June 24, 2024

邮箱验证功能

配置

```
授权密码：FTGAVZFOTINYROSS
```

```
# 在 application.yml 里的 mail 配置
  mail:
    protocol: smtps
    # 配置 smtp 服务器地址
    host: smtp.163.com
    # 发送者邮箱
    username: 15160078109@163.com
    # 配置授权密码
    password: FTGAVZFOTINYROSS
    # 端口号
    port: 587
    # 默认的邮件编码为 UTF-8
    defailt-encoding: UTF-8
```

将项目更新至 GitHub

将本地代码上传到 GitHub 仓库的指令流程如下：

1. 初始化 Git 仓库（如果还没有初始化）

如果你还没有在本地项目中初始化 Git 仓库，可以使用以下命令：

```
git init
```

2. 添加远程仓库

如果你还没有将远程仓库添加到本地仓库，可以使用以下命令：

```
git remote add origin https://github.com/MRYUHUI/Machinery-Mall.git
```

3. 添加所有更改

将所有更改添加到 Git 索引中：

```
# 从工作区添加到暂存区
git add .
```

4. 提交更改

提交更改并附带一条提交信息：

```
# 从暂存区添加到本地库
git commit -m "Your commit message"

# 可与 (3) 合并
git commit -am "Your commit message"
```

5. 推送到远程仓库

将本地仓库中的更改推送到远程仓库。通常，第一次推送需要指定上游分支：

```
git push -u origin main
```

以后推送代码时，只需使用以下命令：

```
git push
```

组员版

1. 将所有更改添加到 Git 索引中

   ```
   git add .
   ```

2. 提交更改并附带一条提交信息：

   ```
   git commit -m "Your commit message"
   ```

3. 推送代码

   ```
   git push
   ```

将 GitHub 仓库更新到本地

打开终端（命令行）。

导航到你的本地仓库目录。

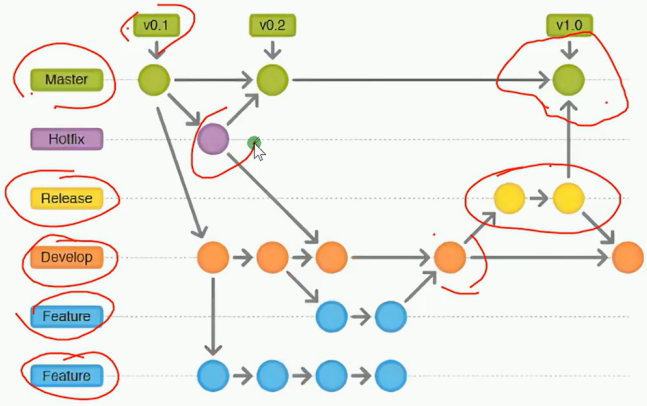
```
cd D:\A_Project\互聯網應用綜合實踐\Machinery-Mall
```

从远程仓库获取并合并更新：

使用`git pull`命令从远程仓库拉取更新并合并到本地分支：

```
git pull origin main
```

GitHub 分支



- **Master分支**：用于版本发布，每一个节点都是可发布版本。
- **Develop分支**：作为开发的主分支始终存在，开发人员应每天拉取远端develop分支，尽早解决冲突。
- **Release分支**：从develop分支创建，作为发布新版本前的准备分支；该分支只进行bug修复和文档修改，待版本稳定后，将该分支合并到master和develop分支，并删除该分支；严禁其他开发分支直接合并到master分支。
- **Feature分支**：从develop分支创建的功能开发分支，用于本地开发使用，开发周期不宜过长；当有功能分支完成，应尽早合并如develop分支，应尽早处理与服务器的冲突；功能完成后，合并到develop分支，并删除该分支。
- **Hotfix分支**：生产环境紧急bug修复分支，从master分支创建，完成bug修改后，合并到master和develop分支，并删除该分支。

分支语句

1. 查看分支：`git branch`
2. 创建分支：`git branch <branch-name>`
3. 切换分支（任选）：
    - `git checkout <branch-name>`
    - `git switch<branch-name>`
4. 创建并切换分支：`git checkout -b <branch-name>`
5. 推送至远程仓库：`git push origin <branch-name>`
6. 合并指定分支到当前分支：`git merge <branch-name>`
7. 删除分支：
    - 删除非合并分支：`git branch -d <branch-name>`
    - 强制删除合并分支：`git branch -D <branch-name>`

分支—多人开发

开发背景

现在有四个开发者：俞晖、刘一霆、张振学、唐行行。其中俞晖是创建者，具有将分支合并到`main`分支的权限。现在远程仓库有一个项目，项目里有一个`Hello.java`文件。任务如下：刘一霆要建立一个分支，负责编写`Walk.java`；张振学要建立一个分支，负责编写`Eat.java`；唐行行要建立分支编写`Hello.java`。最后由俞晖合并。

1. 俞晖克隆远程仓库

如果远程仓库还没有克隆到本地，可以使用以下命令克隆：

```
git clone https://github.com/MRYUHUI/Machinery-Mall.git
cd Machinery-Mall
```

如果已经克隆，可以先拉取最新的更改：

```
git checkout main
git pull origin main
```

2. 刘一霆 创建分支并编写 `Walk.java`

1. **刘一霆 创建新分支**：

   ```
   git checkout -b feature/liu-yiting-walk
   ```

2. **刘一霆 编写 `Walk.java` 文件**：

   创建 `Walk.java` 并编写代码。

3. **刘一霆 提交和推送更改**：

   ```
   git add Walk.java
   git commit -m "Add Walk.java with walking functionality"
   git push origin feature/liu-yiting-walk
   ```

3. 张振学 创建分支并编写 `Eat.java`

1. **张振学 创建新分支**：

   ```
   git checkout -b feature/zhang-zhenxue-eat
   ```

2. **张振学 编写 `Eat.java` 文件**：

   创建 `Eat.java` 并编写代码。

3. **张振学 提交和推送更改**：

```
git add Eat.java
git commit -m "Add Eat.java with eating functionality"
git push origin feature/zhang-zhenxue-eat
```

4. 唐行行 创建分支并修改 `Hello.java`

1. **唐行行 创建新分支：**

```
git checkout -b feature/tang-xingxing-hello
```

2. **唐行行 修改 `Hello.java` 文件：**

   修改 `Hello.java` 并编写代码。

3. **唐行行 提交和推送更改：**

```
git add Hello.java
git commit -m "Modify Hello.java"
git push origin feature/tang-xingxing-hello
```

5. 俞晖 合并分支到 `main`

1. **俞晖 切换到 `main` 分支并更新：**

```
git checkout main
git pull origin main
```

2. **俞晖 合并 刘一霆 的分支：**

```
git merge feature/liu-yiting-walk
```

   解决合并冲突（如果有），然后提交：

```
git add .
git commit -m "Merge feature/liu-yiting-walk into main"
```

3. **俞晖 合并 张振学 的分支：**

```
git merge feature/zhang-zhenxue-eat
```

   解决合并冲突（如果有），然后提交：

```
git add .
git commit -m "Merge feature/zhang-zhenxue-eat into main"
```

4. **俞晖 合并 唐行行 的分支：**

```
git merge feature/tang-xingxing-hello
```

   解决合并冲突（如果有），然后提交：

```
git add .
git commit -m "Merge feature/tang-xingxing-hello into main"
```

5. **俞晖 推送合并后的 `main` 分支到远程仓库**：

```
git push origin main
```

注意

- **克隆或更新本地仓库**：确保每个开发者本地仓库是最新的，为了保证每个开发者的代码库与最新的 `main` 分支保持同步，建议在每天开始工作前执行 `git pull origin main`。
- **组员每天开始工作前**：拉取最新的 `main` 分支，并合并到自己的功能分支，以确保在最新代码基础上进行开发。
- **组长每三天合并一次分支**：组长合并每个开发者的分支到 `main` 分支，并解决可能的合并冲突，然后推送到远程仓库。

每天开始工作前的操作（组员）

1. 切换到 `main` 分支：

```
git checkout main
```

2. 拉取最新的 `main` 分支代码：

```
git pull origin main
```

3. 切换回自己的功能分支：

```
git checkout feature/your-branch-name
```

4. 合并最新的 `main` 分支代码到**自己的分支**：

```
git merge main
```

解决合并冲突（如果有），然后提交：

```
git add .
git commit -m "Merge main into feature/your-branch-name"
```

建立后端项目

编写 application.yml

```yaml
server:
#  设置基地址为 http://localhost:8080/actionmall
  port: 8080
  servlet:
    context-path: /actionmall

spring:
  datasource:
    url: jdbc:mysql://127.0.0.1:3306/mech-mall?
useSSL=false&useUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC&allowPublicKeyRetrieval=true
    username: root
    password: yh020316
    driverClassName: com.mysql.cj.jdbc.Driver
    type: com.alibaba.druid.pool.DruidDataSource
  jackson:
    date-format: yyyy-MM-dd HH:mm:ss
  servlet:
    multipart:
      max-file-size: 120MB
      max-request-size: 120MB

mybatis-plus:
  configuration:
    log-impl: org.apache.ibatis.logging.stdout.StdOutImpl

mybatis:
  type-aliases-package: com.mall.mechmall.domain
  mapper-locations: classpath:mapper/*.xml
```

配置工具类

在 utils/Consts.java 下添加如下代码：

```java
package com.mall.mechmall.utils;

import com.alibaba.fastjson.JSONObject;
import org.springframework.util.FileSystemUtils;

import java.io.File;

public class Consts {
    //    返回碼
    public static final String CODE = "code";
    //    返回信息
    public static final String MSG = "message";

    public static final String SUCCESS = "success";
```

```
    public static final String ERROR = "error";

    public static final String WARNING = "warning";

    public static final String DATA = "data";

    public static final String STATUS = "status";

    public static final String TYPE = "type";

    public static final int CODE_404 = 404;

    public static final int CODE_200 = 200;

    public static final int CODE_500 = 500;

    public static Boolean deleteFile(String path) {
        String filePath = System.getProperty("user.dir") + path;
        File file = new File(filePath);
        if (!file.exists()) {
            return true; // 文件不存在，直接返回true
        }
        return FileSystemUtils.deleteRecursively(new File(filePath));
    }

    public static JSONObject getJson(boolean tag, String msg) {
        JSONObject jsonObject = new JSONObject();
        if (tag) {
            jsonObject.put(Consts.CODE, CODE_200);
            jsonObject.put(Consts.MSG, msg + "成功");
            jsonObject.put(Consts.SUCCESS, true);
            jsonObject.put(TYPE, SUCCESS);
        } else {
            jsonObject.put(Consts.MSG, msg + "失败");
            jsonObject.put(TYPE, ERROR);
            jsonObject.put(Consts.SUCCESS, false);
        }
        return jsonObject;
    }
}
```