

谷粒商城

版本：V 1.0

www.atguigu.com

一、课程简介

1、为什么我们要讲电商？

因为就互联网平台来说，电商网站有很多典型的特征：

- 访问量大
- 数据量大
- 涉及的技术多
- 有一定的业务复杂性
- 涉及支付 考虑一定安全性

2、我们能从这个项目中学到什么？

巩固以前知识，学会应用：

要新掌握的知识

需要掌握的解决方案

课前说明:

二、 IntelliJ idea

1 介绍

IDEA 全称 IntelliJ IDEA, 是 java 语言开发的集成环境, IntelliJ 在业界被公认为最好的 java 开发工具之一, 尤其在智能代码助手、代码自动提示、重构、J2EE 支持、各类版本工具(git、svn、github 等)、JUnit、CVS 整合、代码分析、创新的 GUI 设计等方面的功能可以说是超常的。IDEA 是 JetBrains 公司的产品, 这家公司总部位于捷克共和国的首都布拉格, 开发人员以严谨著称的东欧程序员为主。它的旗舰版本还支持 HTML, CSS, PHP, MySQL, Python 等。免费版只支持 Java 等少数语言



比起 Eclipse 的好处:

2 安装

解压就可以。

3

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载, 可访问百度: [尚硅谷官网](#)

方案一:

前提需要将

0.0.0.0 account.jetbrains.com 添加到 hosts 文件中

第二种方式 需要有网络的情况下才能注册成功

且在注册成功的情况下,没有网络只能打开第一次,如果打开多次,有可能会需要重新联网注册

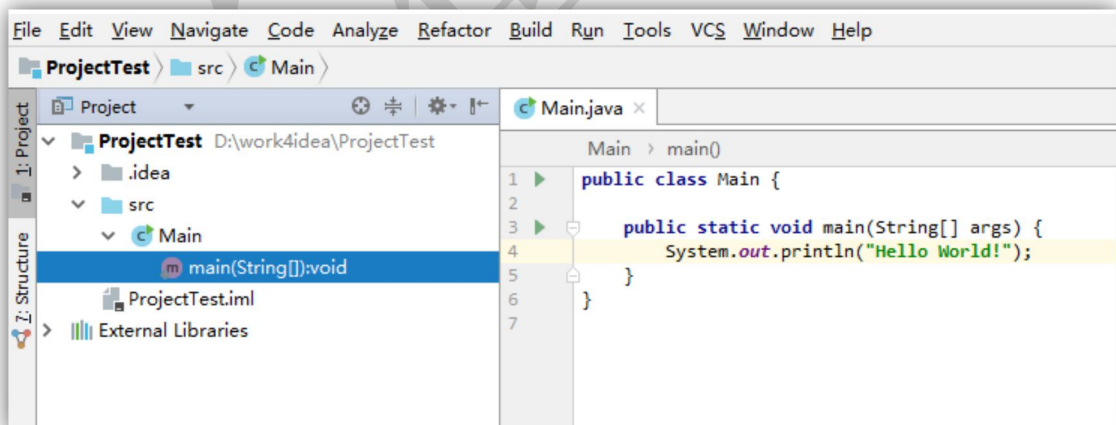
进入 ide 主页面, help-register-license server,然后输入 <http://idea.iteblog.com/key.php>

3 使用

3.1 Project 与 module

在 idea 中没有 workspace 的概念, 每一个窗口只能打开一个 Project。对于单一工程的项目, 直接建一个 Project 在其下面开发就好了。

单一工程的项目:

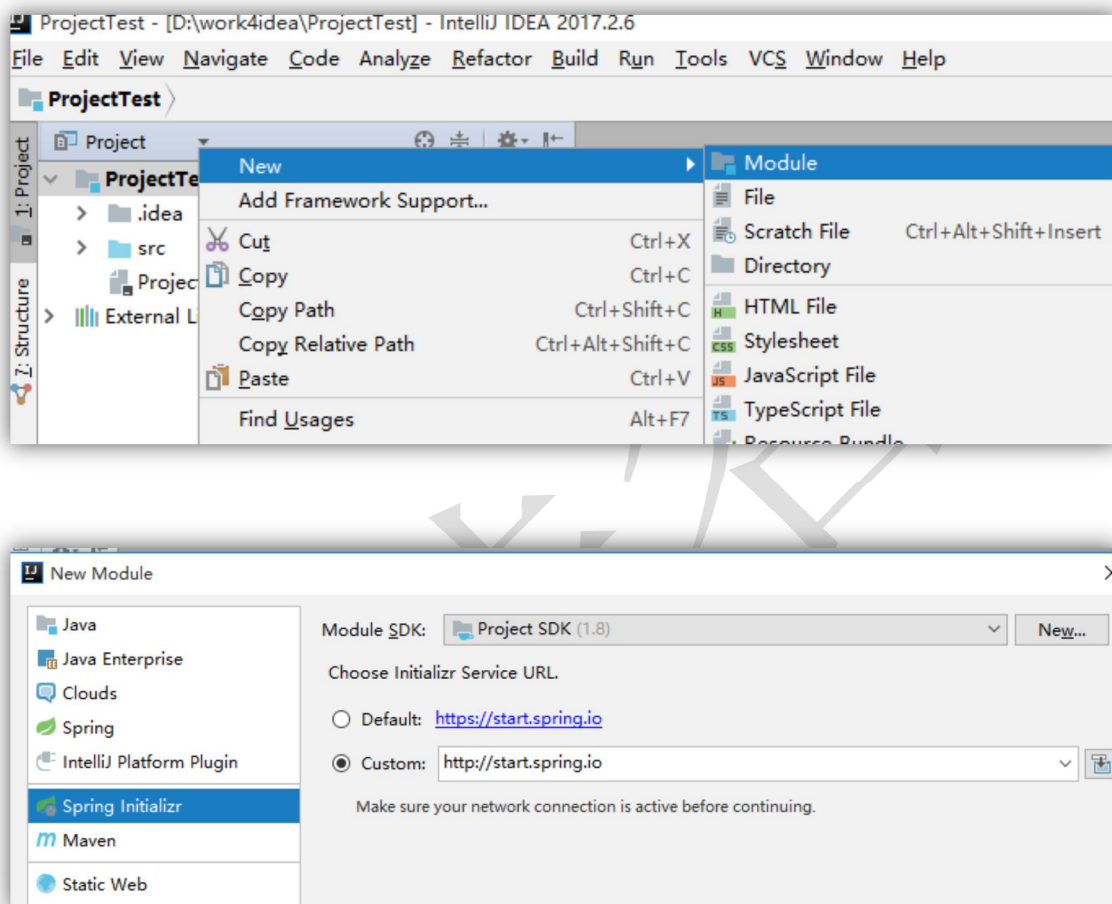


但是我们知道现在稍微大一点的项目都是多项目的分布式部署的,那么岂不是每个子工程都要打开一个窗口?

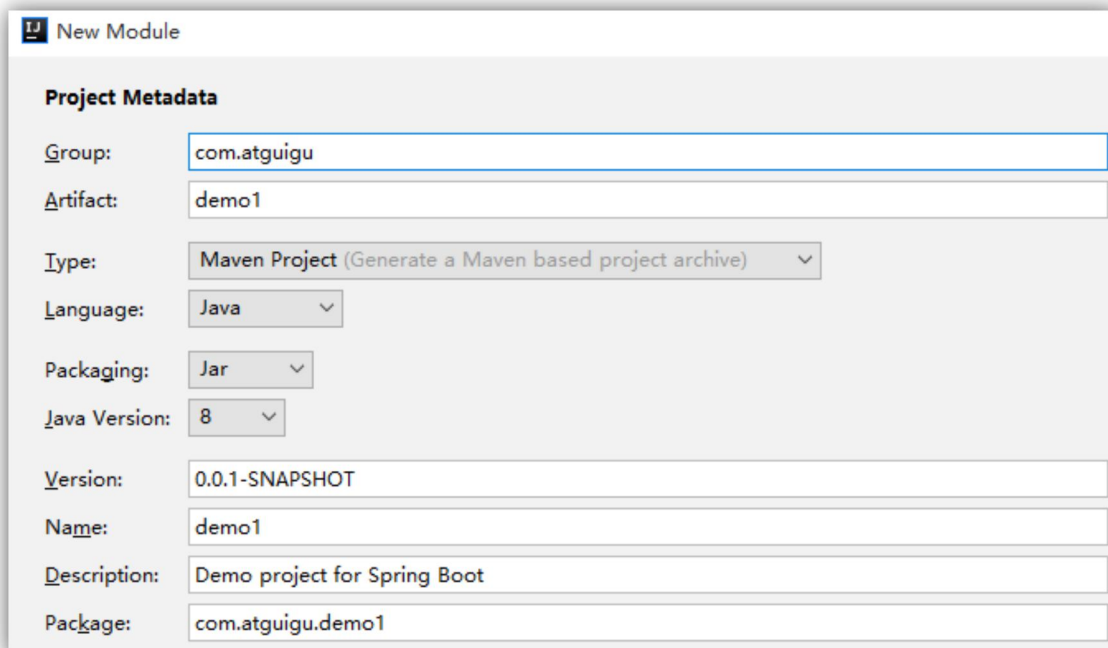
这时候就需要用到 Module 的概念, Module 是项目的子模块,可以独立运行的工程,当一个多项目组成的系统时, Project 下本身可以不拥有代码,而是作为一种顶级的管理目

录，所有的代码都放到各个 module 之中。

下面我们在这个 Project 下增加 Module，



这个时候因为要从网上读取模板所以务必保持联网状态，Spring Initializr 是 springboot 工程的模板。



New Module

Project Metadata

Group:

Artifact:

Type:

Language:

Packaging:

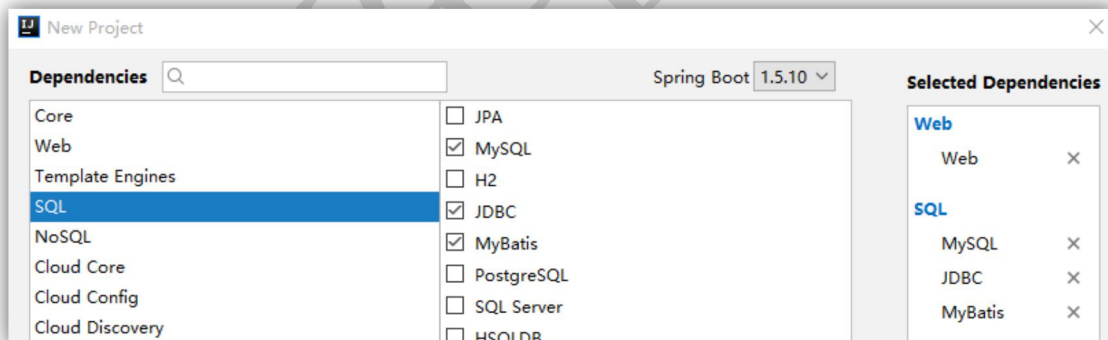
Java Version:

Version:

Name:

Description:

Package:



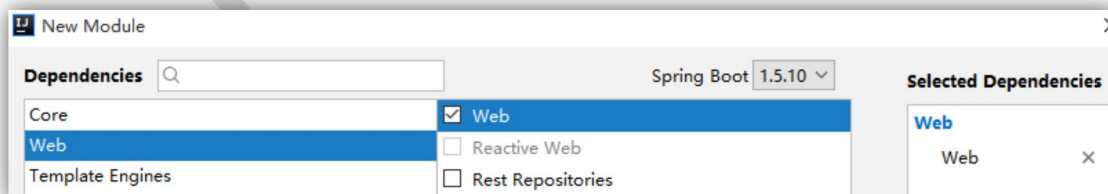
New Project

Dependencies

Spring Boot

Selected Dependencies

Category	Dependency	Action
Web	Web	×
	MySQL	×
SQL	JDBC	×
	MyBatis	×
	MySQL	×



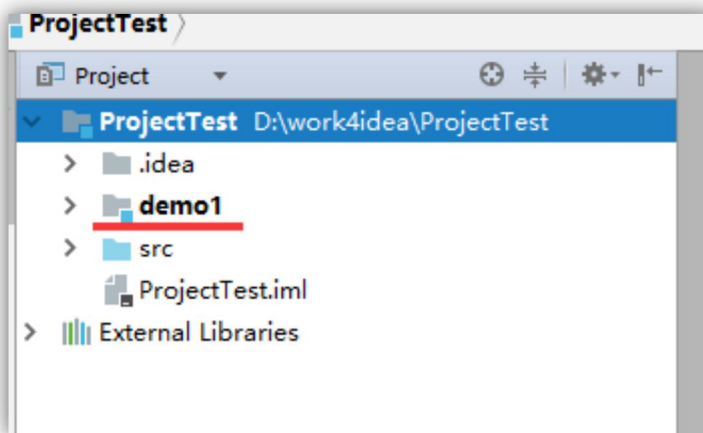
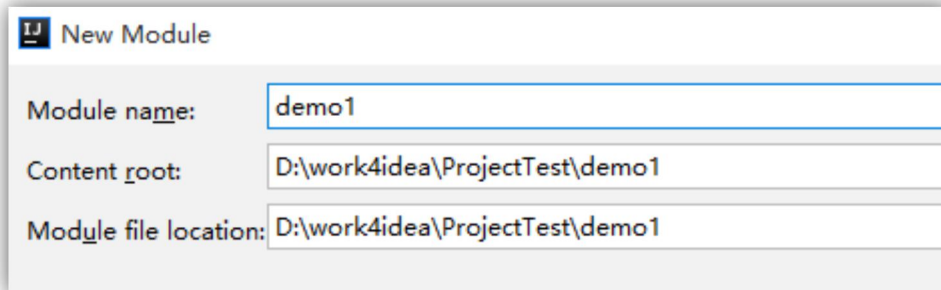
New Module

Dependencies

Spring Boot

Selected Dependencies

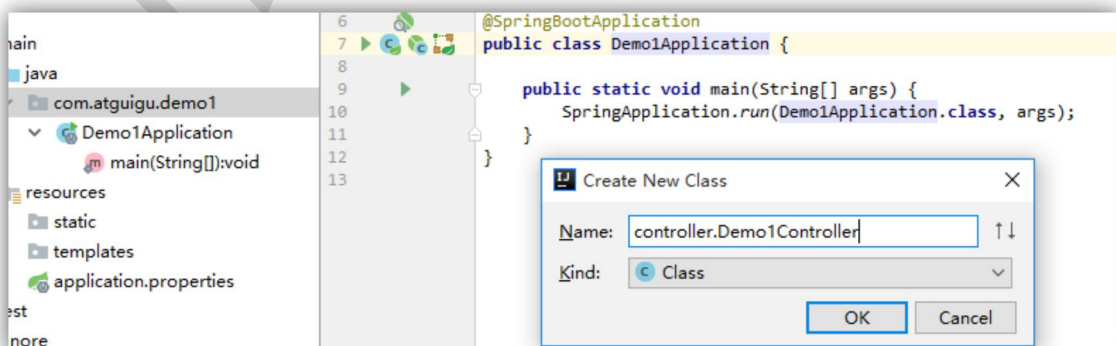
Category	Dependency	Action
Web	Web	×



这时候看到 Project 中多了一个 demo1 的 Module 的。

其实这时候 Project 工程下的 src 就没什么用了，可以删掉。

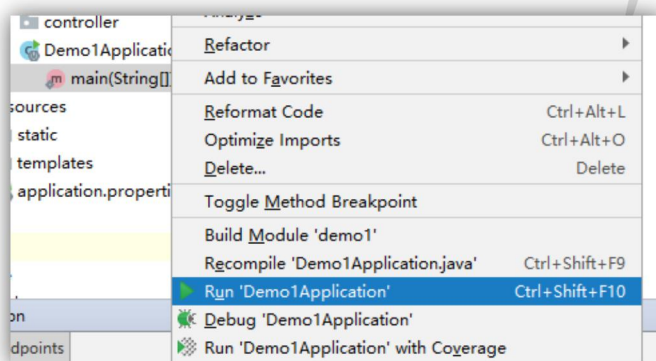
模块建立好了，我们就用 springmvc 标签建一个 controller 看看好不好使。



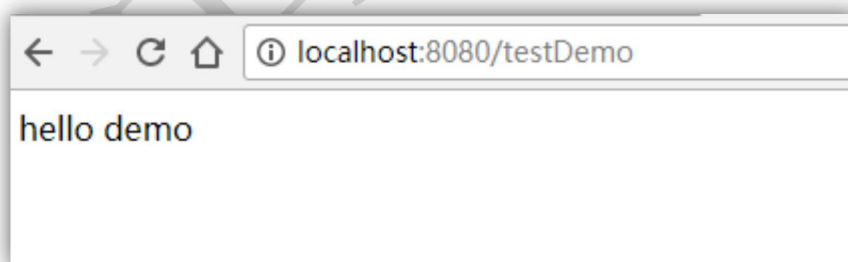
controller 代码

```
@Controller
public class Demo1Controller {
    @ResponseBody
    @RequestMapping("testDemo")
    public String testDemo(){
        return "hello demo";
    }
}
```

运行 Demo1Application 中的 main 方法

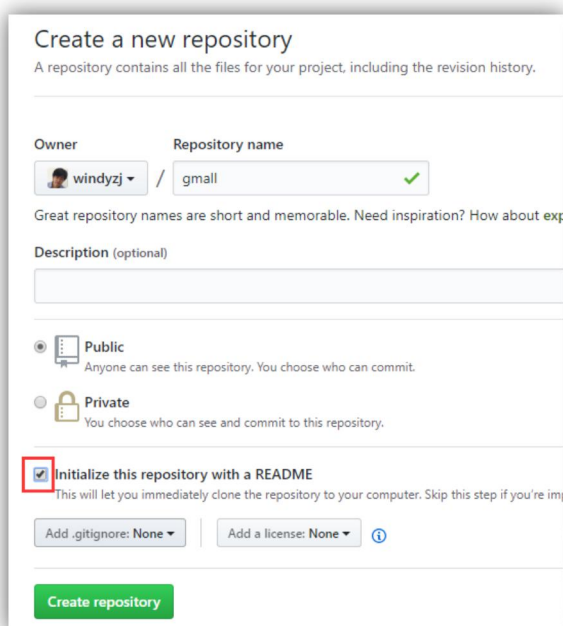


用浏览器测试:

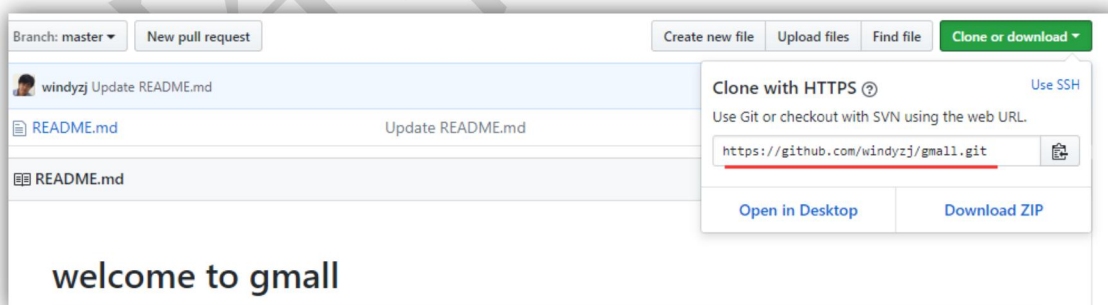


3.2 从 Git 中 clone 项目

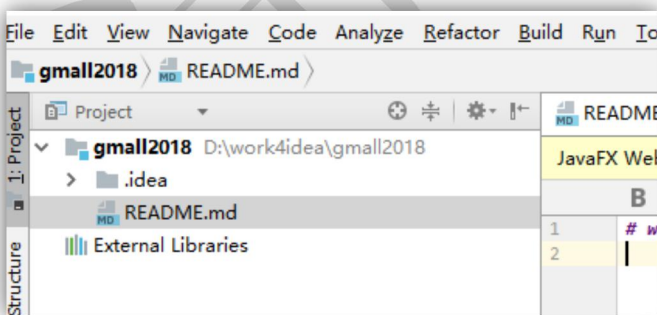
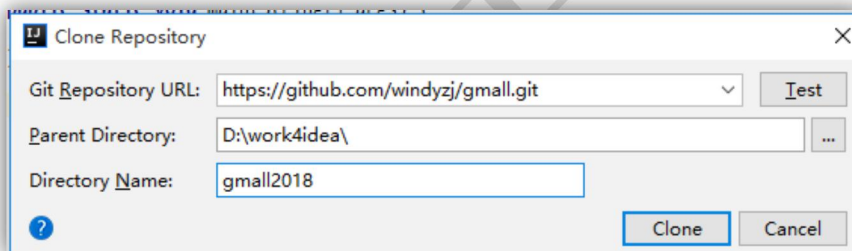
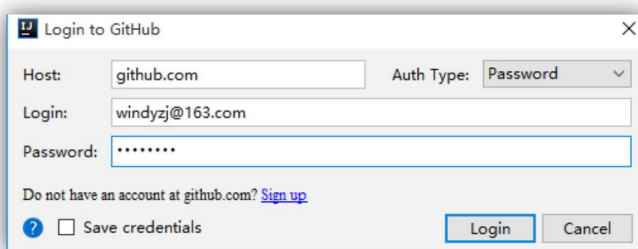
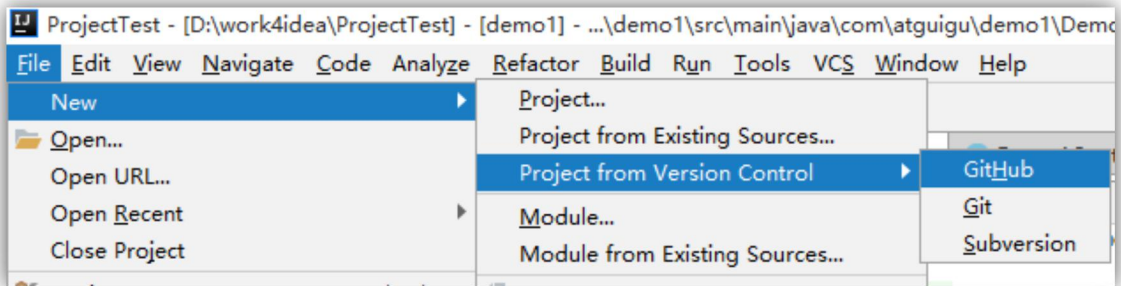
首先要去 GitHub 上创建一个项目



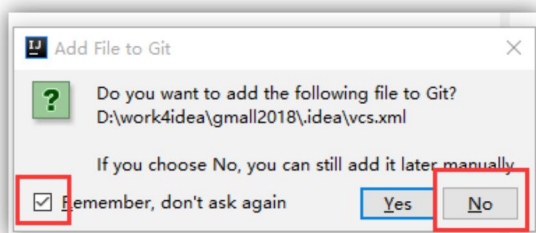
注意的地方是记得加一个 README，这样 clone 下来的工程就不是空的了。



这个就是咱们的仓库地址，咱们来进行第一次复制

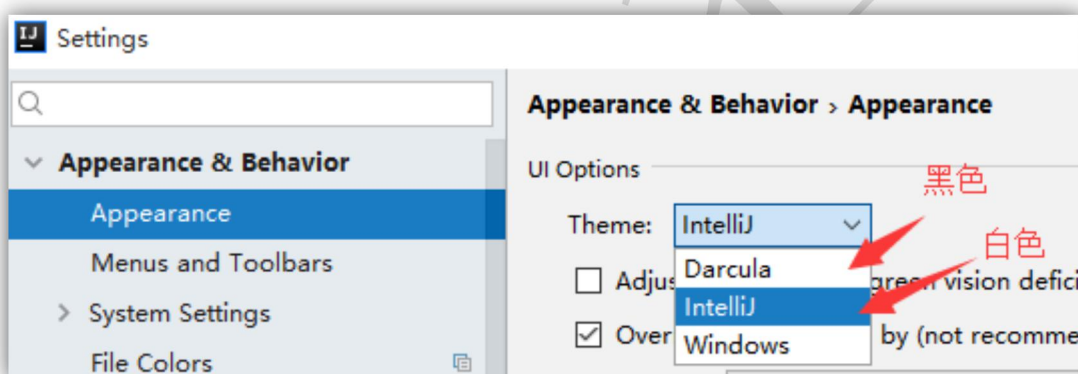


如果弹出提示框如下，问你是否要自动提交某些文件，请一律选 NO,且不再提醒。否则系统会自动提交一些不必要的文件。



4、界面颜色风格

setting->Appearance



5 idea 的快捷键

5.1 常用快捷键

智能补全 引包 alt+Enter

由方法自动生成返回值变量 ctrl+alt+v

跳到方法的实现类 ctrl+alt+b

从实现类跳转到接口 ctrl+u

显示某个接口、抽象类的实现类、子类 ctrl+h

显示最近编辑的文件 ctrl+e

查看方法参数 `ctrl+p`

查看方法文档 `ctrl+q`

复制行 `ctrl+D`

删除行 `ctrl+Y`

跳转到上一个/下一个位置 `ctrl+alt+左右`

大小写切换 `ctrl+shift+u`

5.2 Debug:

F8 执行下一行 (相当于 eclipse 的 F6)

F7 跳入内部 (相当于 eclipse 的 F5)

F9 继续执行 (相当于 eclipse 的 F8)

热部署 `ctrl+shift+F9` (仅 debug 模式)

5.3 搜索

全文搜索文本 `ctrl+shift+f`

全文替换文本 `ctrl+shift+r`

搜索类 `ctrl+n`

任何地方搜索 双击 shift

5.4 快速录入

查看快速录入列表 `ctrl+j`

foreach iter

普通 for 循环 fori

循环数组 itar

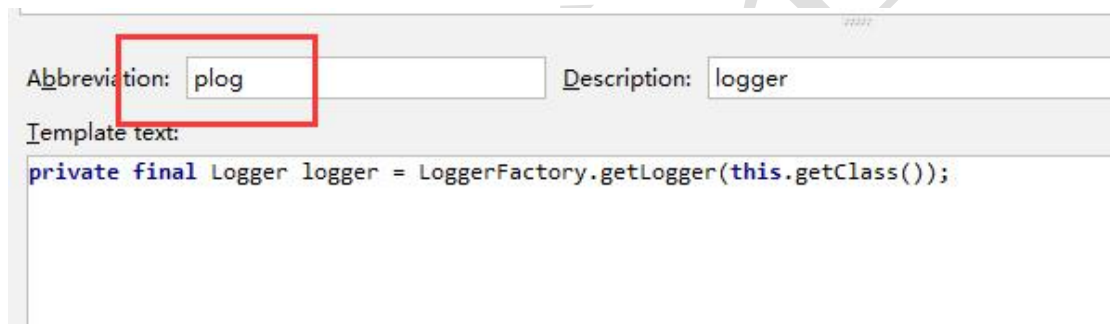
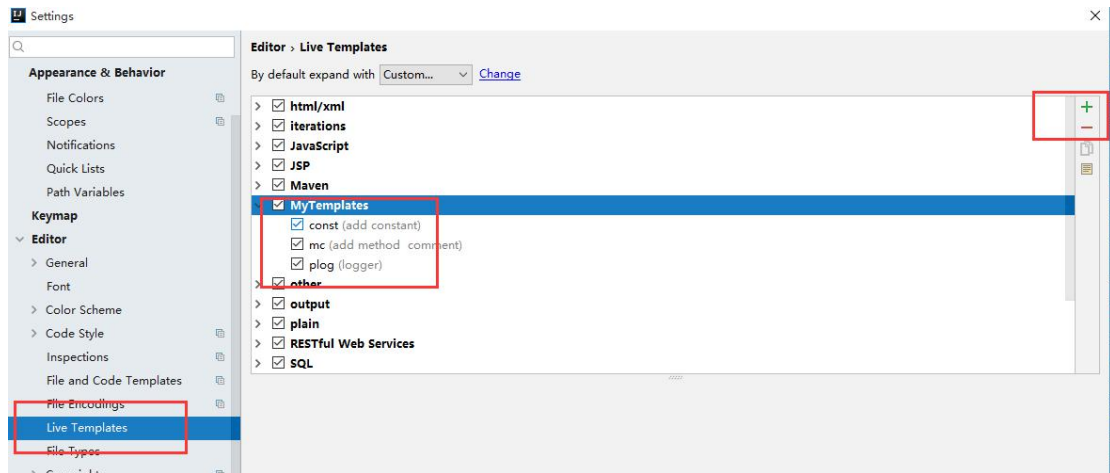
迭代器遍历 itco

psvm 主函数

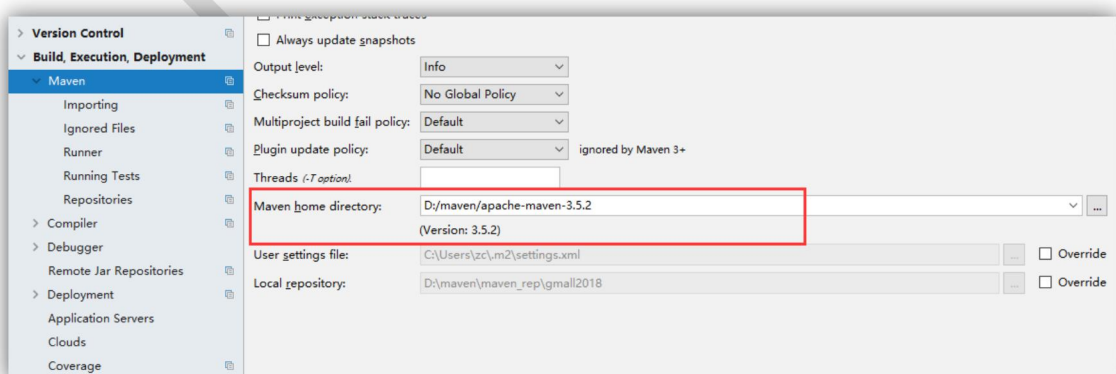
pfs 常量

生成代码块: `try/ if / for/ while/ synchronized`
`ctrl+alt+t`

6 手工加入快捷键模板



7 配置 maven



三、EZDML 工具

配置： 工具—>修改 ini 配置

[DefaultFieldTypes]

[CustFieldTypes]

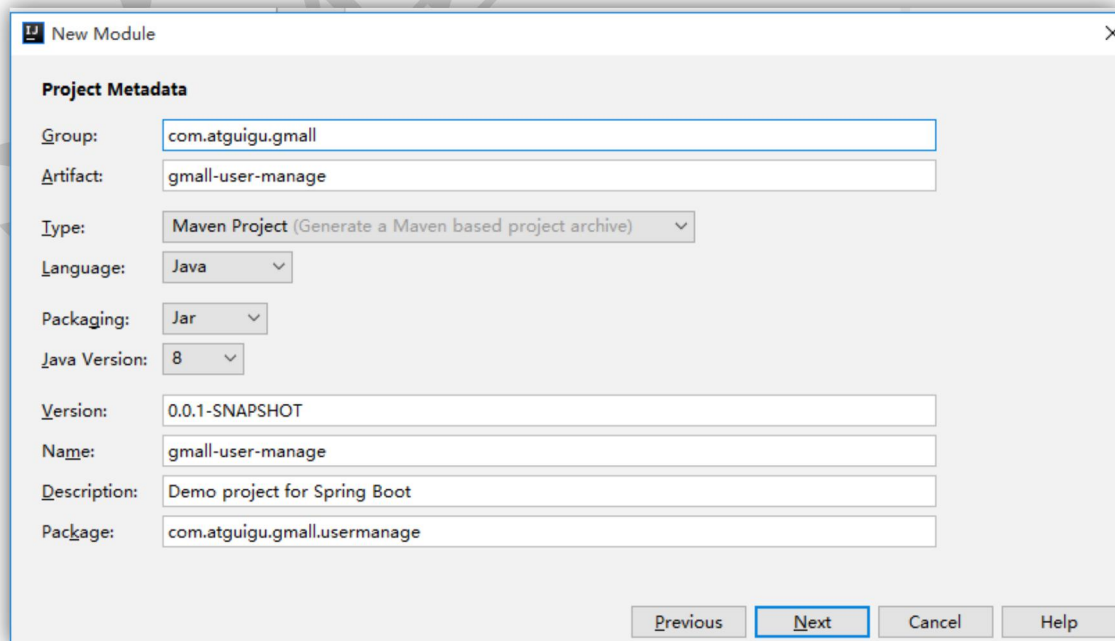
1=bigint

2=decimal

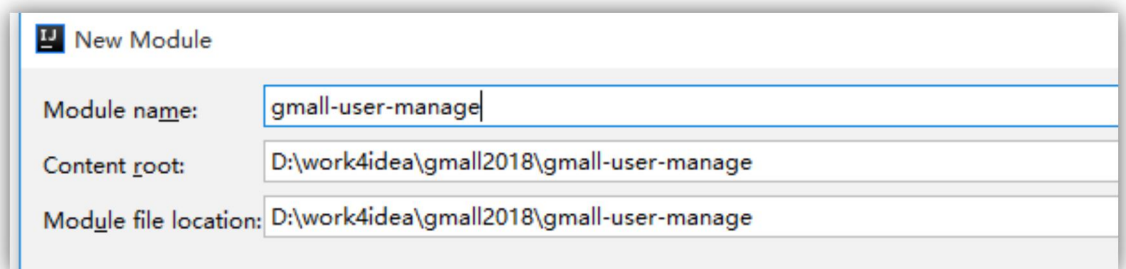
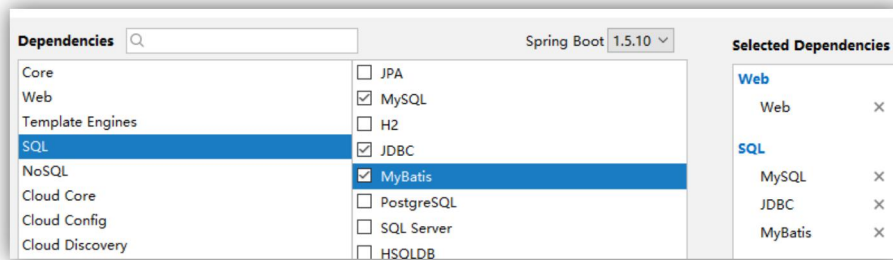
[DbConn]

四 通用 Mapper 的使用

1、搭建 module



依赖选 Web 和 Mysql, Jdbc, MyBatis



注意 Module 位置要在 Project 路径下面

2、配置通用 Mapper

在 pom.xml 文件中，加入

```
<!-- 通用 mapper -->
<dependency>
    <groupId>tk.mybatis</groupId>
    <artifactId>mapper-spring-boot-starter</artifactId>
    <version>1.2.3</version>
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

GmallUserManageApplication.java 中增加注解

```
@SpringBootApplication
```

```
@MapperScan(basePackages = "com.atguigu.gmall.usermanage.mapper")
public class GmallOrderServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(GmallOrderServiceApplication.class, args);
    }
}
```

3、配置数据源

在 application.properties 中

```
spring.datasource.url=jdbc:mysql://localhost:3306/gmall?characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=123123
```

表结构

```
CREATE TABLE `user_info` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '编号',
  `login_name` varchar(200) DEFAULT NULL COMMENT '用户名称',
  `nick_name` varchar(200) DEFAULT NULL COMMENT '用户昵称',
  `passwd` varchar(200) DEFAULT NULL COMMENT '用户密码',
  `name` varchar(200) DEFAULT NULL COMMENT '用户姓名',
  `phone_num` varchar(200) DEFAULT NULL COMMENT '手机号',
  `email` varchar(200) DEFAULT NULL COMMENT '邮箱',
  `head_img` varchar(200) DEFAULT NULL COMMENT '头像',
  `user_level` varchar(200) DEFAULT NULL COMMENT '用户级别',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1000 DEFAULT CHARSET=utf8 COMMENT='用户表'
```


4、代码开发

包	类	说明
controller	UserManageController	web
service	UserManageService	接口
service.impl	UserManageServiceImpl	实现类
bean	UserInfo	实体 bean
mapper	UserInfoMapper	mapper 接口

4.1 bean

```
public class UserInfo implements Serializable{
    @Id
    @Column
    private String id;
    @Column
    private String loginName;
    @Column
    private String nickName;
    @Column
    private String passwd;
    @Column
    private String name;
    @Column
    private String phoneNum;
    @Column
    private String email;
    @Column
    private String headImg;
    @Column
    private String userLevel;
}
```

注意：@Column 和@Id 都是 javax.persistence 包中的
技巧 idea 快捷键：alt+insert 可以快速插入 getter 和 setter

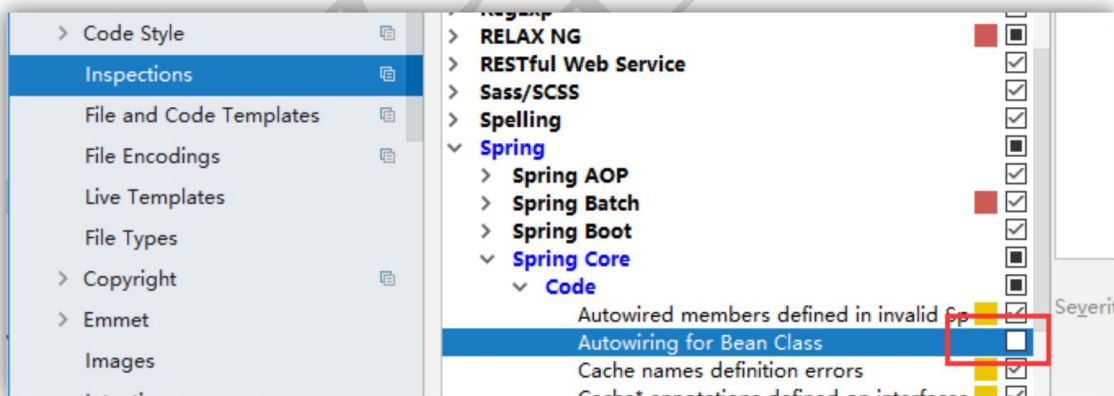
4.2 Mapper

```
public interface UserInfoMapper extends Mapper<UserInfo> {  
}
```

注意：Mapper 也是引用 tk.mybatis.mapper.common.Mapper 包中的

Idea 有的时候校验@Autowired 不准 可以把校验关闭

settings -> Inspections -> spring->spring core -> code-> Autowiring for Bean class



4.4 service

```
public interface UserManagerService {  
  
    public List<UserInfo> getUserInfoList(UserInfo userInfoQuery);  
  
    public UserInfo getUserInfo(UserInfo userInfoQuery);  
  
    public void delete(UserInfo userInfoQuery);  
}
```

```
public void addUserInfo(UserInfo userInfo);

public void updateUserInfo(UserInfo userInfo);

}
```

4.5 ServiceImpl

```
@Service
public class UserManageServiceImpl implements UserManageService {

    @Autowired
    UserInfoMapper userInfoMapper;

    // 查询所有
    public List<UserInfo> getUserInfoList(UserInfo userInfoQuery){
        List<UserInfo> userInfos=null;
        // 查询所有
        //userInfos = userInfoMapper.selectAll();
        // 条件匹配查询
        //userInfos =userInfoMapper.select(userInfoQuery);
        // 特殊条件匹配查询 比如：按姓氏匹配
        Example example=new Example(UserInfo.class);

        example.createCriteria().andLike("loginName", "%"+userInfoQuery.getLoginName()+"%");
        userInfos = userInfoMapper.selectByExample(example);
        return userInfos;
    }

    // 查询单表
    public UserInfo getUserInfo(UserInfo userInfoQuery){
        UserInfo userInfo=null;
        // 按主键查找
        // userInfo = userInfoMapper.selectByPrimaryKey(userInfoQuery.getId());

        // 按所有非空值查询 必须只有一行 否则报错
        userInfo = userInfoMapper.selectOne(userInfoQuery );
        return userInfo;
    }
}
```

```
//增加用户
public void addUserInfo(UserInfo userInfo){
    //会覆盖数据默认值
    userInfoMapper.insert(userInfo);
    // 不会覆盖数据库默认值
    userInfoMapper.insertSelective(userInfo);
}

public void updateUserInfo(UserInfo userInfo){
    //修改用户 依靠主键去查询，然后更新其他值，如果某个值为空，那么原值被清空
    //      userInfoMapper.updateByPrimaryKey(userInfo);
    //修改用户 依靠主键去查询，然后更新其他不为空的值。
    //      userInfoMapper.updateByPrimaryKeySelective(userInfo);

    //修改用户 依靠自定义条件去修改
    Example example=new Example(UserInfo.class);

    example.createCriteria().andLike("loginName", "%"+userInfo.getLoginName()+"%");
    userInfo.setLoginName(null);
    //      userInfoMapper.updateByExample( userInfo,example );
    userInfoMapper.updateByExampleSelective( userInfo,example );
    //
}

public void delete(UserInfo userInfoQuery){
    userInfoMapper.deleteByPrimaryKey(userInfoQuery.getId());
    //按非空值匹配删除
    //      userInfoMapper.delete(userInfoQuery);
    //按条件匹配删除
    //      userInfoMapper.deleteByExample(new Example(UserInfo.class));
}
```

4.6 Controller

```
@RestController
public class UserManagerController {

    @Autowired
    UserManagerService userManagerService;

    @RequestMapping("/users")
    public ResponseEntity<List<UserInfo>> getUserList( UserInfo userInfo){
        List<UserInfo> userInfoList =
userManagerService.getUserInfoList(userInfo);
        return ResponseEntity.ok().body(userInfoList);
    }

    @RequestMapping(value = "/user" ,method = RequestMethod.POST)
    public    ResponseEntity<Void> add(UserInfo userInfo){ ;

        userManagerService.addUserInfo(userInfo);
        return ResponseEntity.ok().build();
    }

    @RequestMapping(value = "/user" ,method = RequestMethod.PUT)
    public    ResponseEntity<Void> update(UserInfo userInfo){
        userManagerService.updateUserInfo(userInfo);
        return ResponseEntity.ok().build();
    }

    @RequestMapping(value = "/user" ,method = RequestMethod.DELETE)
    public    ResponseEntity<Void> delete(UserInfo userInfo){
        userManagerService.delete(userInfo);
        return ResponseEntity.ok().build();
    }

    @RequestMapping(value = "/user" ,method = RequestMethod.GET)
    public    ResponseEntity<UserInfo> getUserInfo(UserInfo userInfoQuery){
        UserInfo userInfo = userManagerService.getUserInfo(userInfoQuery);
        return ResponseEntity.ok().body(userInfo);
    }
}
```

五 hosts 工具



application.properties

```
spring.datasource.url=jdbc:mysql://mysql.server.com:3306/gmall?characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=123123
```