CompSci 316 Discussion 6
Fall 2022

Presenter: Zhe Wang

Session 01: GTAs & UTAs: Chengyu Wu, Tong Lin, Danny Luo, Joyce Wang
Session 02 GTAs & UTAs: Haibo Xiu, Justin Lim
Roadmap

- Fundamentals of Version Control (Git)
- Tutorial for standard group project
  - Setup of Mini Amazon Skeleton
  - Walk through of MVC
  - Walk through of the code base
- Walkthrough of HW4 (will release next week)
- Q&A for milestone 2 and HW4
Fundamentals of Git

- Git manages “repositories” – akin to folders of Word documents that all have revision tracking turned on.
- Many other source control systems have files set up one way on the server for holding the master copy and another way on your local laptop for your local edits.
- Git does not do this – all Git repositories are equally capable, whether on the server or your local laptop – and you should take advantage of this fact!
- Before you commit for the first time (on a given computer):
  
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
Fundamentals of Git

- **Project**
  - Working Directory: All the actual files
  - Index: Staging area for changes
  - Head: Points to the last commit

- **Branch**
  - Local
    - Create: `git branch your_branch_name` or `git checkout -b your_branch_name`
    - Delete: `git branch -d your_branch_name`
    - View all branches: `git branch`
  - Remote
  - Main / Master Branch

- **Commit**
  - Stamp of changes
    - `git commit -m "some meaningful commit message"`

- **PR (Pull Request)**
  - Request for merging changes of your branch into master branch

- **CR (Code Review)**
  - Stakeholders should review your code before approving the changes
Fundamentals of Git Basic Workflow

- Clone a project
  
  ```
  git clone git@gitlab.oit.duke.edu:your_user_name/mini-amazon-skeleton.git
  ```

- Branching
  
  ```
  git branch feature_x or git checkout -b feature_x
  ```

- Make some changes

- Commit
  
  ```
  git add xxx
  git commit -m "yyyy"
  ```

- Push
  
  ```
  git push origin feature_x if feature_x does not exist in remote
  git push if feature_x exists in remote
  ```
Fundamentals of Git

- Sync latest changes from remote and changes local files
  - `git pull` some_remote_name (can think of as `git fetch + git merge`)
- Sync latest changes from remote but does not change local files
  - `git fetch` some_remote_name
- Merge
  - `git merge`
- Open a PR
- Code Review

Duke
Fundamentals of Git

■ Solve Merge Conflict

Practice:

First fork the project as a group, then clone it locally. Create a branch called your_name-dev, add a new file called members.txt in the project’s root directory and add your name to the first line of the file. Stage and Commit your changes then push to remote. Create a pull request and add all your members as reviewers.
Good practices of Git

- Commit in small chunks
- Write meaningful commit log messages
- Commit often – to update the “last known good”
- DO NOT PUSH TO MASTER/MAIN BRANCH DIRECTLY
- Having branches per feature or per developer
- Code Review Carefully
Mini Amazon Skeleton Walkthrough

Flask
- Lightweight full stack framework
  - Can develop standalone REST API backend
  - Or develop server side rendering web app leveraging template
- Please walk through QuickStart Guide

SQLAlchemy
- A Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.
- Please walk through ORM Guide
MVC Architecture

- **Model**
  The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.

- **View**
  Any representation of information such as a chart, diagram or table or web UI.

- **Controller**
  Accepts input and converts it to commands for the model or view.

Duke
Controller

- Backend Endpoint
  - Handling parsing HTTP requests send from frontend
  - Translate requests to SQL query
  - Send SQL query via connector
  - Translating database records to Data Access Objects (DAOs)
  - Returning HTTP response response

```python
@bp.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('index.index'))
    form = LoginForm()
    if form.validate_on_submit():
        user = User.get_by_auth(form.email.data, form.password.data)
        if user is None:
            flash('Invalid email or password')
            return redirect(url_for('users.login'))
        login_user(user)
        next_page = request.args.get('next')
        if not next_page or url_parse(next_page).netloc != '':
            next_page = url_for('index.index')
        return redirect(next_page)
    return render_template('login.html', title='Sign In', form=form)
```
View

- Web UI
- Could be separate frontend like React, Vue.js, Angular or could be server side rendering returning HTML to browser to render such as Django
- Fetch data through sending HTTP requests to backend and rendering response
Model

- Usually represented by classes or objects depending on language
- Containing different methods for different business needs
- Translate between database table rows
- provides an abstract interface to some type of database or other persistence mechanism

```python
class User(UserMixinx):
    def __init__(self, id, email, firstname, lastname):
        self.id = id
        self.email = email
        self.firstname = firstname
        self.lastname = lastname

@staticmethod
    def get_by_auth(email, password):
        rows = app.db.execute(''
SELECT password, id, email, firstname, lastname
FROM Users
WHERE email = :email
'''
            , email=email)
        if not rows:  # email not found
            return None
        elif not check_password_hash(rows[0][0], password):
            # incorrect password
            return None
        else:
            return User(*rows[0][1:])
```
SQLAlchemy

- SQL Connector and Object
- Relational Mapper (ORM)
- Open up connection to PostgreSQL
  - Execute SQL Query
  - Return Database Records
  - Map Database Records to Python Classes/Objects