# 单体到微服务改造

## 第1章 项目概述

### 1.1 房产销售平台

首页轮播、房源列表、房产列表、分页、热门房产（点击率）、按类型排序、

评价、登录、收藏、评论、添加房产、搜索房源、个人信息、我的房产信息、收藏列表、经纪人信息、百科...



单体应用



技术栈

单体架构实现

SSM：SpringBoot、SpringMVC、Mybatis；

数据库设计

本地缓存：Guava Cache本地缓存；

Redis：访问技术、支撑推荐业务；

单元测试：Spring Test；

邮件发送：Spring Mail；

Ajax用于：JQuery Ajax、Spring MVC

分页组件：

单体优缺点

微服务架构实现

微服务优缺点

服务拆分

服务注册发现：Spring Cloud Eureka

远程通信：RestTemplate；

断路器：Spring Cloud Hystrix；防止级联故障

ELK日志检索：定位分布式系统日志汇总处理Elasticsearch、Logstash、Kibana

分布式追踪：全链路追踪，定位性能瓶颈，理清服务依赖；Spring Cloud Sleuth，Zipkin Dashboard监控链路

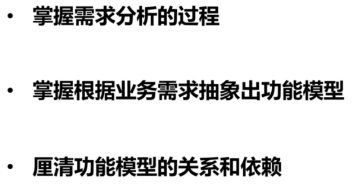
JWT：Token技术，解决分布式环境中的身份认证问题；

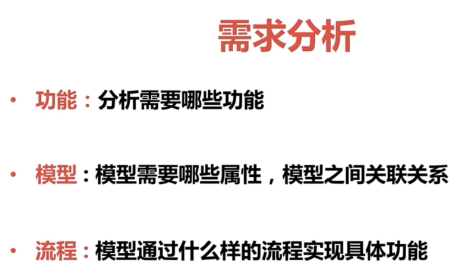
服务监控：服务监控平台，Hystrix Dashboard、Eureka UI

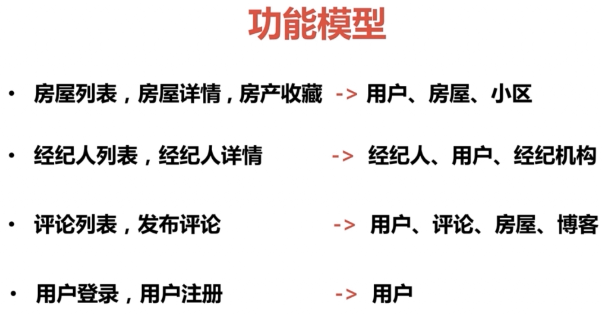


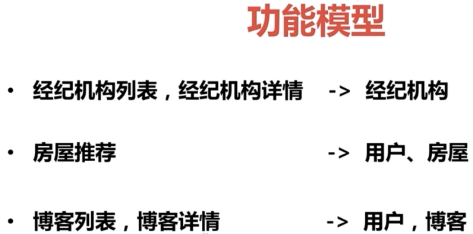
## 第2章 单体架构之项目概要设计和数据表设计

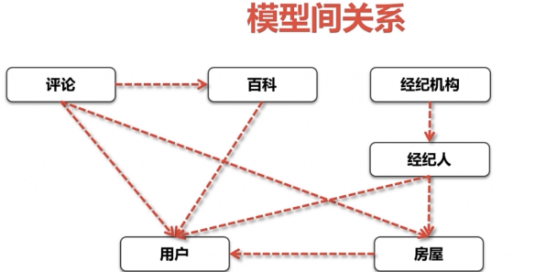
### 2.1项目需求分析





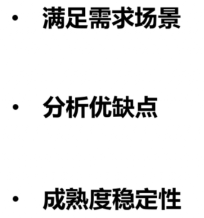


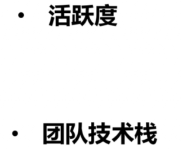


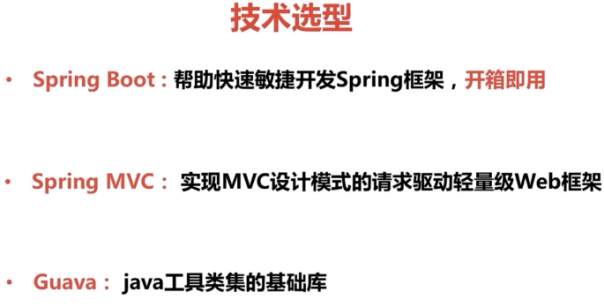


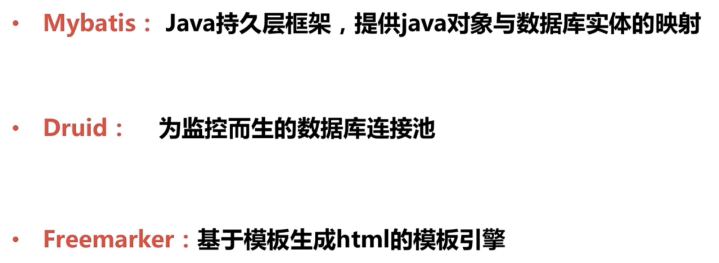
### 2.2单体项目技术选型和架构设计



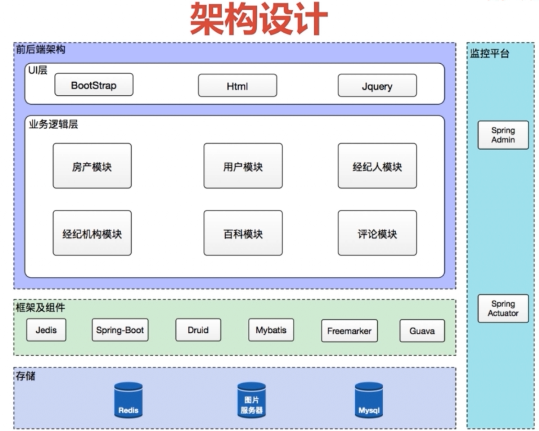






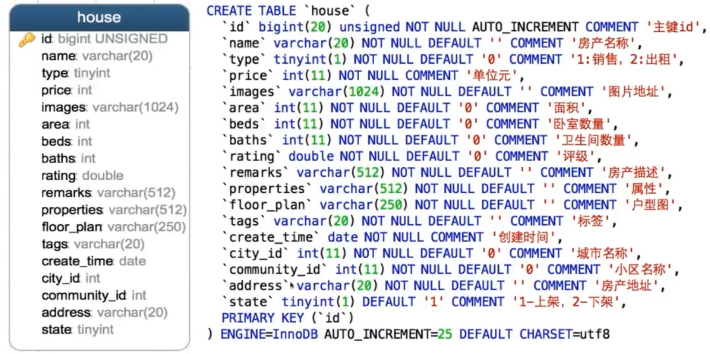






### 2.3数据库设计

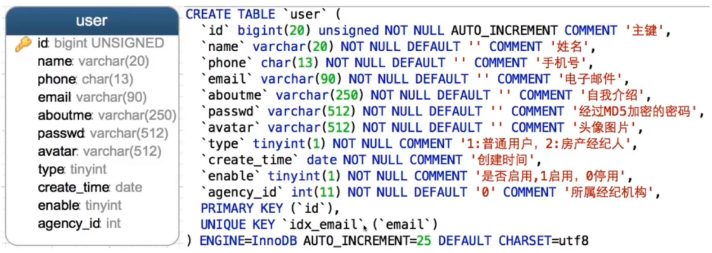
房产表：house



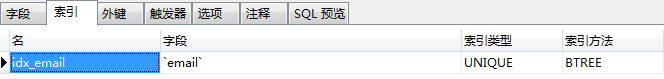
Create\_time默认时间设置：



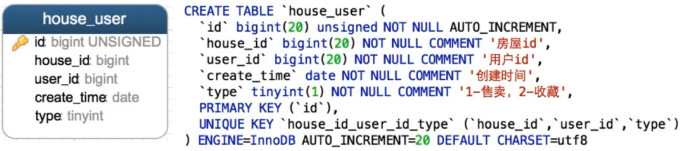
用户表：user



添加索引：



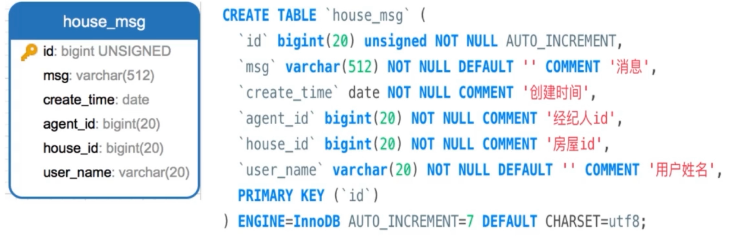
房产用户表：house\_user



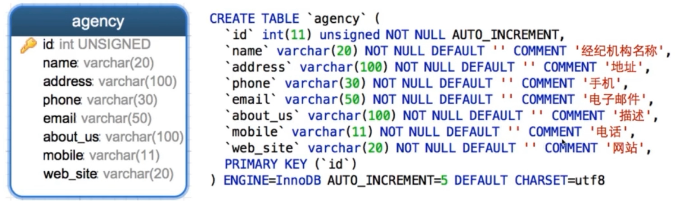
添加索引：



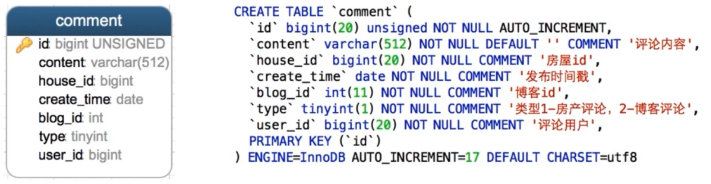
房产留言表：house\_msg



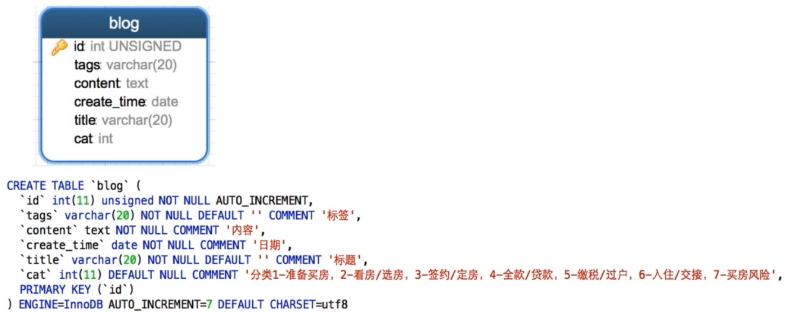
经济机构表：agency



评论表：comment



百科表：blog



## 单体架构之SpringBoot工程框架搭建与技巧

### 3.1创建SpringBoot工程





创建SpringBoot项目



SpringBoot目录结构



Pom.xml实现SpringBoot的两种方式：

①继承spring-boot-starter-parent

|  |
| --- |
| **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-parent</artifactId>**  **<version>1.4.7.RELEASE</version>**  **<type>pom</type>**  **</dependency>** |

②依赖spring-boot-dependencies

|  |
| --- |
| <**dependencyManagement**>  <**dependencies**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-dependencies</**artifactId**>  <**version**>1.4.7.RELEASE</**version**>  <**type**>pom</**type**>  <**scope**>import</**scope**>  </**dependency**>  </**dependencies**> </**dependencyManagement**> |

区别：

覆盖spring-boot提供的依赖的jar包，以mysql驱动包（spring-boot1.4.7.RELEASE提供5.1.42）的版本号为例：

方式①继承spring-boot-starter-parent时，需要将mysql.version进行添加至properties：

|  |
| --- |
| <**properties**>  <**mysql.version**>6.0.6</**mysql.version**>  </**properties**> |

方式②在dependency中添加spring-boot-dependencies依赖，需要在dependecies依赖中添加：

|  |
| --- |
| <**dependency**>  <**groupId**>mysql</**groupId**>  <**artifactId**>mysql-connector-java</**artifactId**>  <**version**>6.0.6</**version**> </**dependency**> |

### 3.2引入内嵌Servlet容器

配置日志级别和格式

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**configuration debug="true" scan="true" scanPeriod="40 seconds"**>  *<!--注意: logback.xml在单体中只为了做测试使用，未来会换成log4j2，课程最后几章也会讲到log4j2及log4j2的相比logback的优势 -->* <**property name="log.base" value="logs"** />   <**contextListener class="ch.qos.logback.classic.jul.LevelChangePropagator"**>  <**resetJUL**>true</**resetJUL**>  </**contextListener**>   <**appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender"**>  <**encoder charset="UTF-8"** >  <**pattern**>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level [%logger{50}:%line] - %msg%n</**pattern**>  <**charset**>UTF-8</**charset**>  </**encoder**>  </**appender**>    <**appender name="FILE-ERROR"  class="ch.qos.logback.core.rolling.RollingFileAppender"**>  <**filter class="ch.qos.logback.classic.filter.LevelFilter"**>  <**level**>ERROR</**level**>  <**onMatch**>ACCEPT</**onMatch**>  <**onMismatch**>DENY</**onMismatch**>  </**filter**>  <**append**>true</**append**>  <**rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy"**>  <**fileNamePattern**>${log.base}/house-error.%d{yyyy-MM-dd}.log  </**fileNamePattern**>  <**maxHistory**>10</**maxHistory**>  </**rollingPolicy**>  <**encoder charset="UTF-8"** >  <**pattern**>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level [%logger{50}:%line] - %msg%n</**pattern**>  <**charset**>UTF-8</**charset**>  </**encoder**>  </**appender**>    <**appender name="FILE-INFO"  class="ch.qos.logback.core.rolling.RollingFileAppender"**>  <**filter class="ch.qos.logback.classic.filter.LevelFilter"**>  <**level**>INFO</**level**>  <**onMatch**>ACCEPT</**onMatch**>  <**onMismatch**>DENY</**onMismatch**>  </**filter**>  <**append**>true</**append**>  <**rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy"**>  <**fileNamePattern**>${log.base}/house-info.%d{yyyy-MM-dd}.log  </**fileNamePattern**>  <**maxHistory**>5</**maxHistory**>  </**rollingPolicy**>  <**encoder charset="UTF-8"** >  <**pattern**>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level [%logger{50}:%line] - %msg%n</**pattern**>  <**charset**>UTF-8</**charset**>  </**encoder**>  </**appender**>   <**appender name="FILE-WARN"  class="ch.qos.logback.core.rolling.RollingFileAppender"**>  <**filter class="ch.qos.logback.classic.filter.LevelFilter"**>  <**level**>WARN</**level**>  <**onMatch**>ACCEPT</**onMatch**>  <**onMismatch**>DENY</**onMismatch**>  </**filter**>  <**append**>true</**append**>  <**rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy"**>  <**fileNamePattern**>${log.base}/house-warn.%d{yyyy-MM-dd}.log  </**fileNamePattern**>  <**maxHistory**>10</**maxHistory**>  </**rollingPolicy**>  <**encoder charset="UTF-8"** >  <**pattern**>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level [%logger{50}:%line] - %msg%n</**pattern**>  <**charset**>UTF-8</**charset**>  </**encoder**>  </**appender**>   <**appender name="FILE-DEBUG"  class="ch.qos.logback.core.rolling.RollingFileAppender"**>  <**filter class="ch.qos.logback.classic.filter.LevelFilter"**>  <**level**>DEBUG</**level**>  <**onMatch**>ACCEPT</**onMatch**>  <**onMismatch**>DENY</**onMismatch**>  </**filter**>  <**append**>true</**append**>  <**rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy"**>  <**fileNamePattern**>${log.base}/house-debug.%d{yyyy-MM-dd}.log  </**fileNamePattern**>  <**maxHistory**>1</**maxHistory**>  </**rollingPolicy**>  <**encoder charset="UTF-8"** >  <**pattern**>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level [%logger{50}:%line] - %msg%n</**pattern**>  <**charset**>UTF-8</**charset**>  </**encoder**>  </**appender**>      <**logger name="com.mooc.house.mapper" level="DEBUG" additivity="false"** >  <**appender-ref ref="FILE-DEBUG"** />  <**appender-ref ref="CONSOLE"** />  </**logger**>   <**root level="INFO"**>  <**appender-ref ref="FILE-DEBUG"** />  <**appender-ref ref="FILE-ERROR"** />  <**appender-ref ref="FILE-INFO"** />  <**appender-ref ref="CONSOLE"** />  </**root**>  </**configuration**> |

定义一个日志过滤器

|  |
| --- |
| @Configuration **public class** FilterBeanConfig {    */\*\*  \* 1.构造filter  \* 2.配置拦截urlPattern  \* 3.利用FilterRegistrationBean进行包装  \** ***@return*** *\*/* @Bean  **public** FilterRegistrationBean logFilter(){  FilterRegistrationBean filterRegistrationBean = **new** FilterRegistrationBean();  filterRegistrationBean.setFilter(**new** LogFilter());  List<String> urList = **new** ArrayList<String>();  urList.add(**"\*"**);  filterRegistrationBean.setUrlPatterns(urList);  **return** filterRegistrationBean;  } } |
| **public class** LogFilter **implements** Filter{    **private** Logger **logger** = LoggerFactory.*getLogger*(LogFilter.**class**);   @Override  **public void** init(FilterConfig filterConfig) **throws** ServletException {    }   @Override  **public void** doFilter(ServletRequest request, ServletResponse response, FilterChain chain)  **throws** IOException, ServletException {  **logger**.info(**"Request--coming"**);  chain.doFilter(request, response);    }   @Override  **public void** destroy() {    }  } |

select VERSION();

8.0.14

**73.11 Use Jetty instead of Tomcat**

[https://docs.spring.io/spring-boot/docs/1.5.4.RELEASE/reference/htmlsingle/#howto-use-jetty-instead-of-tomcat](https://docs.spring.io/spring-boot/docs/1.5.4.RELEASE/reference/htmlsingle/" \l "howto-use-jetty-instead-of-tomcat)

Maven

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  <exclusions>  <exclusion>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-tomcat</artifactId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-jetty</artifactId>  </dependency> |

Gradle

|  |
| --- |
| configurations {  compile.exclude module: "spring-boot-starter-tomcat"  }  dependencies {  compile("org.springframework.boot:spring-boot-starter-web:1.5.4.RELEASE")  compile("org.springframework.boot:spring-boot-starter-jetty:1.5.4.RELEASE")  // ...  } |

### 3.3数据访问层搭建--整合Mybatis

Mybatis-config.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* **<!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd"*>*** <**configuration**>  <**settings**>  *<!-- 配置关闭缓存 -->* <**setting name="cacheEnabled" value="false"**/>  <**setting name="mapUnderscoreToCamelCase" value="true"**/>  <**setting name="useGeneratedKeys" value="true"**/>  <**setting name="defaultExecutorType" value="REUSE"**/>  *<!-- 事务超时时间 -->* <**setting name="defaultStatementTimeout" value="600"**/>  </**settings**>      <**typeAliases**>  <**typeAlias type="com.mooc.house.common.model.User" alias="user"** />  </**typeAliases**>    <**mappers**>  <**mapper resource="mapper/user.xml"** />  </**mappers**>  </**configuration**> |

方式一：XML配置

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"  xmlns:tx="http://www.springframework.org/schema/tx" xmlns:context="http://www.springframework.org/schema/context"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-4.2.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-4.2.xsd  http://www.springframework.org/schema/tx  http://www.springframework.org/schema/tx/spring-tx-4.2.xsd  http://www.springframework.org/schema/data/jpa"**>   *<!-- 数据源 -->* <**bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource"  init-method="init" destroy-method="close"**>  *<!-- 数据库基本信息配置 -->* <**property name="url" value="${druid.url}"** />  <**property name="username" value="${druid.username}"** />  <**property name="password" value="${druid.password}"** />  <**property name = "driverClassName" value = "${druid.driverClassName}"** />   *<!-- 初始化连接数量 -->* <**property name="initialSize" value="${druid.initialSize}"** />  *<!-- 最小空闲连接数 -->* <**property name="minIdle" value="${druid.minIdle}"** />  *<!-- 最大并发连接数 -->* <**property name="maxActive" value="${druid.maxActive}"** />  *<!-- 配置获取连接等待超时的时间 -->* <**property name="maxWait" value="${druid.maxWait}"** />   *<!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->* <**property name="timeBetweenEvictionRunsMillis" value="${druid.timeBetweenEvictionRunsMillis}"** />   *<!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->* <**property name="minEvictableIdleTimeMillis" value="${druid.minEvictableIdleTimeMillis}"** />  <**property name="validationQuery" value="${druid.validationQuery}"** />  <**property name="testWhileIdle" value="${druid.testWhileIdle}"** />  <**property name="testOnBorrow" value="${druid.testOnBorrow}"** />  <**property name="testOnReturn" value="${druid.testOnReturn}"** />   *<!-- 打开PSCache，并且指定每个连接上PSCache的大小 如果用Oracle，则把poolPreparedStatements配置为true，mysql可以配置为false。 -->* <**property name="poolPreparedStatements" value="${druid.poolPreparedStatements}"** />  <**property name="maxPoolPreparedStatementPerConnectionSize"  value="${druid.maxPoolPreparedStatementPerConnectionSize}"** />   *<!-- 配置监控统计拦截的filters -->* <**property name="filters" value="${druid.filters}"** />  <**property name="proxyFilters"**>  <**list**>  <**ref bean="stat-filter"**/>  </**list**>  </**property**>  </**bean**>  <**bean id="stat-filter" class="com.alibaba.druid.filter.stat.StatFilter"**>  <**property name="slowSqlMills" value="5000"**/>  <**property name="logSlowSql" value="true"**/>  <**property name="mergeSql" value="true"**/>  </**bean**>  <**bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"**>  <**property name="dataSource" ref="dataSource"** />  <**property name="configLocation" value="classpath:/mybatis/mybatis-config.xml"**/>  <**property name="mapperLocations" value="classpath:/mapper/\*.xml"** />  </**bean**>   <**bean class="org.mybatis.spring.mapper.MapperScannerConfigurer"**>  <**property name="basePackage" value="com.\*\*.mapper"** />  <**property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"** />  </**bean**>   *<!-- 事务配置 -->* <**bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager"**>  <**property name="dataSource" ref="dataSource"** />  </**bean**>   *<!-- 使用annotation注解方式配置事务 -->* <**tx:annotation-driven transaction-manager="transactionManager"** />  </**beans**> |
| **druid.driverClassName**=**com.mysql.jdbc.Driver  druid.url**=**jdbc:mysql://127.0.0.1:3306/manager?useUnicode=true&characterEncoding=UTF-8 druid.username**=**root druid.password**=**root123 druid.initialSize**=**10 druid.minIdle**=**6 druid.maxActive**=**50 druid.maxWait**=**60000 druid.timeBetweenEvictionRunsMillis**=**60000 druid.minEvictableIdleTimeMillis**=**300000 druid.validationQuery**=**SELECT 'x' druid.testWhileIdle**=**true druid.testOnBorrow**=**false druid.testOnReturn**=**false druid.poolPreparedStatements**=**false druid.maxPoolPreparedStatementPerConnectionSize**=**20 druid.filters**=**wall,stat** |

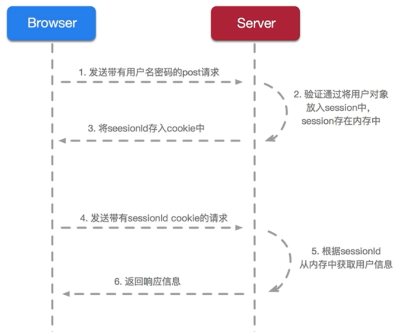
方式二：springboot配置

|  |
| --- |
| *#spring.datasource.url=jdbc:mysql://localhost:3306/user?characterEncoding=UTF-8* **spring.datasource.url**=**jdbc:mysql://localhost:3306/user?serverTimezone=UTC&useSSL=false&characterEncoding=UTF-8 spring.datasource.username**=**root spring.datasource.password**=**1234** *#spring.datasource.driver-class-name=com.mysql.jdbc.Driver* **spring.datasource.driver-class-name**=**com.mysql.cj.jdbc.Driver   mybatis.config-location**=**classpath:/mybatis/mybatis-config.xml** |

## 第10章微服务架构之用户服务（SpringData Redis及JWT等技术）

### 10.4用户服务—基于JWT的Token认证实现登录、鉴权接口

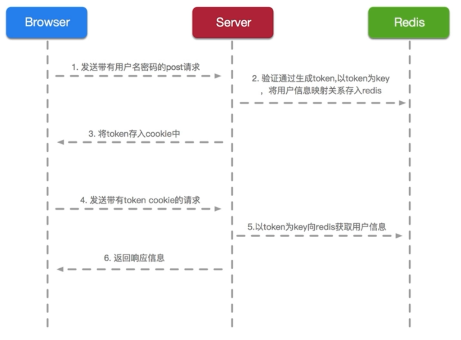
基于Session的身份认证方案



缺点：

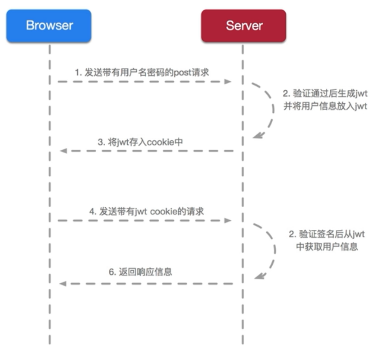
1. Session存于内存，不能跨实例共享，到另外一个实例需要重新登录
2. 高并发场景下，受内存大小限制；
3. Session依赖浏览器的Cookie信息，对于移动端客户很难支持；

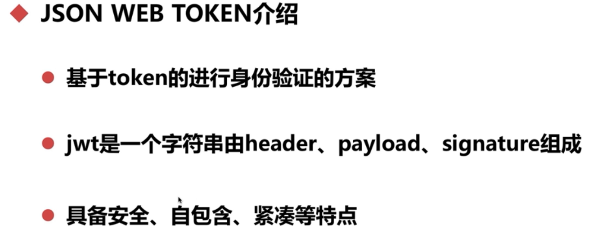
基于token身份认证方案



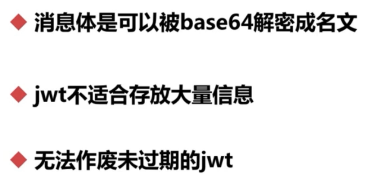
1. 依赖外部存储；
2. 代码相对Session更为复杂；

基于JWT的token身份认证方案





JWT注意事项



可借助Redis过期作废