

CDN大数据实时日志分析项目

一、项目背景

项目开发于2017年，主要为联通包月流量包业务（比如：腾讯定向流量包，爱奇艺定向量包等）提供CDN数据的查询服务，用于支持联通的日常业务需求，包括实时数据查询和报表数据生成。

项目环境背景

联通定向流量包涉及的服务提供商比较多，对应的订购用户和使用量也会比较大，预估每日数据量在8000万条左右，同时由于视频播放app在使用上存在高峰时段所以瞬时访问量会非常大。

项目运作的可行性

基于业务需求和实际的项目环境，需要提供基于大数据级别的业务能力要提供实时数据计算查询又要提供明细数据的存储，同时还要能保证数据的安全性和稳定性。

从项目架构上，在主要节点服务器部署日志采集服务，该服务可以根据业务需求进行日志的处理和合并，再将日志数据发布到消息队列集群中从而保证每条日志处理的正确性和可扩展性，通过部署实时计算集群和可以持久保存数据明细的大数据仓库提供业务需要的数据支撑，从而保证了项目的可行性和可扩展性。

项目优势分析

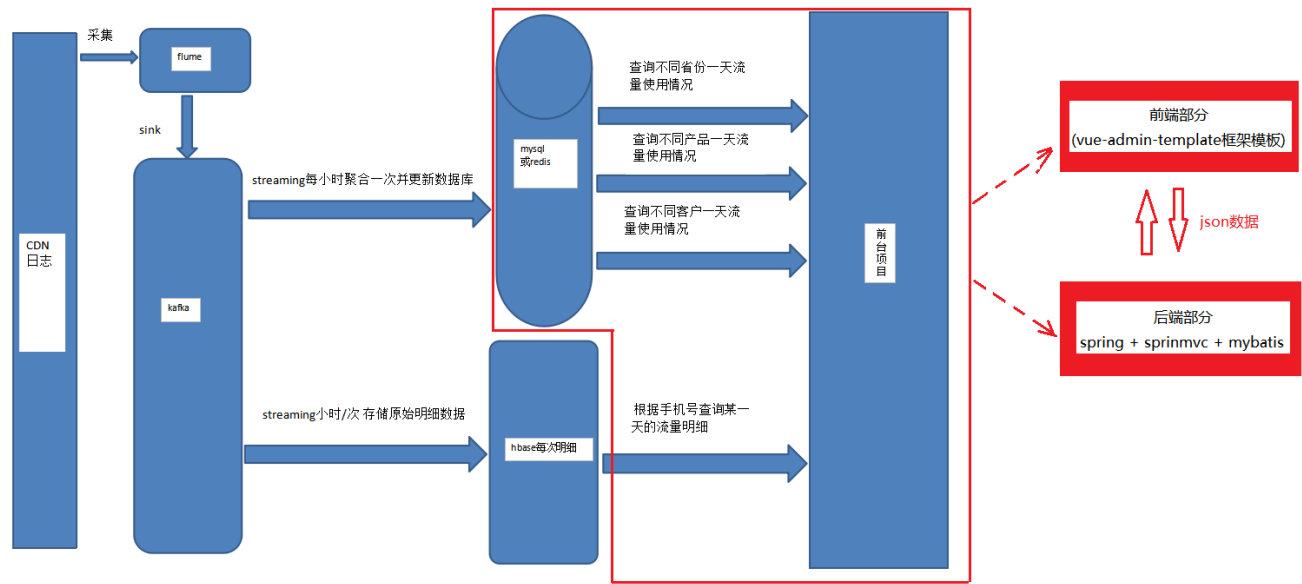
针对实时业务需求提供实时计算集群服务，不保留数据明细只存储各种业务需求对应的计算结果，从而保证对处理速度和查询性能的支持。针对明细查询提供大数据仓库服务，针对目前数据量搭建与之匹配的服务集群，同时可以根据业务未来的用户增长趋势对系统进行弹性扩展保证系统的可用性和稳定性。

二、项目需求说明

参见 [《大数据参考手册》](大数据参考手册.docx)

参见 [《前台项目需求-2.0》](前台项目需求-2.0.doc)

三、系统架构设计



四、开发环境搭建

jdk、tomcat、redis、maven、mysql、powerdesigner、IDEA(easycode插件)、node.js、vscode (Vue插件、Vetur插件、ESLint插件)、git

五、PDM设计

(略)

六、SQL

创建mysql数据库,取名traffic,选择utf-8字符集。并运行批次任务文件traffic.sql,生成表和记录。由于数据量较大,耐心等待几分钟。
![sql源文件](traffic.sql)
共8个表,但我们只使用其中3个,t_pname_traffics、t_province_traffics、t_spname_traffics

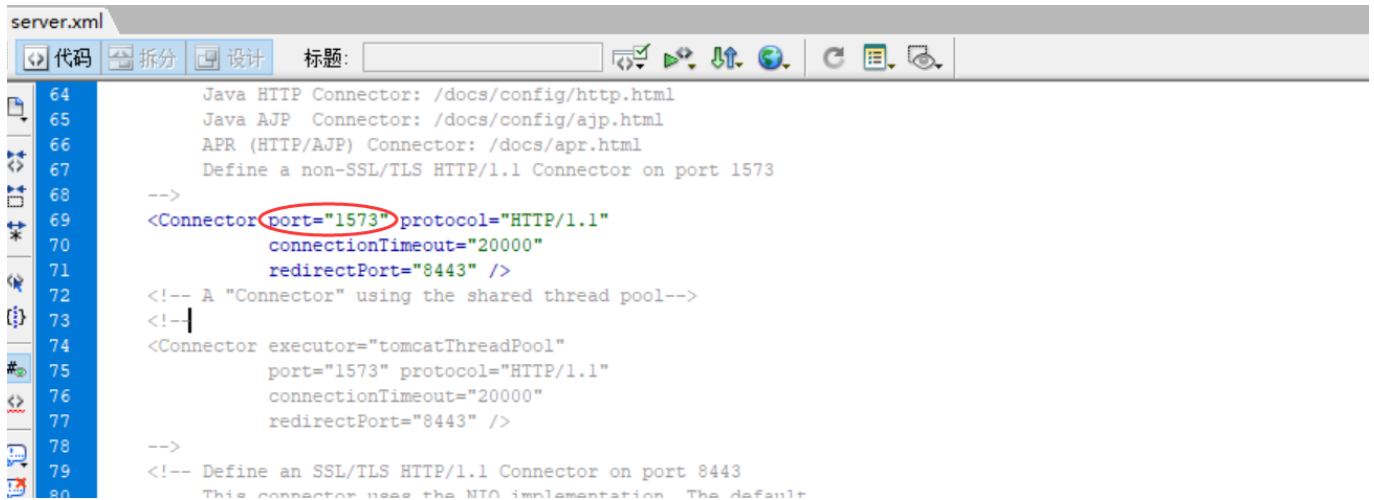
七、后端SSM

安装 JDK

到官网上下载安装文件,JDK 官方网站地址:
<https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>
下载 jdk-8u251-windows-x64.exe 后双击运行,全部默认安装。安装的本地目录:C:\Program Files\java\jdk1.8

安装 Tomcat

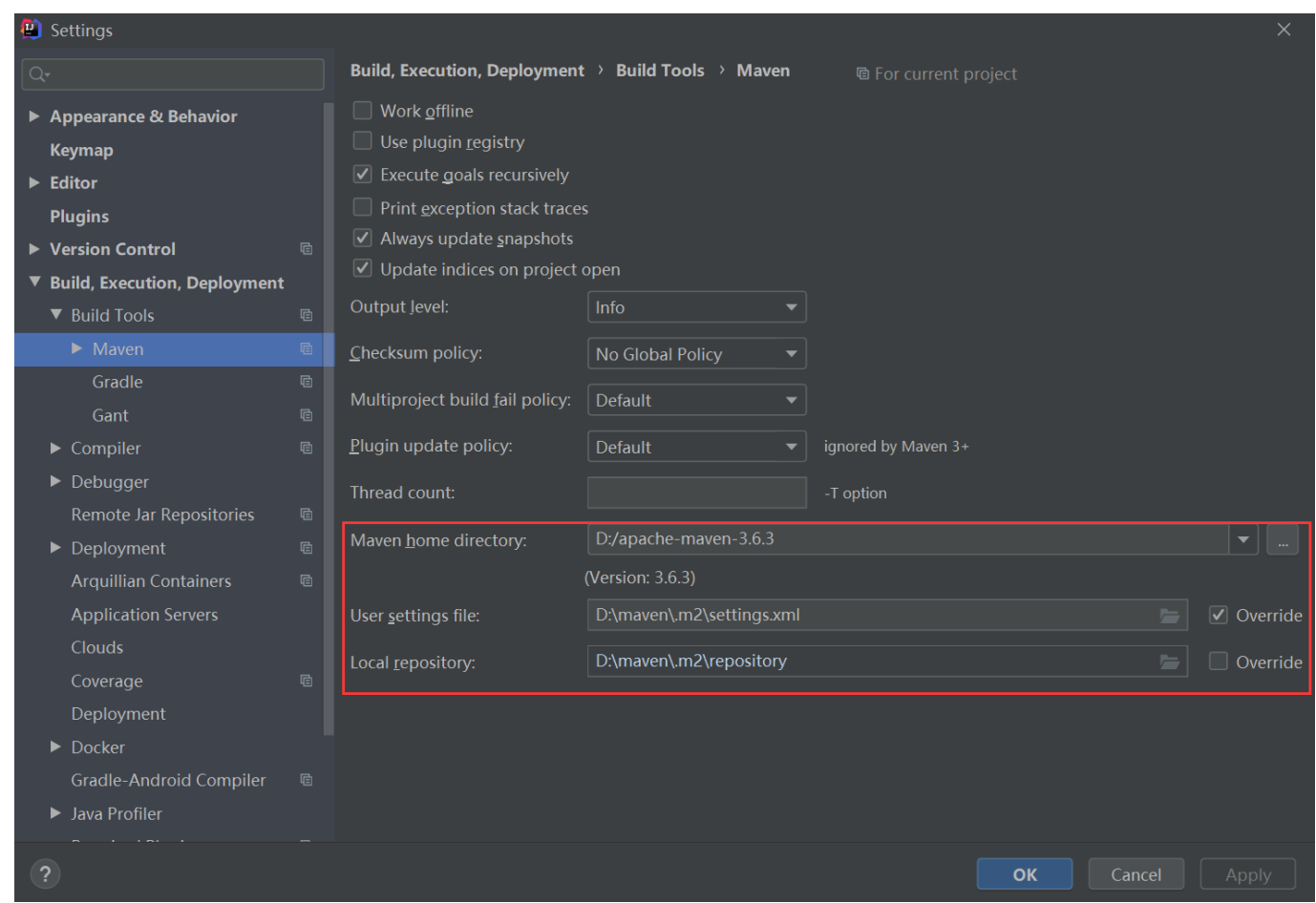
到官网下载安装文件，Tomcat 官方网站地址：<https://tomcat.apache.org/download-80.cgi>
下载 apache-tomcat-8.5.53.exe 后双击运行，全部默认安装。安装的本地目录：C:\Program Files\Apache Software Foundation\Tomcat 8.5
然后打开 Tomcat 的配置文件 C:\Program Files\Apache Software Foundation\Tomcat 8.5\conf\server.xml，把默认端口修改为 1573，防止与其它端口冲突。



```
server.xml
代码  拆分  设计  标题:
64      Java HTTP Connector: /docs/config/http.html
65      Java AJP  Connector: /docs/config/ajp.html
66      APR (HTTP/AJP) Connector: /docs/apr.html
67      Define a non-SSL/TLS HTTP/1.1 Connector on port 1573
68      -->
69      <Connector port="1573" protocol="HTTP/1.1"
70                  connectionTimeout="20000"
71                  redirectPort="8443" />
72      <!-- A "Connector" using the shared thread pool-->
73      <!--|
74      <Connector executor="tomcatThreadPool"
75                  port="1573" protocol="HTTP/1.1"
76                  connectionTimeout="20000"
77                  redirectPort="8443" />
78      -->
79      <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
80      This connector uses the NIO implementation. The default
```

安装 Maven

到官网下载最新版的安装文件，Maven 官方网站地址：<http://maven.apache.org/download.cgi>
下载 apache-maven-3.6.3-bin.tar.gz 后解压缩到本地目录：D:\apache-maven-3.6.3
然后在 IDEA 中配置 maven 路径，File->settings->maven 如图所示：



最后做如下设置：

- 1. 设置要使用的资源仓库（本地仓库）的位置，如 D:\maven.m2\repository；
- 2. 设置要使用的 maven 的 settings.xml 文件位置（默认去 c 盘找，尽可能不要使用默认的），如 D:\maven.m2\settings.xml，可以将 maven 的安装位置下 conf 中的 settings.xml 复制过来并进行修改

```

<!-->
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <!-- localRepository
   | The path to the local repository maven will use to store artifacts.
   |
   | Default: ${user.home}/.m2/repository
  <localRepository>/path/to/local/repo</localRepository>
  -->
  <!--2019.10.18, 设置本地仓库位置-->
  <localRepository>D:/maven/.m2/repository</localRepository>
  <!-- interactiveMode
   | This will determine whether maven prompts you when it needs input. If set to false,
   | maven will use a sensible default value, perhaps based on some other setting, for
   | the parameter in question.
   |
   | Default: true
  <interactiveMode>true</interactiveMode>
  -->

  <!-- offline
   | Determines whether maven should attempt to connect to the network when executing a build.
   | This will have an effect on artifact downloads, artifact deployment, and others.
   |
   | Default: false
  <offline>>false</offline>
  -->

  | however, this repository may have problems with heavy traffic at times, so people have mirrored
  | it to several places.
  |
  | That repository definition will have a unique id, so we can create a mirror reference for that
  | repository, to be used as an alternate download site. The mirror site will be the preferred
  | server for that repository.
  |-->
</mirrors>
  <!-- mirror
   | Specifies a repository mirror site to use instead of a given repository. The repository that
   | this mirror serves has an ID that matches the mirrorOf element of this mirror. IDs are used
   | for inheritance and direct lookup purposes, and must be unique across the set of mirrors.
   |
  <mirror>
    <id>mirrorId</id>
    <mirrorOf>repositoryId</mirrorOf>
    <name>Human Readable Name for this Mirror.</name>
    <url>http://my.repository.com/repo/path</url>
  </mirror>
  -->
  <!--2019.10.18, 设置使用的远程公共仓库, 此处为国内阿里云提供-->
  <mirror>
    <id>nexus-aliyun</id>
    <mirrorOf>central</mirrorOf>
    <name>Nexus aliyun</name>
    <url>http://maven.aliyun.com/nexus/content/groups/public</url>
  </mirror>
</mirrors>

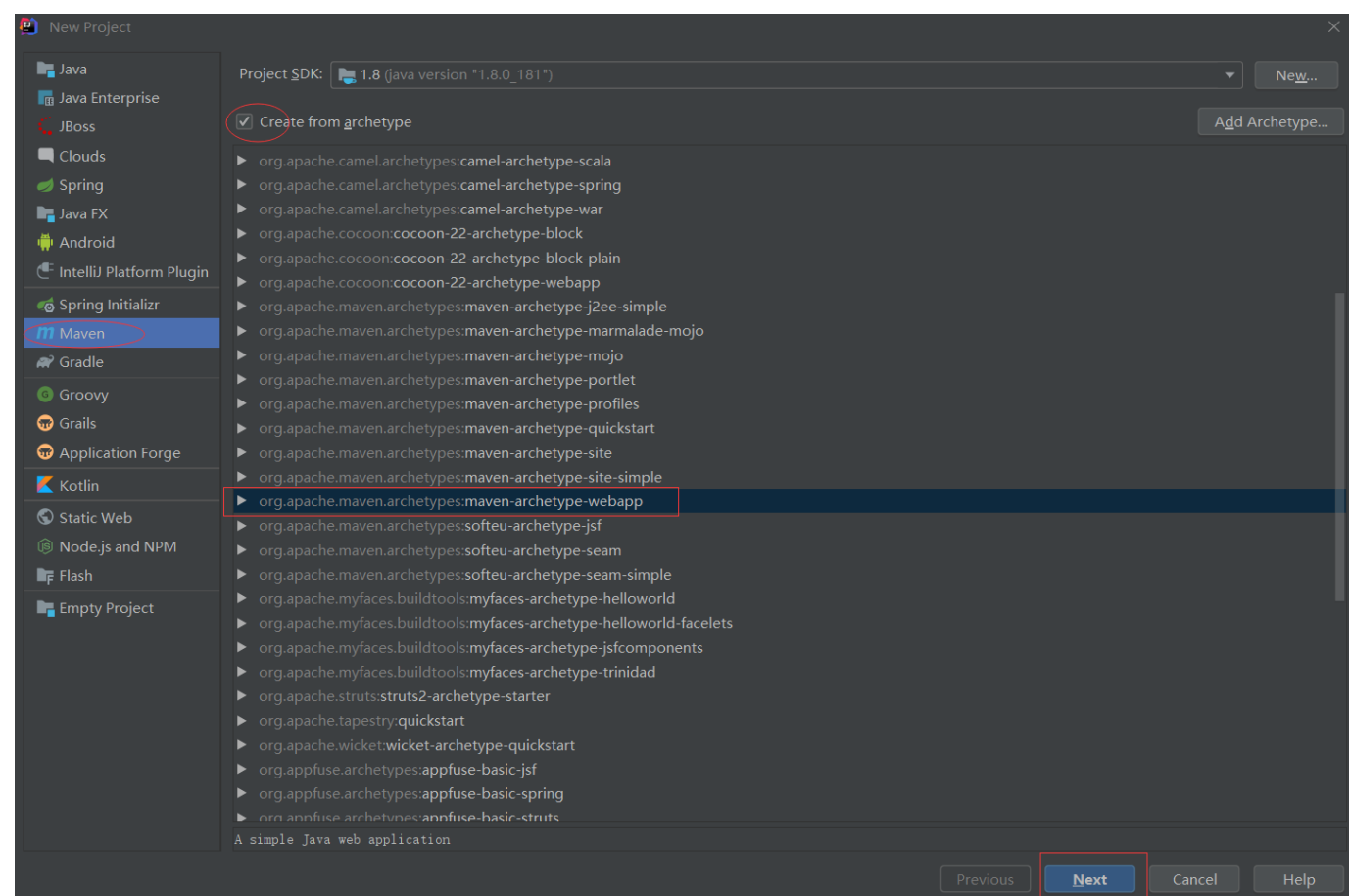
  <!-- profiles
   | This is a list of profiles which can be activated in a variety of ways, and which can modify
   | the build process. Profiles provided in the settings.xml are intended to provide local machine-
   | specific paths and repository locations which allow the build to work in the local environment.
   |
   | For example, if you have an integration testing plugin - like cactus - that needs to know where

```

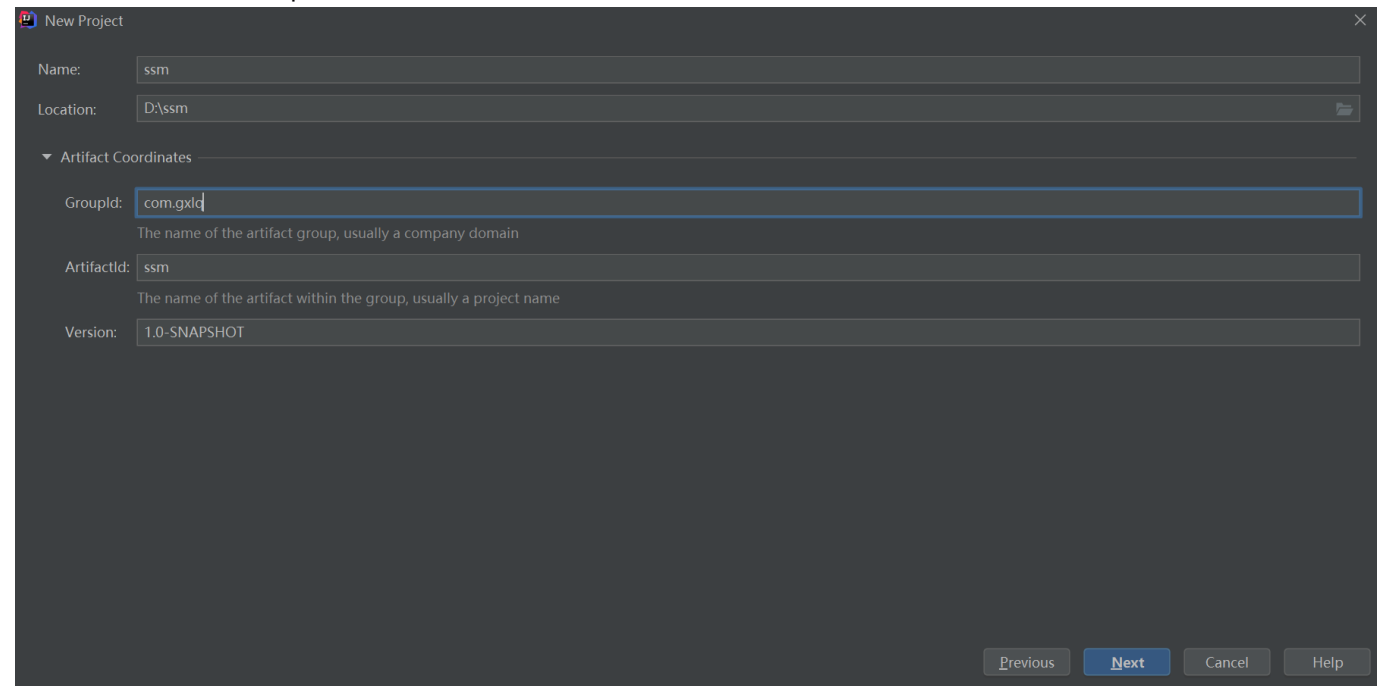
到此，IDEA 配置 maven 完毕。

基于 maven 创建 ssm 项目

在 IDEA 中创建 SSM 项目。选择 File -> New -> Project... , 在左侧导航中选择 Maven，然后在右侧选中某个“原型模板”，此处选择 maven-archetype-webapp，如下图所示：



点击 Next，输入 GroupId 和 artifactId 等项目有关的信息。




```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.gxlq</groupId>
  <artifactId>cdn</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>cdn Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
    <!-- 设置一个spring框架的组件版本号(全局)-->
    <xxx>5.2.9.RELEASE</xxx>
  </properties>

  <dependencies>

    <!-- jackson-databind -->
    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>2.12.3</version>
    </dependency>

    <!-- javax.servlet-api -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>4.0.1</version>
    </dependency>

    <!-- jstl -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>jstl</artifactId>
      <version>1.2</version>
    </dependency>

    <!-- javax.servlet.jsp-api -->
    <dependency>
      <groupId>javax.servlet.jsp</groupId>
      <artifactId>javax.servlet.jsp-api</artifactId>
      <version>2.3.3</version>
```



```
</dependency>

<!-- mybatis-redis -->
<dependency>
  <groupId>org.mybatis.caches</groupId>
  <artifactId>mybatis-redis</artifactId>
  <version>1.0.0-beta2</version>
</dependency>

<!--mybatis-spring -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>2.0.6</version>
</dependency>

<!-- c3p0 -->
<dependency>
  <groupId>com.mchange</groupId>
  <artifactId>c3p0</artifactId>
  <version>0.9.5.5</version>
</dependency>

<!--oracle11g驱动 -->
<dependency>
  <groupId>com.oracle.database.jdbc</groupId>
  <artifactId>ojdbc6</artifactId>
  <version>11.2.0.4</version>
</dependency>

<!-- mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.38</version>
</dependency>

<!-- mybatis3.4 -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.4.6</version>
</dependency>

<!-- log4j -->
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.17</version>
</dependency>

<!-- spring-aspects -->
<dependency>
  <groupId>org.springframework</groupId>
```

```
<artifactId>spring-aspects</artifactId>
<version>${xxx}</version>
</dependency>

<!-- spring-aop -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-aop</artifactId>
  <version>${xxx}</version>
</dependency>

<!-- spring-tx -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-tx</artifactId>
  <version>${xxx}</version>
</dependency>

<!-- spring-webmvc -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>${xxx}</version>
</dependency>

<!-- spring-jdbc -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>${xxx}</version>
</dependency>

<!-- spring-beans -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-beans</artifactId>
  <version>${xxx}</version>
</dependency>

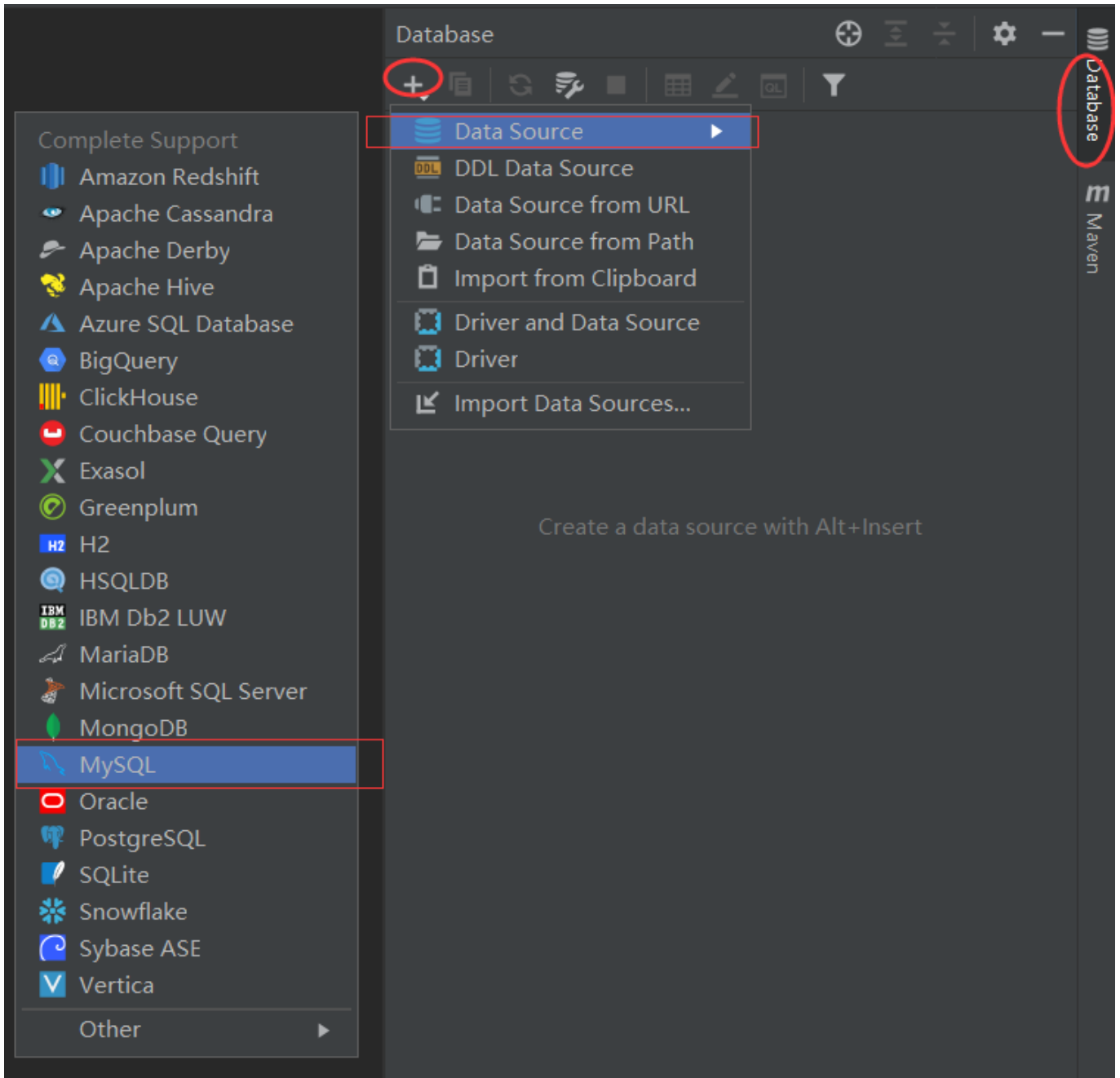
<!--spring-web -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>${xxx}</version>
</dependency>

<!-- spring-core -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>${xxx}</version>
</dependency>
```

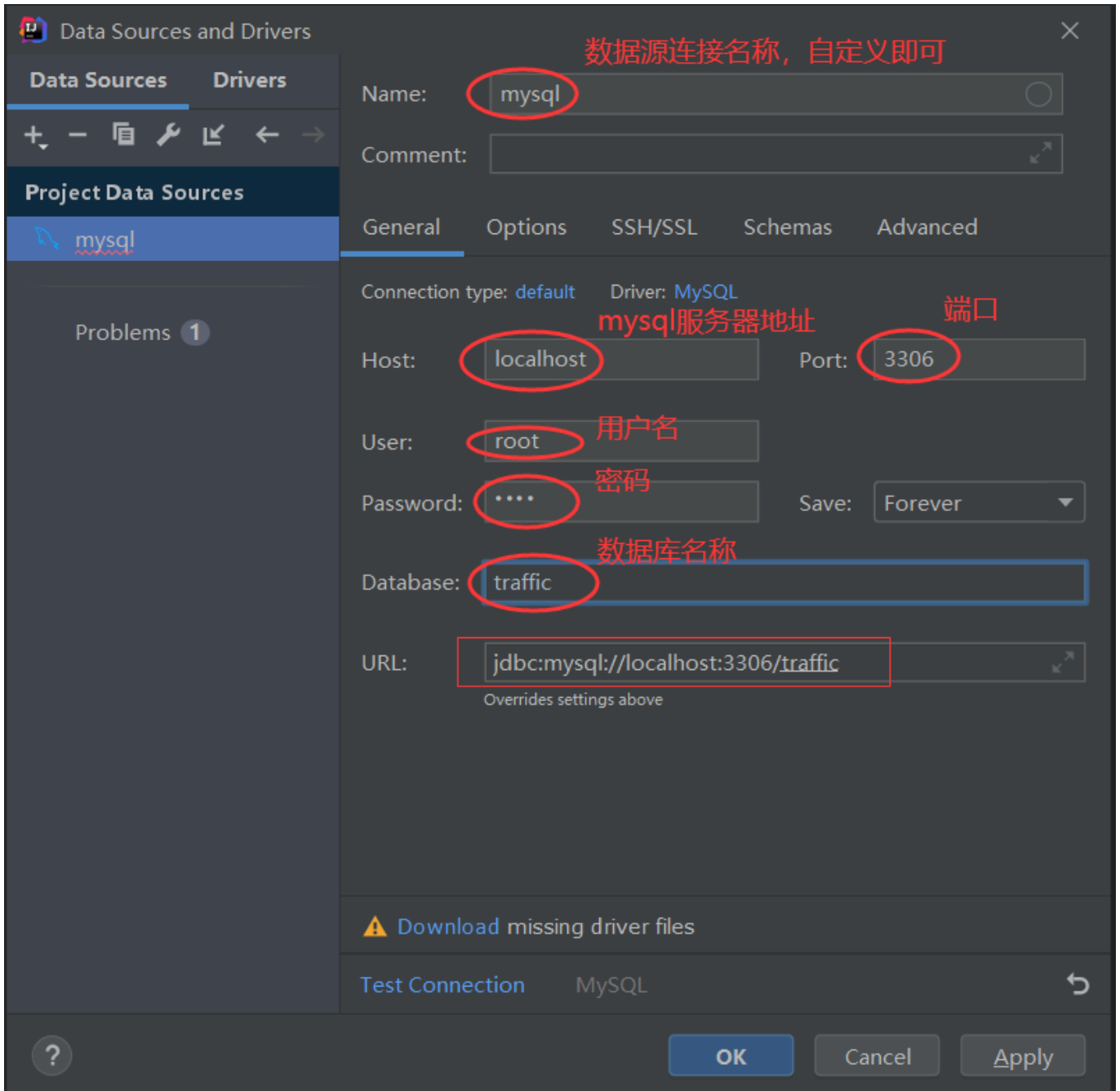
```
<!-- spring-context -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>${xxx}</version>
</dependency>
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
  <scope>test</scope>
</dependency>
</dependencies>

<build>
  <finalName>cdn</finalName>
  <pluginManagement><!-- lock down plugins versions to avoid using Maven
defaults (may be moved to parent pom) -->
    <plugins>
      <plugin>
        <artifactId>maven-clean-plugin</artifactId>
        <version>3.1.0</version>
      </plugin>
      <!-- see http://maven.apache.org/ref/current/maven-core/default-
bindings.html#Plugin_bindings_for_war_packaging -->
      <plugin>
        <artifactId>maven-resources-plugin</artifactId>
        <version>3.0.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
      <plugin>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.22.1</version>
      </plugin>
      <plugin>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.2.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-install-plugin</artifactId>
        <version>2.5.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-deploy-plugin</artifactId>
        <version>2.8.2</version>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
</project>
```

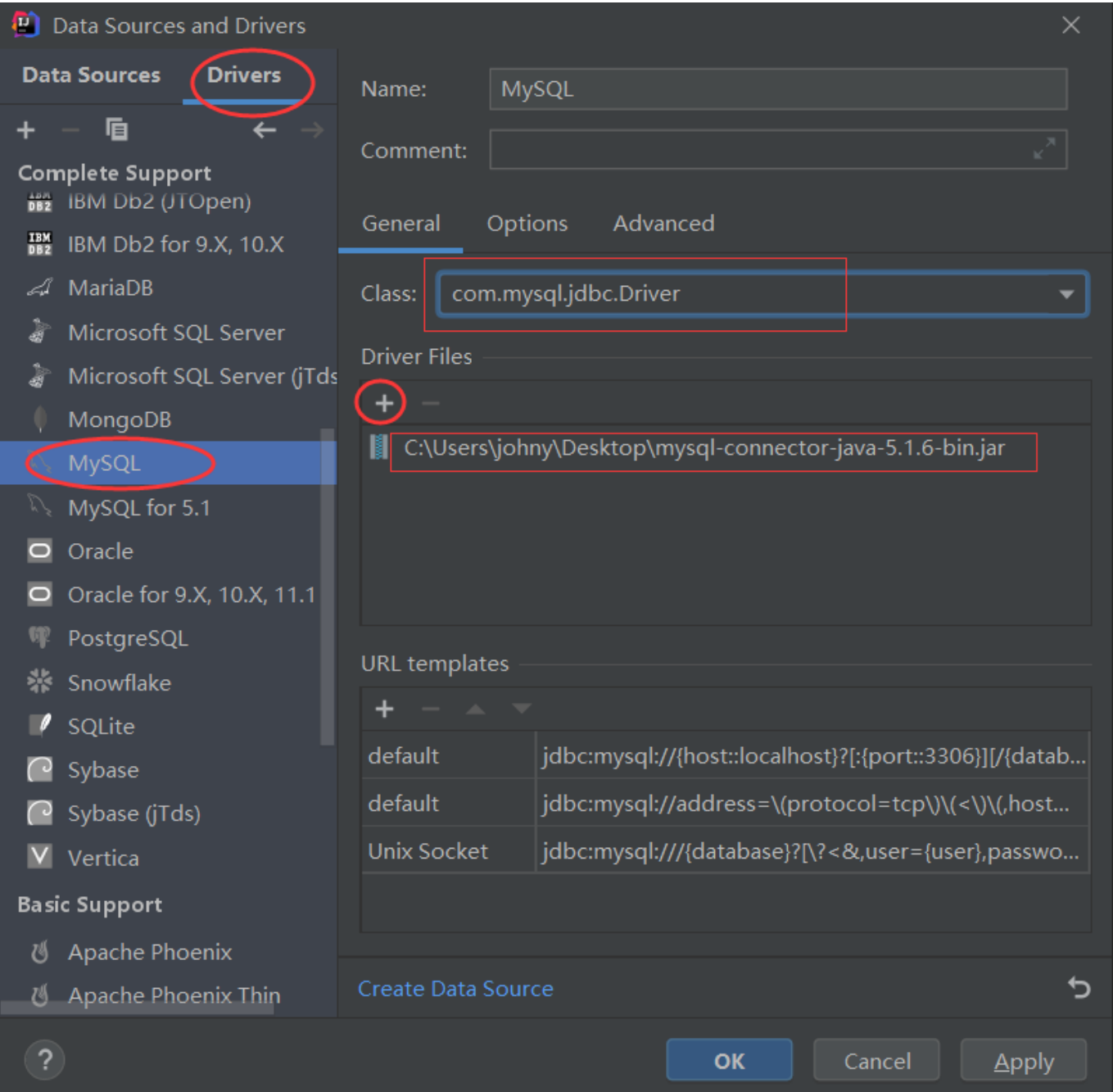
3. 在 java 下创建 com.gxlq 父包
4. 在IDEA中选择 View -> Tool Windows -> Database ,在右侧选择Database图标，打开database管理工具窗口。创建mysql数据库连接。如下图:

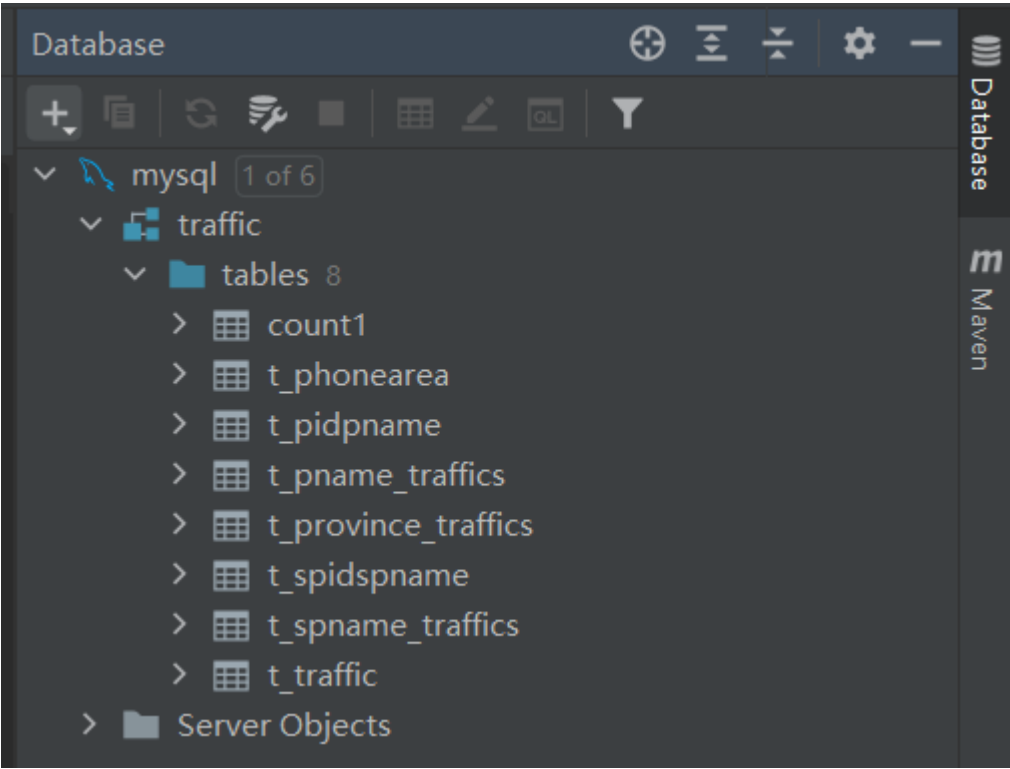


设置mysql数据库连接参数

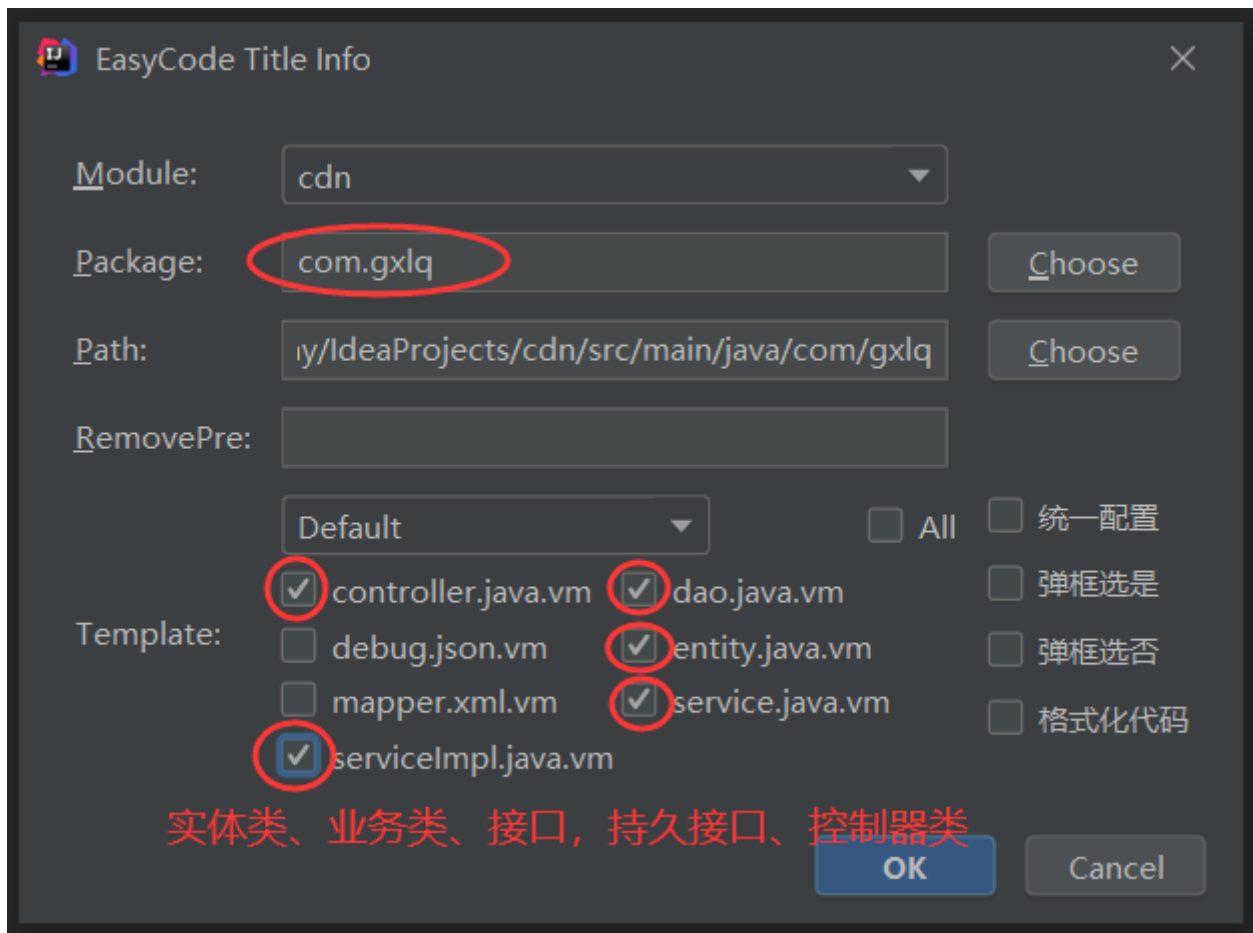
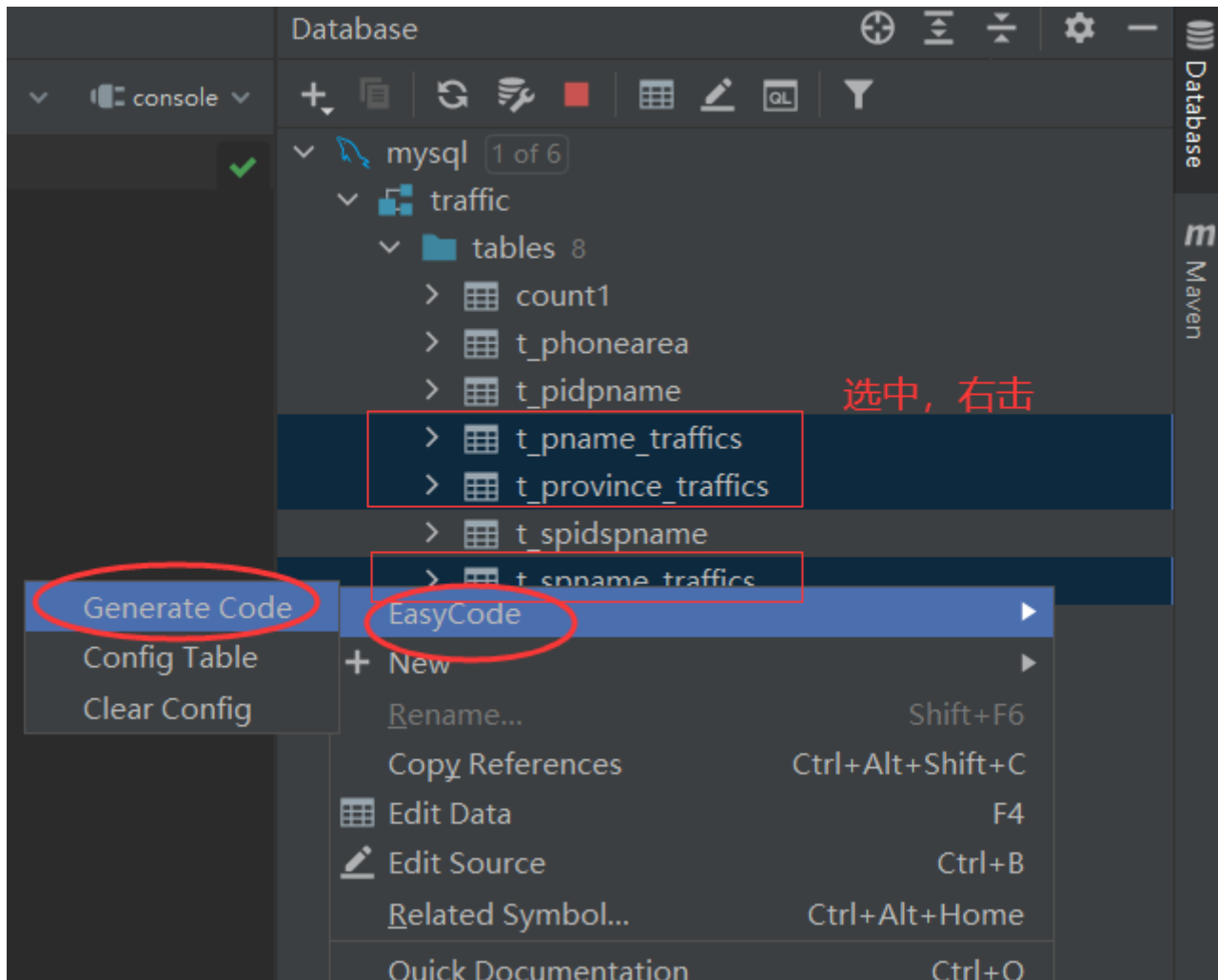


设置mysql数据库驱动





5. 在 IDEA 中安装Easy Code插件，File -> Settings->Plugins... ,生成实体类、业务类、控制器类,如图所示：



实体类 com.gxlq.entity.TPnameTraffics

```
package com.gxlq.entity;

import java.io.Serializable;

/**
 * (TPnameTraffics)实体类
 * 产品日流量信息
 * @author makejava
 * @since 2021-09-05 13:32:34
 */
public class TPnameTraffics implements Serializable {
    private static final long serialVersionUID = 433702769126737317L;
    //产品名称
    private String pname;
    //日流量
    private String traffics;
    //创建时间
    private String createtime;

    public TPnameTraffics() {
    }

    public TPnameTraffics(String pname, String traffics, String createtime) {
        this.pname = pname;
        this.traffics = traffics;
        this.createtime = createtime;
    }

    public String getPname() {
        return pname;
    }

    public void setPname(String pname) {
        this.pname = pname;
    }

    public String getTraffics() {
        return traffics;
    }

    public void setTraffics(String traffics) {
        this.traffics = traffics;
    }

    public String getCreatetime() {
        return createtime;
    }

    public void setCreatetime(String createtime) {
        this.createtime = createtime;
    }
}
```

```
@Override
public String toString() {
    return "TPnameTraffics{" +
        "pname='" + pname + '\'' +
        ", traffics='" + traffics + '\'' +
        ", createtime='" + createtime + '\'' +
        '}';
}
}
```

实体类 com.gxlq.entity.TProvinceTraffics

```
package com.gxlq.entity;

import java.io.Serializable;

/**
 * (TProvinceTraffics)实体类
 * 省份日流量信息
 * @author makejava
 * @since 2021-09-05 13:32:36
 */
public class TProvinceTraffics implements Serializable {
    private static final long serialVersionUID = 561963630805555509L;
    //省份名称
    private String province;
    //日流量
    private String traffics;
    //创建时间
    private String createtime;

    public TProvinceTraffics() {
    }

    public TProvinceTraffics(String province, String traffics, String createtime)
    {
        this.province = province;
        this.traffics = traffics;
        this.createtime = createtime;
    }

    public String getProvince() {
        return province;
    }

    public void setProvince(String province) {
        this.province = province;
    }

    public String getTraffics() {
        return traffics;
    }
}
```

```

    }

    public void setTraffics(String traffics) {
        this.traffics = traffics;
    }

    public String getCreatetime() {
        return createtime;
    }

    public void setCreatetime(String createtime) {
        this.createtime = createtime;
    }

    @Override
    public String toString() {
        return "TProvinceTraffics{" +
            "province='" + province + '\'' +
            ", traffics='" + traffics + '\'' +
            ", createtime='" + createtime + '\'' +
            '}';
    }
}

```

实体类 com.gxlq.entity.TSpnameTraffics

```

package com.gxlq.entity;

import java.io.Serializable;

/**
 * (TSpnameTraffics)实体类
 * 客户日流量信息
 * @author makejava
 * @since 2021-09-05 13:32:36
 */
public class TSpnameTraffics implements Serializable {
    private static final long serialVersionUID = -44939154139560362L;
    //客户名称
    private String spname;
    //日流量
    private String traffics;
    //创建时间
    private String createtime;

    public TSpnameTraffics() {
    }

    public TSpnameTraffics(String spname, String traffics, String createtime) {
        this.spname = spname;
    }
}

```

```

        this.traffics = traffics;
        this.createtime = createtime;
    }

    public String getSpname() {
        return spname;
    }

    public void setSpname(String spname) {
        this.spname = spname;
    }

    public String getTraffics() {
        return traffics;
    }

    public void setTraffics(String traffics) {
        this.traffics = traffics;
    }

    public String getCreatetime() {
        return createtime;
    }

    public void setCreatetime(String createtime) {
        this.createtime = createtime;
    }

    @Override
    public String toString() {
        return "TSpnameTraffics{" +
            "spname='" + spname + '\'' +
            ", traffics='" + traffics + '\'' +
            ", createtime='" + createtime + '\'' +
            '}';
    }
}

```

持久层接口 com.gxlq.dao.TPnameTrafficsDao

```

package com.gxlq.dao;

import com.gxlq.entity.TPnameTraffics;
import org.apache.ibatis.annotations.CacheNamespace;
import org.apache.ibatis.annotations.Param;
import org.apache.ibatis.annotations.Select;
import org.mybatis.caches.redis.RedisCache;

import java.util.List;

```

```
/**
 * (TPnameTraffics)表数据库访问层
 *
 * @author makejava
 * @since 2021-09-05 13:32:34
 */
//该注解表示，当前映射接口中的查询方法均支持mybatis二级缓存，此处使用redis作为二级缓存
，替代mybatis自带二级缓存
@CacheNamespace(size = 512,implementation = RedisCache.class)
public interface TPnameTrafficsDao {

    /**
     * 通过ID查询单条数据
     * @return 实例对象
     */
    TPnameTraffics queryById( );

    /**
     * 统计总行数
     * @param tPnameTraffics 查询条件
     * @return 总行数
     */
    long count(TPnameTraffics tPnameTraffics);

    /**
     * 新增数据
     * @param tPnameTraffics 实例对象
     * @return 影响行数
     */
    int insert(TPnameTraffics tPnameTraffics);

    /**
     * 批量新增数据 ( MyBatis原生foreach方法 )
     * @param entities List<TPnameTraffics> 实例对象列表
     * @return 影响行数
     */
    int insertBatch(@Param("entities") List<TPnameTraffics> entities);

    /**
     * 批量新增或按主键更新数据 ( MyBatis原生foreach方法 )
     * @param entities List<TPnameTraffics> 实例对象列表
     * @return 影响行数
     * @throws org.springframework.jdbc.BadSqlGrammarException 入参是空List的时候会
     抛SQL语句错误的异常，请自行校验入参
     */
    int insertOrUpdateBatch(@Param("entities") List<TPnameTraffics> entities);

    /**
     * 修改数据
     * @param tPnameTraffics 实例对象
     * @return 影响行数
     */
    int update(TPnameTraffics tPnameTraffics);
```

```

/**
 * 通过主键删除数据
 * @return 影响行数
 */
int deleteById( );

/**
 * 查询不同产品每天使用的流量
 */
@Select("select pname,traffics from t_pname_traffics where createtime=#{createtime}")
public List<TPnameTraffics> find(@Param("createtime") String createtime);

/**
 * 查询不同产品某个时间段的使用的流量
 * @param b 起始时间
 * @param e 结束时间
 * @return
 */
@Select("select pname,sum(traffics) as traffics from t_pname_traffics where createtime between #{b} and #{e} group by pname")
public List<TPnameTraffics> findx(@Param("b") String b, @Param("e") String e);
}

```

持久层接口 com.gxlq.dao.TProvinceTrafficsDao

```

package com.gxlq.dao;

import com.gxlq.entity.TProvinceTraffics;
import org.apache.ibatis.annotations.CacheNamespace;
import org.apache.ibatis.annotations.Param;
import org.apache.ibatis.annotations.Select;
import org.mybatis.caches.redis.RedisCache;

import java.util.List;

/**
 * (TProvinceTraffics)表数据库访问层
 *
 * @author makejava
 * @since 2021-09-05 13:32:36
 */
//该注解表示，当前映射接口中的查询方法均支持mybatis二级缓存，此处使用redis作为二级缓存，替代mybatis自带二级缓存
@CacheNamespace(size = 512,implementation = RedisCache.class)
public interface TProvinceTrafficsDao {

    /**
     * 通过ID查询单条数据
     *
     * @return 实例对象
     */
}

```

```
    */
    TProvinceTraffics queryById( );

    /**
     * 统计总行数
     *
     * @param tProvinceTraffics 查询条件
     * @return 总行数
     */
    long count(TProvinceTraffics tProvinceTraffics);

    /**
     * 新增数据
     *
     * @param tProvinceTraffics 实例对象
     * @return 影响行数
     */
    int insert(TProvinceTraffics tProvinceTraffics);

    /**
     * 批量新增数据 ( MyBatis原生foreach方法 )
     *
     * @param entities List<TProvinceTraffics> 实例对象列表
     * @return 影响行数
     */
    int insertBatch(@Param("entities") List<TProvinceTraffics> entities);

    /**
     * 批量新增或按主键更新数据 ( MyBatis原生foreach方法 )
     *
     * @param entities List<TProvinceTraffics> 实例对象列表
     * @return 影响行数
     * @throws org.springframework.jdbc.BadSqlGrammarException 入参是空List的时候会
    抛SQL语句错误的异常，请自行校验入参
     */
    int insertOrUpdateBatch(@Param("entities") List<TProvinceTraffics> entities);

    /**
     * 修改数据
     *
     * @param tProvinceTraffics 实例对象
     * @return 影响行数
     */
    int update(TProvinceTraffics tProvinceTraffics);

    /**
     * 通过主键删除数据
     *
     * @return 影响行数
     */
    int deleteById( );

    /**
     * 查询不同的省份每天使用的流量
```

```

    */
    @Select("select province,traffics from t_province_traffics where createtime=#{createtime}")
    public List<TProvinceTraffics> find(@Param("createtime") String createtime);

    /**
     * 查询不同的省份某个时间段使用的流量
     * @param b 起始时间
     * @param e 结束时间
     * @return
     */
    @Select("select province,sum(traffics) as traffics from t_province_traffics
where createtime between #{b} and #{e} group by province")
    public List<TProvinceTraffics> findx(@Param("b") String b, @Param("e") String
e);
}

```

持久层接口 com.gxlq.dao.TSpnameTrafficsDao

```

package com.gxlq.dao;

import com.gxlq.entity.TSpnameTraffics;
import org.apache.ibatis.annotations.CacheNamespace;
import org.apache.ibatis.annotations.Param;
import org.apache.ibatis.annotations.Select;
import org.mybatis.caches.redis.RedisCache;

import java.util.List;

/**
 * (TSpnameTraffics)表数据库访问层
 *
 * @author makejava
 * @since 2021-09-05 13:32:36
 */
//该注解表示，当前映射接口中的查询方法均支持mybatis二级缓存，此处使用redis作为二级缓存
，替代mybatis自带二级缓存
@CacheNamespace(size = 512,implementation = RedisCache.class)
public interface TSpnameTrafficsDao {

    /**
     * 通过ID查询单条数据
     *
     * @return 实例对象
     */
    TSpnameTraffics queryById( );

    /**
     * 统计总行数
     *
     * @param tSpnameTraffics 查询条件

```



```

    * @return 总行数
    */
    long count(TSpnameTraffics tSpnameTraffics);

    /**
     * 新增数据
     *
     * @param tSpnameTraffics 实例对象
     * @return 影响行数
     */
    int insert(TSpnameTraffics tSpnameTraffics);

    /**
     * 批量新增数据 ( MyBatis原生foreach方法 )
     *
     * @param entities List<TSpnameTraffics> 实例对象列表
     * @return 影响行数
     */
    int insertBatch(@Param("entities") List<TSpnameTraffics> entities);

    /**
     * 批量新增或按主键更新数据 ( MyBatis原生foreach方法 )
     *
     * @param entities List<TSpnameTraffics> 实例对象列表
     * @return 影响行数
     * @throws org.springframework.jdbc.BadSqlGrammarException 入参是空List的时候会
    抛SQL语句错误的异常，请自行校验入参
     */
    int insertOrUpdateBatch(@Param("entities") List<TSpnameTraffics> entities);

    /**
     * 修改数据
     *
     * @param tSpnameTraffics 实例对象
     * @return 影响行数
     */
    int update(TSpnameTraffics tSpnameTraffics);

    /**
     * 通过主键删除数据
     *
     * @return 影响行数
     */
    int deleteById( );

    /**
     * 查询不同的客户每天使用的流量
     */
    @Select("select spname,traffics from t_spname_traffics where createtime=#{createtime}")
    public List<TSpnameTraffics> find(@Param("createtime") String createtime);

    /**
     * 询不同的客户某个时间段使用的流量

```

```

    * @param b 起始时间
    * @param e 结束时间
    * @return
    */
    @Select("select spname,sum(traffics) as traffics from t_spname_traffics where
createtime between #{b} and #{e} group by spname")
    public List<TSpnameTraffics> findx(@Param("b") String b, @Param("e") String
e);
}

```

业务层接口 com.gxlq.service.TPnameTrafficsService

```

package com.gxlq.service;

import com.gxlq.entity.TPnameTraffics;

import java.util.List;

/**
 * (TPnameTraffics)表服务接口
 *
 * @author makejava
 * @since 2021-09-05 13:32:35
 */
public interface TPnameTrafficsService {

    /**
     * 通过ID查询单条数据
     *
     * @return 实例对象
     */
    TPnameTraffics queryById( );

    /**
     * 新增数据
     *
     * @param tPnameTraffics 实例对象
     * @return 实例对象
     */
    TPnameTraffics insert(TPnameTraffics tPnameTraffics);

    /**
     * 修改数据
     *
     * @param tPnameTraffics 实例对象
     * @return 实例对象
     */
    TPnameTraffics update(TPnameTraffics tPnameTraffics);

    /**
     * 通过主键删除数据

```

```
    *  
    * @return 是否成功  
    */  
    boolean deleteById( );  
  
    public List<TPnameTraffics> find(String createtime);  
    public List<TPnameTraffics> findx(String b, String e);  
  
}
```

业务层接口 com.gxlq.service.TProvinceTrafficsService

```
package com.gxlq.service;  
  
import com.gxlq.entity.TProvinceTraffics;  
  
import java.util.List;  
  
/**  
 * (TProvinceTraffics)表服务接口  
 *  
 * @author makejava  
 * @since 2021-09-05 13:32:36  
 */  
public interface TProvinceTrafficsService {  
  
    /**  
     * 通过ID查询单条数据  
     *  
     * @return 实例对象  
     */  
    TProvinceTraffics queryById( );  
  
    /**  
     * 新增数据  
     *  
     * @param tProvinceTraffics 实例对象  
     * @return 实例对象  
     */  
    TProvinceTraffics insert(TProvinceTraffics tProvinceTraffics);  
  
    /**  
     * 修改数据  
     *  
     * @param tProvinceTraffics 实例对象  
     * @return 实例对象  
     */  
    TProvinceTraffics update(TProvinceTraffics tProvinceTraffics);  
  
    /**  
     * 通过主键删除数据
```

```
    *
    * @return 是否成功
    */
    boolean deleteById( );

    public List<TProvinceTraffics> find(String createtime);
    public List<TProvinceTraffics> findx(String b, String e);

}
```

业务层接口 com.gxlq.service.TSpnameTrafficsService

```
package com.gxlq.service;

import com.gxlq.entity.TSpnameTraffics;

import java.util.List;

/**
 * (TSpnameTraffics)表服务接口
 *
 * @author makejava
 * @since 2021-09-05 13:32:36
 */
public interface TSpnameTrafficsService {

    /**
     * 通过ID查询单条数据
     *
     * @return 实例对象
     */
    TSpnameTraffics queryById( );

    /**
     * 新增数据
     *
     * @param tSpnameTraffics 实例对象
     * @return 实例对象
     */
    TSpnameTraffics insert(TSpnameTraffics tSpnameTraffics);

    /**
     * 修改数据
     *
     * @param tSpnameTraffics 实例对象
     * @return 实例对象
     */
    TSpnameTraffics update(TSpnameTraffics tSpnameTraffics);

    /**
     * 通过主键删除数据

```

```

    *
    * @return 是否成功
    */
    boolean deleteById( );

    public List<TSpnameTraffics> find(String createtime);
    public List<TSpnameTraffics> findx(String b, String e);

}

```

业务层类 com.gxlq.service.impl.TPnameTrafficsServiceImpl

```

package com.gxlq.service.impl;

import com.gxlq.entity.TPnameTraffics;
import com.gxlq.dao.TPnameTrafficsDao;
import com.gxlq.service.TPnameTrafficsService;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import javax.annotation.Resource;
import java.util.List;

/**
 * (TPnameTraffics)表服务实现类
 *
 * @author makejava
 * @since 2021-09-05 13:32:35
 */
@Service("tPnameTrafficsService")
//该注解可以添加在类上或某个方法，添加在类上表示当前类的所有方法均由spring框架管理事务，
//每个方法运行前会开启一个事务，每个方法结束关闭该事务
//事务的提交和回滚由spring框架代理实现
@Transactional
public class TPnameTrafficsServiceImpl implements TPnameTrafficsService {
    @Resource
    private TPnameTrafficsDao tPnameTrafficsDao;

    /**
     * 通过ID查询单条数据
     *
     * @return 实例对象
     */
    @Override
    public TPnameTraffics queryById( ) {
        return this.tPnameTrafficsDao.queryById();
    }

    /**
     * 新增数据

```

```

    *
    * @param tPnameTraffics 实例对象
    * @return 实例对象
    */
    @Override
    public TPnameTraffics insert(TPnameTraffics tPnameTraffics) {
        this.tPnameTrafficsDao.insert(tPnameTraffics);
        return tPnameTraffics;
    }

    /**
     * 修改数据
     *
     * @param tPnameTraffics 实例对象
     * @return 实例对象
     */
    @Override
    public TPnameTraffics update(TPnameTraffics tPnameTraffics) {
        this.tPnameTrafficsDao.update(tPnameTraffics);
        return this.queryById();
    }

    /**
     * 通过主键删除数据
     *
     * @return 是否成功
     */
    @Override
    public boolean deleteById( ) {
        return this.tPnameTrafficsDao.deleteById() > 0;
    }

    @Override
    public List<TPnameTraffics> find(String createtime) {
        return tPnameTrafficsDao.find(createtime);
    }

    @Override
    public List<TPnameTraffics> findx(String b, String e) {
        return tPnameTrafficsDao.findx(b, e);
    }
}

```

业务层类 com.gxlq.service.impl.TProvinceTrafficsServiceImpl

```

package com.gxlq.service.impl;

import com.gxlq.entity.TProvinceTraffics;
import com.gxlq.dao.TProvinceTrafficsDao;
import com.gxlq.service.TProvinceTrafficsService;
import org.springframework.stereotype.Service;

```

```
import org.springframework.transaction.annotation.Transactional;

import javax.annotation.Resource;
import java.util.List;

/**
 * (TProvinceTraffics)表服务实现类
 *
 * @author makejava
 * @since 2021-09-05 13:32:36
 */
@Service("tProvinceTrafficsService")
@Transactional
public class TProvinceTrafficsServiceImpl implements TProvinceTrafficsService {
    @Resource
    private TProvinceTrafficsDao tProvinceTrafficsDao;

    /**
     * 通过ID查询单条数据
     *
     * @return 实例对象
     */
    @Override
    public TProvinceTraffics queryById( ) {
        return this.tProvinceTrafficsDao.queryById();
    }

    /**
     * 新增数据
     *
     * @param tProvinceTraffics 实例对象
     * @return 实例对象
     */
    @Override
    public TProvinceTraffics insert(TProvinceTraffics tProvinceTraffics) {
        this.tProvinceTrafficsDao.insert(tProvinceTraffics);
        return tProvinceTraffics;
    }

    /**
     * 修改数据
     *
     * @param tProvinceTraffics 实例对象
     * @return 实例对象
     */
    @Override
    public TProvinceTraffics update(TProvinceTraffics tProvinceTraffics) {
        this.tProvinceTrafficsDao.update(tProvinceTraffics);
        return this.queryById();
    }

    /**
     * 通过主键删除数据
     *

```

```

        * @return 是否成功
        */
        @Override
        public boolean deleteById( ) {
            return this.tProvinceTrafficsDao.deleteById() > 0;
        }

        @Override
        public List<TProvinceTraffics> find(String createtime) {
            return tProvinceTrafficsDao.find(createtime);
        }

        @Override
        public List<TProvinceTraffics> findx(String b, String e) {
            return tProvinceTrafficsDao.findx(b,e);
        }
    }

```

业务层类 com.gxlq.service.impl.TSpnameTrafficsServiceImpl

```

package com.gxlq.service.impl;

import com.gxlq.entity.TSpnameTraffics;
import com.gxlq.dao.TSpnameTrafficsDao;
import com.gxlq.service.TSpnameTrafficsService;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import javax.annotation.Resource;
import java.util.List;

/**
 * (TSpnameTraffics)表服务实现类
 *
 * @author makejava
 * @since 2021-09-05 13:32:36
 */
@Service("tSpnameTrafficsService")
@Transactional
public class TSpnameTrafficsServiceImpl implements TSpnameTrafficsService {
    @Resource
    private TSpnameTrafficsDao tSpnameTrafficsDao;

    /**
     * 通过ID查询单条数据
     *
     * @return 实例对象
     */
    @Override
    public TSpnameTraffics queryById( ) {
        return this.tSpnameTrafficsDao.queryById();
    }

```



```
}

/**
 * 新增数据
 *
 * @param tSpnameTraffics 实例对象
 * @return 实例对象
 */
@Override
public TSpnameTraffics insert(TSpnameTraffics tSpnameTraffics) {
    this.tSpnameTrafficsDao.insert(tSpnameTraffics);
    return tSpnameTraffics;
}

/**
 * 修改数据
 *
 * @param tSpnameTraffics 实例对象
 * @return 实例对象
 */
@Override
public TSpnameTraffics update(TSpnameTraffics tSpnameTraffics) {
    this.tSpnameTrafficsDao.update(tSpnameTraffics);
    return this.queryById();
}

/**
 * 通过主键删除数据
 *
 * @return 是否成功
 */
@Override
public boolean deleteById( ) {
    return this.tSpnameTrafficsDao.deleteById() > 0;
}

@Override
public List<TSpnameTraffics> find(String createtime) {
    return tSpnameTrafficsDao.find(createtime);
}

@Override
public List<TSpnameTraffics> findx(String b, String e) {
    return tSpnameTrafficsDao.findx(b,e);
}
}
```

控制层类 com.gxlq.controller.TPnameTrafficsController

```
package com.gxlq.controller;
```

```

import com.gxlq.entity.TPnameTraffics;
import com.gxlq.service.TPnameTrafficsService;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import javax.annotation.Resource;
import javax.servlet.http.HttpServletRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * (TPnameTraffics)表控制层
 *
 * @author makejava
 * @since 2021-09-05 13:32:32
 */
@Controller
public class TPnameTrafficsController {
    /**
     * 业务对象
     */
    @Resource
    private TPnameTrafficsService tPnameTrafficsService;

    @RequestMapping(value = "/tpnametraffics",method = RequestMethod.GET)
    //该注解用于配合jackson-databind使用
    @ResponseBody
    public Map<String,Object> list(HttpServletRequest request){
        String b = request.getParameter("b");
        String e = request.getParameter("e");
        List<TPnameTraffics> pnts = new ArrayList<>();
        if(e!=null && !e.equals("") && !e.equals("null")){
            pnts = tPnameTrafficsService.findx(b, e);
        }else{
            pnts = tPnameTrafficsService.find(b);
        }
        Map<String,Object> msg = new HashMap<>();
        msg.put("code",20000);
        msg.put("data",pnts);
        return msg;
    }
}

```

控制层类 com.gxlq.controller.TPProvinceTrafficsController

```
package com.gxlq.controller;
```

```

import com.gxlq.entity.TProvinceTraffics;
import com.gxlq.service.TProvinceTrafficsService;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import javax.annotation.Resource;
import javax.servlet.http.HttpServletRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * (TProvinceTraffics)表控制层
 *
 * @author makejava
 * @since 2021-09-05 13:32:36
 */
@Controller
public class TProvinceTrafficsController {
    /**
     * 服务对象
     */
    @Resource
    private TProvinceTrafficsService tProvinceTrafficsService;

    @RequestMapping(value = "/tprovincetraffics",method = RequestMethod.GET)
    @ResponseBody
    public Map<String,Object> list(HttpServletRequest request){
        String b = request.getParameter("b");
        String e = request.getParameter("e");
        List<TProvinceTraffics> pts = new ArrayList<>();
        if(e!=null && !e.equals("") && !e.equals("null")){
            pts = tProvinceTrafficsService.findx(b,e);
        }else{
            pts = tProvinceTrafficsService.find(b);
        }
        Map<String,Object> msg = new HashMap<>();
        msg.put("code",20000);
        msg.put("data",pts);
        return msg;
    }
}

```

控制层类 com.gxlq.controller.TSpnameTrafficsController

```

package com.gxlq.controller;

import com.gxlq.entity.TSpnameTraffics;

```

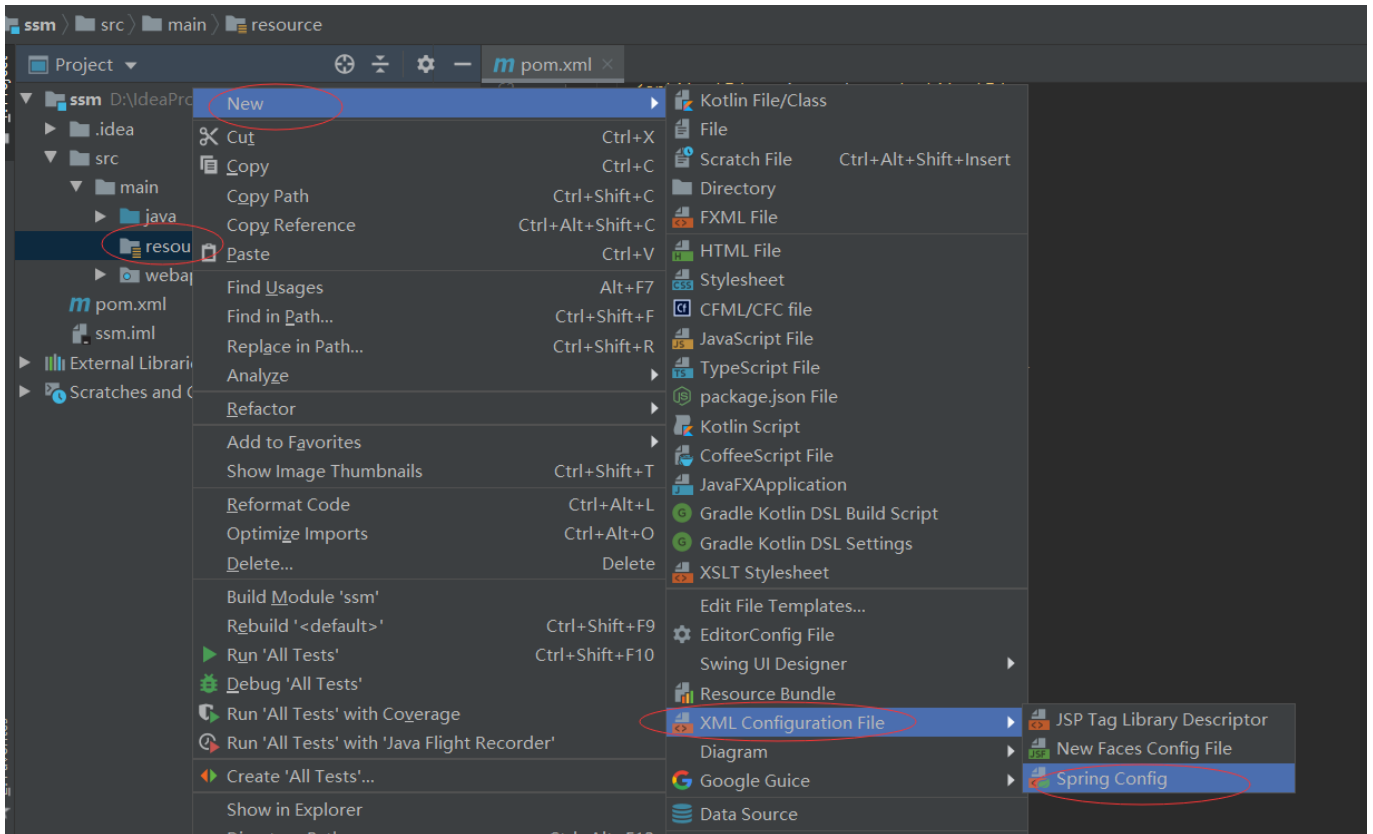
```
import com.gxlq.service.TSpnameTrafficsService;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import javax.annotation.Resource;
import javax.servlet.http.HttpServletRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * (TSpnameTraffics)表控制层
 *
 * @author makejava
 * @since 2021-09-05 13:32:36
 */
@Controller
public class TSpnameTrafficsController {
    /**
     * 服务对象
     */
    @Resource
    private TSpnameTrafficsService tSpnameTrafficsService;

    @RequestMapping(value = "/tspsnametraffics",method = RequestMethod.GET)
    @ResponseBody
    public Map<String,Object> list(HttpServletRequest request){
        String b = request.getParameter("b");
        String e = request.getParameter("e");
        List<TSpnameTraffics> spnts = new ArrayList<>();
        if(e!=null && !e.equals("") && !e.equals("null")){
            spnts = tSpnameTrafficsService.findx(b,e);
        }else{
            spnts = tSpnameTrafficsService.find(b);
        }
        Map<String,Object> msg = new HashMap<>();
        msg.put("code",20000);
        msg.put("data",spnts);
        return msg;
    }
}
```

6. 在 resources 下创建 spring 框架的配置文件 applicationContext.xml。



在 spring 配置文件中配置数据源连接（如 MySQL）、扫描注解的父包路径，对应标签的名称空间会自动导入
在 spring 配置文件中配置 myBatis 会话工厂，持久层 Bean，事务管理等。

```
<?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:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd">
```

<!-- 添加扫描配置，告诉spring框架到哪些包(仅指定父包即可)中去扫描IOC注解，并实现自动依赖注入-->

```
<context:component-scan base-package="com.gx1q"></context:component-scan>
<!-- 数据源C3P0连接池-->
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
  <property name="driverClass" value="com.mysql.jdbc.Driver"/>
  <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/traffic"/>
  <property name="user" value="root"/>
  <property name="password" value="root"/>
  <property name="maxPoolSize" value="30"/>
</bean>
```

<!-- 事务管理器，此处使用的是封装jdbc的事务管理机制，间接使用oracleDBMS的事务管理机制，需要指定对应哪个数据源连接池进行事务管理-->

```

    <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"></property>
</bean>
<tx:annotation-driven transaction-manager="transactionManager">
</tx:annotation-driven>

<!-- 该对象用于对mybatis会话工厂相关参数进行配置 -->
<bean id="cfg" class="org.apache.ibatis.session.Configuration">
    <!-- 显示sql语句相关日志信息到控制台 -->
    <property name="logImpl"
value="org.apache.ibatis.logging.stdout.StdOutImpl"></property>
</bean>
<!--基于spring框架整个mybatis，创建mybatis会话工厂，依赖于C3P0连接池-->
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource"></property>
    <property name="configuration" ref="cfg"></property>
</bean>

<!--持久层对象，底层使用了反射机制，依赖于会话工厂，spring框架负责该对象的创建，默认为单例singleton，scope="prototype"可以设置非单例-->
<bean id="tPnameTrafficsDao"
class="org.mybatis.spring.mapper.MapperFactoryBean">
    <property name="sqlSessionFactory" ref="sqlSessionFactory"></property>
    <property name="mapperInterface" value="com.gxlq.dao.TPnameTrafficsDao">
</property>
</bean>

    <bean id="tProvinceTrafficsDao"
class="org.mybatis.spring.mapper.MapperFactoryBean">
    <property name="sqlSessionFactory" ref="sqlSessionFactory"></property>
    <property name="mapperInterface"
value="com.gxlq.dao.TProvinceTrafficsDao"></property>
</bean>

    <bean id="tSpnameTrafficsDao"
class="org.mybatis.spring.mapper.MapperFactoryBean">
    <property name="sqlSessionFactory" ref="sqlSessionFactory"></property>
    <property name="mapperInterface" value="com.gxlq.dao.TSpnameTrafficsDao">
</property>
</bean>

</beans>

```

7. 在 web.xml 文件中配置 spring 框架监听器和 springMVC 的总控制器。

```

<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
    <display-name>Archetype Created Web Application</display-name>

```

```

<!--若让项目在web环境下依托spring框架正常运行，须指定spring框架的配置文件路径及监听器-->
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath:applicationcontext.xml</param-value>
</context-param>
<!--spring框架监听器-->
<listener>
  <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

<!--springmvc 框架的总控制器，实质为servlet-->
<servlet>
  <servlet-name>zyj</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
  <!--指定springmvc框架配置文件的位置，默认/WEB-INF/总控制器的名称-servlet.xml-->
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <!--<param-value>/WEB-INF/springmvc.xml</param-value-->
    <!--此处的路径为项目resources目录下-->
    <param-value>classpath:springmvc.xml</param-value>
  </init-param>

  <!--服务器启动时加载该servlet的优先次序，“1”优先级最高-->
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>zyj</servlet-name>
  <!--指定springmvc框架对哪些http请求进行拦截处理，即，什么样的请求才走springmvc框
架，此处/表示 所有请求-->
  <url-pattern>/</url-pattern>
</servlet-mapping>

<!--session过期时间，30分钟-->
<session-config>
  <session-timeout>30</session-timeout>
</session-config>
</web-app>

```

8. 在 resources 下创建 springmvc 框架的配置文件 springmvc.xml，并进行配置。

```

<?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:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context

```

```

https://www.springframework.org/schema/context/spring-context.xsd">
  <!--指定springmvc框架对应控制器的位置，即去哪些包下扫描控制器类-->
  <context:component-scan base-package="com.gxlq.controller">
</context:component-scan>
  <!--用于自动注册DefaultAnnotationHandlerMapping和AnnotationMethodHandlerAdapter
两个bean,是springMVC为@Controller分发请求所必须的。-->
  <mvc:annotation-driven></mvc:annotation-driven>
</beans>

```

9. 在 resources 下创建 log4j日志框架 的配置文件 log4j.properties，并进行配置。

```

log4j.rootLogger=DEBUG,zhaoliu
## zhangshan
log4j.appender.zhangshan=org.apache.log4j.ConsoleAppender
log4j.appender.zhangshan.layout=org.apache.log4j.HTMLLayout
## lisi
log4j.appender.lisi=org.apache.log4j.FileAppender
log4j.appender.lisi.layout=org.apache.log4j.PatternLayout
log4j.appender.lisi.layout.ConversionPattern=[QC]%p[%t]%C.%M(%L)|%m%n
log4j.appender.lisi.File=D:\\logs\\test.log

##log4j.appender.lisi.layout.ConversionPattern=%d-[TS]%p%t%c-%m%n
## wangwu
log4j.appender.wangwu=org.apache.log4j.DailyRollingFileAppender
log4j.appender.wangwu.layout=org.apache.log4j.TTCCLayout
log4j.appender.wangwu.File=D:\\test.log

## zhaoliu
log4j.appender.zhaoliu=org.apache.log4j.ConsoleAppender
log4j.appender.zhaoliu.layout=org.apache.log4j.PatternLayout
log4j.appender.zhaoliu.layout.ConversionPattern=[QC]%p[%t]%C.%M(%L)|%m%n

```

10. 在 resources 下创建 redis服务器 的配置文件 redis.properties，并进行配置。

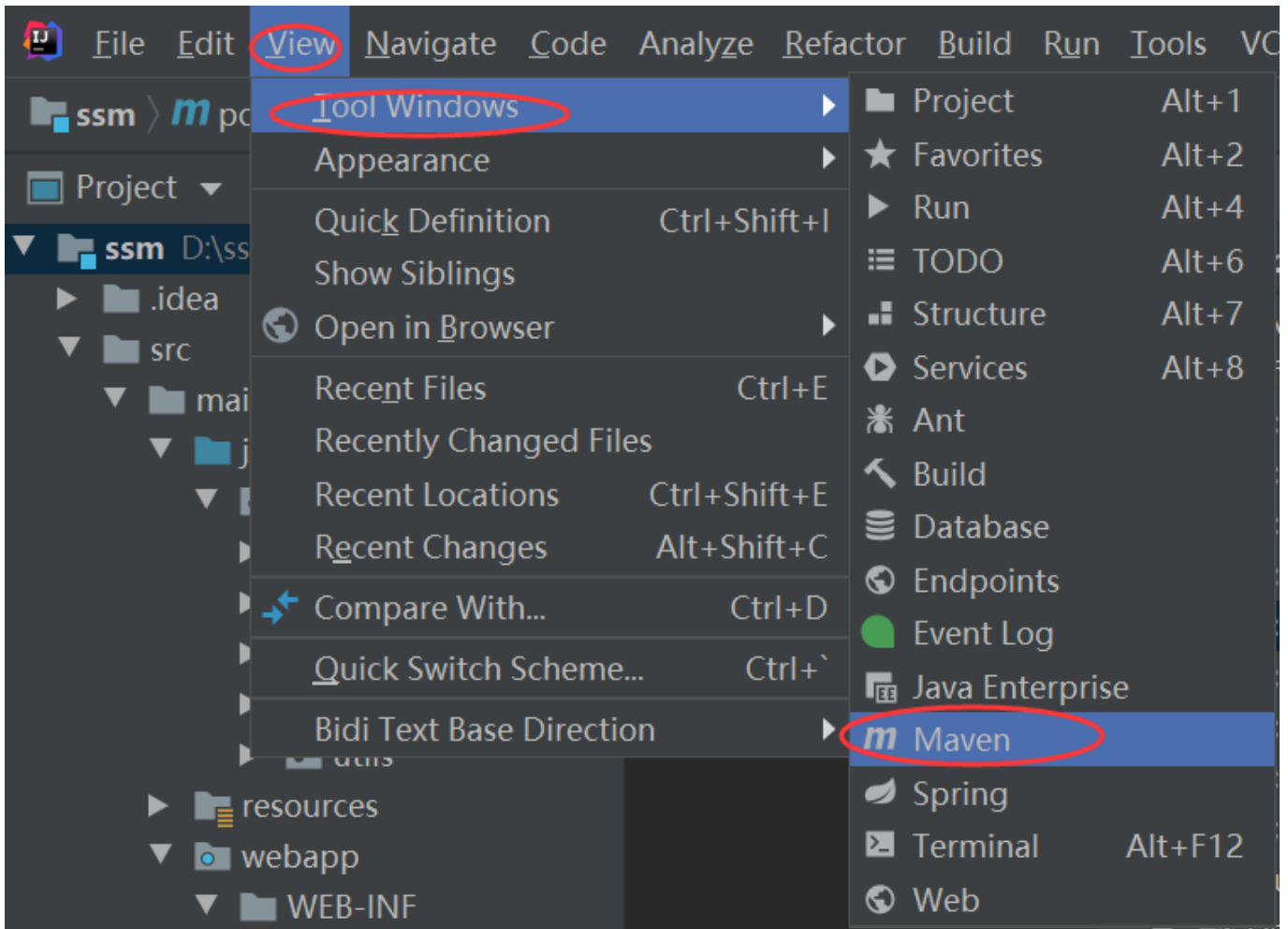
```

host=localhost
port=6379
connectionTimeout=5000
soTimeout=5000
password=
database=0
clientName=

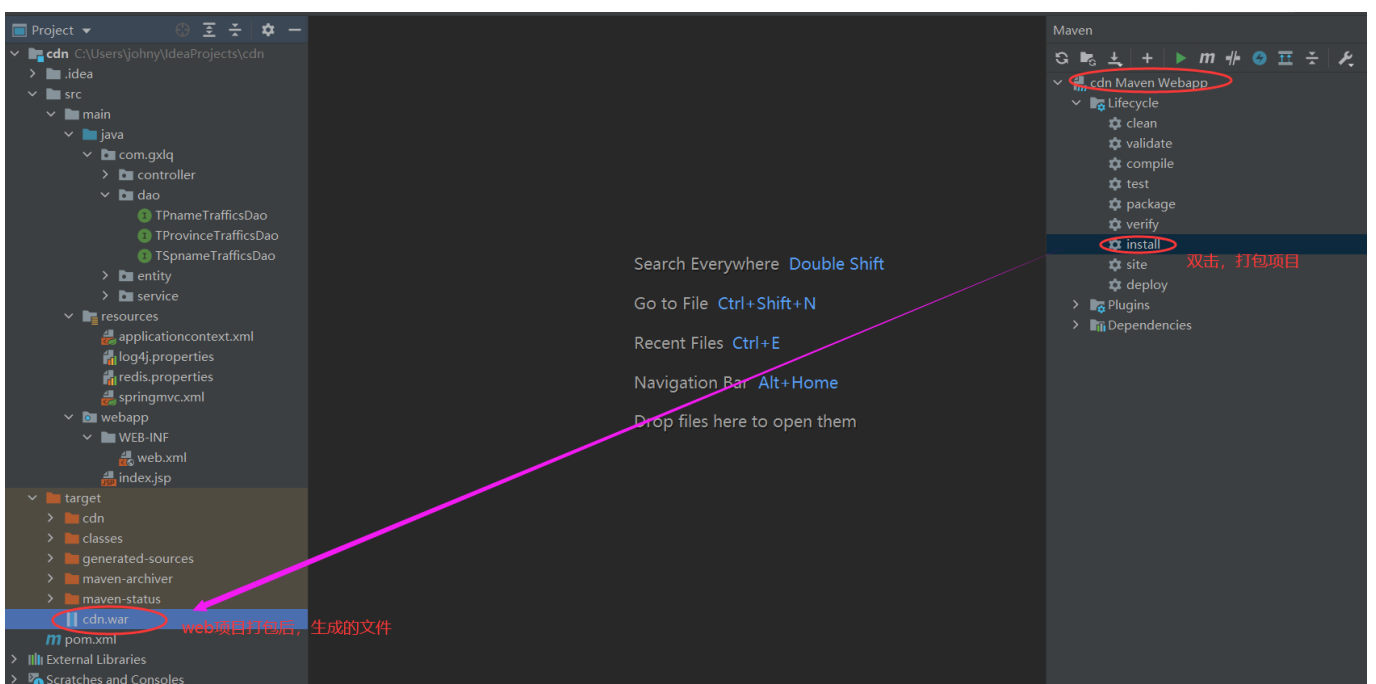
```

八、后端打包部署

在 IDEA 中把项目打包为 war。选择 View -> Tool Windows -> Maven，在右侧会显示出 Maven 管理窗口。



利用 `maven package` 或 `install` 一下，在项目文件夹 `/target/` 下可以找到打包后生成的 `cdn.war` 文件。



把 `cdn.war` 复制到 Tomcat 服务器 `C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps` 下，然后双击 `C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\Tomcat8.exe` 启动 Tomcat 服务器。

在浏览器的地址栏中输入 `url: http://localhost:1573/cdn/tprovincetraffics?b=20200309` 显示如下效果，表示后台成功从数据源中获取到了数据信息。



```
{
  "code": 20000,
  "data": [
    {
      "province": "河北省",
      "traffics": 11008667,
      "createtime": null
    },
    {
      "province": "安徽省",
      "traffics": 9312684.0,
      "createtime": null
    },
    {
      "province": "新疆维吾尔自治区",
      "traffics": 3604938.2,
      "createtime": null
    },
    {
      "province": "广东省",
      "traffics": 20625054,
      "createtime": null
    },
    {
      "province": "山东省",
      "traffics": 17883834,
      "createtime": null
    },
    {
      "province": "陕西省",
      "traffics": 8351245.0,
      "createtime": null
    },
    {
      "province": "河南省",
      "traffics": 17199704,
      "createtime": null
    },
    {
      "province": "黑龙江省",
      "traffics": 4218398.0,
      "createtime": null
    },
    {
      "province": "湖南省",
      "traffics": 9818846.0,
      "createtime": null
    },
    {
      "province": "辽宁省",
      "traffics": 6181527.5,
      "createtime": null
    },
    {
      "province": "江西省",
      "traffics": 2920873.2,
      "createtime": null
    },
    {
      "province": "上海市",
      "traffics": 7174596.5,
      "createtime": null
    },
    {
      "province": "贵州省",
      "traffics": 3969506.5,
      "createtime": null
    },
    {
      "province": "内蒙古自治区",
      "traffics": 3965067.8,
      "createtime": null
    },
    {
      "province": "山西省",
      "traffics": 1915042.2,
      "createtime": null
    },
    {
      "province": "四川省",
      "traffics": 3836647.0,
      "createtime": null
    },
    {
      "province": "北京市",
      "traffics": 10396440,
      "createtime": null
    },
    {
      "province": "江苏省",
      "traffics": 13872780,
      "createtime": null
    },
    {
      "province": "吉林省",
      "traffics": 10712207,
      "createtime": null
    },
    {
      "province": "浙江省",
      "traffics": 7473656.0,
      "createtime": null
    },
    {
      "province": "青海省",
      "traffics": 1164907.2,
      "createtime": null
    },
    {
      "province": "福建省",
      "traffics": 11134730,
      "createtime": null
    },
    {
      "province": "甘肃省",
      "traffics": 1624121.0,
      "createtime": null
    },
    {
      "province": "天津市",
      "traffics": 3446924.8,
      "createtime": null
    },
    {
      "province": "重庆市",
      "traffics": 6046721.5,
      "createtime": null
    },
    {
      "province": "宁夏回族自治区",
      "traffics": 1672444.1,
      "createtime": null
    },
    {
      "province": "海南省",
      "traffics": 10075594,
      "createtime": null
    },
    {
      "province": "广西壮族自治区",
      "traffics": 3354701.0,
      "createtime": null
    },
    {
      "province": "湖北省",
      "traffics": 16376587,
      "createtime": null
    },
    {
      "province": "云南省",
      "traffics": 2043353.6,
      "createtime": null
    },
    {
      "province": "河南省",
      "traffics": 1931025.6,
      "createtime": null
    },
    {
      "province": "西藏自治区",
      "traffics": 297077.25,
      "createtime": null
    }
  ]
}
```

九、前端

安装 Node.js

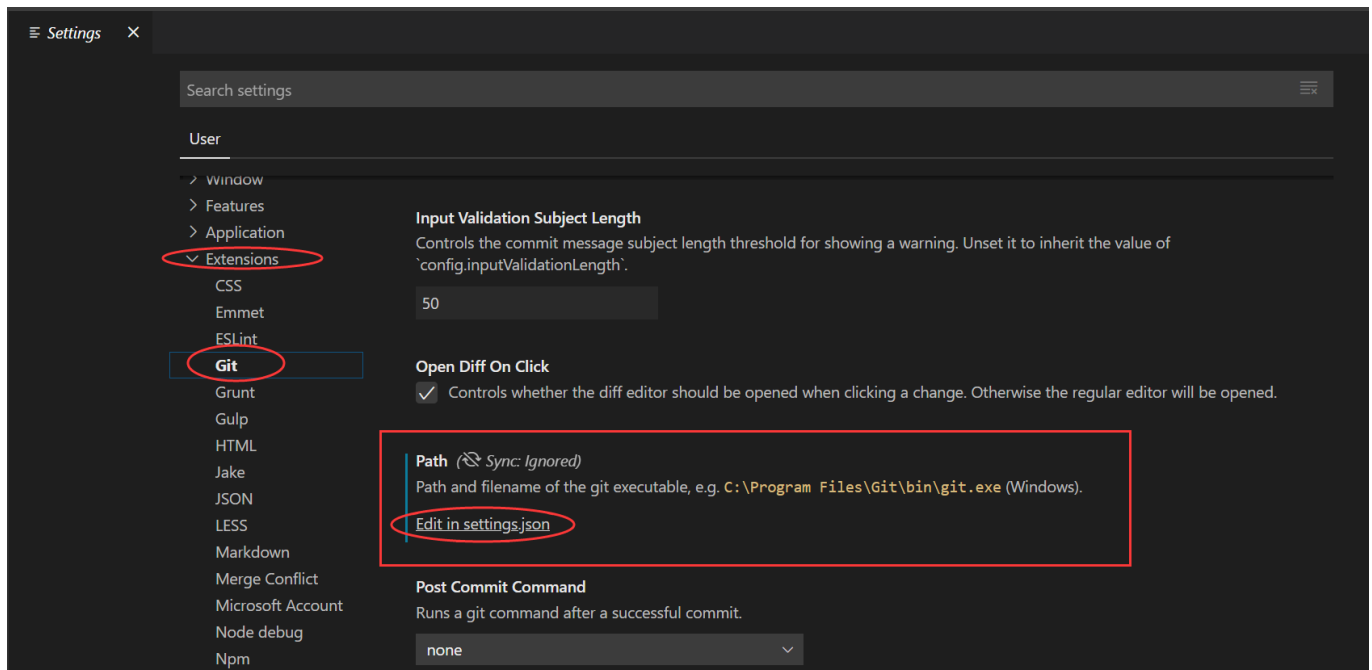
为了能正常使用 `npm` 指令，需要在电脑上安装 `Node.js`。

到官网下载安装文件，`Node.js` 官方网站地址：<https://nodejs.org/en/download/>，下载 `node-v12.16.1-x64.msi` 之后，全部默认安装，选择路径 `C:\Program Files\nodejs`

安装 Git

到官网下载安装文件，`Git` 官方下载地址 <https://git-scm.com/download>，下载 `Git-2.24.0-64-bit.exe` 之后，全部默认安装，选择路径 `C:\Program Files\Git`

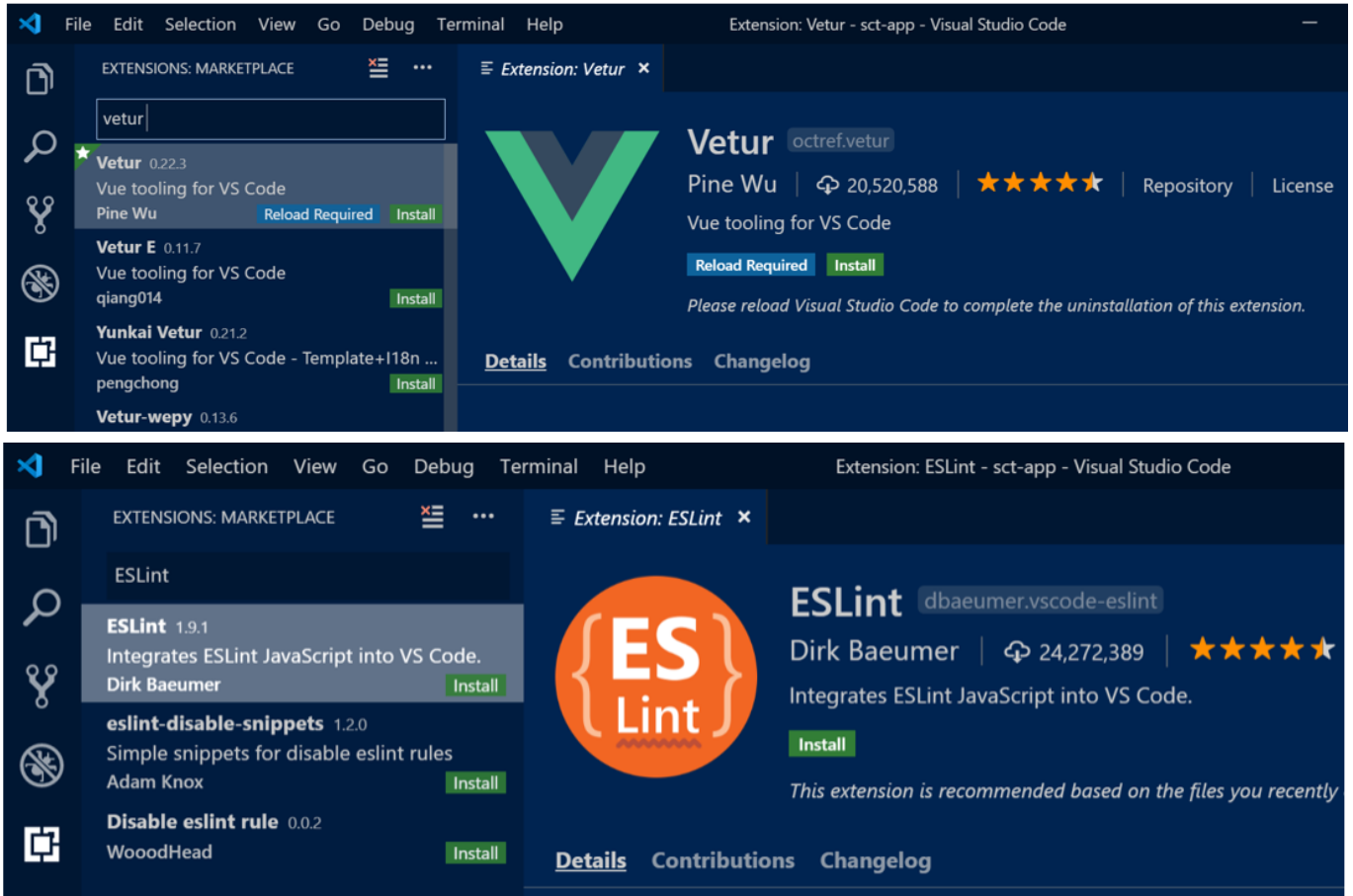
然后在 `VSCode` 中配置 `Git` 的路径就可以了，在 `VSCode` 中选择 `File -> Preferences -> Settings`，找到 `Extensions` 下的 `Git`，编辑 `settings.json` 文件，如图所示：



```
{
  "explorer.confirmDelete": false,
  "git.path": "C:/Program Files/Git/bin/git.exe",
  "git.enableSmartCommit": true
}
```

在 VSCode 中安装 Vue 插件

插件 `vetur`，实现支持 `vue` 文件的代码高亮，插件 `ESLint` 支持 `js` 文件的脚本检测。
在 VSCode 中，点击左边的 `Extensions` 图标，输入 `vetur`，找到对应版本然后点击 `install` 即可。

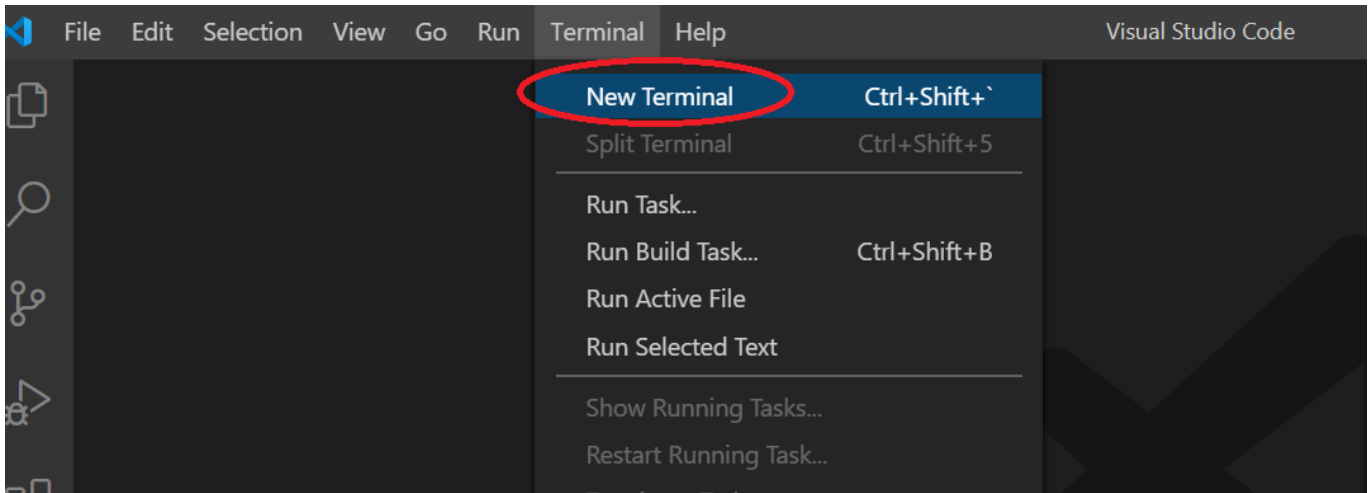


安装 `vue-cli` 插件 (脚手架) ，在 VSCode 中选择 `Terminal -> New Terminal` ，在终端中输入指令 `npm install -g vue-cli` 。

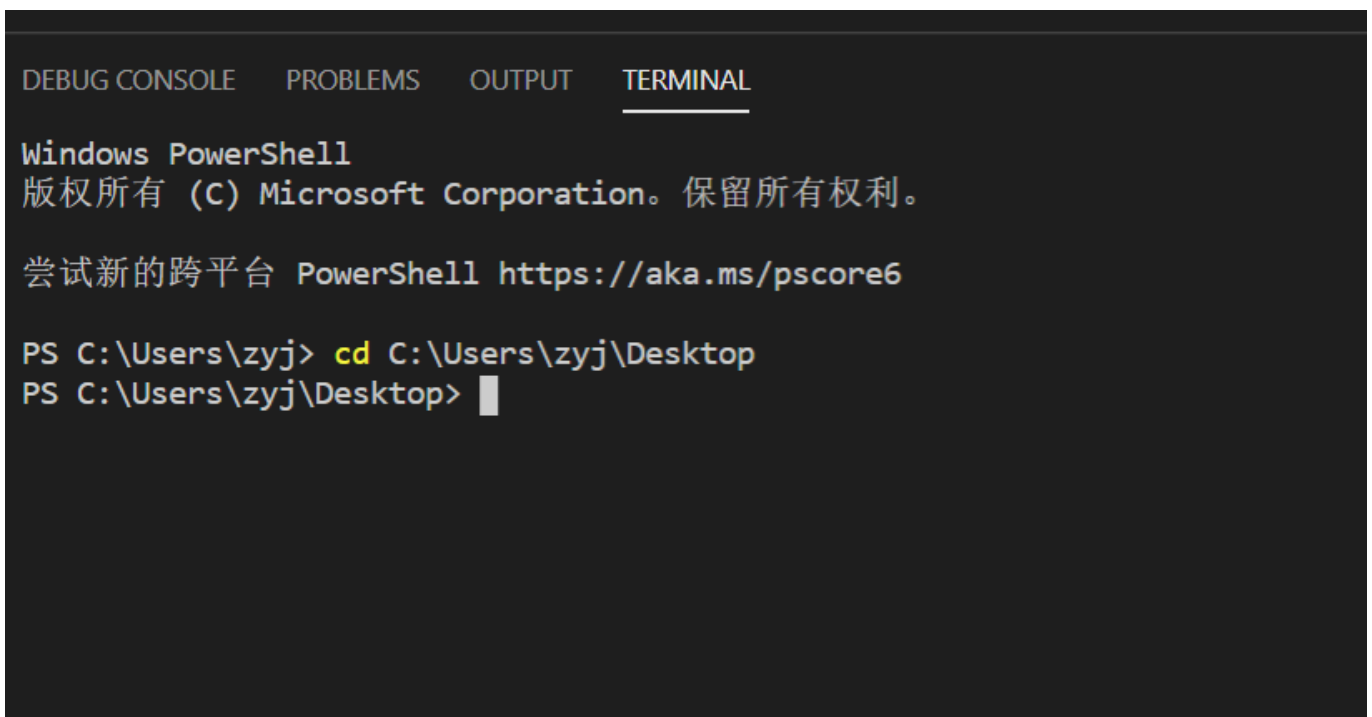
从 Github 上克隆项目

在 VSCode 中从 Github 上直接克隆项目 `vue-admin-template` ，这是一个 Vue admin 管理后台，我们在其上做二次开发。该项目的资源地址为：<https://github.com/PanJiaChen/vue-admin-template.git> 。

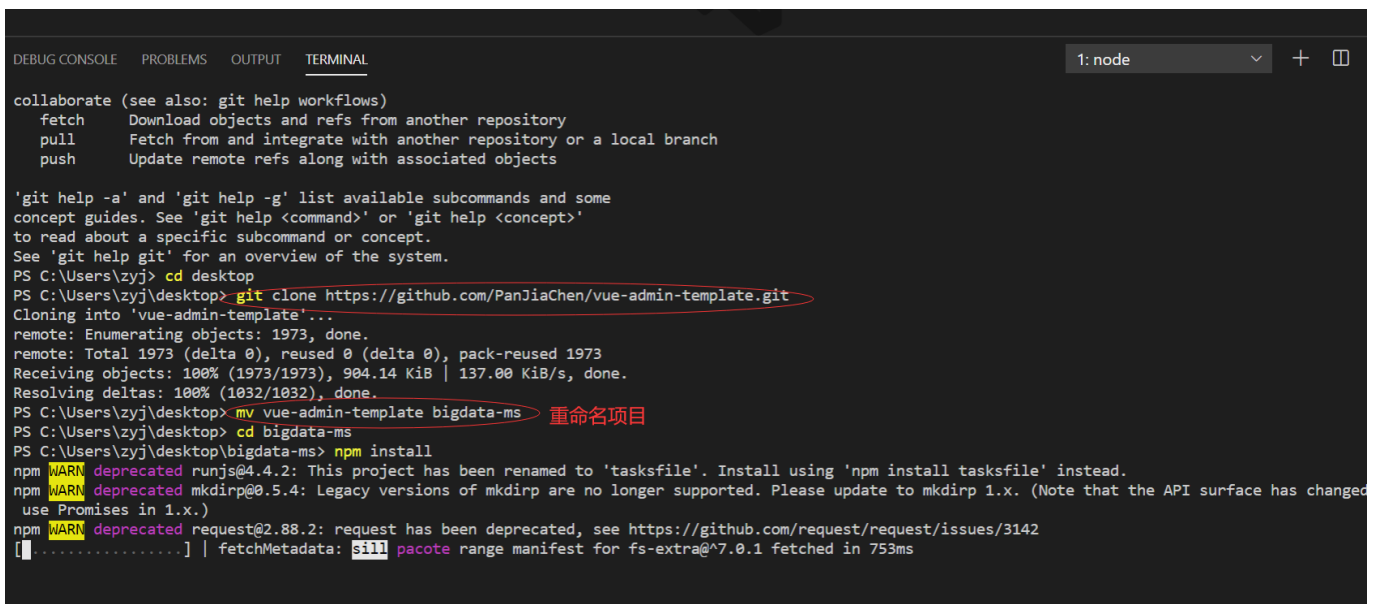
1. 在 VSCode 中打一个命令行终端窗口，也可以使用 `ctrl + ~` 。



2. 切换到存放项目的文件夹



3. 克隆并编译项目



注意：下载过程中如果有的插件加载错误，可以使用淘宝代理 `npm install -g cnpm --registry=https://registry.npm.taobao.org`

前端代码的编写

在 VSCode 中打开 `bigdata-ms` 项目，File -> Open Folder...，目录结构如下图所示：

- bulid	构建脚本目录
- mock	mock数据目录
- node_modules	依赖的node工具包目录
- public	对外公开的资源
- src	vue项目的核心文件目录
- api	各种接口，ajax组件也在其中
- assets	图片等静态资源，这里的资源会被webpack构建
- components	各种公共组件，非公共组件在各自view下维护
- icons	svg 图标
- layout	布局
- router	前端路由
- store	应用级数据（存储）
- styles	各种样式
- utils	公共工具，非公共工具在各自view下维护
- views	页面目录
- App.vue	页面级Vue根组件
- main.js	页面入口js文件
- permission.js	认证入口
- settings.js	配置
- tests	测试相关
- .editorconfig	协同团队开发人员之间的代码的风格及样式规范化的一个工具,确保不同的编辑器和IDE之间定义和维护一致的编码样式
- .env.development	vue项目开发环境下的环境变量
- .env.production	vue项目生产环境下的环境变量
- .env.staging	vue项目测试环境下的环境变量
- .eslintignore	指定哪些目录忽略eslint语法检测
- .eslintrc.js	指定eslint语法检测的规则，规范编码习惯，便于以后项目维护和优化
- .gitignore	指定哪些文件不需要添加到git的版本管理中
- .travis.yml	yaml语法写出来的配置文件，用于描述如何持续构建，支持各种语言，各种系统环境
- .babel.config.js	Babel是一个JS编译器，Vue项目普遍使用ES6语法，若要兼容低版本浏览器，就需要引入Babel，将ES6转换为ES5
- jest.config.js	facebook出品的一款测试框架jest的配置文件
- jsconfig.json	指定根目录和javascript服务提供的功能选项
- license	项目版权声明文件
- package-lock.json	记录模块与模块之间的依赖关系，锁定包的版本，记录项目所依赖的第三方包的树状结构和包的下载地址，加快重新安装的下速度
- package.json	在拷贝项目时不需要拷贝node_modules文件夹（太慢），为了解决安装的依赖与原项目依赖及版本保持一致，在package.json中记录了当前项目的信息，拷贝项目后，直接执行npm install会自动安装package.json文件中记录的依赖
- postcss.config.js	一个用javascript工具和插件转换css代码的工具
- readme-zh.md	阅读指南文件（中文版）
- readme.md	阅读指南文件
- vue.config.js	vue项目的配置文件，包括跨域访问、项目打包设置、其它项目特征设置等

vue-admin-template 目录结构说明

1. 安装 `v-charts` 插件。在 VSCode 中打一个命令行终端窗口，运行如下命令：

```
npm i v-charts echarts -S
```

注意：默认下载的echart为最新版本（可能产生版本兼容问题），建议修改package.json文件中的 "echarts": "^4.7.0"

2. 修改路由文件 `/src/router/index.js`，去掉多余的路由，添加新的路由。

```
import Vue from "vue";
import Router from "vue-router"; // 引入插件

Vue.use(Router); // 安装插件

/* Layout */
import Layout from "@layout";

/**
```

```

* Note: sub-menu only appear when route children.length >= 1
* Detail see: https://panjiachen.github.io/vue-element-admin-site/guide/essentials/router-and-nav.html
*
* hidden: true                if set true, item will not show in the
sidebar(default is false)
* alwaysShow: true           if set true, will always show the root menu
*                             if not set alwaysShow, when item has more than
one children route,
*                             it will becomes nested mode, otherwise not show
the root menu
* redirect: noRedirect        if set noRedirect will no redirect in the
breadcrumb
* name: 'router-name'         the name is used by <keep-alive> (must set!!!)
* meta : {
  roles: ['admin','editor']   control the page roles (you can set multiple
roles)
  title: 'title'              the name show in sidebar and breadcrumb
(recommend set)
  icon: 'svg-name'            the icon show in the sidebar
  breadcrumb: false           if set false, the item will hidden in
breadcrumb(default is true)
  activeMenu: '/example/list' if set path, the sidebar will highlight the path
you set
}
*/

/**
* constantRoutes
* a base page that does not have permission requirements
* all roles can be accessed
* 路由表
*/
export const constantRoutes = [
  {
    path: "/login",
    component: () => import("@/views/login/index"),
    hidden: true,
  },

  {
    path: "/404",
    component: () => import("@/views/404"),
    hidden: true,
  },

  {
    path: "/",
    component: Layout,
    redirect: "/dashboard",
    children: [
      {
        path: "dashboard",
        name: "Dashboard",

```

```

        component: () => import("@/views/dashboard/index"),
        meta: { title: "欢迎界面", icon: "dashboard" },
    },
],
},

// {
//   path: '/example',
//   component: Layout,
//   redirect: '/example/table',
//   name: 'Example',
//   meta: { title: 'Example', icon: 'example' },
//   children: [
//     {
//       path: 'table',
//       name: 'Table',
//       component: () => import('@/views/table/index'),
//       meta: { title: 'Table', icon: 'table' }
//     },
//     {
//       path: 'tree',
//       name: 'Tree',
//       component: () => import('@/views/tree/index'),
//       meta: { title: 'Tree', icon: 'tree' }
//     }
//   ]
// },

// {
//   path: '/form',
//   component: Layout,
//   children: [
//     {
//       path: 'index',
//       name: 'Form',
//       component: () => import('@/views/form/index'),
//       meta: { title: 'Form', icon: 'form' }
//     }
//   ]
// },

// {
//   path: '/nested',
//   component: Layout,
//   redirect: '/nested/menu1',
//   name: 'Nested',
//   meta: {
//     title: 'Nested',
//     icon: 'nested'
//   },
//   children: [
//     {
//       path: 'menu1',
//       component: () => import('@/views/nested/menu1/index'), // Parent

```



```

router-view
//      name: 'Menu1',
//      meta: { title: 'Menu1' },
//      children: [
//          {
//              path: 'menu1-1',
//              component: () => import('@/views/nested/menu1/menu1-1'),
//              name: 'Menu1-1',
//              meta: { title: 'Menu1-1' }
//          },
//          {
//              path: 'menu1-2',
//              component: () => import('@/views/nested/menu1/menu1-2'),
//              name: 'Menu1-2',
//              meta: { title: 'Menu1-2' },
//              children: [
//                  {
//                      path: 'menu1-2-1',
//                      component: () => import('@/views/nested/menu1/menu1-2/menu1-2-
1'),
//                      name: 'Menu1-2-1',
//                      meta: { title: 'Menu1-2-1' }
//                  },
//                  {
//                      path: 'menu1-2-2',
//                      component: () => import('@/views/nested/menu1/menu1-2/menu1-2-
2'),
//                      name: 'Menu1-2-2',
//                      meta: { title: 'Menu1-2-2' }
//                  }
//              ]
//          },
//          {
//              path: 'menu1-3',
//              component: () => import('@/views/nested/menu1/menu1-3'),
//              name: 'Menu1-3',
//              meta: { title: 'Menu1-3' }
//          }
//      ]
//  },
//  {
//      path: 'menu2',
//      component: () => import('@/views/nested/menu2/index'),
//      meta: { title: 'menu2' }
//  }
//  ]
// },

// {
//     path: 'external-link',
//     component: Layout,
//     children: [
//         {
//             path: 'https://panjiachen.github.io/vue-element-admin-site/#/',

```

```

//      meta: { title: 'External Link', icon: 'link' }
//    }
//  ]
// },
{
  path: '/research',
  component: Layout,
  redirect: '/research/respage01',
  name: 'Research',
  alwaysShow: true,
  meta: { title: '流量统计', icon: 'form' },
  children: [
    {
      path: 'pnametraffics',
      name: 'Pnametraffics',
      component: () => import('@views/pnametraffics/index'),
      meta: { title: '按产品统计', icon: 'Bar chart' }
    },
    {
      path: 'provincetraffics',
      name: 'provincetraffics',
      component: () => import('@views/provincetraffics/index'),
      meta: { title: '按省份统计', icon: 'Bar chart' }
    },
    {
      path: 'spnametraffics',
      name: 'spnametraffics',
      component: () => import('@views/spnametraffics/index'),
      meta: { title: '按用户统计', icon: 'Bar chart' }
    },
    {
      path: 'tel',
      name: 'tel',
      component: () => import('@views/tel/index'),
      meta: { title: '按手机号查询明细', icon: 'table' }
    }
  ]
},
// 404 page must be placed at the end !!!
{ path: "*", redirect: "/404", hidden: true },
];

// 初始化路由
const createRouter = () =>
  new Router({
    // mode: 'history', // require service support
    scrollBehavior: () => ({ y: 0 }),
    routes: constantRoutes,
  });
// 得到路由
const router = createRouter();

// 重置路由
// Detail see: https://github.com/vuejs/vue-router/issues/1234#issuecomment-
```

```

357941465
export function resetRouter() {
  const newRouter = createRouter();
  router.matcher = newRouter.matcher; // reset router
}

// 导出路由
export default router;

```

3. 在 /src/views/ 目录下创建 4 个视图文件夹，并在每个文件夹中创建一个视图文件，分别为 pnametraffics/index.vue、provincetraffics/index.vue、spnametraffics/index.vue、tel/index.vue。

/src/views/pnametraffics/index.vue 的代码：

```

<template>
  <div class="app-container">
    <el-form ref="form" :model="form" :rules="dateRules" label-width="120px">
      <el-form-item label="选择时间">
        <el-col :span="11">
          <el-form-item prop="date1">
            <el-date-picker
              ref="date1"
              v-model="form.date1"
              name="date1"
              :editable="false"
              type="date"
              placeholder="开始"
              style="width: 100%;"
              value-format="yyyyMMdd"
            />
          </el-form-item>
        </el-col>
        <el-col :span="2" class="line"></el-col>
        <el-col :span="11">
          <el-form-item prop="date2">
            <el-date-picker
              ref="date2"
              v-model="form.date2"
              name="date2"
              :editable="false"
              type="date"
              placeholder="结束(可选)"
              style="width:100%;"
              value-format="yyyyMMdd"
            />
          </el-form-item>
        </el-col>
      </el-form-item>
      <el-form-item>
        <!-- <el-button type="primary" @click="onSubmit">查询</el-button> -->
        <el-button type="primary" @click.native.prevent="onSubmit">查询</el-

```

```

button>
  <el-button @click="onCancel">取消</el-button>
</el-form-item>
</el-form>
<ve-histogram :data="chartData" :extend="vchartsConfig.extend" />
</div>
</template>

<script>
import VeHistogram from 'v-charts/lib/histogram'
import { getXxx } from '@api/pnametraffics'
export default {
  components: { VeHistogram },
  data() {
    const validateDate1 = (rule, value, callback) => {
      if (!value) {
        callback(new Error('请选择开始时间'))
      } else {
        callback()
      }
    }
  }
  const validateDate2 = (rule, value, callback) => {
    if (this.form.date1 && value) {
      if (this.form.date1 > value) {
        callback(new Error('结束时间不能小于开始时间，请重新选择'))
      } else {
        callback()
      }
    } else {
      callback()
    }
  }
  return {
    // 表单验证
    dateRules: {
      date1: [
        { required: true, trigger: 'blur', validator: validateDate1 }
      ],
      date2: [{ validator: validateDate2 }]
    },
    // 表单相关内容
    form: {
      name: '',
      region: '',
      date1: '',
      date2: '',
      delivery: false,
      type: [],
      resource: '',
      desc: ''
    },
    // v-charts相关内容
    chartData: {
      columns: ['pname', '流量值(单位：兆)'],

```

```

        rows: []
      },
      vchartsConfig: {
        extend: {
          legend: {},
          // 柱形区域
          grid: {
            show: true,
            backgroundColor: '#FFF6F3'
          },
          // 柱子
          series(v) {
            v.forEach((i) => {
              // i.barWidth = 10 // 某个柱子的宽度，默认为自适应
              // i.barGap = 0
              // i.barCategoryGap = 30 // 同一类目之间的间距，若一个类目下存在多个柱子，可以使用barGap设置间距
              // 柱条最小高度，可用于防止某数据项的值过小而影响交互
              i.barMinHeight = 2
              i.itemStyle = {}
              i.label = {}
            })
            return v
          },
          // x轴的样式
          xAxis: {
            axisLabel: {
              margin: 5,
              interval: 0,
              rotate: 25
            }
          },
          yAxis: {
            axisLabel: {}
          }
        }
      }
    },
    mounted() {
      console.log('xxx')
    },
    methods: {
      onSubmit() {
        this.$refs.form.validate((valid) => {
          if (valid) {
            this.$message('已提交!')
            // getXxx().then(function(msg) {
            //   console.log(msg.data)
            // }).catch(function(error) {
            var b = this.form.date1
            var e = this.form.date2
            getXxx(b, e).then((response) => {
              this.chartData.rows = []
            })
          }
        })
      }
    }
  }
}

```

```

        var dataShow = response.data // 清空上次显示的数据，避免数据合并
        console.log(dataShow)
        for (var i = 0; i < dataShow.length; i++) {
            this.chartData.rows.push({
                pname: dataShow[i].pname,
                '流量值(单位：兆)': dataShow[i].traffics
            })
        }
    }).catch(function(error) {
        console.log(error)
    })
} else {
    console.log('error submit!!')
    return false
}
}
})
},
onCancel() {
    this.$message({
        message: '已取消!',
        type: 'warning'
    })
}
}
}
</script>
<style scoped>
    .line {
        text-align: center;
    }
</style>

```

/src/views/provincetraffics/index.vue 的代码：

```

<template>
  <div class="app-container">
    <el-form ref="form" :model="form" :rules="dateRules" label-width="120px">
      <el-form-item label="选择时间">
        <el-col :span="11">
          <el-form-item prop="date1">
            <el-date-picker
              ref="date1"
              v-model="form.date1"
              name="date1"
              :editable="false"
              type="date"
              placeholder="开始"
              style="width: 100%;"
              value-format="yyyyMMdd"
            />
          </el-form-item>

```

```

    </el-col>
    <el-col :span="2" class="line"></el-col>
    <el-col :span="11">
      <el-form-item prop="date2">
        <el-date-picker
          ref="date2"
          v-model="form.date2"
          name="date2"
          :editable="false"
          type="date"
          placeholder="结束(可选)"
          style="width:100%;"
          value-format="yyyyMMdd"
        />
      </el-form-item>
    </el-col>
  </el-form-item>
  <el-form-item>
    <el-button type="primary" @click.native.prevent="onSubmit">查询</el-
button>
    <el-button @click="onCancel">取消</el-button>
  </el-form-item>
</el-form>
<ve-histogram :data="chartData" :extend="vchartsConfig.extend" />
</div>
</template>

<script>
import VeHistogram from 'v-charts/lib/histogram'
import { getXxx } from '@api/provincetraffics'
export default {
  components: { VeHistogram },
  data() {
    const validateDate1 = (rule, value, callback) => {
      if (!value) {
        callback(new Error('请选择开始时间'))
      } else {
        callback()
      }
    }
  }
  const validateDate2 = (rule, value, callback) => {
    if (this.form.date1 && value) {
      if (this.form.date1 > value) {
        callback(new Error('结束时间不能小于开始时间，请重新选择'))
      } else {
        callback()
      }
    } else {
      callback()
    }
  }
  return {
    // 表单验证
    dateRules: {

```

```

    date1: [
      { required: true, trigger: 'blur', validator: validateDate1 }
    ],
    date2: [{ validator: validateDate2 }]
  },
  // 表单相关内容
  form: {
    name: '',
    region: '',
    date1: '',
    date2: '',
    delivery: false,
    type: [],
    resource: '',
    desc: ''
  },
  // v-charts相关内容
  chartData: {
    columns: ['province', '流量值(单位: 兆)'],
    rows: []
  },
  vchartsConfig: {
    extend: {
      // 柱形区域
      grid: {
        show: true,
        backgroundColor: '#FFF6F3'
      },
      // 柱子
      series(v) {
        v.forEach((i) => {
          i.barWidth = 10
          i.barGap = 1
          // 柱条最小高度, 可用于防止某数据项的值过小而影响交互
          i.barMinHeight = 2
          i.itemStyle = {}
          i.label = {}
        })
        return v
      },
      // x轴的样式
      xAxis: {
        axisLabel: {
          margin: 5,
          interval: 0,
          rotate: 30
        }
      },
      yAxis: {
        axisLabel: {}
      }
    }
  }
}

```



```

    },
    mounted() {
      console.log('xxx')
    },
    methods: {
      onSubmit() {
        this.$refs.form.validate((valid) => {
          if (valid) {
            this.$message('已提交!')
            var b = this.form.date1
            var e = this.form.date2
            getXxx(b, e).then((response) => {
              this.chartData.rows = []
              var dataShow = response.data // 清空上次显示的数据，避免数据合并
              for (var i = 0; i < dataShow.length; i++) {
                this.chartData.rows.push({
                  province: dataShow[i].province,
                  '流量值(单位:兆)': dataShow[i].traffics
                })
              }
            }).catch(function(error) {
              console.log(error)
            })
          } else {
            console.log('error submit!!')
            return false
          }
        })
      },
      onCancel() {
        this.$message({
          message: '已取消',
          type: 'warning'
        })
      }
    }
  }
</script>
<style scoped>
  .line {
    text-align: center;
  }
</style>

```

/src/views/spnametraffics/index.vue 的代码：

```

<template>
  <div class="app-container">
    <el-form ref="form" :model="form" :rules="dateRules" label-width="120px">
      <el-form-item label="选择时间">
        <el-col :span="11">

```

```

        <el-form-item prop="date1">
          <el-date-picker
            ref="date1"
            v-model="form.date1"
            name="date1"
            :editable="false"
            type="date"
            placeholder="开始"
            style="width: 100%;"
            value-format="yyyyMMdd"
          />
        </el-form-item>
      </el-col>
    <el-col :span="2" class="line">-</el-col>
    <el-col :span="11">
      <el-form-item prop="date2">
        <el-date-picker
          ref="date2"
          v-model="form.date2"
          name="date2"
          :editable="false"
          type="date"
          placeholder="结束(可选)"
          style="width:100%;"
          value-format="yyyyMMdd"
        />
      </el-form-item>
    </el-col>
  </el-form-item>
  <el-form-item>
    <el-button type="primary" @click.native.prevent="onSubmit">查询</el-button>
    <el-button @click="onCancel">取消</el-button>
  </el-form-item>
</el-form>
<ve-histogram :data="chartData" :extend="vchartsConfig.extend" />
</div>
</template>

<script>
import VeHistogram from 'v-charts/lib/histogram'
import { getXxx } from '@api/spnametraffics'
export default {
  components: { VeHistogram },
  data() {
    const validateDate1 = (rule, value, callback) => {
      if (!value) {
        callback(new Error('请选择开始时间'))
      } else {
        callback()
      }
    }
  }
  const validateDate2 = (rule, value, callback) => {
    if (this.form.date1 && value) {

```

```
        if (this.form.date1 > value) {
            callback(new Error('结束时间不能小于开始时间，请重新选择'))
        } else {
            callback()
        }
    } else {
        callback()
    }
}
return {
    // 表单验证
    dateRules: {
        date1: [
            { required: true, trigger: 'blur', validator: validateDate1 }
        ],
        date2: [{ validator: validateDate2 }]
    },
    // 表单相关内容
    form: {
        name: '',
        region: '',
        date1: '',
        date2: '',
        delivery: false,
        type: [],
        resource: '',
        desc: ''
    },
    // v-charts相关内容
    chartData: {
        columns: ['spname', '流量值'],
        rows: []
    },
    vchartsConfig: {
        extend: {
            // 柱形区域
            grid: {
                show: true,
                backgroundColor: '#FFF6F3'
            },
            // 柱子
            series(v) {
                v.forEach((i) => {
                    i.barWidth = 20
                    i.barGap = 1
                    // 柱条最小高度，可用于防止某数据项的值过小而影响交互
                    i.barMinHeight = 2
                    i.itemStyle = {
                        // barBorderRadius: [10, 10, 10, 10],
                        color: '#FF6633'
                        // borderWidth: 0
                    }
                    i.label = {}
                })
            }
        }
    }
}
```

```
        return v
    },
    // x轴的样式
    xAxis: {
        axisLabel: {
            margin: 5,
            interval: 0,
            rotate: 0
        }
    },
    yAxis: {
        axisLabel: {
            formatter: function(value) {
                return value + ' M'
            }
        }
    }
}
},
mounted() {
    console.log('xxx')
},
methods: {
    onSubmit() {
        this.$refs.form.validate((valid) => {
            if (valid) {
                this.$message('已提交')
                var b = this.form.date1
                var e = this.form.date2
                getXxx(b, e).then((response) => {
                    this.chartData.rows = []
                    var dataShow = response.data // 清空上次显示的数据，避免数据合并
                    for (var i = 0; i < dataShow.length; i++) {
                        this.chartData.rows.push({
                            spname: dataShow[i].spname,
                            '流量值': dataShow[i].traffics
                        })
                    }
                }).catch(function(error) {
                    console.log(error)
                })
            } else {
                console.log('error submit!!!')
                return false
            }
        })
    },
    onCancel() {
        this.$message({
            message: '已取消',
            type: 'warning'
        })
    }
}
```

```

    }
  }
}
</script>
<style scoped>
  .line {
    text-align: center;
  }
</style>

```

/src/views/tel/index.vue 的代码：

```

<template>
  <div class="app-container">
    <el-form ref="form" :model="form" :rules="dateRules" label-width="120px">
      <el-form-item label="输入手机号" prop="tn">
        <el-input ref="tn" v-model="form.tn" name="tn" style="width:91%" />
      </el-form-item>
      <el-form-item label="选择时间">
        <el-col :span="11">
          <el-form-item prop="date1">
            <el-date-picker
              ref="date1"
              v-model="form.date1"
              name="date1"
              type="date"
              :editable="false"
              placeholder="开始"
              style="width: 80%;"
              value-format="yyyyMMdd"
            />
          </el-form-item>
        </el-col>
        <el-col :span="2" class="line"></el-col>
        <el-col :span="11">
          <el-form-item prop="date2">
            <el-date-picker
              ref="date2"
              v-model="form.date2"
              name="date2"
              type="date"
              :editable="false"
              placeholder="结束(可选)"
              style="width:80%;"
              value-format="yyyyMMdd"
            />
          </el-form-item>
        </el-col>
      </el-form-item>
      <el-form-item>
        <el-button type="primary" @click.native.prevent="onSubmit">查询</el-

```

```
button>
  <el-button @click="onCancel">取消</el-button>
</el-form-item>
</el-form>
<el-table
  v-loading="listLoading"
  :data="list"
  element-loading-text="Loading"
  border
  fit
  highlight-current-row
>
  <el-table-column
    width="140"
    align="center"
    label="手机号"
    prop="phones:phone"
    :formatter="fone"
  />
  <el-table-column label="产品名" align="center" prop="pids:pname" />
  <el-table-column
    label="客户"
    width="200"
    align="center"
    prop="spids:spname"
  />
  <el-table-column
    label="流量 ( 兆 ) "
    width="140"
    align="center"
    prop="traffics:traffic"
  />
  <el-table-column
    label="访问时间"
    width="200"
    align="center"
    prop="times:time"
    :formatter="ftwo"
  />
  <el-table-column
    align="center"
    label="省份城市"
    width="200"
    prop="phones:province"
    :formatter="fthree"
  />
</el-table>
</div>
</template>

<script>
import { getXxx } from "@api/tel";

export default {
```

```
filters: {
  statusFilter(status) {},
},
data() {
  const validateTn = (rule, value, callback) => {
    if (!value) {
      callback(new Error("请输入手机号"));
    } else if (!/^1[3456789]\d{9}$/.test(value)) {
      callback(new Error("手机号格式不正确"));
    } else {
      callback();
    }
  };
  const validateDate1 = (rule, value, callback) => {
    if (!value) {
      callback(new Error("请选择开始时间"));
    } else {
      callback();
    }
  };
  const validateDate2 = (rule, value, callback) => {
    if (this.form.date1 && value) {
      if (this.form.date1 > value) {
        callback(new Error("结束时间不能小于开始时间，请重新选择"));
      } else {
        callback();
      }
    } else {
      callback();
    }
  };
  return {
    // 表单验证
    dateRules: {
      tn: [{ required: true, trigger: "blur", validator: validateTn }],
      date1: [
        { required: true, trigger: "blur", validator: validateDate1 },
      ],
      date2: [{ validator: validateDate2 }],
    },
    // 表单相关内容
    form: {
      tn: "",
      date1: "",
      date2: "",
    },
    // 表格相关内容
    list: null,
    listLoading: false,
  };
},
methods: {
  fone(row, column, cellValue, index) {
    const msg = row[column.property];
  }
}
```

```
    if (msg === undefined) {
      return "";
    }
    return msg.indexOf("-") > 0 ? msg.substring(0, msg.indexOf("-")) : msg;
  },
  ftwo(row, column, cellValue, index) {
    const msg = row[column.property];
    if (msg === undefined) {
      return "";
    }
    console.log(msg);
    const y = msg.substr(0, 4);
    const m = msg.substr(4, 2);
    const d = msg.substr(6, 2);
    const h = msg.substr(8, 2);
    return y + "年" + m + "月" + d + "日" + h + "时";
  },
  fthree(row, column, cellValue, index) {
    const msg = row[column.property];
    if (msg === undefined) {
      return "";
    }
    return msg.indexOf("-") > 0 ? msg.substring(0, msg.indexOf("-")) : msg;
  },
  onSubmit() {
    this.$refs.form.validate((valid) => {
      if (valid) {
        this.$message("已提交");
        var b = this.form.date1;
        var e = this.form.date2;
        var tn = this.form.tn;

        this.listLoading = true; // 表格数据加载中
        getXxx(b, e, tn)
          .then((response) => {
            console.log(response.data);
            this.list = response.data;
            // console.log(response.data.items)
            // this.list = response.data.items
            this.listLoading = false;
          })
          .catch(function (error) {
            console.log(error);
          });
      } else {
        console.log("error submit!!");
        return false;
      }
    });
  },
  onCancel() {
    this.$message({
      message: "已取消",
      type: "warning",
    });
  }
}
```



```
    });  
  },  
},  
};  
</script>
```

4. 在 /src/api/ 目录下创建 4 个 JS 文件，用于异步访问后端项目的接口，分别为 pnametraffics.js、provincetraffics.js、spnametraffics.js、tel.js。

/src/api/pnametraffics.js 的代码：

```
import request from '@/utils/request'  
  
export function getXxx(b, e) {  
  return request({  
    url: '/api/tpnametraffics?b=' + b + '&e=' + e,  
    method: 'get'  
  })  
}
```

/src/api/provincetraffics.js 的代码：

```
import request from '@/utils/request'  
  
export function getXxx(b, e) {  
  return request({  
    url: '/api/tprovincetraffics?b=' + b + '&e=' + e,  
    method: 'get'  
  })  
}
```

/src/api/spnametraffics.js 的代码：

```
import request from '@/utils/request'  
  
export function getXxx(b, e) {  
  return request({  
    url: '/api/tspnametraffics?b=' + b + '&e=' + e,  
    method: 'get'  
  })  
}
```

/src/api/tel.js 的代码：

```
import request from "@/utils/request";

export function getXxx(b, e, tn) {
  return request({
    url: "/api/tel?b=" + b + "&e=" + e + "&tn=" + tn,
    method: "get",
  });
}
```

5. 设置跨域访问后端项目，在 /vue.config.js 中进行配置，代码如下：

```
"use strict";
const path = require("path");
const defaultSettings = require("../src/settings.js");

function resolve(dir) {
  return path.join(__dirname, dir);
}

const name = defaultSettings.title || "vue Admin Template"; // page title

// If your port is set to 80,
// use administrator privileges to execute the command line.
// For example, Mac: sudo npm run
// You can change the port by the following methods:
// port = 9528 npm run dev OR npm run dev --port = 9528
const port = process.env.port || process.env.npm_config_port || 9528; // dev port

// All configuration item explanations can be find in
https://cli.vuejs.org/config/
module.exports = {
  /**
   * You will need to set publicPath if you plan to deploy your site under a sub
   path,
   * for example GitHub Pages. If you plan to deploy your site to
   https://foo.github.io/bar/,
   * then publicPath should be set to "/bar/".
   * In most cases please use '/' !!!
   * Detail: https://cli.vuejs.org/config/#publicpath
   */
  publicPath: "/",
  outputDir: "dist",
  assetsDir: "static",
  lintOnSave: process.env.NODE_ENV === "development",
  productionSourceMap: false,
  devServer: {
    port: port,
    open: true,
    overlay: {
      warnings: false,

```

```
    errors: true,
  },
  // 主要实现跨域访问某个外网资源
  proxy: {
    // 配置跨域
    '/dev-api/api': {
      target: 'http://localhost:1573/cdn',
      // 这里后台的地址模拟的;应该填写你们真实的后台接口
      ws: true,
      changOrigin: true,
      // 允许跨域
      pathRewrite: {
        '^/dev-api/api': '/'
        // 请求的时候使用这个api就可以
      },
      secure: false
    }
  },
  before: require("./mock/mock-server.js"),
},
configureWebpack: {
  // provide the app's title in webpack's name field, so that
  // it can be accessed in index.html to inject the correct title.
  name: name,
  resolve: {
    alias: {
      "@": resolve("src"),
    },
  },
},
chainWebpack(config) {
  config.plugins.delete("preload"); // TODO: need test
  config.plugins.delete("prefetch"); // TODO: need test

  // set svg-sprite-loader
  config.module.rule("svg").exclude.add(resolve("src/icons")).end();
  config.module
    .rule("icons")
    .test(/\.(svg)$/i)
    .include.add(resolve("src/icons"))
    .end()
    .use("svg-sprite-loader")
    .loader("svg-sprite-loader")
    .options({
      symbolId: "icon-[name]",
    })
    .end();

  // set preserveWhitespace
  config.module
    .rule("vue")
    .use("vue-loader")
    .loader("vue-loader")
    .tap((options) => {
```

```

        options.compilerOptions.preserveWhitespace = true;
        return options;
    })
    .end();

config
// https://webpack.js.org/configuration/devtool/#development
.when(process.env.NODE_ENV === "development", (config) =>
    config.devtool("cheap-source-map")
);

config.when(process.env.NODE_ENV !== "development", (config) => {
    config
        .plugin("ScriptExtHtmlWebpackPlugin")
        .after("html")
        .use("script-ext-html-webpack-plugin", [
            {
                // `runtime` must same as runtimeChunk name. default is `runtime`
                inline: /runtime\..*\..js$/,
            },
        ])
        .end();
    config.optimization.splitChunks({
        chunks: "all",
        cacheGroups: {
            libs: {
                name: "chunk-libs",
                test: /[\\/]node_modules[\\/]/,
                priority: 10,
                chunks: "initial", // only package third parties that are initially
dependent
            },
            elementUI: {
                name: "chunk-elementUI", // split elementUI into a single package
                priority: 20, // the weight needs to be larger than libs and app or it
will be packaged into libs or app
                test: /[\\/]node_modules[\\/]_?element-ui(.*)/, // in order to adapt
to cnpm
            },
            commons: {
                name: "chunk-commons",
                test: resolve("src/components"), // can customize your rules
                minChunks: 3, // minimum common number
                priority: 5,
                reuseExistingChunk: true,
            },
        },
    });
    config.optimization.runtimeChunk("single");
});
},
};
};

```

6. 在项目中添加其它 icon

首先，到图标库下载需要的图标如阿里的<http://iconfont.cn/home/index>；然后，点击下载图标，随后选择以 svg 下载，把下载的.svg 文件复制到项目的 icons/svg 下；最后，在项目的页面中 icon 属性上使用对应的名称即可。

7. 修改登录页面 /src/views/login/index.vue，代码如下：

```
<template>
  <div class="login-container">
    <el-form
      ref="loginForm"
      :model="loginForm"
      :rules="loginRules"
      class="login-form"
      auto-complete="on"
      label-position="left"
    >
      <div class="title-container">
        <h3 class="title">CDN大数据分析系统</h3>
      </div>

      <el-form-item prop="username">
        <span class="svg-container">
          <svg-icon icon-class="user" />
        </span>
        <el-input
          ref="username"
          v-model="loginForm.username"
          placeholder="Username"
          name="username"
          type="text"
          tabindex="1"
          auto-complete="on"
        />
      </el-form-item>

      <el-form-item prop="password">
        <span class="svg-container">
          <svg-icon icon-class="password" />
        </span>
        <el-input
          :key="passwordType"
          ref="password"
          v-model="loginForm.password"
          :type="passwordType"
          placeholder="Password"
          name="password"
          tabindex="2"
          auto-complete="on"
          @keyup.enter.native="handleLogin"
        />
      </el-form-item>
    </div>
  </div>
```

70 / 81

```

    ]
  },
  loading: false,
  passwordType: 'password',
  redirect: undefined
}
},
watch: {
  $route: {
    handler: function(route) {
      this.redirect = route.query && route.query.redirect
    },
    immediate: true
  }
},
methods: {
  showPwd() {
    if (this.passwordType === 'password') {
      this.passwordType = ''
    } else {
      this.passwordType = 'password'
    }
    this.$nextTick(() => {
      this.$refs.password.focus()
    })
  },
  handleLogin() {
    this.$refs.loginForm.validate((valid) => {
      if (valid) {
        this.loading = true
        this.$store.dispatch('user/login', this.loginForm).then(() => {
          this.$router.push({ path: this.redirect || '/' })
          this.loading = false
        }).catch(() => {
          this.loading = false
        })
      } else {
        console.log('error submit!!')
        return false
      }
    })
  }
}
}
</script>

<style lang="scss">
/* 修复input 背景不协调 和光标变色 */
/* Detail see https://github.com/PanJiaChen/vue-element-admin/pull/927 */

$bg: #283443;
$light_gray: #fff;
$cursor: #fff;

```

```
@supports (-webkit-mask: none) and (not (cater-color: $cursor)) {
  .login-container .el-input input {
    color: $cursor;
  }
}

/* reset element-ui css */
.login-container {
  .el-input {
    display: inline-block;
    height: 47px;
    width: 85%;

    input {
      background: transparent;
      border: 0px;
      -webkit-appearance: none;
      border-radius: 0px;
      padding: 12px 5px 12px 15px;
      color: $light_gray;
      height: 47px;
      caret-color: $cursor;

      &:-webkit-autofill {
        box-shadow: 0 0 0px 1000px $bg inset !important;
        -webkit-text-fill-color: $cursor !important;
      }
    }
  }
}

.el-form-item {
  border: 1px solid rgba(255, 255, 255, 0.1);
  background: rgba(0, 0, 0, 0.1);
  border-radius: 5px;
  color: #454545;
}
}
</style>

<style lang="scss" scoped>
  $bg: #2d3a4b;
  $dark_gray: #889aa4;
  $light_gray: #eee;

  .login-container {
    min-height: 100%;
    width: 100%;
    background-color: $bg;
    overflow: hidden;

    .login-form {
      position: relative;
      width: 520px;
      max-width: 100%;
    }
  }
}
```



```
padding: 160px 35px 0;  
margin: 0 auto;  
overflow: hidden;  
}  
  
.tips {  
font-size: 14px;  
color: #fff;  
margin-bottom: 10px;  
  
span {  
  &:first-of-type {  
    margin-right: 16px;  
  }  
}  
}  
  
.svg-container {  
padding: 6px 5px 6px 15px;  
color: $dark_gray;  
vertical-align: middle;  
width: 30px;  
display: inline-block;  
}  
  
.title-container {  
position: relative;  
  
.title {  
font-size: 30px;  
color: $light_gray;  
margin: 0px auto 40px auto;  
text-align: center;  
font-weight: bold;  
}  
}  
  
.show-pwd {  
position: absolute;  
right: 10px;  
top: 7px;  
font-size: 16px;  
color: $dark_gray;  
cursor: pointer;  
user-select: none;  
}  
}  
</style>
```

8. 修改欢迎页面 /src/views/dashboard/index.vue · 代码如下：

74 / 81

```

    &-text {
      font-size: 30px;
      line-height: 46px;
    }
  }
</style>

```

注意：必须在 `/src/assets/` 下创建目录 **welcome**，并在该目录下创建 5 个图片 **1.png**、**2.png**、**3.png**、**4.png**、**5.png**

9. 修改退出页面 `/src/layout/components/Navbar.vue`，代码如下：

```

<template>
  <div class="navbar">
    <hamburger
      :is-active="sidebar.opened"
      class="hamburger-container"
      @toggleClick="toggleSideBar"
    />

    <breadcrumb class="breadcrumb-container" />

    <div class="right-menu">
      <el-dropdown class="avatar-container" trigger="click">
        <div class="avatar-wrapper">
          
          <i class="el-icon-caret-bottom" />
        </div>
        <el-dropdown-menu slot="dropdown" class="user-dropdown">
          <!-- <router-link to="/">
            <el-dropdown-item>
              欢迎界面
            </el-dropdown-item>
          </router-link> -->
          <!-- <a target="_blank" href="https://github.com/PanJiaChen/vue-admin-
template/">
            <el-dropdown-item>Github</el-dropdown-item>
          </a> -->
          <a target="_blank" href="http://www.lanqiao.org/">
            <el-dropdown-item>技术支持</el-dropdown-item>
          </a>
          <el-dropdown-item divided @click.native="logout">
            <span style="display:block;">退出</span>
          </el-dropdown-item>
        </el-dropdown-menu>
      </el-dropdown>
    </div>
  </div>
</template>

<script>
import { mapGetters } from 'vuex'

```

```
import Breadcrumb from '@components/Breadcrumb'
import Hamburger from '@components/Hamburger'

export default {
  components: {
    Breadcrumb,
    Hamburger
  },
  computed: {
    ...mapGetters(['sidebar', 'avatar'])
  },
  methods: {
    toggleSideBar() {
      this.$store.dispatch('app/toggleSideBar')
    },
    async logout() {
      await this.$store.dispatch('user/logout')
      this.$router.push(`/login?redirect=${this.$route.fullPath}`)
    }
  }
}
</script>
```

```
<style lang="scss" scoped>
.navbar {
  height: 50px;
  overflow: hidden;
  position: relative;
  background: #fff;
  box-shadow: 0 1px 4px rgba(0, 21, 41, 0.08);

  .hamburger-container {
    line-height: 46px;
    height: 100%;
    float: left;
    cursor: pointer;
    transition: background 0.3s;
    -webkit-tap-highlight-color: transparent;

    &:hover {
      background: rgba(0, 0, 0, 0.025);
    }
  }

  .breadcrumb-container {
    float: left;
  }

  .right-menu {
    float: right;
    height: 100%;
    line-height: 50px;

    &:focus {
```

```
        outline: none;
    }

    .right-menu-item {
        display: inline-block;
        padding: 0 8px;
        height: 100%;
        font-size: 18px;
        color: #5a5e66;
        vertical-align: text-bottom;

        &.hover-effect {
            cursor: pointer;
            transition: background 0.3s;

            &:hover {
                background: rgba(0, 0, 0, 0.025);
            }
        }
    }
}

.avatar-container {
    margin-right: 30px;

    .avatar-wrapper {
        margin-top: 5px;
        position: relative;

        .user-avatar {
            cursor: pointer;
            width: 40px;
            height: 40px;
            border-radius: 10px;
        }

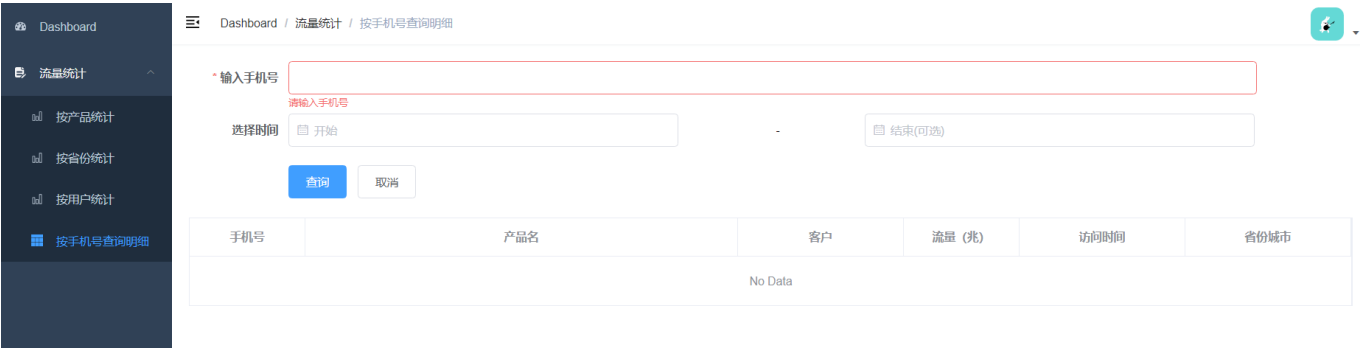
        .el-icon-caret-bottom {
            cursor: pointer;
            position: absolute;
            right: -20px;
            top: 25px;
            font-size: 12px;
        }
    }
}
}
}
}
</style>
```

前端项目的部署运行

编译项目，在 VSCode 中打一个命令行终端窗口，运行如下命令：

```
npm run dev
```

自动打开浏览器窗口，并显示如下界面：



到目前为止，我们使用 npm 对 Vue 项目进行编译部署，如果使用第三方服务器部署 Vue 项目，可以先使用 VSCode 把 Vue 项目打包到 dist 目录，使用如下命令：

```
npm run build:prod
```

在 Vue 项目根目录下生成 dist 目录，把其中的所有文件发布到第三方服务器上就可以了。

十、前端打包发布

把前端部分发布到 nginx 服务器上

- 1. 把 dist 文件夹，放置在某个磁盘，如 d:\dist
- 2. 修改 nginx 服务器 配置文件 nginx.conf，如下

```
#user nobody;
worker_processes 1;

#error_log logs/error.log;
#error_log logs/error.log notice;
#error_log logs/error.log info;

#pid logs/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include mime.types;
    default_type application/octet-stream;

    #log_format main '$remote_addr - $remote_user [$time_local] "$request" '
    #                '$status $body_bytes_sent "$http_referer" '
    #                '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log logs/access.log main;

    sendfile on;
    #tcp_nopush on;

    #keepalive_timeout 0;
    keepalive_timeout 65;

    #gzip on;

    server {
        listen 8888;
        server_name localhost;

        #charset koi8-r;

        #access_log logs/host.access.log main;

        location / {
            root d:\\dist;
            index index.html index.htm;
        }

        #error_page 404 /404.html;

        # redirect server error pages to the static page /50x.html
        #
        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
            root html;
        }
    }
}
```

```
# proxy the PHP scripts to Apache listening on 127.0.0.1:80
#
#location ~ /\.php$ {
#    proxy_pass    http://127.0.0.1;
#}
# 反向代理 · 支持跨域 ( 访问后端项目接口 )
location /prod-api/api {
    proxy_pass    http://localhost:1573/cdn/;
}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
#
#location ~ /\.php$ {
#    root          html;
#    fastcgi_pass  127.0.0.1:9000;
#    fastcgi_index index.php;
#    fastcgi_param SCRIPT_FILENAME    /scripts$fastcgi_script_name;
#    include       fastcgi_params;
#}

# deny access to .htaccess files, if Apache's document root
# concurs with nginx's one
#
#location ~ /\.ht {
#    deny  all;
#}
}

# another virtual host using mix of IP-, name-, and port-based configuration
#
#server {
#    listen      8000;
#    listen      somename:8080;
#    server_name somename alias another.alias;

#    location / {
#        root    html;
#        index   index.html index.htm;
#    }
#}

# HTTPS server
#
#server {
#    listen      443 ssl;
#    server_name localhost;

#    ssl_certificate      cert.pem;
#    ssl_certificate_key  cert.key;

#    ssl_session_cache    shared:SSL:1m;
#    ssl_session_timeout  5m;
```



```
#    ssl_ciphers  HIGH:!aNULL:!MD5;
#    ssl_prefer_server_ciphers  on;

#    location / {
#        root    html;
#        index  index.html index.htm;
#    }
#}
}
```

3. 运行 nginx.exe 启动 nginx 服务器，在地址栏中访问 <http://localhost:8888>

总结

本手册把 CDN 项目的前端部分开发过程进行了详细的介绍，给出了大部分的核心代码。当然有些代码片断是可以根据需要进行修改的，仅供参考使用。