JAVA / GIT

SOLUTIONS

home / solutions / java-git

#solutions #java #git #oop

1. FIRST JAVA PROJECT

- Create Project
- Create Application.java
- Edit Application.java

```
public class Application {
   public static void main(String args[]) {
       System.out.println("Hello World");
   }
}
```

- Run Application.main()

2. INITIALIZE GIT REPOSITORY

```
cd directory/of/my/project/helloworld
git init
ls -la
git status
git status -s
echo "out" >> .gitignore     # this appends out to the .gitignore file
echo ".idea" >> .gitignore   # this appends .idea to the .gitignore
file
echo "*.iml" >> .gitignore   # this appends *.iml* to the .gitignore
file
git status
git status
git add .gitignore src             # add files to the staging area
git status
```

```
git commit -m "Initial commit"     # commit the new version
git status
```

3. MAKING ANOTHER COMMIT

- Create file first.txt with some text

```
git add first.txt
git commit -m "Added first.txt"
git status
```

4. INSPECTING THE LOG

```
git log
git log --oneline -1
git log -1 -p
```

5. ADDING AND REMOVING

- Create file second.txt with some text
- Delete file first.txt

```
git add first.txt second.txt
git status
git commit -m "Added second.txt and removed first.txt"
git status
git log
```

6. PARTIALLY STAGED FILES AND DIFF

- Create file third.txt with some text

```
git status
git add third.txt
```

- Add more text to file third.txt

```
git status
git commit -m "Added text to third.txt"
git status
git diff third.txt
git add third.txt
git commit -m "Added more text to third.txt"
git status
git log
```

7. BRANCHING AND MERGING

```
git log --oneline
git branch
git branch testing
git branch
git checkout testing
git branch
```

- Add more text to file third.txt

```
git add third.txt
git commit -m "Added even more text to third.txt"
git log --oneline
git checkout master
git log --oneline
```

- Create file LICENSE with text: “This is the license text for our application”.

```
git add LICENSE
git commit -m "Added Added LICENSE file"
git merge testing
git branch -d testing
```

8. HANDLING MERGE CONFLICTS

```
git branch testing
```

- Edit Application.java to:

```
public class Application {
    public static void main(String[] args) {
        System.out.println("Thank you for using the Hello World App");
    }
}
```

```
git add Application.java
git commit -m "Added a welcoming message"
git checkout testing
```

- Edit Application.java to:

```
public class Application {
    public static void main(String[] args) {
        System.out.println("Hello World");
        System.out.println("Version 1.0");
    }
}
```

```
git add Application.java
git commit -m "Added the version number"
git checkout master
git merge testing
```

- Edit Application.java to:

```
public class Application {
    public static void main(String[] args) {
        System.out.println("Thank you for using the Hello World App");
        System.out.println("Version 1.0");
    }
}
```

```
git status
git add src/Application.java
git status
git commit -m "Merge branch 'testing'"   # commit resolved conflicts
and end the merge
git status
git log --oneline
git branch -d testing
```

9. USING REMOTES

- Register for a new GitHub account (if needed).
- Create a new **empty** repository with the name "helloworld".

```
git remote add origin git@github.com:yourusername/helloworld.git
git remote -v
git branch
git push origin master
git branch -u origin/master
git push
```

- Open the github page for your repository.
- Press the "Add a README" button, and then "Commit new file".
- This will add a "README.md" file to our repository.

```
git fetch
git merge origin/master
```

10. GIT CLONE AND IMPORT

- Close your project on IntelliJ and remove it from the recent project list.
- Delete your project directory.

```
git clone git@github.com:arestivo/helloworld.git
cd helloworld
ls -la
```

- Import Project into IDEA

Copyright © André Restivo