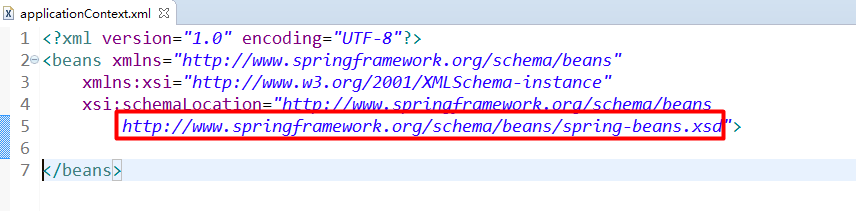
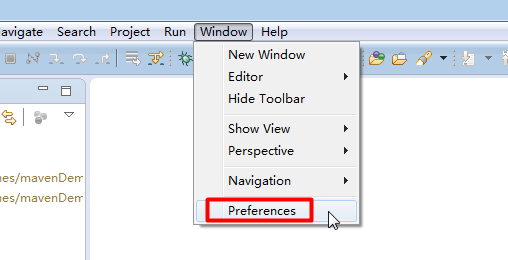
# 如何配置XML自动提示？

* 如果电脑可以联网，自动从网上下载 DTD，在编写xml文件时就有了提示。
* 如果电脑没有联网，在本地配置DTD文件位置，提供本地提示。
* 配置XML自动提示步骤：

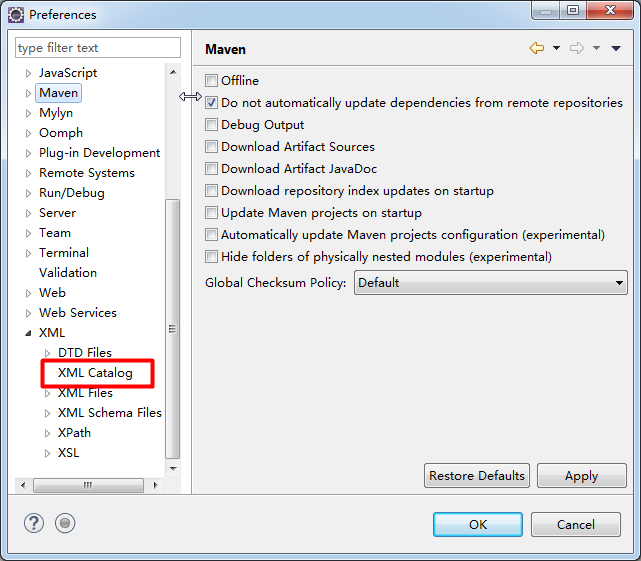
**第一步**：从applicationContext.xml中复制dtd网址：



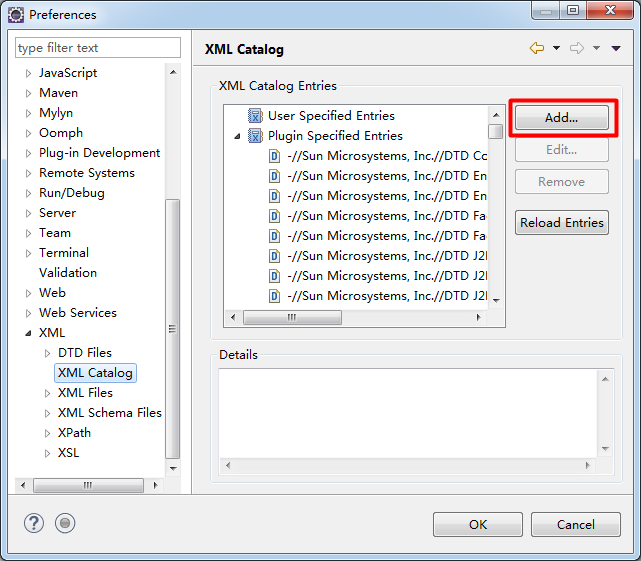
**第二步**：选择window-Preferences

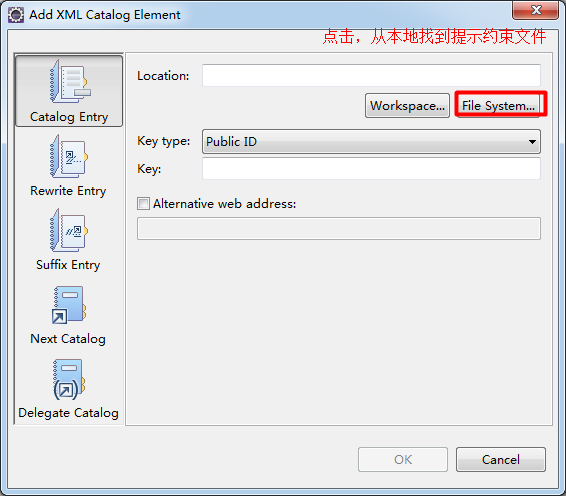


选择Xml Catalog

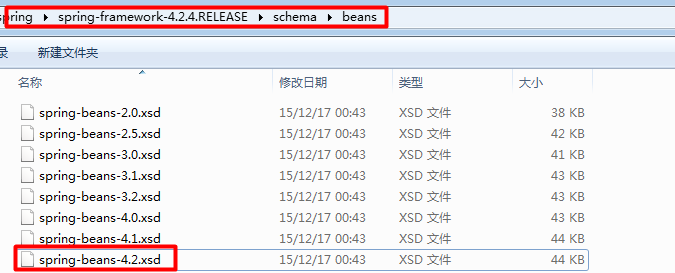


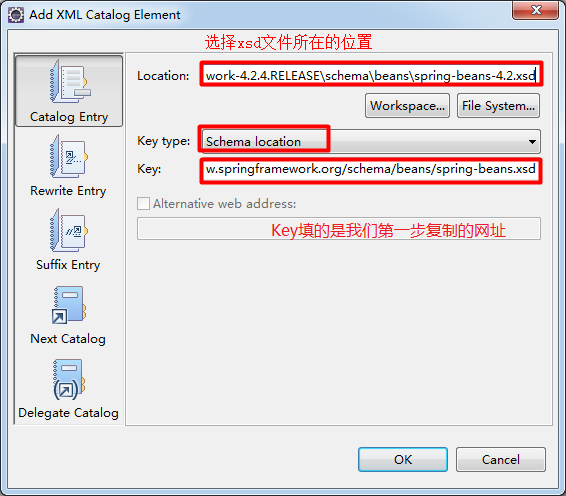
点击“Add”



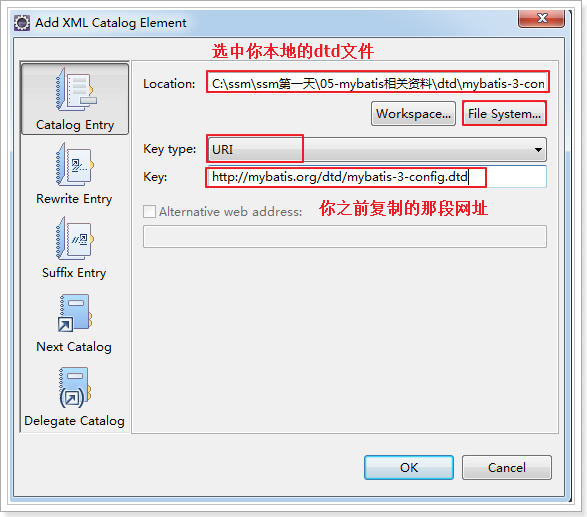


* 提示约束文件在官方包的schema目录中：

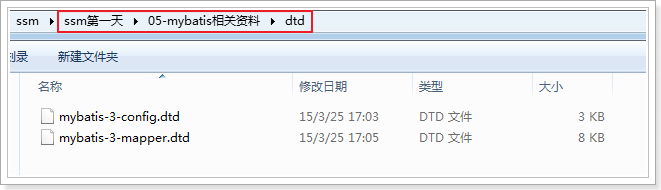




对于dtd文件，如mybatis配置文件中有所涉及，操作与上述有所不同，具体如下图



本地的dtd文件都在“05-mybatis相关资料中”：



到此，本地提示就已经配置好啦！

# 读取外部配置文件

<properties resource=/>

# 引入映射文件

<mapper resource=/>

# 各个注解的作用

* Configuration注解 声明一个类作为配置类，代替xml文件
* PropertySource注解 指定外部属性文件
* ConfigurationProperties注解 声明当前类为属性读取类
* @ComponentScan注解告知Spring哪些package下，会被Spring自动
* JSONFiled注解 的作用对象：1、Field 2、Setter Getter方法

FastJson在进行操作时，是根据getter setter方法进行的，并不是依据Field

* Data注解在类上；提供类所有属性的 getting 和 setting 方法，此外还提供了equals、canEqual、hashCode、toString 方法
* setter getter注解在类上，提供类的所有属性的getting 和 setting 方法，所有说@Data作用要比@Setter @Getter大
* @Entity注释指名这是一个实体Bean，@Table注释指定了Entity所要映射带数据库表，其中@Table.name()用来指定映射表的表名。

如果缺省@Table注释，系统默认采用类名作为映射表的表名。实体Bean的每个实例代表数据表中的一行数据，行中的一列对应实例中的一个属性。

@Column注释定义了将成员属性映射到关系表中的哪一列和该列的结构信息，属性如下：

1）name：映射的列名。如：映射tbl\_user表的name列，可以在name属性的上面或getName方法上面加入；

2）unique：是否唯一；

3）nullable：是否允许为空；

4）length：对于字符型列，length属性指定列的最大字符长度；

5）insertable：是否允许插入；

6）updatetable：是否允许更新；

7）columnDefinition：定义建表时创建此列的DDL；

8）secondaryTable：从表名。如果此列不建在主表上（默认是主表），该属性定义该列所在从表的名字。

@Id注释指定表的主键，它可以有多种生成方式：

1）TABLE：容器指定用底层的数据表确保唯一；

2）SEQUENCE：使用数据库德SEQUENCE列莱保证唯一（Oracle数据库通过序列来生成唯一ID）；

1. IDENTITY：使用数据库的IDENTITY列莱保证唯一；

4）AUTO：由容器挑选一个合适的方式来保证唯一；

5）NONE：容器不负责主键的生成，由程序来完成。

@GeneratedValue注释定义了标识字段生成方式。

@Temporal注释用来指定java.util.Date或java.util.Calender属性与数据库类型date、time或timestamp中的那一种类型进行映射。

@Temporal(value=TemporalType.TIME)

在Springboot中的启动类上加上

@PropertySource注解 可以读取相应的配置文件

@PropertySource({"classpath:application.yml","classpath:parameter.properties"})

如果需要使用一些固定数据 可以在配置文件中以key-value形式配置好，在相应启动类上加入@PropertySource注解，在需要用到该常量时，使用

@Value("${dataTypeCode\_so}")  
public String dataTypeCode;

@SpringCloudApplication 注解 包含了@SpringBootApplication @EnableDiscoveryClient @EnableCircurtBreaker 三个注解，而这三个注解正是每个微服务必须的注解 所以可以使用

@SpringCloudApplication注解来替代

# 通过切面来实现拦截器的功能

Point：就是当请求进入时，不会直接进入对应的controller，而是编写一个通过切面来实现拦截器的一个类，在类中定义具体需要拦截的路径，譬如对应包下的所有controller。

另外对于SpringBoot而言，可以通过 RequestContextHolder 来获取 HttpRequest 和 HttpResponse，具体方法如下示例

ServletRequestAttributes attributes = (ServletRequestAttributes) RequestContextHolder.*getRequestAttributes*();  
HttpServletRequest request = attributes.getRequest();  
String url = request.getRequestURI();  
String servletPath = request.getServletPath();

# JSONArray JSONObject的用法

JSONArray jsonArray = JSONArray.*parseArray*(jsonData);  
for (int i = 0; i < jsonArray.size(); i++) {  
 JSONObject jsonObject = jsonArray.getJSONObject(i);  
 String shopId = jsonObject.getString("shopId");

}

JSONArray中等于是有多个JSONObject 然后可以获取到JSONObject中的每个key的value值

【对于jsonObject.get()的方法，如果说获取不到当前key,则直接为null，并不会出现异常】

【对于JSONObject.parseObject(String jsonData)，需要保证的是，jsonData的数据是json的格式，即{}，如果说jsonData是个数组，[{},{}]，此时要么将jsonData先通过JSONArray获取每一个JSONObject,要么手动put，添加一个key,value，value即为jsonData】

# Kafka

* 下载Kafka发行版本的tar文件 解压到相应目录
* bin/zookeeper-server-start.sh config/zookeeper.properties【如果系统安装了Zookeeper，则直接启动该服务，如果没有安装，可以启动与Kafka一同打包的zookeeper，即为以上命令】
* zookeeper启动后，启动Kafka服务

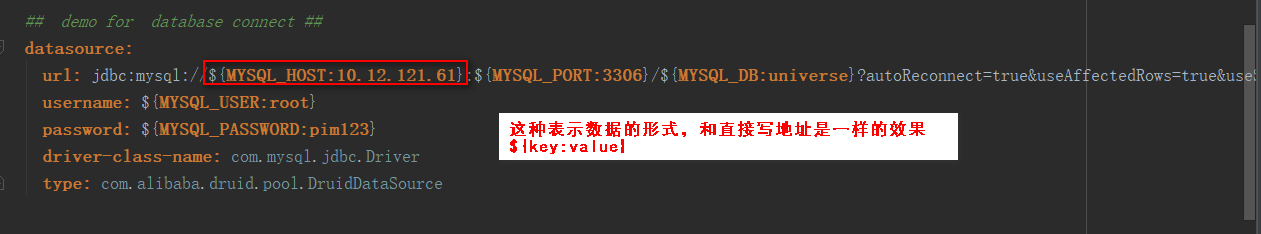
bin/kafka-server-start.sh config/server.properties

* 创建一个topic，bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test
* 输入命令 bin/kafka-topics.sh --list --zookeeper localhost:2181，即可看到刚刚创建的topic test
* 启动生产者，输入需要发送的信息，bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test
* 启动消费者，bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning 即可获取刚才发送的消息

# 代码书写json格式的数据

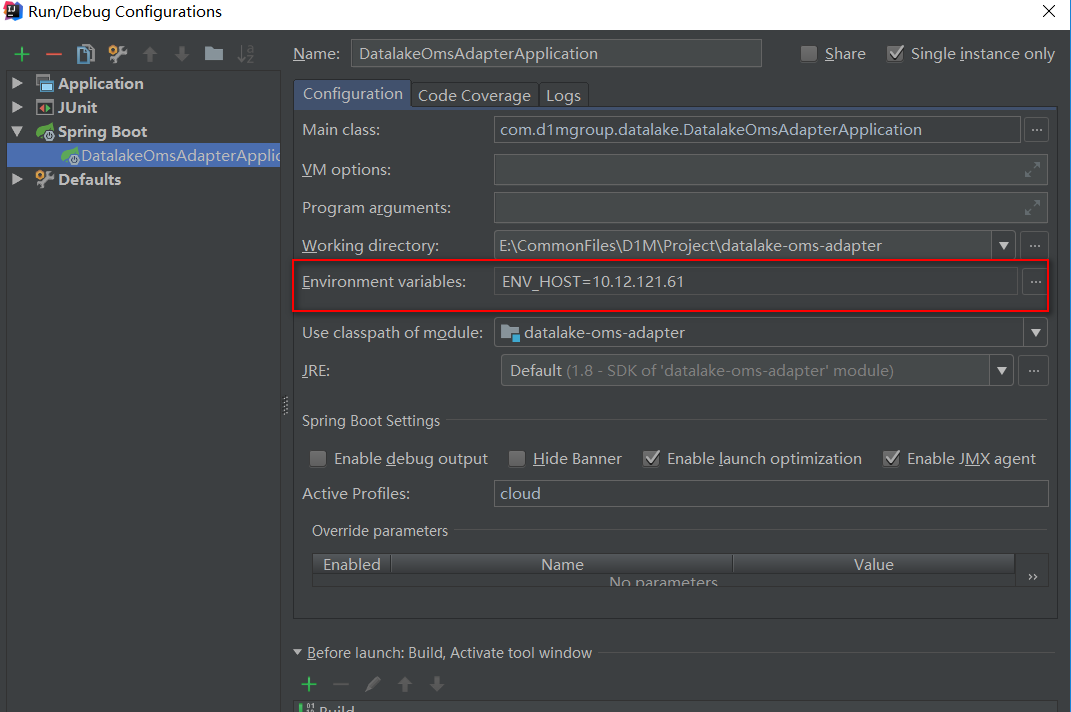
使用 \"字段值\" 来表示 即添加两对\"

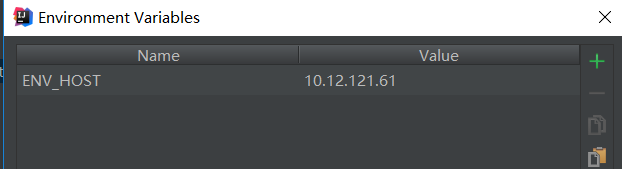
# 配置文件中参数的书写



使用${key:value}的形式来代替直接写value 作用是一样的，key的值没有影响，只是个说明

如果说value也是通过${value}来代替，那么此时，因为配置文件是无法读取相应具体数据的，所以需要进行配置，简单的方法就是直接在该配置文件中声明该参数的值，但通常都是需要直接配置environment variables 具体如下图





添加相应数值即可

StringUtils.isBlank 和StringUtils.isEmpty的区别是 前者不仅判断了字符串是否为空，长度是否为0 同时也判断了如果是空字符串，但长度不为零也是false 但对于后者，如果是空字符串 但长度不为0 结果却是true