Dubbo

http://dubbo.apache.org/zh-cn/docs/user/quick-start.html

Zookeeper在windows下的配置

1 先解压zookeeper压缩文件

2 进入到conf目录, 设置zoo.cfg文件，dataDir=F:\zookeeper-3.4.14\data

3 设置环境变量ZOOKEEPER\_HOME=F:\zookeeper-3.4.14

4在path变量中添加路径 ;F:\zookeeper-3.4.14\bin

5 设置JAVA\_HOME

服务端的启动： zkServer

客户端的启动： zkCli

查询zookeeper下的目录： ls /

dubbo-demo-api: 服务端和消费端都需要使用的api接口。

dubbo-demo-provider: 服务端通过dubbo框架，提供api接口的实现。

dubbo-demo-consumer: 消费端通过duboo框架，使用api接口， 进行消费。

Maven的设置，

1. 解压maven压缩文件
2. M2\_HOME=F:\apache-maven-3.6.1-bin\apache-maven-3.6.1
3. path=;F:\apache-maven-3.6.1-bin\apache-maven-3.6.1\bin
4. mvn -version

在eclipse中设置maven

Window->preferences->maven->installation->add...

添加上面的maven

配置maven镜像

cd F:\apache-maven-3.6.1-bin\apache-maven-3.6.1\conf

修改settings.xml文件

<mirrors>

<mirror>

<id>alimaven</id>

<mirrorOf>\*</mirrorOf>

<name>aliyun maven</name>

<url>http://maven.aliyun.com/nexus/content/groups/public</url>

</mirror>

</mirrors>

学习springboot

New project->spring starter project

https://start.spring.io

Provider配置：

server.port=8333

spring.dubbo.application.name=provider

spring.dubbo.application.registry=zookeeper://127.0.0.1:2181

Consumer的配置

server.port=8330

spring.dubbo.application.name=consumer

spring.dubbo.application.registry=zookeeper://127.0.0.1:2181

@RestController

public class HelloController {

@Reference

private IHello helloService;

@RequestMapping("/hello")

public String hello() {

return helloService.hello();

}

}

服务端和消费端依赖的包

<dependency>

<groupId>com.alibaba.spring.boot</groupId>

<artifactId>dubbo-spring-boot-starter</artifactId>

<version>2.0.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.101tec/zkclient -->

<dependency>

<groupId>com.101tec</groupId>

<artifactId>zkclient</artifactId>

<version>0.11</version>

</dependency>

<dependency>

<groupId>com.example</groupId>

<artifactId>springboot-interface</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

Springboot启动程序：

@SpringBootApplication

@EnableDubboConfiguration

public class SpringbootProviderApplication {

public static void main(String[] args) {

SpringApplication.run(SpringbootProviderApplication.class, args);

}

}

Sso单点登录

<https://hellokoding.com/hello-single-sign-on-sso-with-json-web-token-jwt-spring-boot/>

<https://www.jianshu.com/p/576dbf44b2ae>

cookie.setPath（）的用法

正常的cookie只能在一个应用中共享，即一个cookie只能由创建它的应用获得。  
1.可在同一应用服务器内共享方法：设置cookie.setPath("/");  
    本机tomcat/webapp下面有两个应用：cas和webapp\_b，  
    1）原来在cas下面设置的cookie，在webapp\_b下面获取不到，path默认是产生cookie的应用的路径。  
    2）若在cas下面设置cookie的时候，增加一条cookie.setPath("/");或者cookie.setPath("/webapp\_b/");就可以在webapp\_b下面获取到cas设置的cookie了。  
    3）此处的参数，是相对于应用服务器存放应用的文件夹的根目录而言的(比如tomcat下面的webapp)，因此cookie.setPath("/");之后，可以在webapp文件夹下的所有应用共享cookie，而cookie.setPath("/webapp\_b/");是指cas应用设置的cookie只能在webapp\_b应用下的获得，即便是产生这个cookie的cas应用也不可以。  
    4）设置cookie.setPath("/webapp\_b/jsp")或者cookie.setPath("/webapp\_b/jsp/")的时候，只有在webapp\_b/jsp下面可以获得cookie，在webapp\_b下面但是在jsp文件夹外的都不能获得cookie。  
    5）设置cookie.setPath("/webapp\_b");，是指在webapp\_b下面才可以使用cookie，这样就不可以在产生cookie的应用cas下面获取cookie了  
    6）有多条cookie.setPath("XXX");语句的时候，起作用的以最后一条为准。  
    6）设置多个path的方法？？？  
  
2.跨域共享cookie的方法：设置cookie.setDomain(".jszx.com");  
    A机所在的域：home.langchao.com,A有应用cas  
    B机所在的域：jszx.com，B有应用webapp\_b  
    1）在cas下面设置cookie的时候，增加cookie.setDomain(".jszx.com");，这样在webapp\_b下面就可以取到cookie。  
    2）这个参数必须以“.”开始。  
    3）输入url访问webapp\_b的时候，必须输入域名才能解析。比如说在A机器输入：http://lc-bsp.jszx.com:8080/webapp\_b,可以获取cas在客户端设置的cookie，而B机器访问本机的应用，输入：http://localhost:8080/webapp\_b则不可以获得cookie。  
    4）设置了cookie.setDomain(".jszx.com");，还可以在默认的home.langchao.com下面共享。  
    5）设置多个域的方法？？？  
最近的工作经常用到cookie，看了一些资料，虽然原来依赖cookie比较多的方案已经改成现在这个了，但是当时看得一些东西还是最好记下来，免得过些天就一干二净了，怪可惜的。

下面实验报告任选一个：

分布式框架dubbo实验报告一

1. 使用springboot和dubbo框架实现分布式消费和服务
2. 使用json web token实现单点登录功能

分布式框架dubbo实验报告二

1. 登录服务
2. 数据库访问服务， 完成insert, update, delete, select
3. 前台浏览服务

注重功能的实现， 不关注界面！