# Flink的大数据实时城市交通监控平台**



## 1.1 项目整体介绍

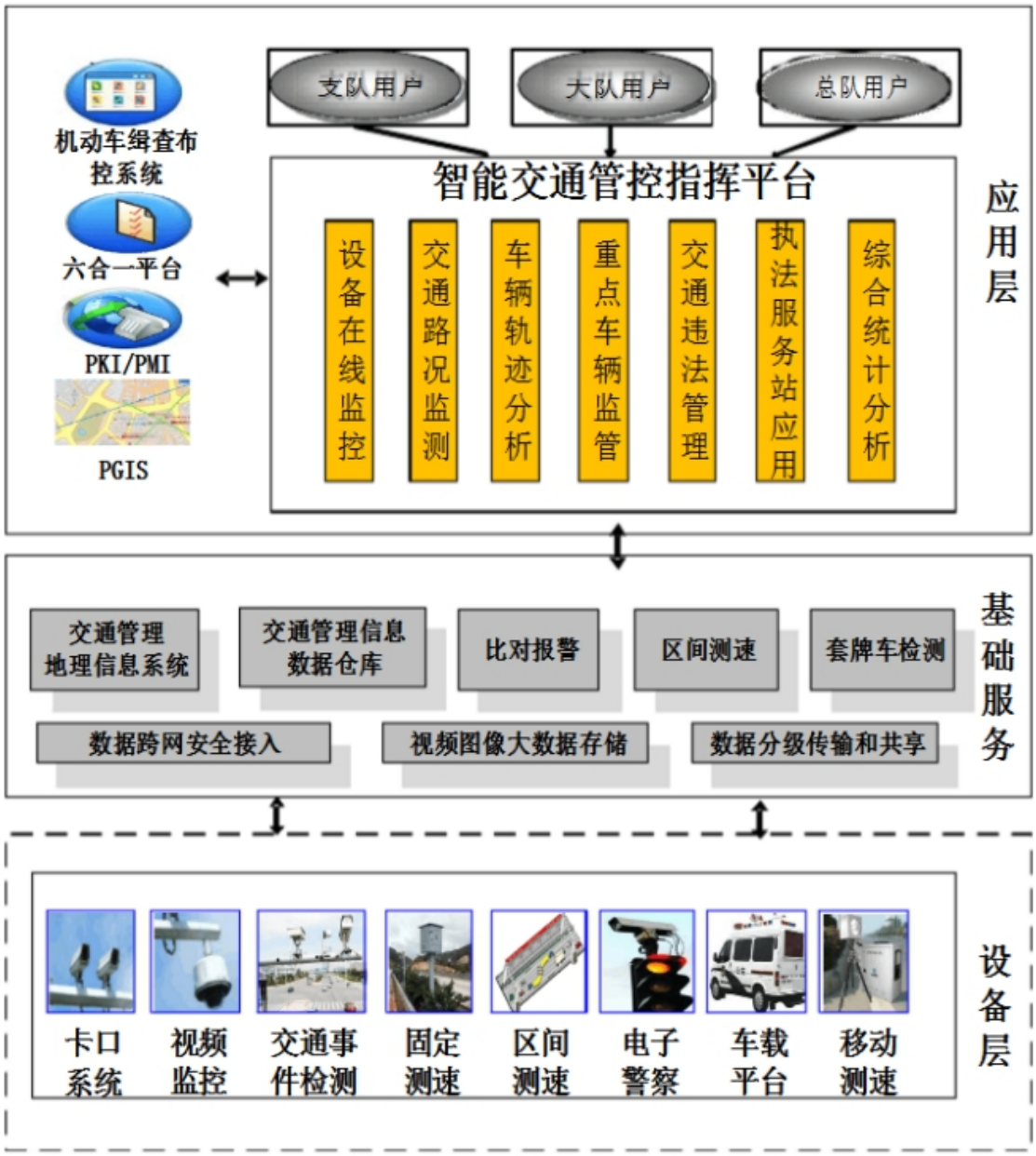
近几年来，随着国内经济的快速发展，高速公路建设步伐不断加快，全国机动车辆、驾驶员数量迅速增长，交通管理工作日益繁重，压力与日俱增。为了提高公安交通管理工作的科学化、现代化水平，缓解警力不足，加强和保障道路交通的安全、有序和畅通，减少道路交通违法和事故的发生，全国各地建设和使用了大量的“电子警察”、“高清卡口”、“固定式测速”、“区间测速”、“便携式测速”、“视频监控”、“预警系统”、“能见度天气监测系统”、“LED信息发布系统”等交通监控系统设备。尽管修建了大量的交通设施，增加了诸多前端监控设备，但交通拥挤阻塞、交通安全状况仍然十分严重。由于道路上交通监测设备种类和生产厂家繁多，目前还没有一个统一的数据采集和交换标准，无法对所有的设备、数据进行统一、高效的管理和应用，造成各种设备和管理软件混用的局面，给使用单位带来了很多不便，使得国家大量的基础建设投资未达到预期的效果。各交警支队的设备大都采用本地的数据库管理，交警总队无法看到各支队的监测设备及监测信息，严重影响对全省交通监测的宏观管理；目前网络状况为设备专网、互联网、公安网并存的复杂情况，需要充分考虑公安网的安全性，同时要保证数据的集中式管理；监控数据需要与“六合一”平台、全国机动车稽查布控系统等的数据对接，迫切需要一个全盘考虑面向交警交通行业的智慧交通管控指挥平台系统。

智慧交通管控指挥平台建成后，达到了以下效果目标：

1. 交通监视和疏导：通过系统将监视区域内的现场图像传回指挥中心，使管理人员直接掌握车辆排队、堵塞、信号灯等交通状况，及时调整信号配时或通过其他手段来疏导交通，改变交通流的分布，以达到缓解交通堵塞的目的。
2. 交通警卫：通过突发事件的跟踪，提高处置突发事件的能力。
3. 建立公路事故、事件预警系统的指标体系及多类分析预警模型，实现对高速公路通行环境、交通运输对象、交通运输行为的综合分析和预警，建立真正意义上的分析及预警体系。
4. 及时准确地掌握所监视路口、路段周围的车辆、行人的流量、交通治安情况等，为指挥人员提供迅速直观的信息从而对交通事故和交通堵塞做出准确判断并及时响应。
5. 收集、处理各类公路网动静态交通安全信息，分析研判交通安全态势和事故隐患，并进行可视化展示和预警提示。
6. 提供接口与其他平台信息共享和关联应用，基于各类动静态信息的大数据分析处理，实现交通违法信息的互联互通、源头监管等功能。

### 1.1.1 项目架构

本项目是与公安交通管理综合应用平台、机动车缉查布控系统等对接的，并且基于交通部门现有的数据平台上，进行的数据实时分析项目。



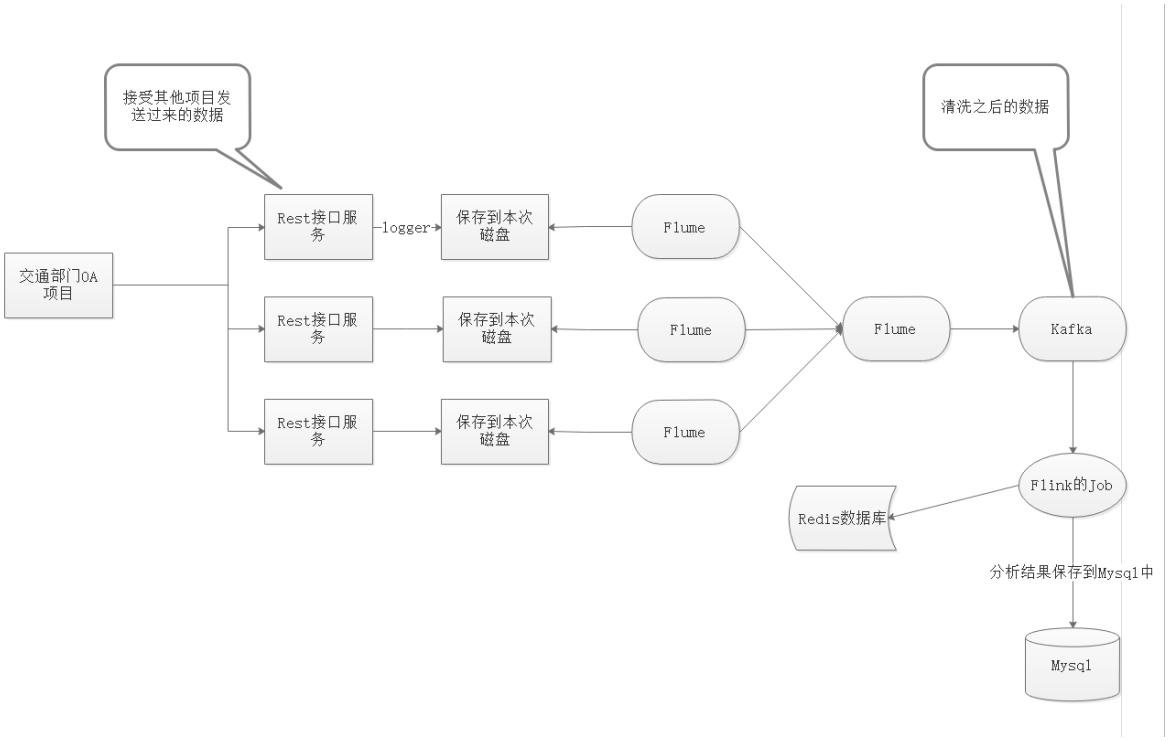

1) **相关概念**

- 卡口：道路上用于监控的某个点，可能是十字路口，也可能是高速出口等。

- 通道：每个卡口上有多个摄像头，每个摄像头有拍摄的方向。这些摄像头也叫通道。
- “违法王“车辆： 该车辆违法未处理超过50次以上的车。

- 摄像头拍照识别：
  - 一次拍照识别：经过卡口摄像头进行的识别，识别对象的车辆号牌信息、车辆号牌颜色信息等，基于车辆号牌和车辆颜色信息，能够实现基本的违法行为辨识、车辆黑白名单比对报警等功能。
  - 二次拍照识别：可以通过时间差和距离自动计算出车辆的速度。

## 1.1.2 项目数据流




实时处理流程如下：

http请求 -->数据采集接口-->数据目录--> flume监控目录[监控的目录下的文件是按照日期分的] -->Kafka [也会放在HDFS中，就是上面做的] -->Flink分析数据 --> Mysql[给运营中心使用]

## 1.1.3 项目主要模块

本项目的主要模块有三个方向：

1) 实时卡口监控分析：

依托卡口云管控平台达到降事故、保畅通、服务决策、引领实战的目的，最

大限度指导交通管理工作。丰富了办案手段，提高了办案效率、节省警力资源，最终达

到牵引警务模式的变革。

利用摄像头拍摄的车辆数据来分析每个卡口车辆超速监控、卡口拥堵情况监控、每个区域卡口车流量TopN统计。

2) 实时智能报警：

该模块主要针对路口一些无法直接用单一摄像头拍摄违章的车辆，通过海量数据分析并实时智能报警。

在一时间段内同时在 2 个区域出现的车辆记录则为可能为套牌车。这个模块包括：实时套牌分析，实时危险驾驶车辆分析。

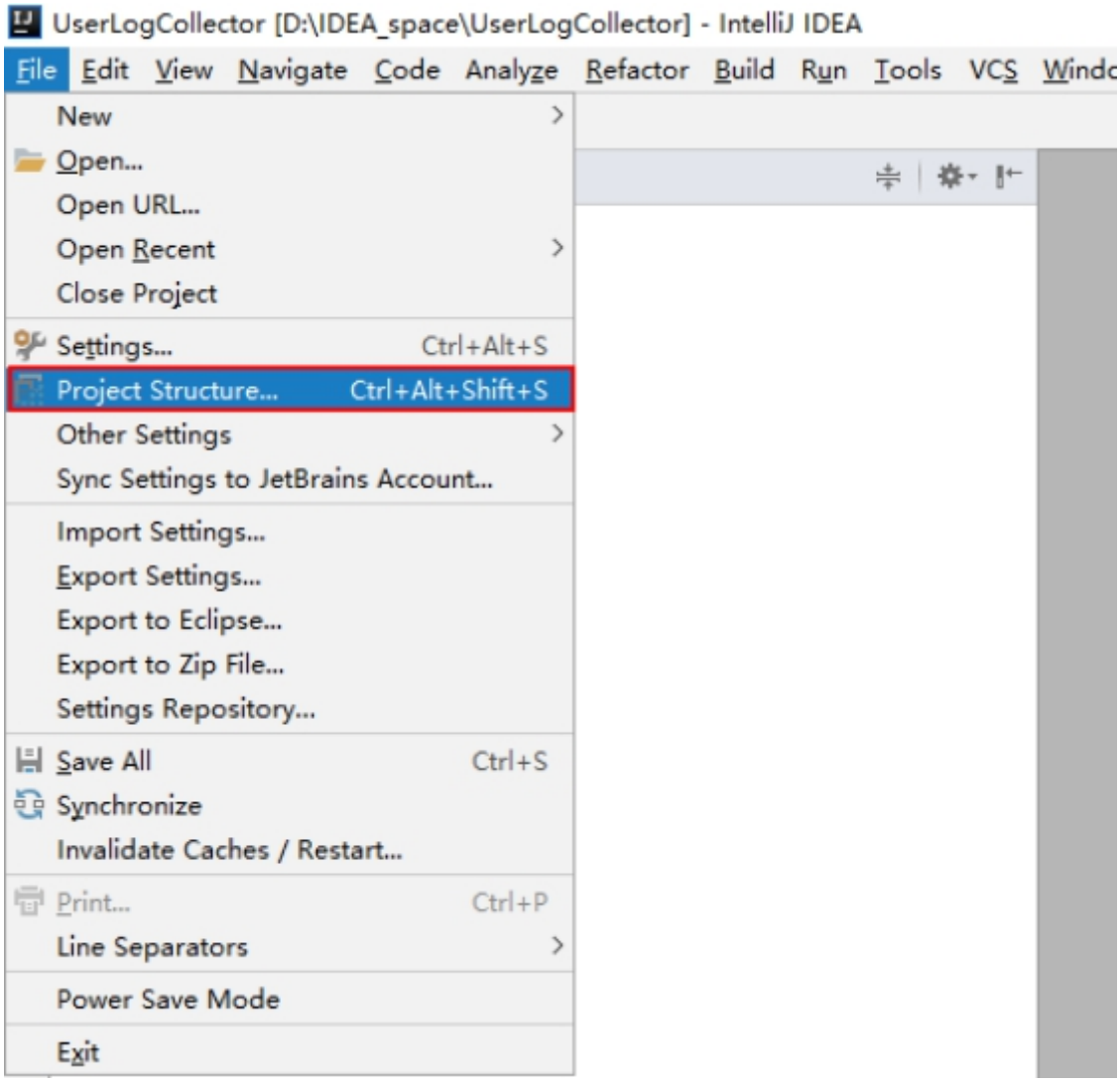
3) 智能车辆布控：

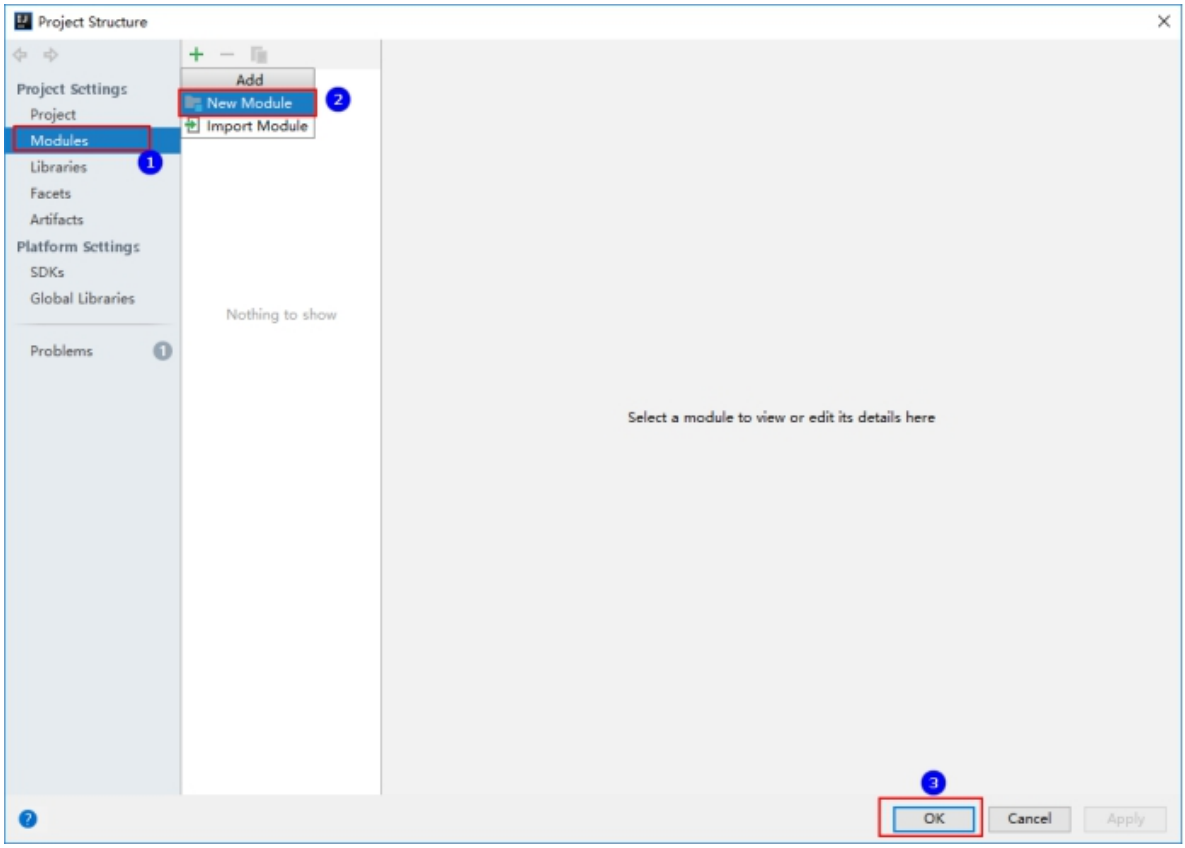
该模块主要从整体上实时监控整个城市的车辆情况，并且对整个城市中出现“违法王”的车辆进行布控。

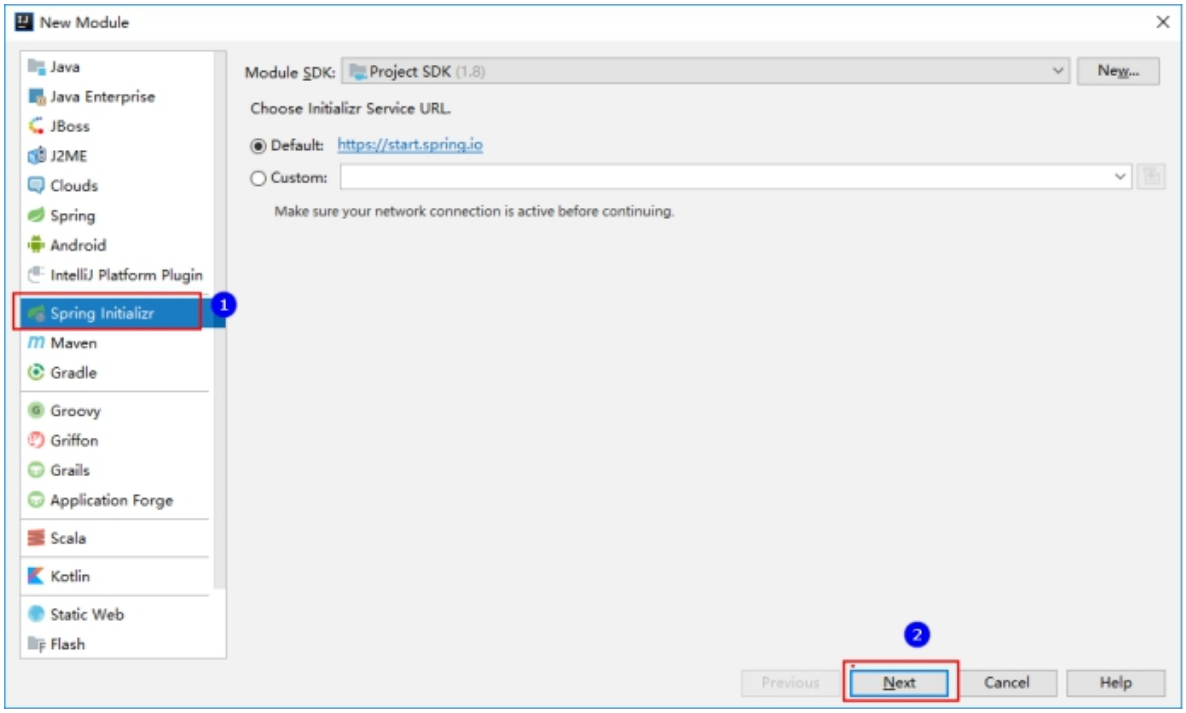
主要功能包括：单一车辆轨迹跟踪布控，“违法王”轨迹跟踪布控，实时车辆分布分析，实时外地车分布分析。

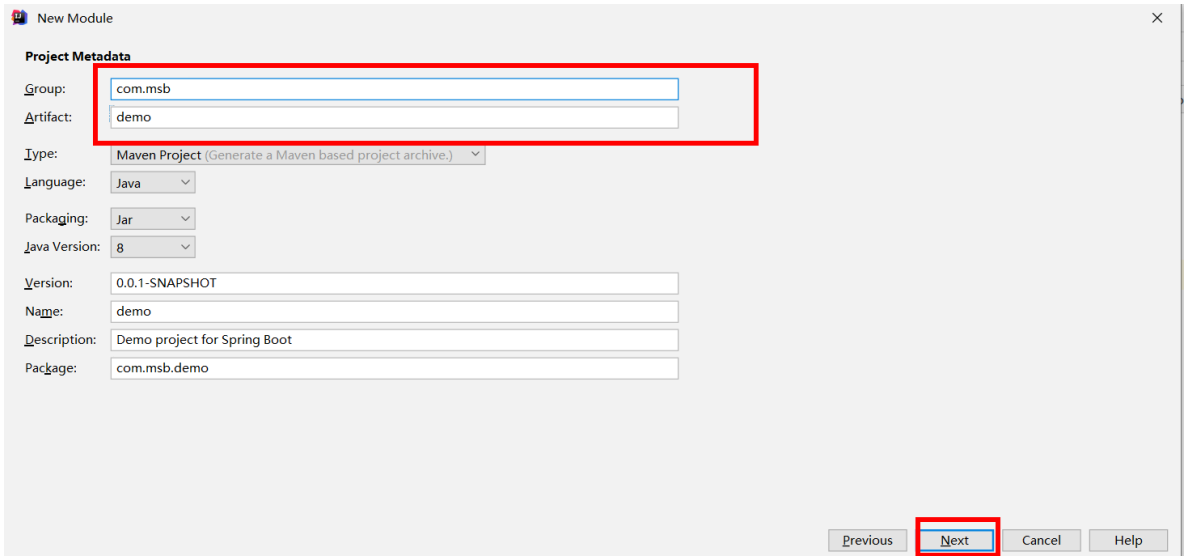
# 1.2数据采集

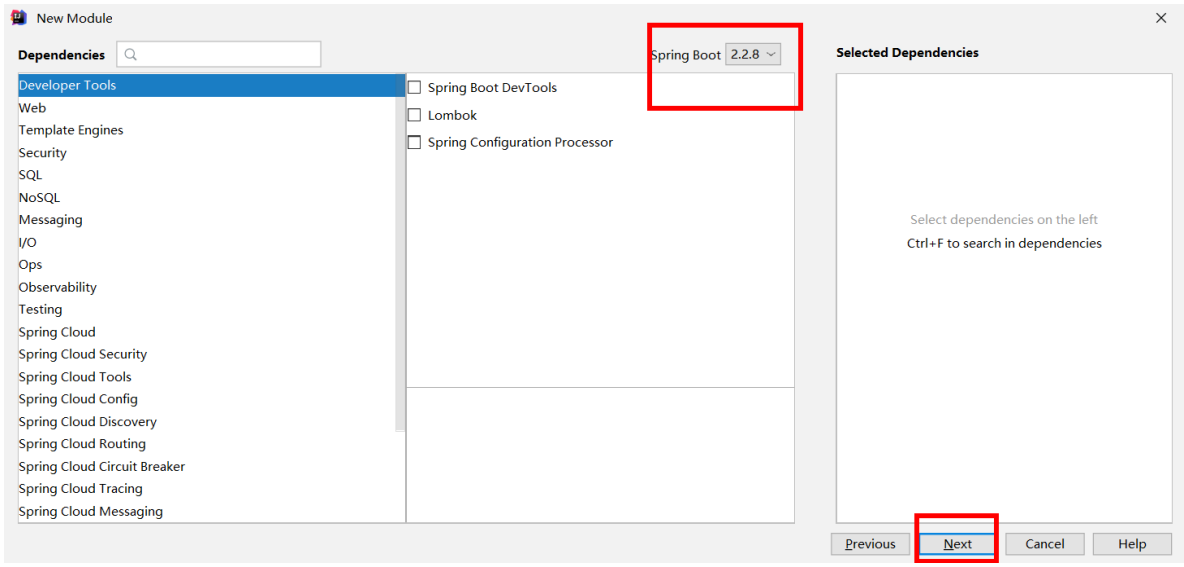
## 1.2.1创建Maven项目，并添加Spring-boot的模块

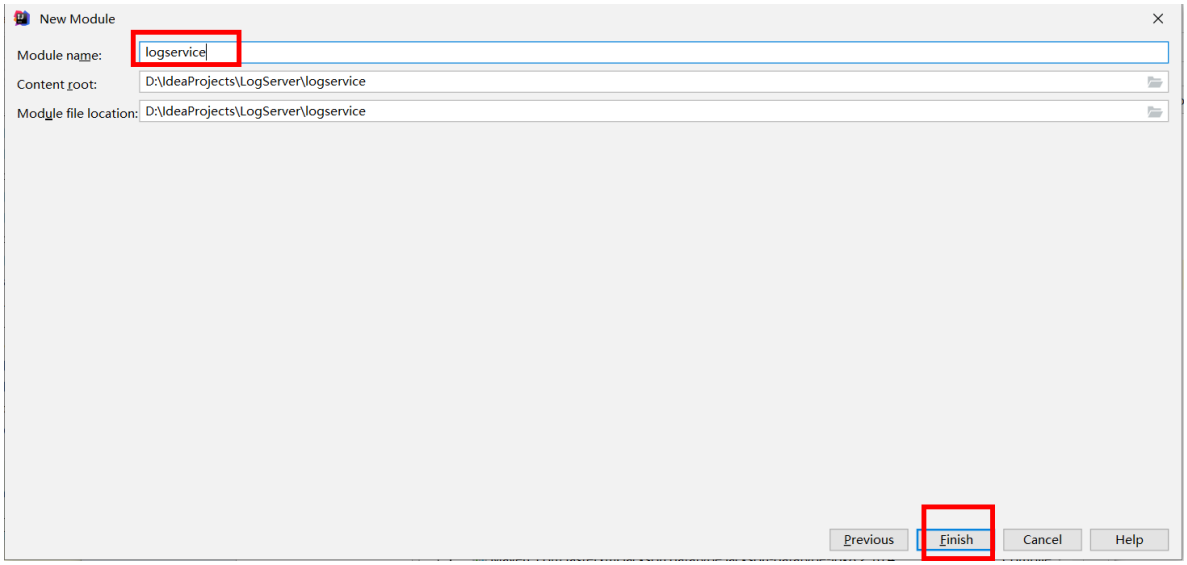

UserLogCollector [D:\IDEA_space\UserLogCollector] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Windo
New
Open...
Open URL...
Open Recent
Close Project
Settings... Ctrl+Alt+S
Project Structure... Ctrl+Alt+Shift+S
Other Settings
Sync Settings to JetBrains Account...
Import Settings...
Export Settings...
Export to Eclipse...
Export to Zip File...
Settings Repository...
Save All Ctrl+S
Synchronize
Invalidate Caches / Restart...
Print... Ctrl+P
Line Separators
Power Save Mode
Exit


Project Structure
Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems
Add
New Module
Import Module
Nothing to show
Select a module to view or edit its details here
OK
Cancel
Apply


New Module
Java
Java Enterprise
JBoss
J2ME
Clouds
Spring
Android
IntelliJ Platform Plugin
Spring Initializr
Maven
Gradle
Groovy
Griffon
Grails
Application Forge
Scala
Kotlin
Static Web
Flash
Module SDK:
Project SDK (1.8)
New...
Choose Initializr Service URL.
Default:
https://start.spring.io
Custom:
Make sure your network connection is active before continuing.
1
2
Previous
Next
Cancel
Help


New Module
Project Metadata
Group:
com.msb
Artifact:
demo
Type:
Maven Project (Generate a Maven based project archive.)
Language:
Java
Packaging:
Jar
Java Version:
8
Version:
0.0.1-SNAPSHOT
Name:
demo
Description:
Demo project for Spring Boot
Package:
com.msb.demo
Previous
Next
Cancel
Help


New Module
Dependencies
Spring Boot 2.2.8
Selected Dependencies
Developer Tools
Web
Template Engines
Security
SQL
NoSQL
Messaging
I/O
Ops
Observability
Testing
Spring Cloud
Spring Cloud Security
Spring Cloud Tools
Spring Cloud Config
Spring Cloud Discovery
Spring Cloud Routing
Spring Cloud Circuit Breaker
Spring Cloud Tracing
Spring Cloud Messaging
Spring Boot DevTools
Lombok
Spring Configuration Processor
Select dependencies on the left
Ctrl+F to search in dependencies
Previous
Next
Cancel
Help




在项目中找到Application类，这个类就是SpringBoot的启动类，通常是*Application的命名。入口类里有一个main方法，其实就是一个标准的Java应用的入口方法。在main方法中使用SpringApplication.run启动Spring Boot项目。Sping Boot内嵌了servlet容器，默认tomcat。

@SpringBootApplication是Spring Boot的核心注解，是一个组合注解。

在当前类中直接右键run main主方法即是启动SpringBoot。

在resources配置目录中有一个名称为application.properties或者名称为application.yaml的文件，这个文件是全局配置文件，可以在这个文件中对一些默认的配置项进行配置修改。配置项可参考：

https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/#appendix

项目中一般为不同的环境配置不同的配置文件，这些配置文件也是存在于resources目录下，使用application-{profile}.properties指定不同环境配置文件。

这里我们在resources目录下增加了开发环境(dev)和生产环境(prod)的配置文件，配置文件名称分别为：application-dev.properties和application-prod.properties，并通过在application.properties中设置spring.profiles.active=dev来指定当前环境为开发环境。Spring Boot为我们做了很多自动化的配置，搭建快速方便。

可以在application.properties配置debug=true，当启动SpringBoot后，查看当前项目中已启用和未启用的自动配置。

## 1.2.2添加依赖支持web和配置

在pom.xml文件中加入spring-boot-starter-web的依赖：

```
<dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-configuration-processor</artifactId>
            <optional>true</optional>
        </dependency>
        <!--<dependency>-->
            <!--<groupId>com.google.code.gson</groupId>-->
            <!--<artifactId>gson</artifactId>-->
        <!--</dependency>-->
```

```
        <!-- 解析json -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.36</version>
    </dependency>
```

### 1.2.3开发数据采集的服务器