

WEB APPLICATION DEVELOPMENT

DR. ALISTAIR MORRISON

*Joao Almeida-Domingues**

University of Glasgow

January 15th, 2020 – March 25th, 2020

CONTENTS

1	Introduction	2
2	Development Environment	3
2.1	Version Control Systems	3
2.2	Git	4
2.3	Package Management	5
2.4	Virtual Environments	5
3	Django - Introduction	6
3.1	Design Philosophy	6
3.2	Essential (pre-packaged) Modules	6
3.3	MVT	6

These lecture notes were collated by me from a mixture of sources , the two main sources being the lecture notes provided by the lecturer and the content presented in-lecture. All other referenced material (if used) can be found in the *Bibliography* and *References* sections.

The primary goal of these notes is to function as a succinct but comprehensive revision aid, hence if you came by them via a search engine , please note that they're not intended to be a reflection of the quality of the materials referenced or the content lectured.

Lastly, with regards to formatting, the pdf doc was typeset in L^AT_EX, using a modified version of Stefano Maggiolo's [class](#)

*2334590D@student.gla.ac.uk

1 INTRODUCTION

1.1 definition. Web Application is a distributed information management system. In essence it is an application , usually spread around several machines, which allows users to query and manipulate data, and which is then displayed via the browser

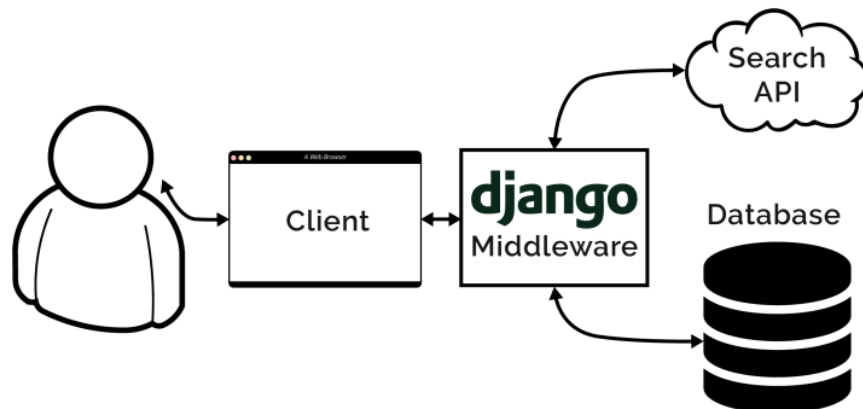
1.2 remark. The distinction between a website and a web app is less clear nowadays, but the main feature of a web app seems to be its interactive nature, as opposed to the more expository nature of simple websites

1.3 definition. Distributed Information Management System (DIM) a group of machines working together but appearing to the user as a single entity. These systems have major advantages, such as concurrent and independent operation. One machine can fail or be upgraded without the system failing

Architecture

A common system architecture for a DIM is represented in the diagram below. Web development has a wide range of tools and new frameworks and languages come into being every other week. In this class, we'll focus on the tools listed below but in general each tool group performs certain tasks which are essential to each bit of the system

1.4 remark. Most web apps follow a 3-tier architecture : Presentation, Application, Data. The rango app will also include an external service , the Bing search API



Overview of the 3-tier system architecture for our Rango application.

1.5 definition. User machine or person which initiates contact with the client

1.6 definition. Client program sitting on the user device which sends/accepts requests/responses and acts on the messages by communicating to them the user and/or by altering state in some way

1.7 remark. A request is usually done via HTTP where the data to be sent is

embedded, and responses are also sent via HTTP with content usually delivered in a markup language like XHTML or JSON

1.8 definition. Middleware Responsible for accepting and sending responses on server side and for coordinating messages

1.9 remark. In this course we'll be using Django


1.10 remark. The database is usually kept in a separate node

Tools Covered

Proficiency expected

- | | |
|-------------------|--------------------|
| 1. Python | 8. AJAX |
| 2. Django | 9. Github |
| 3. HTML/CSS | 10. Pip |
| 4. HTTP GET/POST | 11. PythonAnywhere |
| 5. XML/XHTML/JSON | 12. IDLE |
| 6. JS | 13. Venv |
| 7. JQuery | 14. Draw.io |

2 DEVELOPMENT ENVIRONMENT

 Git , package management , pip , IDE , Django , SE best practices , repository , pip , virtual environment , version control

Over the years web applications' development adopted many practices from software engineering, in order to maximize productivity. It is essential to setup your development environment appropriately. Below we explore common tools and practices widely used in industry

2.1 Version Control Systems

2.1 definition. VCS software which makes tracking and managing changes to documents easy

2.2 definition. Repository where copies of a project/file being tracked are stored

2.3 definition. Snapshot the state of the file at a point in time

VC is widely used not only in industry but in personal and open-source projects. It's main advantages being the fact that it makes collaboration easy since everyone in a team can read from and write to the same repository ; the

fact that it allows for several working versions of a project to be stored separately without having to save a copy of every single file (instead pointers for existing files are simply stored with every new commit); and also the fact that it prevents against loss of code, specially when stored in a remote repository

2.2 Git

The most widely used VCS is Git, invented by Linus Torvalds in 2005. Before, the most widely used software was SVN. They differ mainly in how changes are distributed to the shared version. SVN uses a single central repository, which means that your data is stored in a single place in a server. Git, on the other hand, allows developers to keep a local copy of the shared repository, and allows for greater granularity when deciding which changes to apply and when, by offering a staging area. When ready to share the changes, developers *push* their local repo to the *remote*/shared one

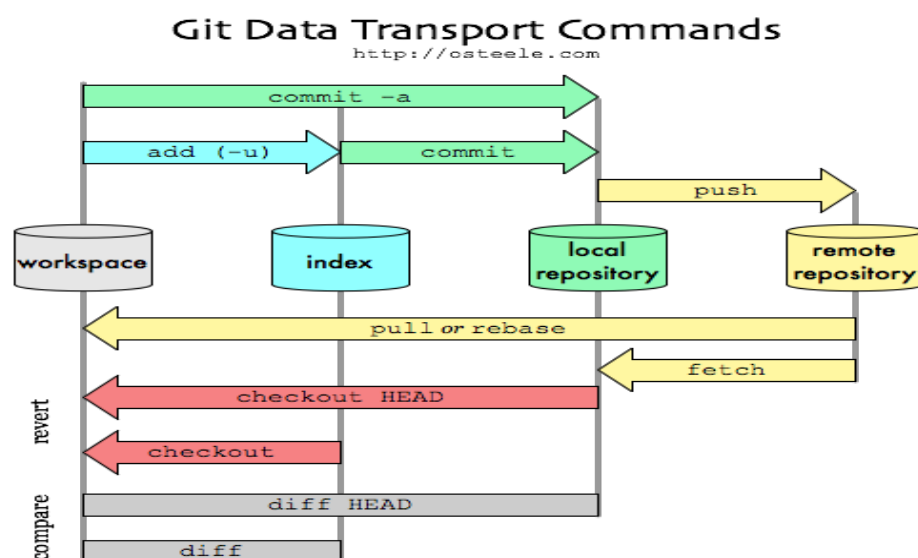
Git Workflow

Git uses 3 local spaces:

Workspace is your working directory, where `.git` is created when initializing the repository, along with some other needed files for version control

Index is the staging area, where files being tracked are added prior to committing. It consists of a binary file with a list of all the files in the workspace

Local Repository is where the actual copies of the files for all branches (both local and remote) exist



2.4 remark. Git mantra : "*commit early, commit often*"

2.5 remark. See [1] for further details

2.3 Package Management

2.6 definition. **Package Manager** software tool that automates the process of installing, upgrading and configuring software libraries

Prior to the existence of package managers, running other developers' apps could be incredibly difficult ; not because there was something inherently wrong in its code base, but because as apps became more complex, they relied on public libraries, each with their own specific version, and keeping track of it all and setting them up individually was not the easiest of tasks [3] . PMs help overcome this difficulties by partially recreating the creator's environment. Hence, the process becomes *defined* , since a list of the required packages is distributed along with the rest of the software ; *repeatable* and *exportable* since the list of the required packages and their versions is passed onto the package manager and can be easily exported

2.7 remark. In this course we'll be using pip the *de facto* PM for python-based apps

2.4 Virtual Environments

2.8 definition. **Virtual Environment** a local environment that is configured to provide access to libraries, settings, hardware

Installing software system-wide is not considered a best-practice. Often, environments will clash, given that, as discussed above, different apps might require different versions of the same software. In order to avoid this, it is best to setup a virtual environment for your project. This way we satisfy each project dependencies, while avoiding system-wide conflicts

2.9 remark. Anaconda will be used in the lab machines. Use VirtualEnv with VirtualEnvWrapper in yours

3 DJANGO - INTRODUCTION

Django is a web framework which makes building web apps simpler and faster, by providing:

- a division between logic and presentation components
- auto-generation of web admin
- many APIs for common tasks
- extra URL mapping control

URL-Django-HTML vs URL-HTML

3.1 Design Philosophy

Django follows the *loose coupling* principle, which states that different layers of the app should not know about each other's details ; and the *DRY* principle which states that "*Every piece of knowledge must have a single, unambiguous, authoritative representation within a system*", i.e one should know exactly where to lookup required data without having to repeat it in logically separate modules

3.2 Essential (pre-packaged) Modules

- Create Read Update Delete Admin Interface
- Authorization Systems
- Form, Session Handling
- Syndication Awareness (e.g. RSS)
- Caching
- Internationalization

3.3 MVT

3.1 definition. ORM a programming technique for converting data of different types using OO concepts and languages

Django uses a variant of the MVC design pattern - *MVT*. The Model-View-Template is a *data-driven* framework which separates an app into the following logical units

1. **Models** : is an abstraction of the database, where data structures are represented via python objects

This abstraction is achieved via *Object Relational Mapping* , where one can query the database via the usual object dot access notation, and Django in the background, converts both the objects and our queries into valid SQL (or whatever DB language the project is using)

2. **Views** : provide both the representation logic and app logic. It's here that one handles HTTP requests/responses via response objects. It is also here that one can query data via the models' API, where a template populated with results is returned for presenting
3. **Templates** : provide the presentation layout (decoupled from the data), composed of static HTML, with the possibility of using the *django template language* to include some dynamic elements

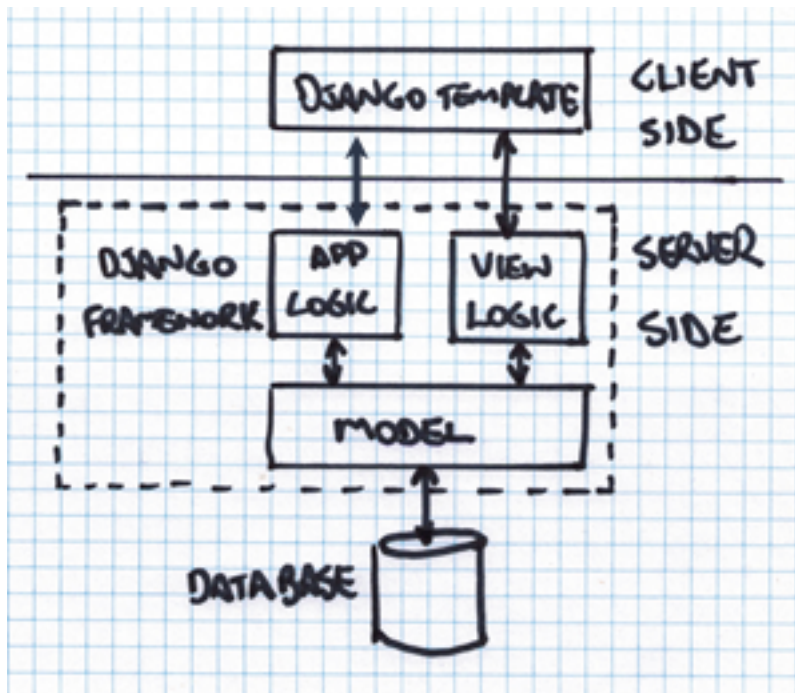


Figure 1: Django MVT

3.2 remark. Views are not queried directly via the client, instead the `urls.py` file allows for finer control, pairing a given URL (or URL pattern) to a certain view.

REFERENCES

REFERENCES

Almeida-Domingues: Knowledge Base **knowBase**

Joao Almeida-Domingues. *Knowledge Base*. URL: <https://app.gitbook.com/@knowbase/s/workspace/git>.

Kozlovski: A Thorough Introduction to Distributed Systems **kozlovski'2019**

Stanislav Kozlovski. *A Thorough Introduction to Distributed Systems*. June 2019. URL: <https://www.freecodecamp.org/news/a-thorough-introduction-to-distributed-systems-3b91562c9b3c/>.

Ovens: The evolution of package managers **ovens**

Steve Ovens. *The evolution of package managers*. URL: <https://opensource.com/article/18/7/evolution-package-managers>.