Building Web Applications

Internet Technologies

ITECH

Question

What is a
Web App?

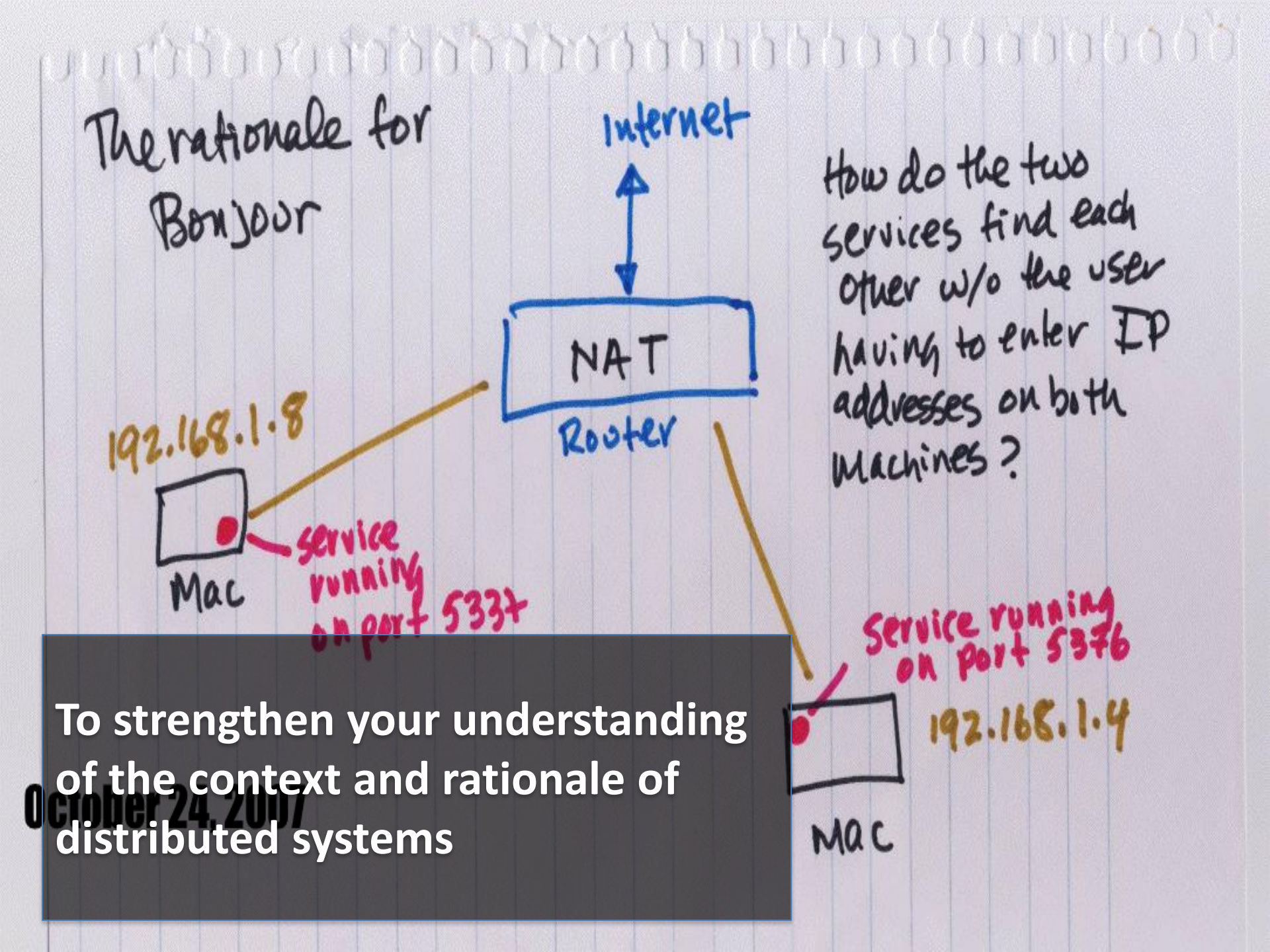
Examples?

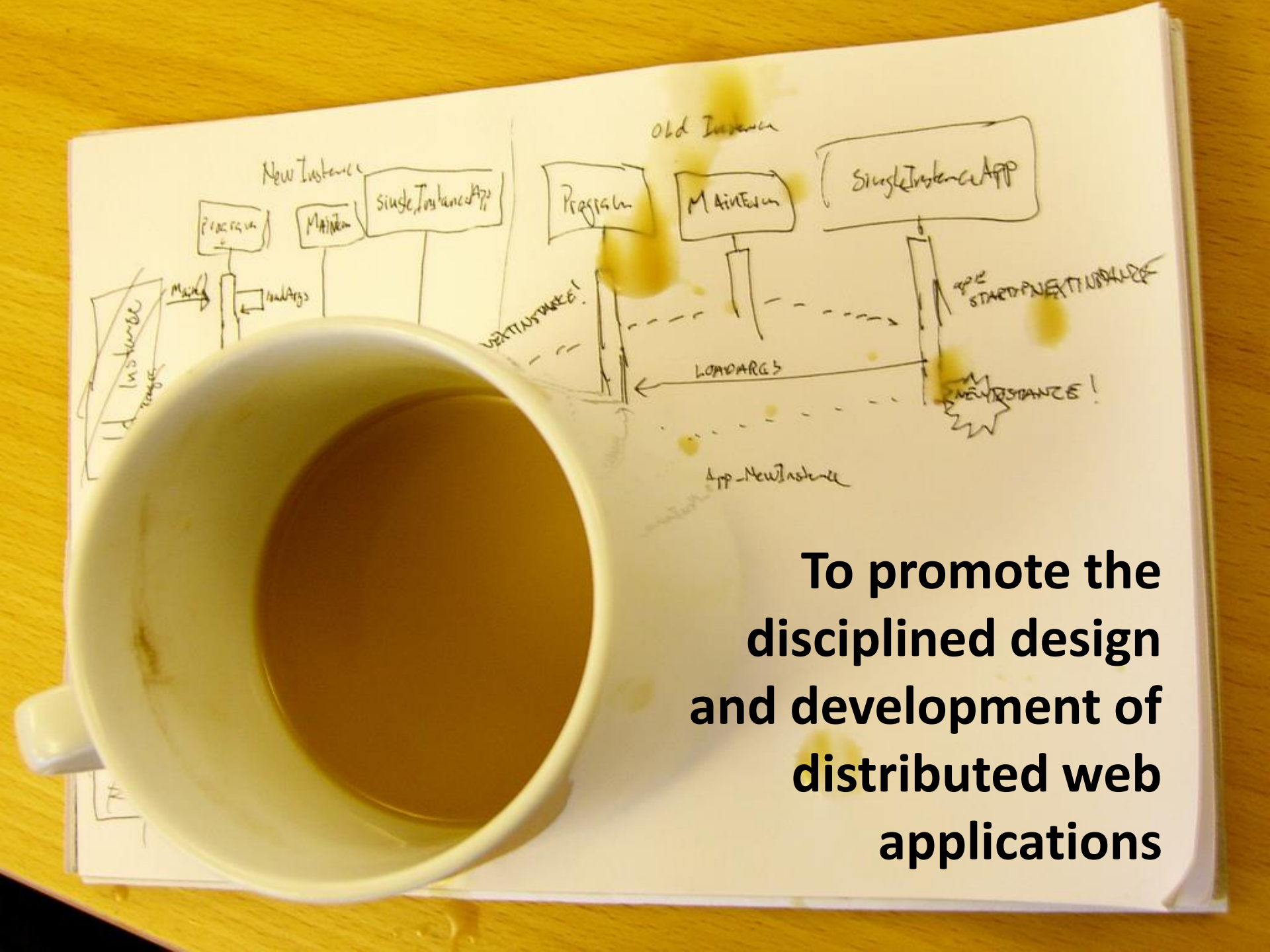
What is a Web App

- It is a distributed information management system (DIM)
- Enables the management, sharing, finding and presentation
- Does so over a network, in a distributed fashion
- Typically has many users, often geographically separated
- Ideally DIMs enable users to access timely, relevant and useful information in a seamless manner
- Think MyCampus ☺

COURSE AIMS


To provide an overview of web application development
WAVERLEIGH
YOUNGSON
BINNSMEAD
BROOKLYN
GLADSTONE
MARYSVILLE
BOYLE
REED
WOODSTOCK
ARLETA
FOSTER
LENTS
LLEWELLYN
WOODMERE
CLINTON KELLY
LEWIS
LANE
DUNIWAY
NORTH SELLWOOD
ERROL HEIGHTS
HARNEY
WHITMAN
SELLWOOD
BATTIN
ARDENWALD
WICHITA
BELL
MILWAUKIE
KING ROAD


The rationale for Bonjour
Internet
NAT
Router
How do the two services find each other w/o the user having to enter IP addresses on both machines?
192.168.1.8
Mac
service running on port 5337
Service running on port 5376
192.168.1.4
Mac
To strengthen your understanding of the context and rationale of distributed systems
October 24, 2007


To promote the disciplined design and development of distributed web applications

To understand the messaging and protocols used as a communication mechanisms in web applications


To develop the ability to implement and deploy distributed web applications

Types of Architectures

Information Architecture

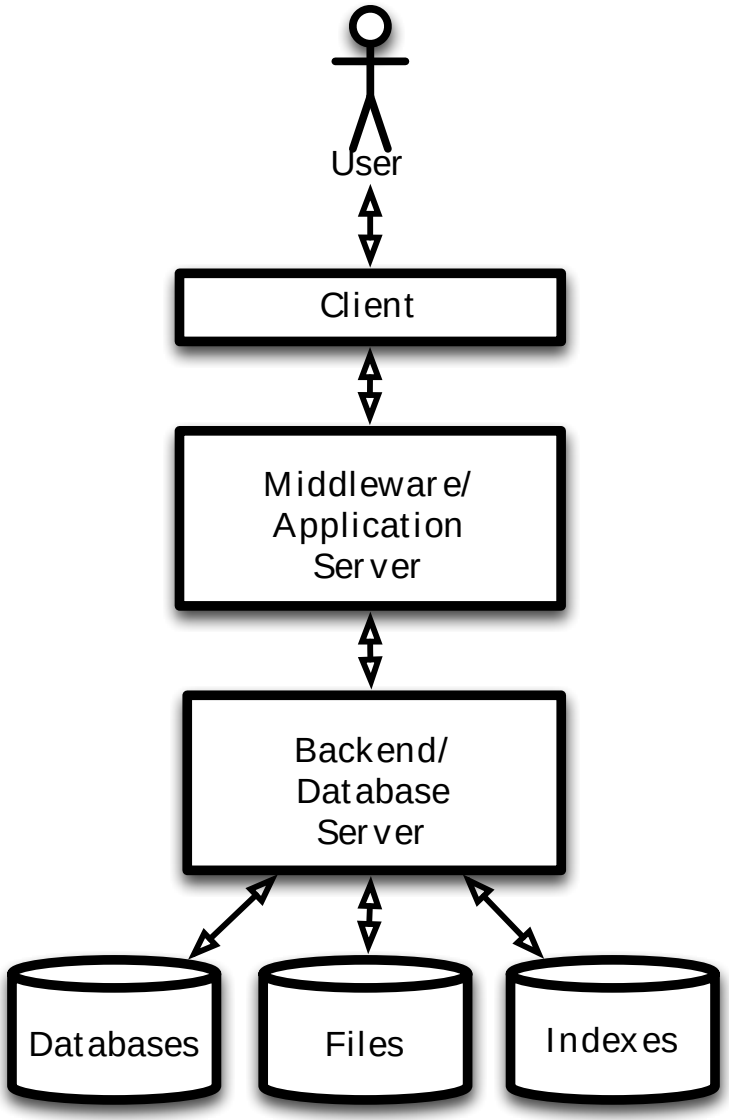
System Architecture

Design Elements

- Information Architecture (user focused)
  - User Personas
  - User needs matrix
  - Site design / URL design
  - Wireframes and Walkthroughs
- System Architecture (system focused)
  - Specifications and Requirements
  - High Level System Architecture
  - Sequence Diagrams
  - Entity Relationship Diagrams

SYSTEM COMPONENTS

High Level System Architecture



User and Client

The **User** could be human or machine, which

- Initiates and interacts with the client
- Ranges in skills and abilities
- Has a number of requirements that need to be satisfied

The **Client** is a program sitting on a client device which

- sends **request messages**
- accepts **response messages**
- acts on the message
  - either communicating to the user
  - or affecting the environment in some way

Messaging

The **request message** is sent to the server from a user agent to:

- ask for some information
- send some information to be stored
  - either from user input
  - or from some device (sensor)

The **response message** is sent from the server to a user agent to:

- return the requested information
- affect some change in the environment

Middleware & Backends

The **Middleware/Application Server** is the central component which

- accepts **request messages** from clients
- returns **response messages**
- co-ordinates the application components

The **Backend/Database** is typically on a separate node and

- stores the data for the application
- provides the data when needed
- needs to scale and be reliable
- could be a database, an index, a flat file


Web Dev Complexity

Web Development Complexity

Collision of Languages
- Markup Languages
- Programming Languages
- Database Query Languages

Shifting Standards
- Document Object Model
- XML/JSON

Web Browser Compatibility
- Browser wars encourage new ‘unique’ features

HTTP is a Stateless Protocol
- But most applications require the persistence of state

Signs of Hope

Web development has become a serious business

- Despite the poor platform, there is incentive to persevere

Inline with this, the methods of development are maturing and increasingly adopting good practices of 'classical' Software Engineering:

- Application Programmer Interfaces (APIs)
- Libraries
- Frameworks
- Tools
- Standards

Tools

Web Development Tools

Web development tool support is not yet as advanced as with classic software development

- Most languages have several complex IDEs (Integrated Development Environments) to chose from
  - Eclipse, .Net Framework, Django, etc

The nature of web development is disjoint

- a developer must become familiar with a set of distinct and (typically) non-integrated tools

What you code in…

An important tool is the text editor or IDE

The choice of text editor and your expertise in its usage can seriously affect your productivity

- Syntax Awareness
- Auto-completion
- Snippets
- Scripts & Macros
- Integration with other development tools (revision control)

Some IDEs have plugins for scripting languages (e.g. Eclipse has a PHP plug-in)

Checking your code

Interpreted languages lack the compilation stage where errors and warnings can be raised

Frustratingly, scripts will just run until they reach an error and fall over

- If you are lucky, there will be an error message
  - But it won't make much sense most of the time

There are a wide range of tools that will perform static analysis of scripts to spot errors

- PyLint / JSLint
- PHP_CodeSniffer (works on PHP, Javascript & CSS)
- Chrome's Developer Tools, Firebug for Firefox

Complying with Standards

There are a variety of tools to check that certain parts of a web application conforms to standards

- (X)HTML
  - http://validator.w3.org/
- CSS
  - http://jigsaw.w3.org/css-validator/
- XML Feeds (RSS/Atom)
  - http://validator.w3.org/feed/
- JavaScript Code Quality Tool
  - http://www.jslint.com/


Learning
Challenges

Main Technologies in this Course

- Python
- Django
- Javascript
- JQuery
- CSS
- HTTP GET/POST
- XML/JSON
- XHTML
- AJAX

- As well as:
  - GitHub
  - Pip
  - PythonAnywhere
  - PyCharm
  - Virtual Environments
  - Draw.io

Summary

Web application development is complex and difficult due to:

- Convergence of technologies (markup, scripting, database)
- Variations in platform (browsers/standards)
- Rate of change for expectations of functionality
  - web 2.0 / Ajax / web apps / etc

Increasingly, as the area matures, established software engineering methods are beginning to be incorporated, acting as complexity countermeasures

- But be aware, they may introduce new problems

Assessments

- (1) Lab Exercises (10%)
  - development of the Rango application
- (2) Group Project (40%)
  - Design specification (10%)
  - Project presentation (5%)
  - Project application (25%)
- (3) Exam (50%)
  - Multiple-choice and open questions
  - **Includes code-based questions!!!**

Deadlines

13 Feb

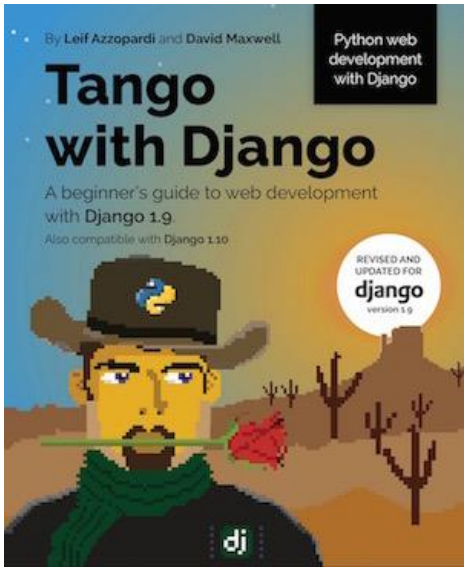
06 Feb

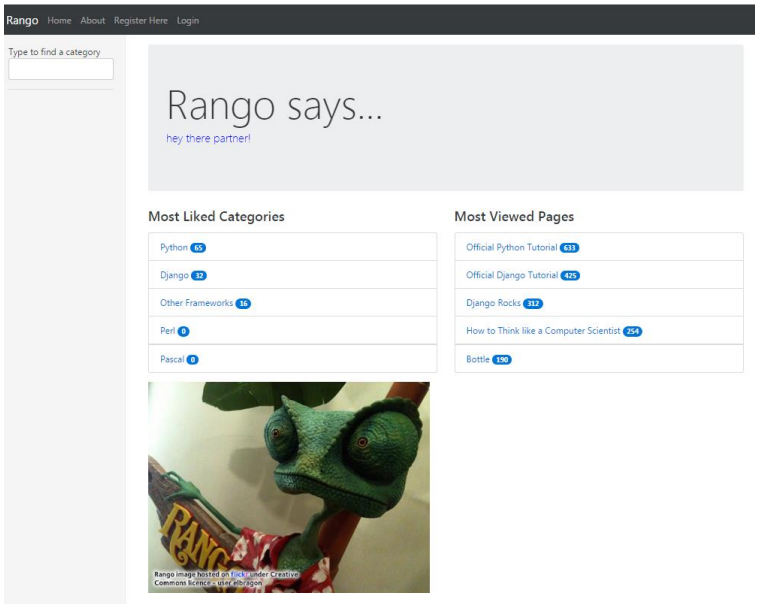
Week 10 lab

15 Mar

Rango application

- Main focus of the lab sessions in weeks 1-6
- To be developed individually
- Will be marked using automated tests
- Checkpoint 1: 23 Jan – formative feedback
- Checkpoint 2: 13 Feb – summative feedback





ITECH Project

- In a group of 4, you will be responsible for the design, development and deployment of a web application using Python / Django
- This will be the main focus of lab sessions 6-10
- Choice of what to build is up to you
  - but it will have to be designed well!
  - design specification assesses this
- You will present your application in week 10 (8 minutes presentation)
- And submit the code by 13 March

Quizzes

- Lecture quizzes will aim to reinforce concepts
- Not about attendance monitoring
- But, strong correlation between attendance and performance in the final exam
- Typically two quizzes per lecture
- Can only be answered during scheduled lectures
- They are not assessed!
- But similar to types of questions asked in the final exam

Quiz: dry run

- Via your smartphone, laptop or tablet, connect to the internet using WiFi (e.g., eduroam) or 3G/4G

- In your smartphone, tablet, laptop:
  - Go to: https://classresponse.gla.ac.uk/
  - Enter your GUID and password
  - Enter the session ID

Quiz Question (ID: 1129)

How many of the following technologies have you used before?

- Python
- Django
- HTML
- CSS
- HTTP GET/POST
- XML/JSON
- XHTML
- JavaScript
- JQuery
- AJAX

Go to: https://classresponse.gla.ac.uk/
Enter your GUID and password
Enter the session ID

A. 0-2
B. 3-5
C. 6-8
D. 9-10

Development Environment

Internet Technologies

ITECH

Setting Up

- It is good Software Engineering Practice to:
  - Use a Version Control System
  - Use a Package Manager
  - Use a Virtual Environment
  - Use a Integrated Development Environment

Version Control

- Is all about managing multiple versions of documents and code
  - It is common practice in Industry and Open Source projects to use Version control
  - Essential for teams but also useful for individual projects
- There have been many VC systems, i.e.
  - CVS, Subversion (SVN), Mercurial, Git, etc

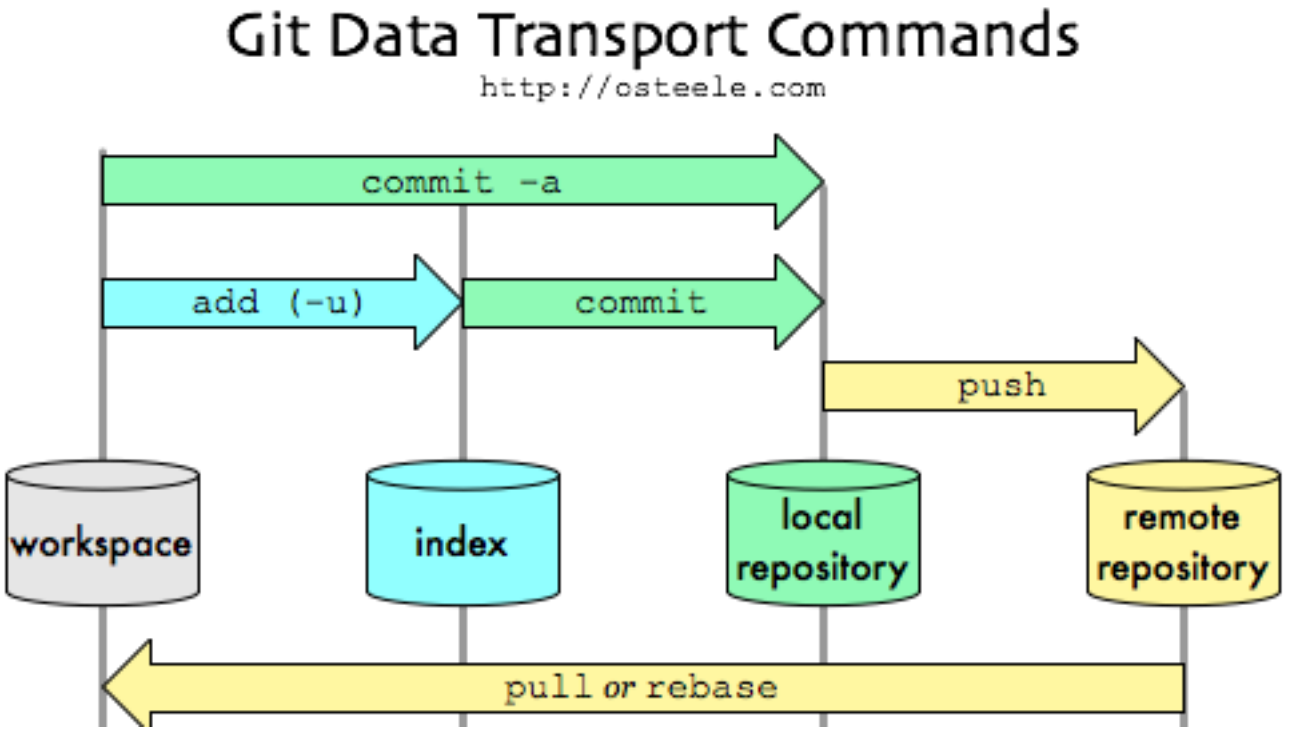
Why use Version Control?

- Access to older (working) versions of your code
- Greatly simplifies concurrent work
- Enables changes to be merged (easily)
- Keeps track of different versions and releases
- Safeguards your code against disaster
  - (assuming the repo is in the cloud)
- Enables exploratory work (branching)

Git

- Git is one of the newer VC systems which has several benefits over older ones (CVS/SVN)
  - More efficient, Better workflow, More Flexible, etc
- We will be using Git and GitHub
- Download Git from:
  - http://git-scm.com/downloads
- How to install:
  - http://stackoverflow.com/questions/315911/git-for-beginners-the-definitive-practical-guide#323764
- Get started:
  - https://www.git-tower.com/learn/git/ebook/en/mac/basics/getting-ready#start

Git – How it Works



- For ITECH, your repository will be on GitHub https://github.com/username/repo_name

Common Git Commands

- git clone <remote_repository>
  - Make a copy of the repository (done once)
- git pull
  - Get the latest changes and merge with your local repo
  - Work on your files
- git status
  - Find out what files you changed,
- git add <filename>
  - Add the files you want to commit
- git commit –m “what bugs you fixed”
  - Add the changes to the local repo
- git push
  - Uploads your changes to the remote repository

Git Tips

- Always Pull to make sure you are working on the latest version
- Commit early, Commit often, and then Push your changes frequently

- The biggest hassle is dealing with merge conflicts
  - If the remote repo has changed, it is your responsibility to merge the versions
  - So communicate that with your team

Package Manager

- PMs are software tools that automate the process of installing, upgrading and configuring software libraries.
- It tracks the packages installed and their dependencies
  - If pre-requisite packages are not installed it will install them too
- They help to overcome the nightmare of managing libraries and setting up software
  - Defined: the list of packages is defined.
  - Repeatable: easy to install the same set of libraries and versions
  - Managed: stored in the package manager and exportable

Pip: Python Package Manager

- PyPI: Python Package Index is used to install and manage Python software packages
- Pip is a recursive acronym: Pip Installs Packages
- Using Pip reduces development set up hassles
  - No need to mess around with path issues
  - No need to worry about what version of the library is used (it is recorded)
  - Easy to export and share the “requirements” i.e. the set of libraries used
  - Easy to install the same set of libraries on another machine
  - Works in conjunction with Virtual Environments

Virtual Environments

- VEs refer to any software, program or system that implements, manages and controls multiple virtual environment instances.
- When considered in terms of Software Engineering context, it is an environment that is configured to provide access to suite of libraries, settings, hardware

Python Virtual Environments

- Virtualenv: is a tool to keep the dependencies required by different projects in separate places
  - It solves the problem that you have used version X.1 in project A, but now you use X.2 in project B.
  - It isolates the different environments and lets you switch between them easily
    - It is not complete isolation
    - i.e. it doesn't simulate the production environment as the libraries when they are built are dependent on the compiler and hardware

Python Virtual Environment

- Main Advantages
  - Separation of package installation – you can use different packages sets for each project
  - Separation of python versions – you can use different python versions for each project
  - However, Virtualenv is difficult to use, but virtualenvwrapper makes things very simple.
- Main Disadvantages
  - It does not keep existing python packages installed in other environments, need to install everything again.
  - It becomes a problem when 2 or more VE start at the same time

More about…

- For more about Version Control, GIT, PIP:
  - http://betterexplained.com/articles/a-visual-guide-to-version-control/
  - http://betterexplained.com/articles/intro-to-distributed-version-control-illustrated/
  - http://betterexplained.com/articles/aha-moments-when-learning-git/
  - https://python-packaging-user-guide.readthedocs.org/en/latest/
  - http://docs.python-guide.org/en/latest/dev/virtualenvs/
  - http://www.simononsoftware.com/virtualenv-tutorial-part-2/

Lab session

- 84 machines in BO1028 and ~115 students
  - Deficit: 31 computers
- Last digit in your student id:
  - Even -> 3pm-4pm
  - Odd -> 4pm-5pm
    - **After 10 mins: first come, first served**
- Download the Lab Briefing document from the course Moodle page
- Work your way through the document
- David, Addulwhab and me will be there!

Lab session

- In this week:
  - “Labs Briefing” (Moodle)
  - “Environment Setup” document (Moodle)
  - “Rango Application” document (Moodle)
- If you have not programmed in Python before…
  - Go over Python tutorials on your own!
    - Check links o n Labs Briefing doc