Lecture 19
Web Application Frameworks

Boriana Koleva
Room: C54
Email: bnk@cs.nott.ac.uk

Overview

- Web applications overview
- Introduction to Model-View-Controller (MVC)
- Overview of web application frameworks
- Introduction to TurboGears
  - video

WPS so far

- Technologies:
  - HTML
  - CSS
  - JavaScript
  - DOM, Dynamic HTML
  - PHP
  - XML
- May seem complicated already
- But still not everything (not by far!)
- How to possibly get it all under one hood?

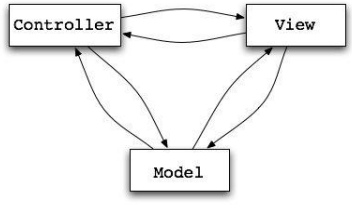
Webapps summary

- Accessed with a Web Browser (client)
- Over a network
- Code is mainly run on server
- Exception: JavaScript (also: Java applets, Flash,..)
- Data is mainly stored on server
- Webapps can be updated easily…
  ..without updating the clients!

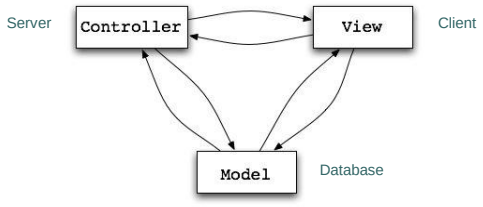
General 3 tiered structure

- First tier: client side code (web-browser), e.g. (X)HTML, JavaScript, Java applets, Flash
- Second tier: server side code, e.g. C/C++, Perl, PHP, Java, Ruby, Python
- Third tier: server side database

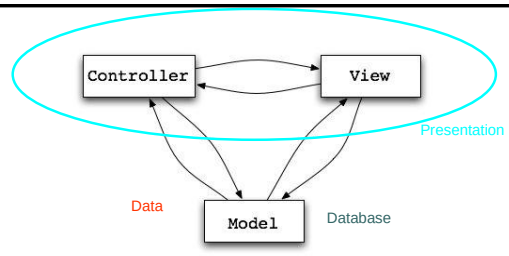
Model View Controller



- Architectural Pattern from Smalltalk (1979)
- Decouples data and presentation
- Eases the development



- First thought (ok, but not far enough):
  - Tier 1: View (Client)
  - Tier 2: Controller (Server)
  - Tier 3: Model (Database)

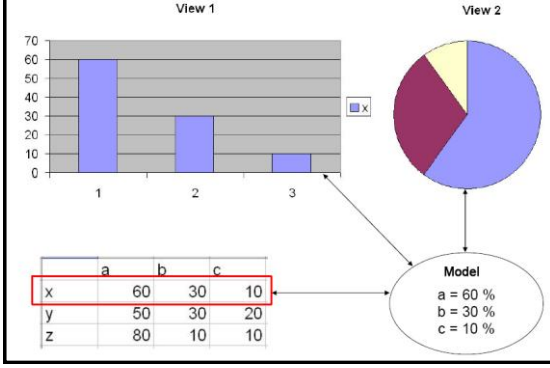


- Presentation:
  - View is the user interface (e.g. button)
  - Controller is the code (e.g. callback for button)
- Data:
  - Model is the database

Example Control Flow in MVC

- User interacts with the VIEW UI
- CONTROLLER handles the user input (often a callback function attached to UI elements)
- CONTROLLER updates the MODEL
- VIEW uses MODEL to generate new UI
- UI waits for user interaction

MVC – general example



Web application framework

- Software framework that is designed to support the development of dynamic websites
- Aims to alleviate the overhead associated with common activities
  - libraries for database access
  - templating
  - session management
  - code reuse

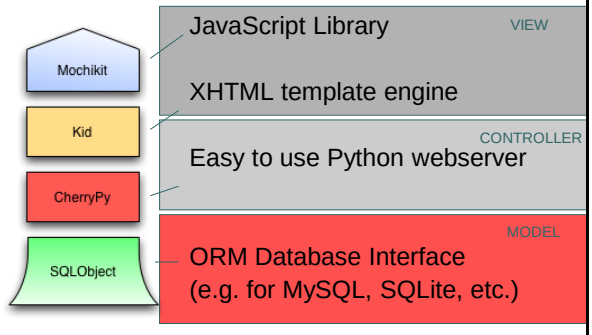
Web App Framework Architecture

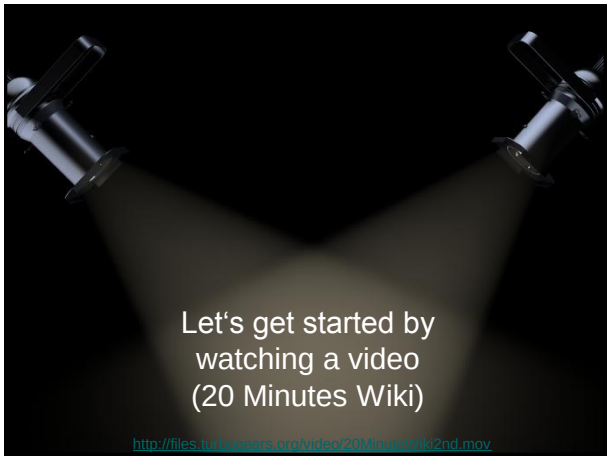
- Most frameworks follow the Model View Controller (MVC) architectural pattern
- Most MVC frameworks follow a push-based architecture
  - use actions that do the required processing, and then "push" the data to the view layer to render the results
- Alternative pull-based architecture
  - start with the view layer, which can then "pull" results from multiple controllers as needed

Popular web application frameworks

- http://en.wikipedia.org/wiki/Ruby_on_rails (Ruby)
  - http://www.rubyonrails.org/screencasts
- http://en.wikipedia.org/wiki/Cake_php (PHP)
  - http://cakephp.org/screencasts
- http://en.wikipedia.org/wiki/Turbogears (Python)
  - http://showmedo.com/videos/series?name=TurboGears20MinWiki
- http://en.wikipedia.org/wiki/Django_%28web_framework%29 (Python)
  - http://www.throwingbeans.org/django_screencasts.html
- http://en.wikipedia.org/wiki/Google_App_Engine (Python, Django)
  - http://www.youtube.com/watch?v=3Ztr-HhWX1c

Introduction to TurboGears (1.x series)





So what was that?

- Created skeleton files with startup script
- Defined Data-Model
  - created database tables from model
  - created seeding data in toolbox webapp
- Wrote View template in XHTML
- Wrote Controller code in Python
  - Index, edit, save
- At this point he had a working system
  - Several iterations to add all features
- Finally wrote AJAX code (Javascript) in template

Benefits

- Local development
  - No need to upload to server
- Quick turn around times
  - No need to compile
  - As CherryPy watches the file-system and reloads when sources are changed
- Database query and update was easy
  - No need to hand-write SQL
  - But could be done, if necessary

Summary

- Web applications
  - Client, Server, Database
  - Easy to maintain, harder to write
- Model – View – Controller
  - Eases web application development
- TurboGears
  - MVC WebApp Framework written in Python
  - www.turbogears.org