



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

软件工程

## Lab 3: Git 实战



## 实验目标

- 熟练掌握git的基本指令和分支管理指令；
  - 掌握git支持软件配置管理的核心机理；
  - 在实践项目中使用git/github管理自己的项目源代码。
- 
- 本次实验由个人单独完成。

## 在本地机器上安装Git

- **Linux:**
  - \$ yum install git-core #在redhat等系统下用yum
  - \$ apt-get install git-core #在debian, ubuntu等系统下用apt-get
- **Windows:** <https://git-scm.com/download> 下载git客户端安装包，在本地机器安装
- 在Github上申请个人账号，作为Git远程服务器：  
Github(<http://www.github.com>)
- 说明：
  - Linux和Windows下的Git安装操作二选一即可
  - 在Github上申请账号
  - 使用命令行方式完成实验，避免图形界面下的操作。

## 实验场景(1)：仓库创建与提交

- **R0**：在进行每次git操作之前，随时查看工作区、暂存区、git仓库的状态，确认项目里的各文件当前处于什么状态；
- **R1**：本地初始化一个git仓库，将自己在Lab1中所创建项目的全部源文件加入进去，纳入git管理；
- **R2**：提交；
- 手工对Lab1的3个文件进行修改；
- **R3**：查看上次提交之后都有哪些文件修改、具体修改内容是什么；
- **R4**：重新提交；
- 再次对Lab1的3个文件进行修改；
- **R5**：重新提交
- **R6**：把最后一次提交撤销；
- **R7**：查询该项目的全部提交记录；

## 实验场景(1)：推送到GitHub上

**R8：**在GitHub上创建名为Lab1的仓库，并在本地仓库建立相应的远程仓库；

**R9：**将之前各步骤得到的本地仓库全部内容推送到GitHub的Lab1仓库；

## 实验场景(2)：分支管理

- R1: 获得本地Lab1仓库的全部分支，切换至分支master;
- R2: 在master基础上建立两个分支B1、B2;
- R3: 在B2分支基础上创建一个新分支C4;
- R4: 在C4上，对4个文件进行修改并提交;
- R5: 在B1分支上对同样的4个文件做不同修改并提交;
- R6: 将C4合并到B1分支，若有冲突，手工消解;
- R7: 在B2分支上对3个文件做修改并提交;
- R8: 查看目前哪些分支已经合并、哪些分支尚未合并;
- R9: 将已经合并的分支删除，将尚未合并的分支合并到一个新分支上，分支名字为你的学号;
- R10: 将本地以你的学号命名的分支推送到GitHub上自己的仓库内;
- R11: 查看完整的版本变迁树;
- R12: 在Github上以web页面的方式查看你的Lab1仓库的当前状态。

## 实验场景(3)：远程仓库与远程分支

1. 获取自己Lab1的搭档的GitHub上的Lab1仓库URL，在本地自己的Lab1仓库建立自己命名的分支指向它；
2. 查询目前本地已配置了哪个(些)远程仓库，查看各自详细信息以及权限等；
3. 将自己搭档的Lab1数据从GitHub复制到本地；
4. 查看本地仓库当前的远程分支；
5. 在自己命名的分支上选择任意3个文件做改动提交，然后将本地以自己命名的分支推送到搭档的GitHub仓库中——有无写入权限？
6. 任选Lab1中3个文件做改动，将结果推送到自己的GitHub上，从GitHub上获取搭档的最新推送；
7. 查看本地仓库当前的远程分支；
8. 将自己搭档对Lab1做出的变化合并到自己的本地仓库的“学号”分支；
9. 再次将当前本地仓库的全部内容推送至自己的GitHub；
10. 查看本地仓库当前完整的版本变迁树；
11. 不再关注搭档的GitHub Lab1仓库；
12. 做完上述各步骤之后，将本地仓库的HEAD切换回master分支，确保自己本地文件系统恢复到Lab1结束时的状态。

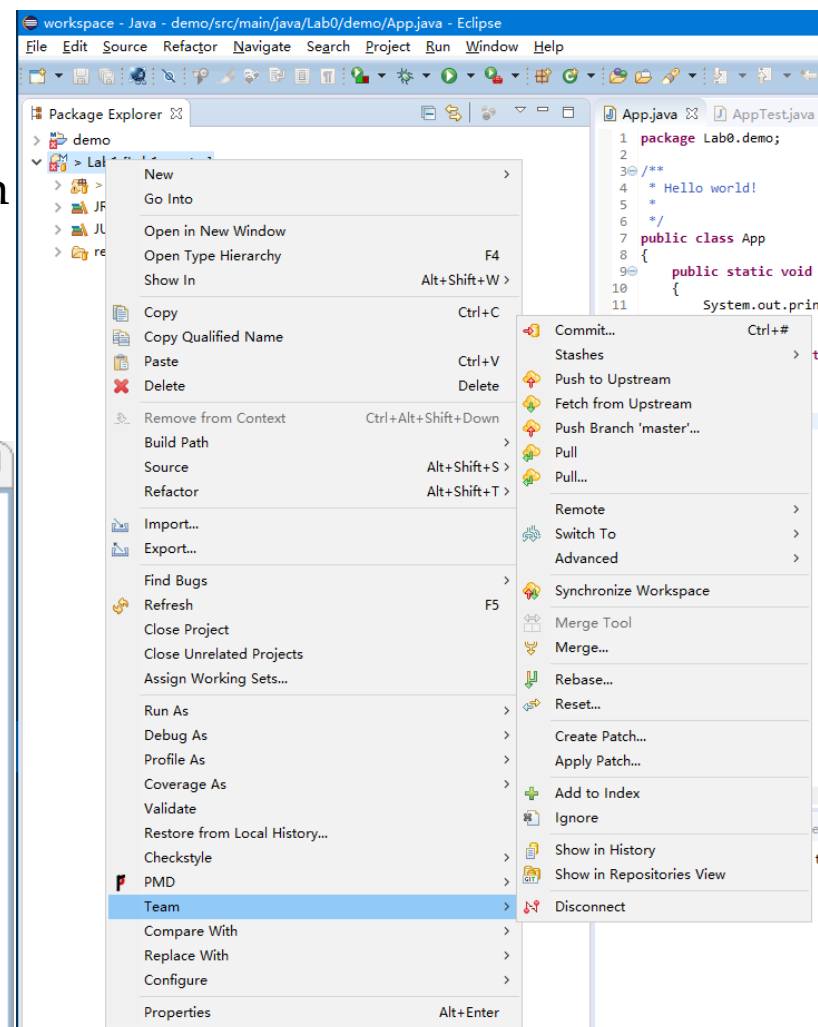
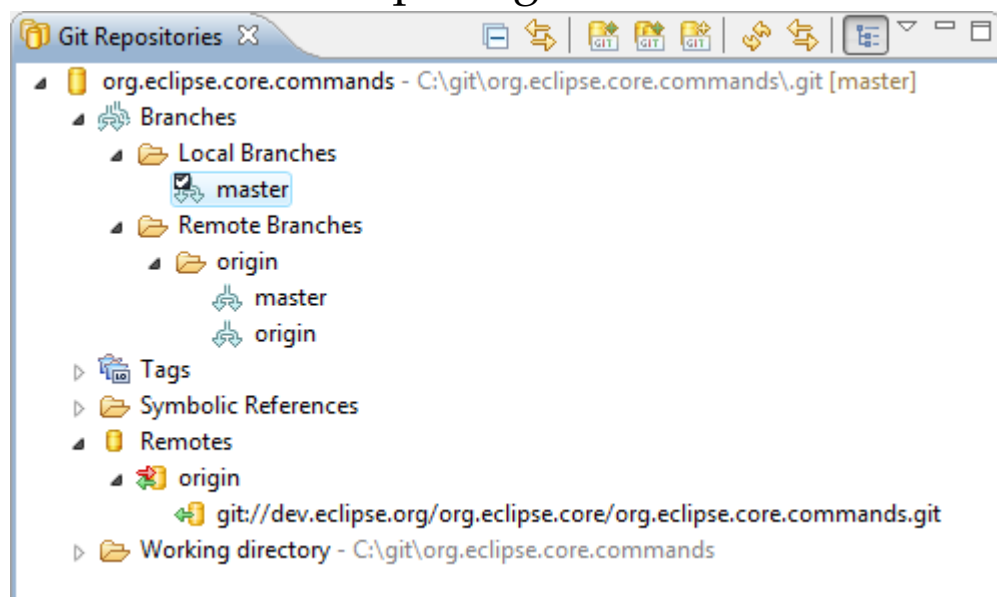
## 实验场景4: Git高级指令

- 将GitHub上的jQuery项目 <https://github.com/jquery/jquery> 克隆至本地;
- 使用git命令完成以下任务:
  1. 按时间从早到晚的次序列出该项目内的所有commit (该项目的最后一次commit应该在最后出现)
  2. 选定一个commit, 找出其父commit、提交者信息、作者信息、日期;
  3. 查询由 m.goleb@gmail.com 作为提交者的全部commit, 每行一条, 展示SHA、提交时间、message, 从晚到早的次序排序;
  4. 获得该仓库内的所有tag, 按时间从早到晚排序;
  5. 任选两个时间相邻的commit, 找出它们的哪些文件做了修改, 并分别给出增加的文件、删除的文件、修改的文件、重命名的文件、代码变化的行数、变化的代码内容;
  6. 选定一个commit, 选定其中一个文件, 查询该文件的每一行的owner;
  7. 查询该仓库当前HEAD的位置;



## 在Eclipse中安装eGit

- eGit是git在Eclipse IDE中的插件，在Eclipse内直接使用git进行代码版本控制。
  - <https://www.eclipse.org/egit>
  - 可看作git在Eclipse中的可视化workbench
  - 教程: [http://wiki.eclipse.org/EGit User Guide](http://wiki.eclipse.org/EGit_User_Guide)
  - 注: 新版Eclipse中git支持已经内置



## 实验场景5：在Eclipse中使用Git管理Lab2

1. 在Eclipse中，将自己的Lab2纳入git管理；
2. 对Lab2进行若干修改，使用eGit对其进行提交操作；
3. 将Lab2内容推送至个人GitHub的Lab2仓库。

## 评判标准

- 是否完成了实验要求的各项任务；
- 是否正确使用了恰当的git指令完成任务；
- 是否可在Eclipse中安装和配置Git客户端并用其管理自己的代码。

## 提交方式

- 请遵循实验报告模板撰写。
- 提交日期：第7周周一晚(10月16日 23:55)
- 提交实验报告到CMS：
  - 实验报告：命名规则 “学号-Lab3-report.doc”
- 本次实验由个人单独完成。



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

软件工程

结束

