秒得到，进而影响业务处理速度。

4-实验题.

选择方案一

代码仓库地址如下(已添加老师为成员)：

https://gitee.com/lhfhlhfl/chapter1-Web_backend.git

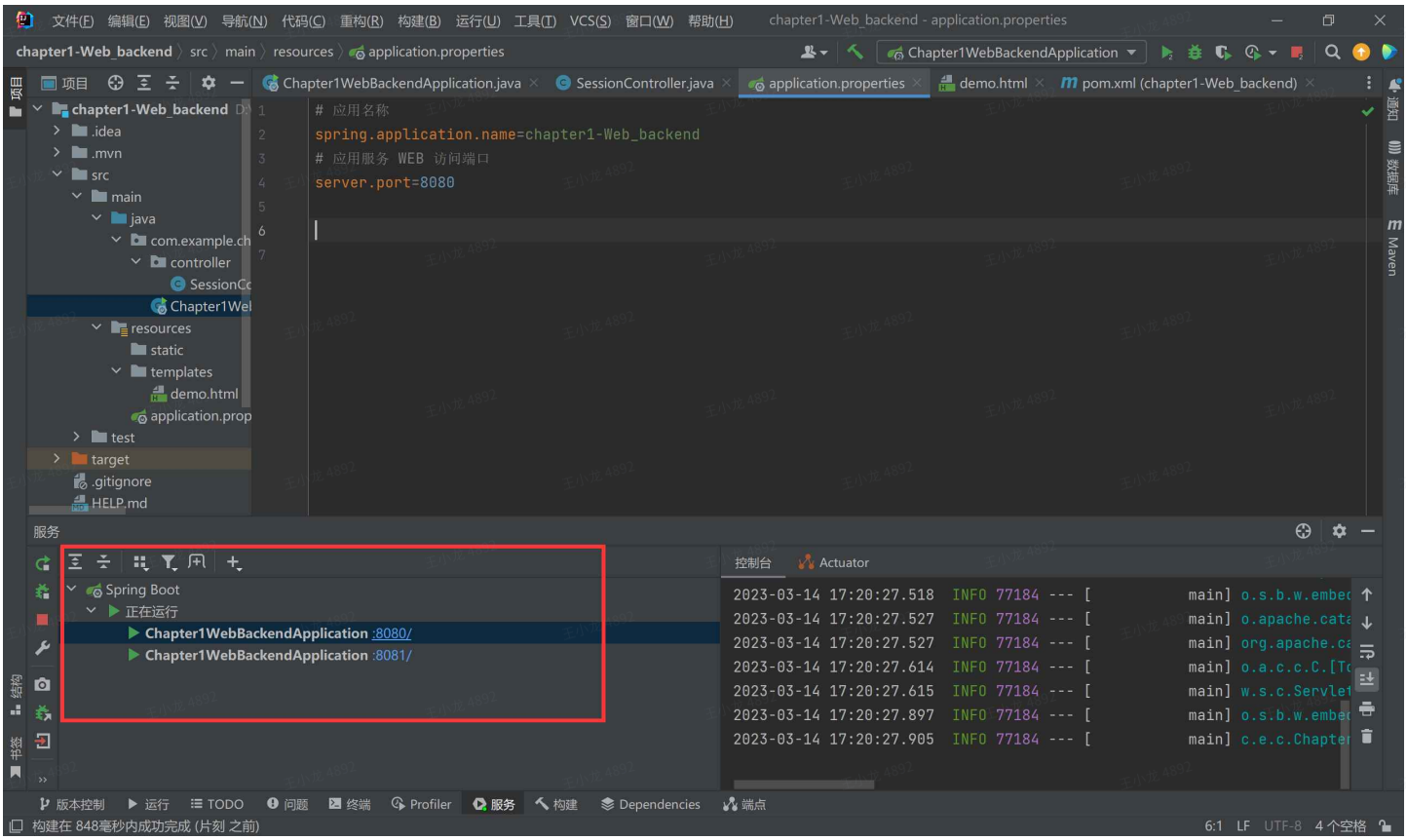
<1>

编写一个示例Web应用，实现/session/set?key={}&value={} 和/session/get?key={}两个后端http接口（无需编写前端页面）接口功能分别是将数据存入读取当前请求相关的session代码如下：

```
0 个用法
@RestController
public class SessionController {

    // 设置session属性
    0 个用法
    @GetMapping("/session/set")
    public String setSession(@RequestParam String key, @RequestParam String value, HttpSession session) {
        session.setAttribute(key, value);
        return "Session attribute " + key + " set to " + value;
    }

    // 获取session属性
    0 个用法
    @GetMapping("/session/get")
    public String getSession(@RequestParam String key, HttpSession session) {
        Object value = session.getAttribute(key);
        if (value == null) {
            return "Session attribute " + key + " not found";
        } else {
            return "Session attribute " + key + " is " + value;
        }
    }

}
```

<2>

使用不同的端口，启动两个Web应用服务实例如下：



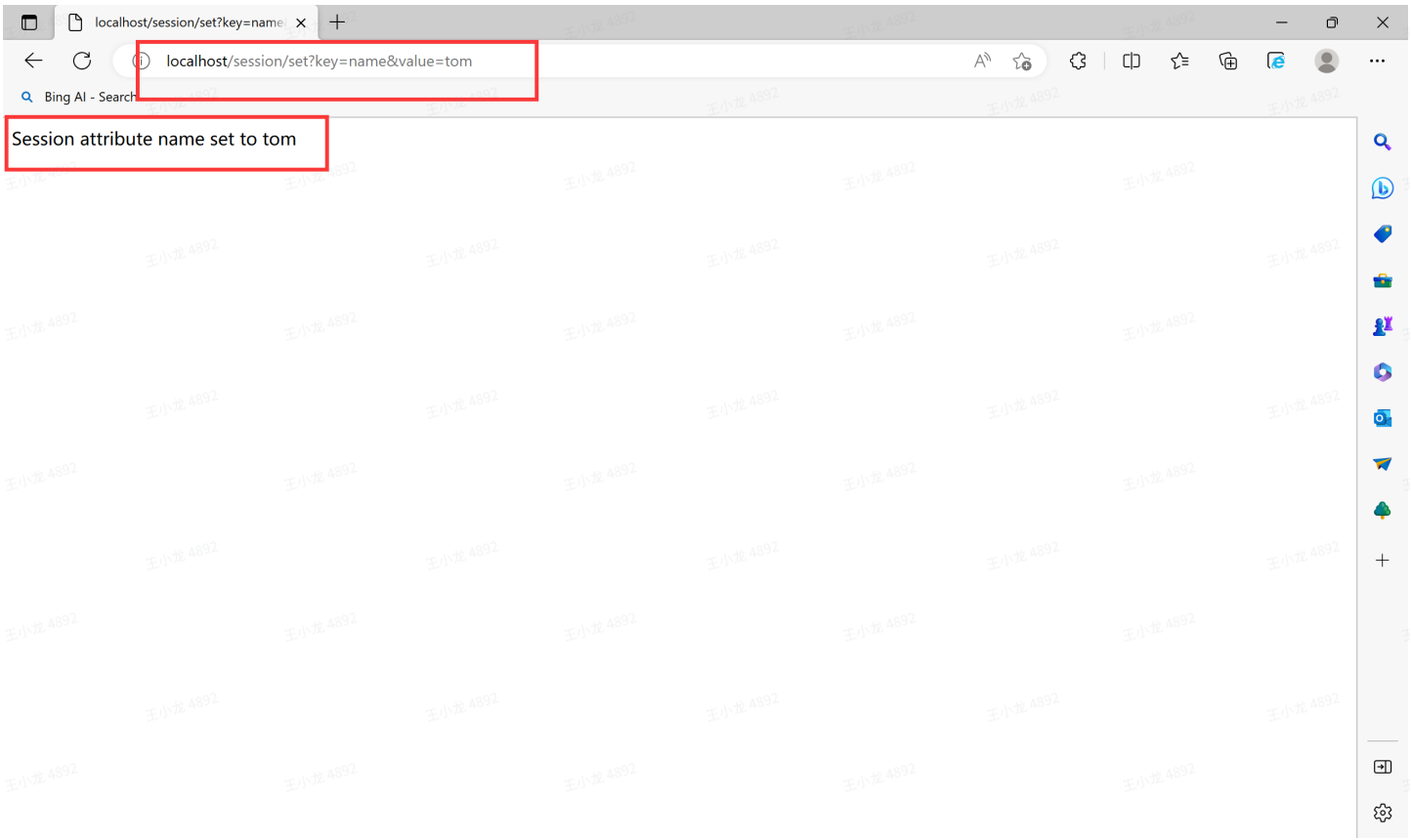
<3>

使用Nginx配置负载均衡，将请求负载转发到两个应用服务实例上，如下操作：

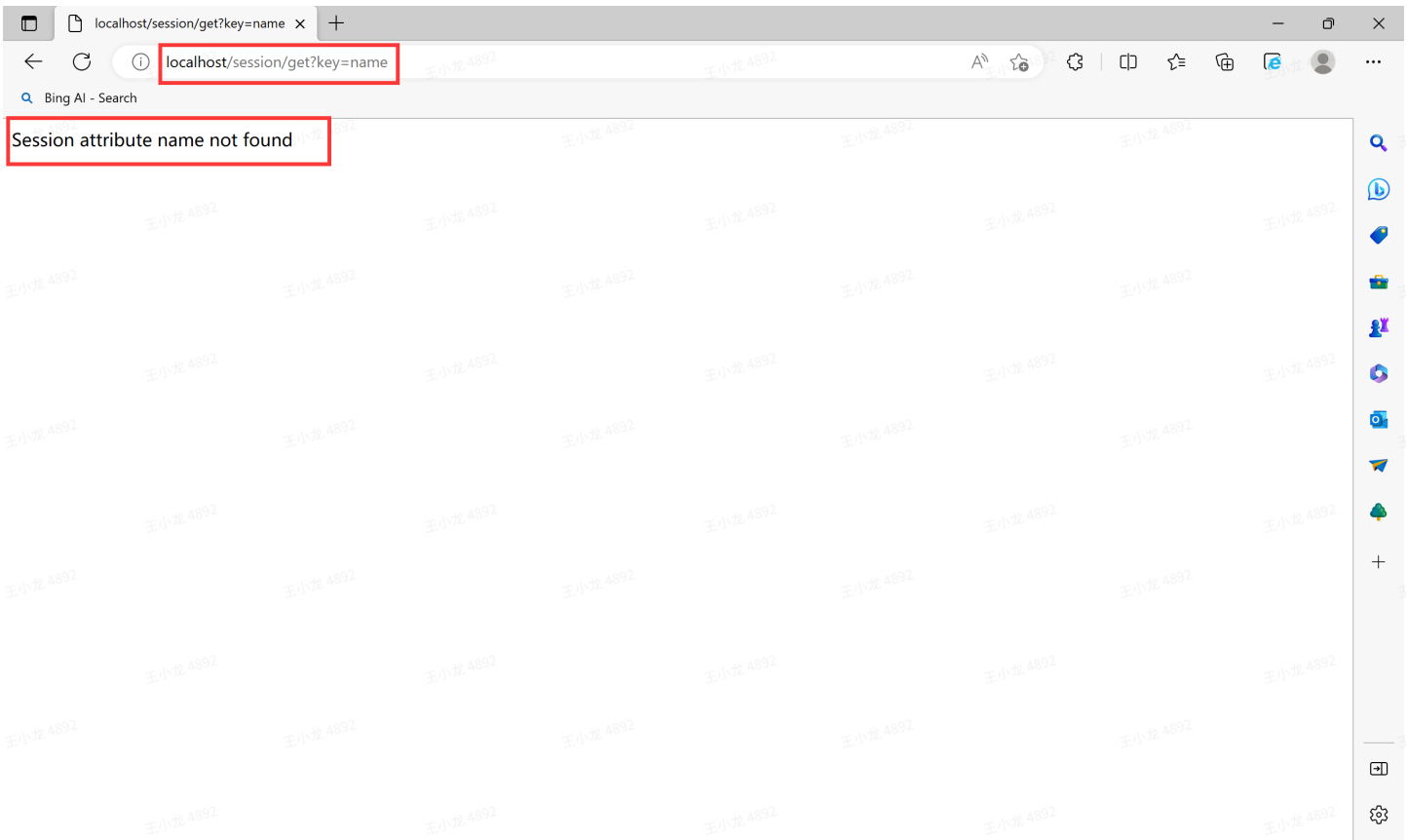
第一步，先编辑nginx.conf里的配置如下，然后打开nginx：

```
upstream pancm{
    server 127.0.0.1:8080;
    server 127.0.0.1:8081;
}
server {
    listen       80;
    server_name  127.0.0.1;

    location / {
        root   html;
        proxy_pass http://pancm;
        index  index.html index.htm;
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   html;
    }
}
```

第二步，访问查看运行状况，发现第一次访问正确返回了：



然后，在访问get的页面，发现无法返回数据，说明出现了问题



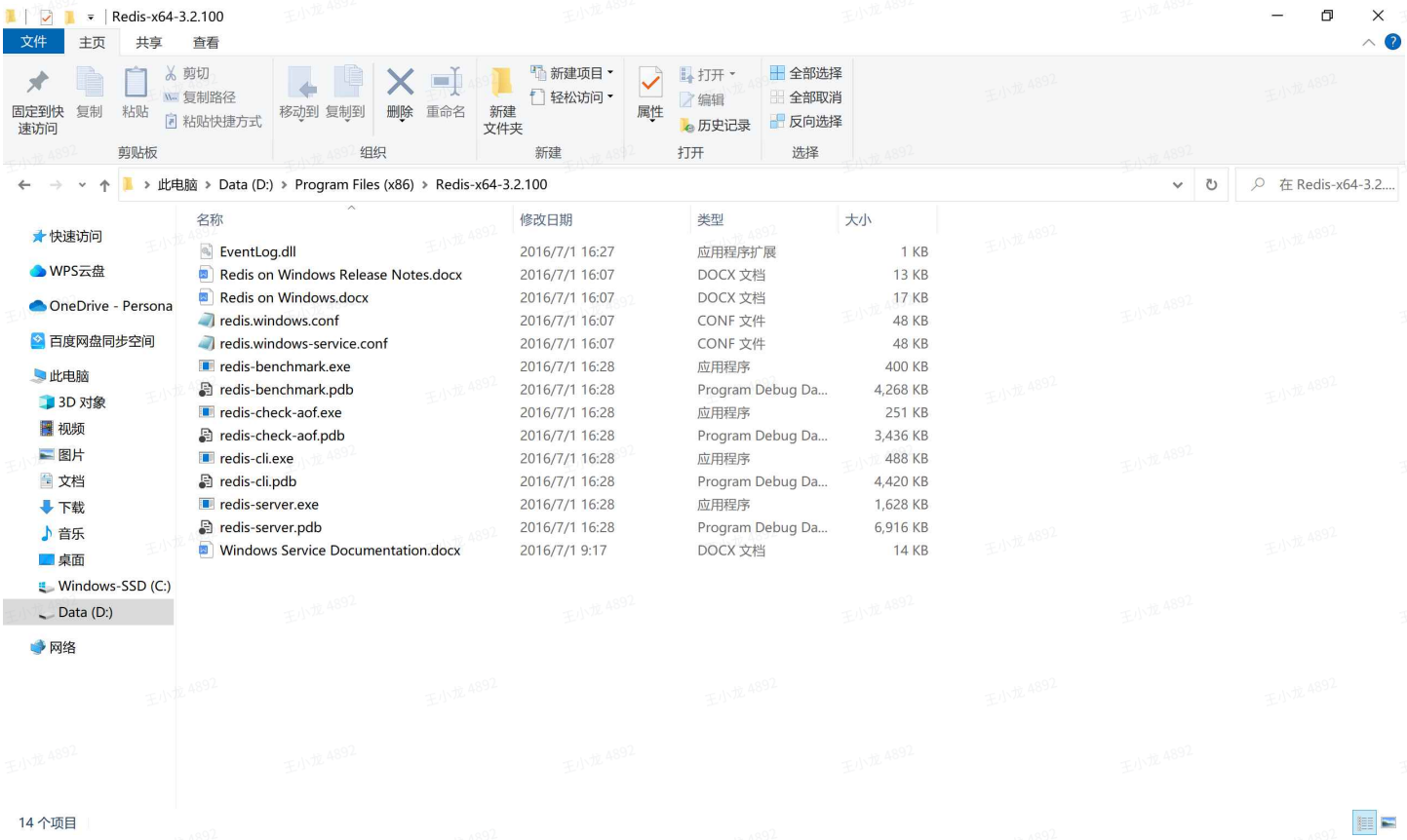
经过分析，可以知道，如果不采用session共享，即每个应用服务实例都有自己的session存储空间，那么可能会发生用户登录后刷新页面或访问其他页面时，可能会被分配到另一个应用服务实例，导致登录信息失效或异常。

用户在多个页面之间切换时，可能会被分配到不同的应用服务实例，导致数据不一致或冲突。

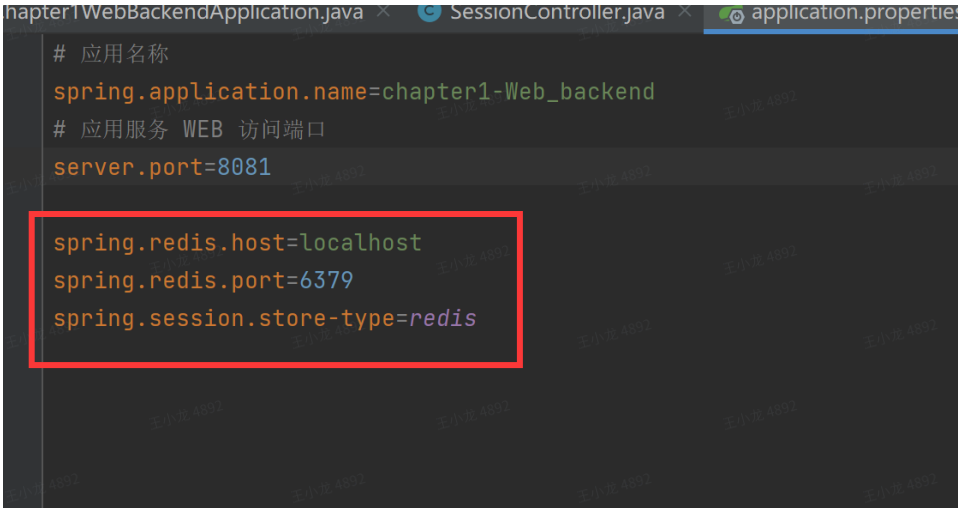
所以猜测上述结果就是存入与读取访问到了不同服务器的结果。

<4>

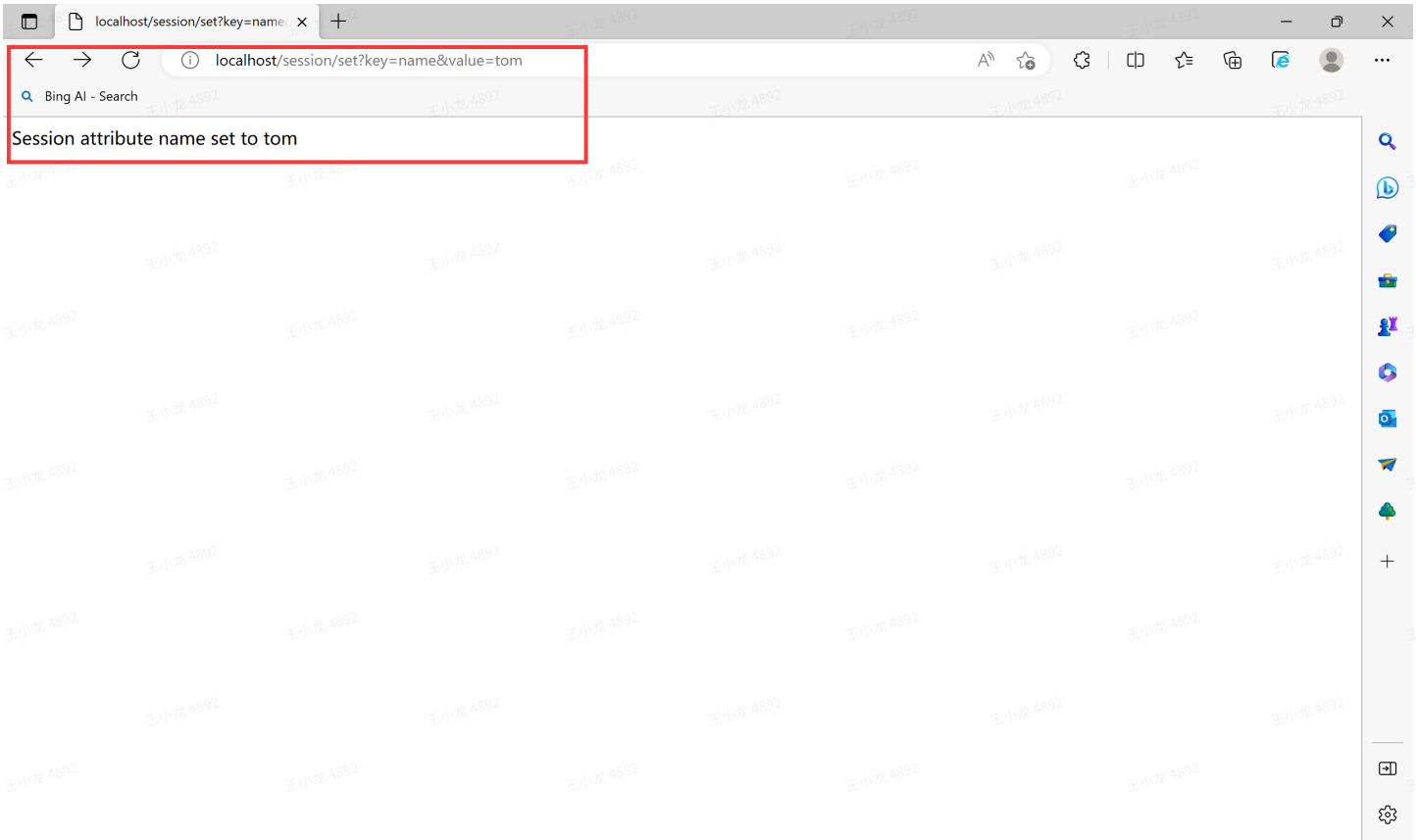
本地安装redis如下：



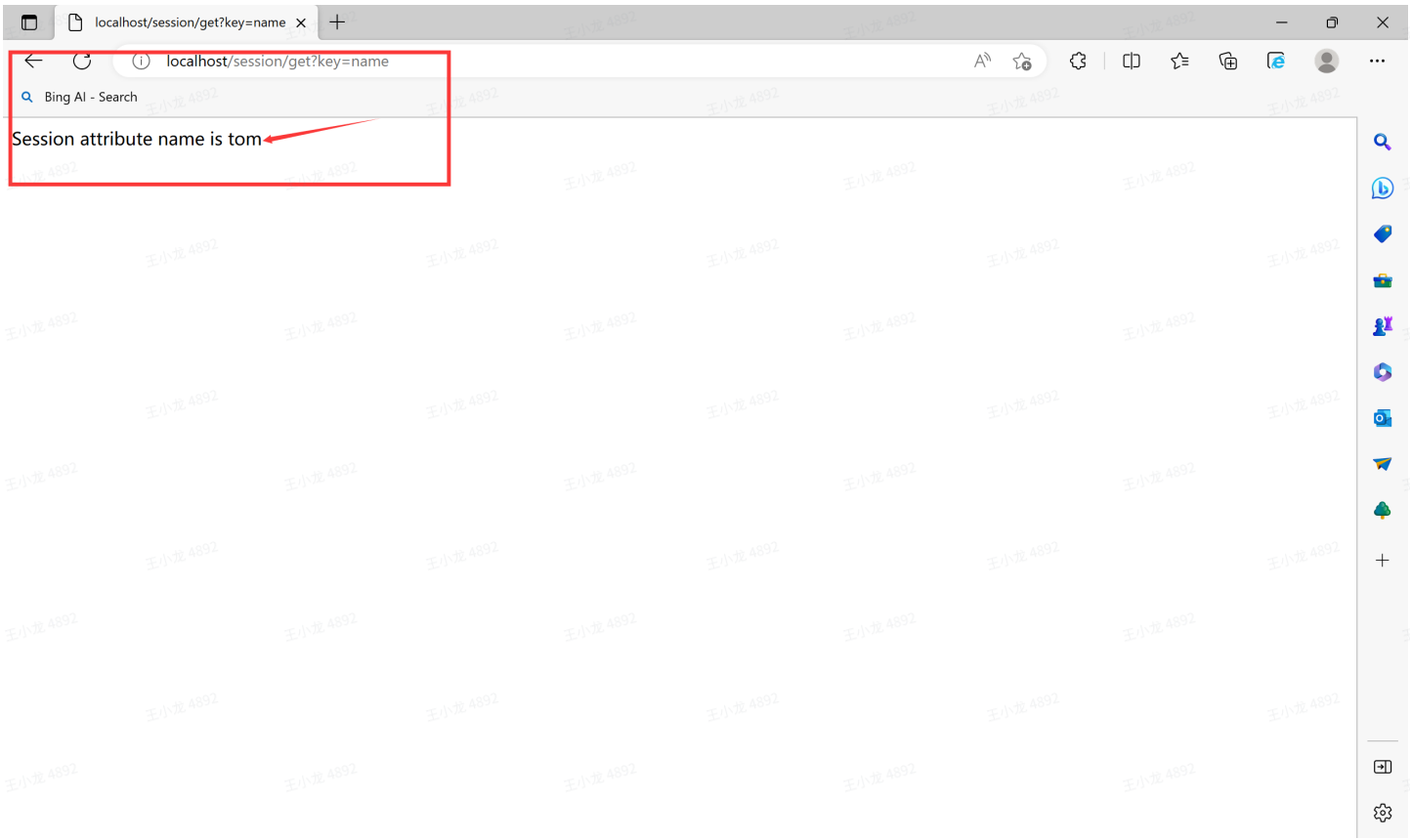
项目引入spring data redis 和spring session如下：



再次打开nginx，访问页面如下：



然后访问get页面如下，发现正确返回了所需数据，由此可以知道实现了session共享，解决了上述session存取的问题：


localhost/session/get?key=name
localhost/session/get?key=name
Bing AI - Search
Session attribute name is tom