SUSS
SINGAPORE UNIVERSITY
OF SOCIAL SCIENCES

ICT239 Web Programming

Back End Development – Flask I
Seminar 4

RECAP: SEMINAR OVERVIEW

FRONT END DEVELOPMENT – HTML & CSS – LEARNING OBJECTIVES

1. Understand Document Object Model (DOM)

2. Use the Javascript Programming Language

3. Code in Javascript: Using Variables, Handling Conditions, Loops, Performing

4. String, Numeric Operations, Using Simple Arrays Data Structures

5. Use and Apply Complex Data structures - Javascript Objects

6. Use Advanced Javascript: JQuery and Client Side APIs

7. Understand the different purposes of Client side vs Server side programming

SEMINAR OVERVIEW

BACKEND DEVELOPMENT – LEARNING OBJECTIVES

1. Differentiate Static vs Dynamic Sites
2. Understand the different purposes of Client side vs Server side programming
3. Learn about Web Application Programming Frameworks
4. Deploy a Web Application - Flask

SEMINAR 4: Pre-Reading Reference

Server-side website programming

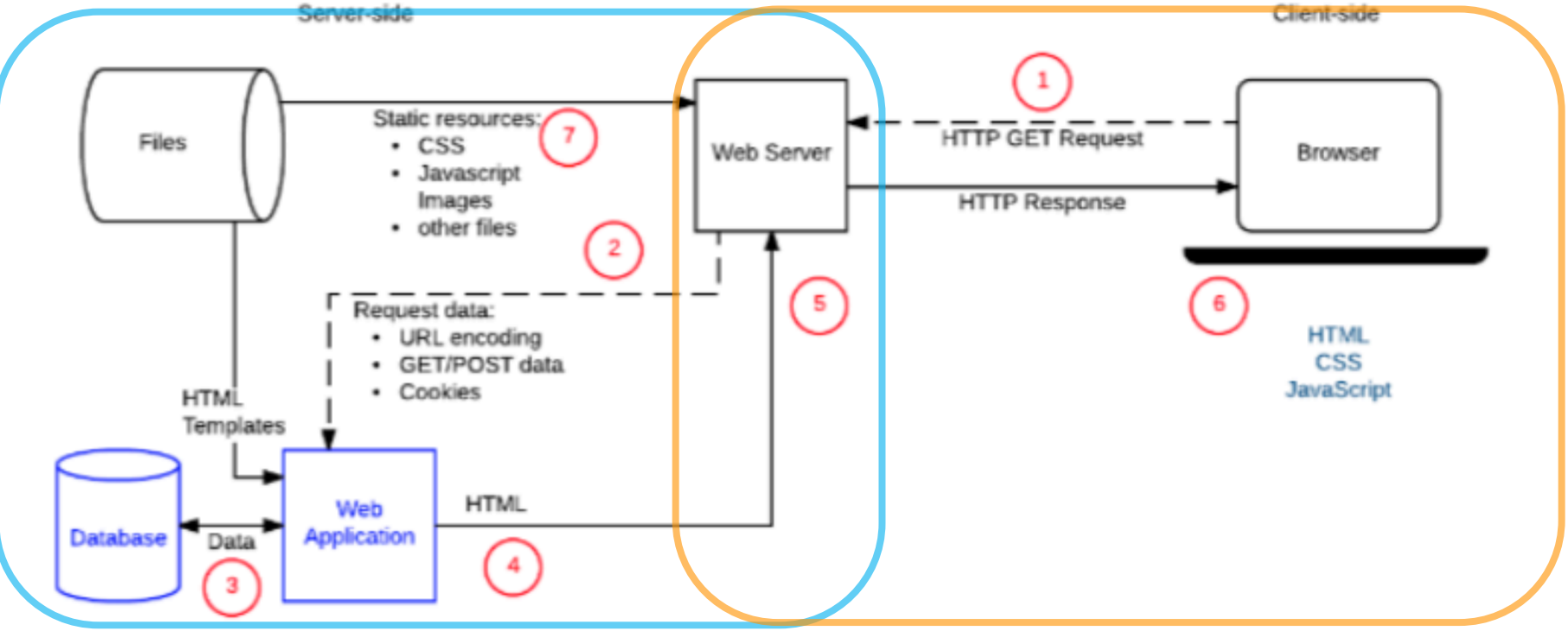
https://developer.mozilla.org/en-US/docs/Learn/Server-side (dated 28 Mar 2020)

Section	Note
First steps	First steps overview Introduction to the server-side Client-Server overview Server-side web frameworks Website security
Internet Resources	
How to build a web application using Flask and deploy it to the cloud - by Salvador Villalon	https://www.freecodecamp.org/news/how-to-build-a-web-application-using-flask-and-deploy-it-to-the-cloud-3551c985e492/

RECAP: Client and Server

Overview

http:/www.server.com/best?team=my_team&show=11



Server / Backend Programming

Figure 2: Dynamic website architecture
Source: mozilla.org

Client / Frontend Programming

https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

Introduction to Server Side

What is webserver ?

Hardware – Computer connected into Internet / Cloud Server

Software – HTTP Server software, interprets URLs, uses HTTP Protocol

Server Side programming

- Logic, routine to generate different type of response content, according to different request inputs (from client / frontend)
- Validates data
- Store session / state information
- Implements the business logic
- Read/Updates data in the database
- Returns the required data back to the client
- Notification & communications
- Data Analysis

Introduction to Server Side

Server side

Many program languages / frameworks

- Java / Spring
- C# / .NET
- Ruby / RoR
- Javascript / NodeJS
- Python / Django / Flask

Common functions / benefits

- Efficient storage and delivery of info
- Customised user experience
- Control assess to information
- Session / State Management for different concurrent users
- Notifications & Communications
- Data Analysis

Introduction to Server Side

Topic 2: Server Side Frameworks

Make developer's life easier!

- Pre-defined functions/libraries for common tasks
  - Request and Response objects handling
  - Defining URL routes and handlers
  - Database / Model operations
  - User authentications
  - Session handling
  - Request and Response objects handling
- Scaffolding / code generation
- Logging and debugging
- REST support (REST APIs)
- Unit Testing

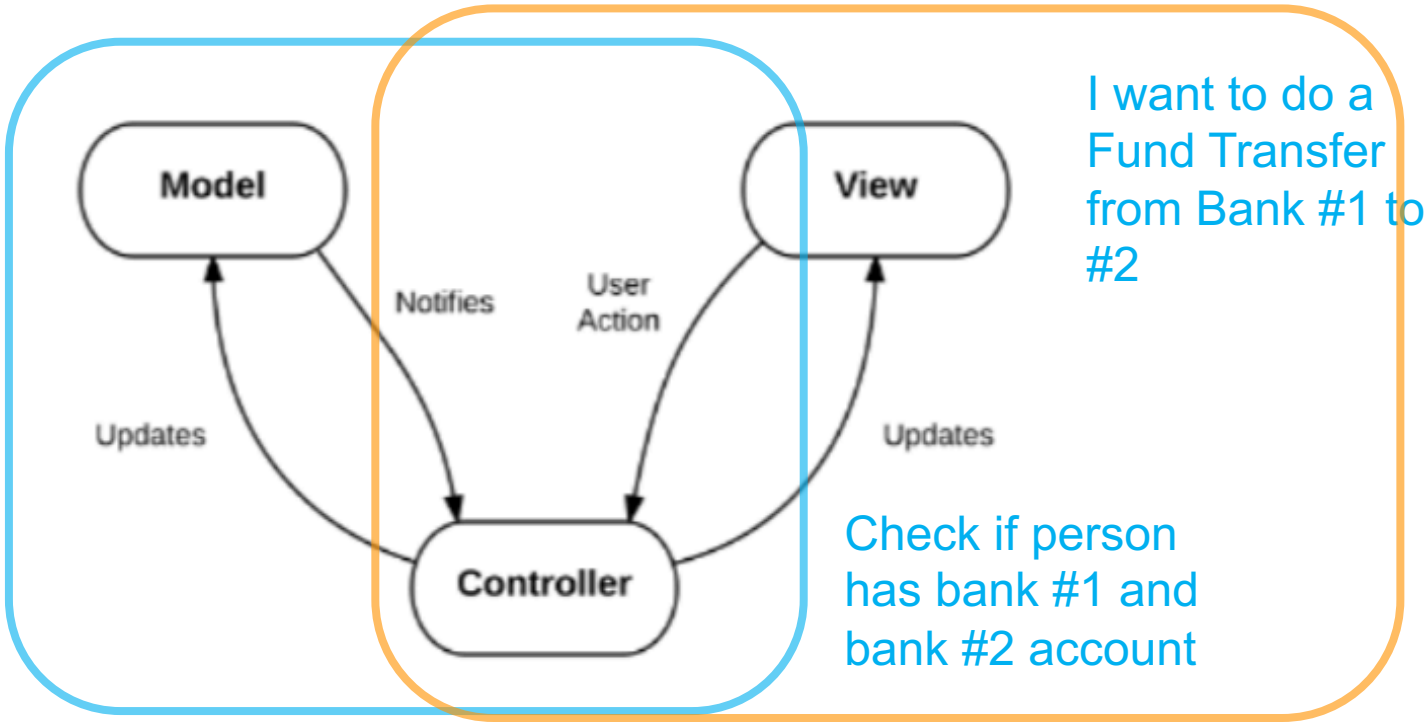
Client and Server

Model View Controller (MVC) Architecture

A software design pattern used to organise application logic into 3 parts

- Modular
- Collaboration
- Reuse

bank #1 - $100
Bank #2 + $100



I want to do a Fund Transfer from Bank #1 to #2

Check if person has bank #1 and bank #2 account

Server / Backend Programming

Figure 3: MVC architecture
Source: Wikipedia

Client / Frontend Programming

http:/www.server.com/login/123

HTTP GET/POST

HTTP Reply

Internet

HTTP Server

WSGI

Middleware

Django

Flask

```
@app.route("/login/<int:id>"):
def login(id):
```

Routing (urls.py)

```
def login(id):
…
return render_template(profile.html’,
userid=id)
```

Template

View

app.py

```
def adduser(id):
…
new_user = User(username=username)
db.session.add(new_user)
```

models.py

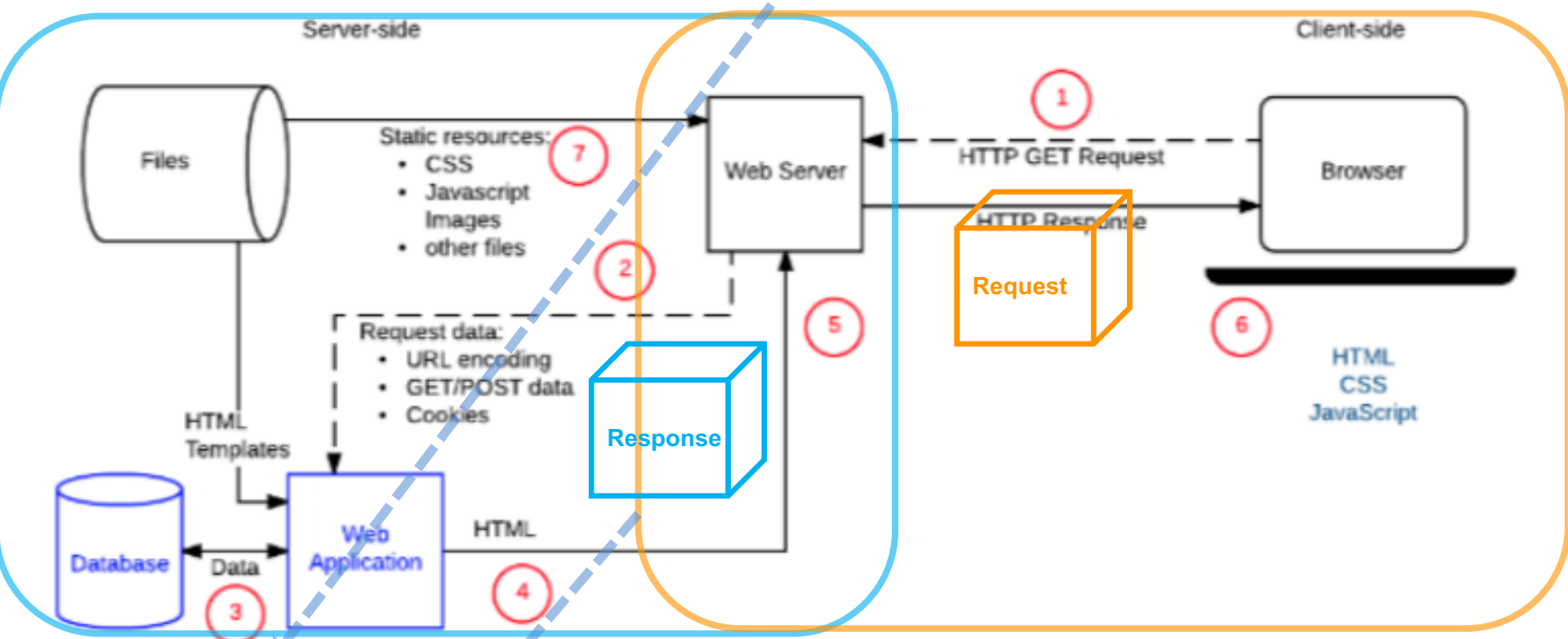
```
class User(db.Model):

    """Model for user accounts."”
    __tablename__ = 'users'

    id = db.Column(db.Integer, primary_key=True)
```

Model

Database

URLs, Routes, Controller

http:/www.server.com/best?team=my_team&show=11



Route = /best

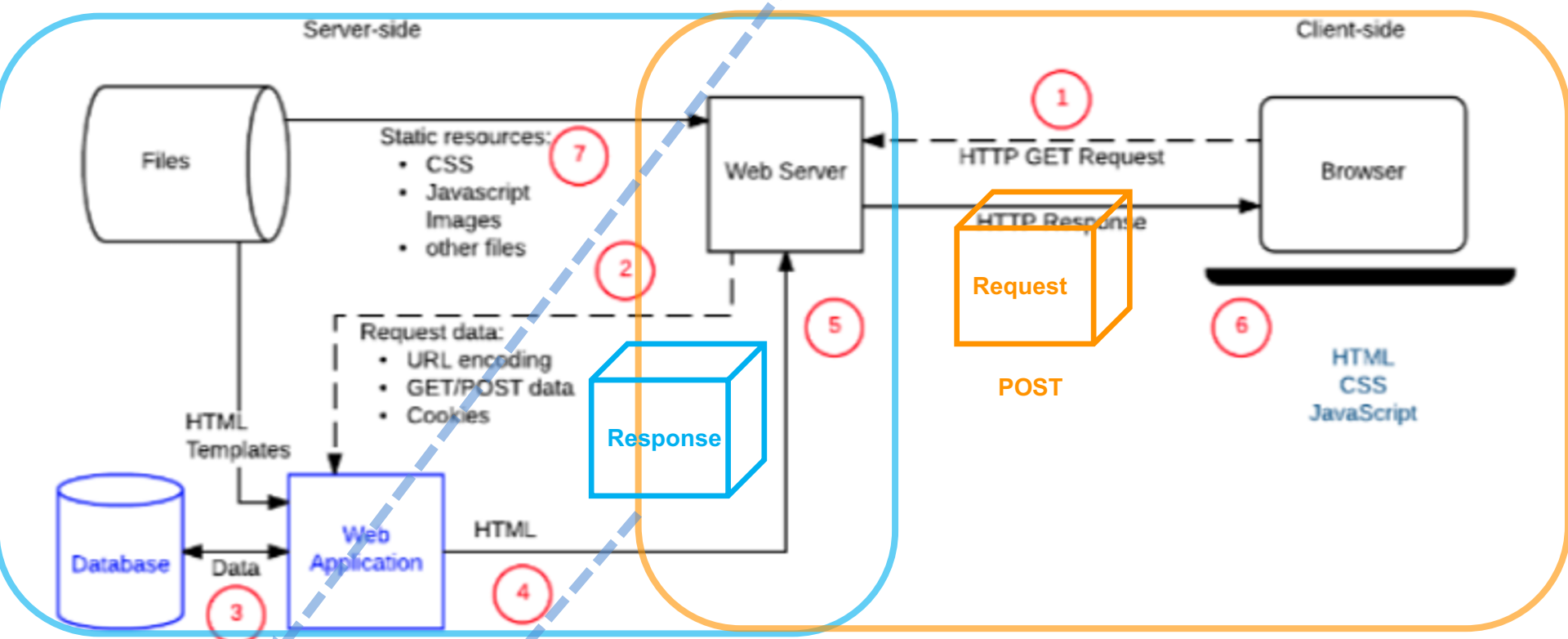
```
@app.route("/best")
def best():
return ("Here's the best team's data")
```

https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

SUSS
SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

Request and Response

http:/www.server.com/login?team=mctan011&show=11

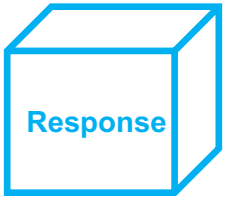


Route = /best

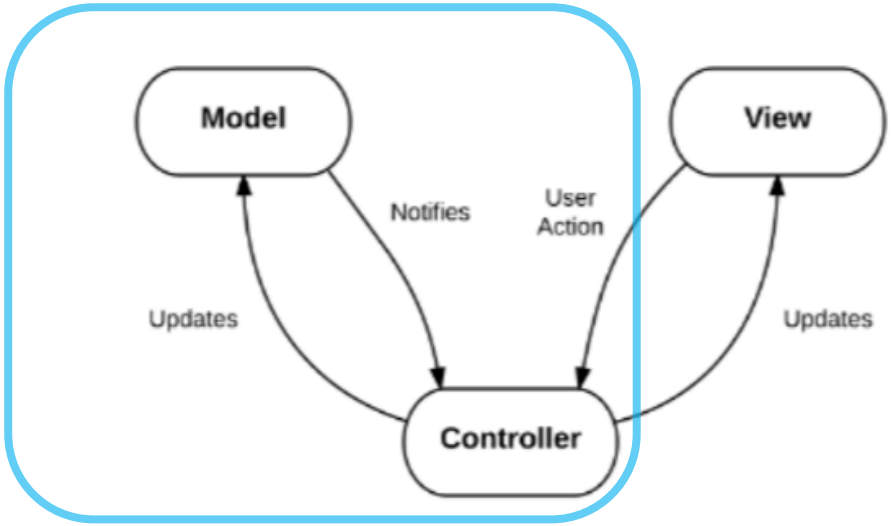
```
@app.route("/best", methods = ["POST"])
def add():
    name = request.form["team"]
    email = request.form["show"]
    return ('Information for best team and it's 11 members")
```

ORMs and Database access

http:/www.server.com/best?team=my_team&show=11

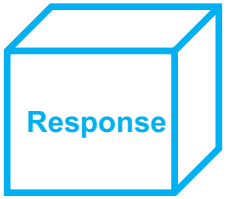



```
@app.route("/best")
def best():
    # Searches the database for teams which are ranked “best”, then displays the top 11
    db = get_db()
    cur = db.execute('select * from team where ranking=“best” limit 11’)
    team_list = cur.fetchall()
    return (”Here’s the best team’s data”)
```



Rendering Data

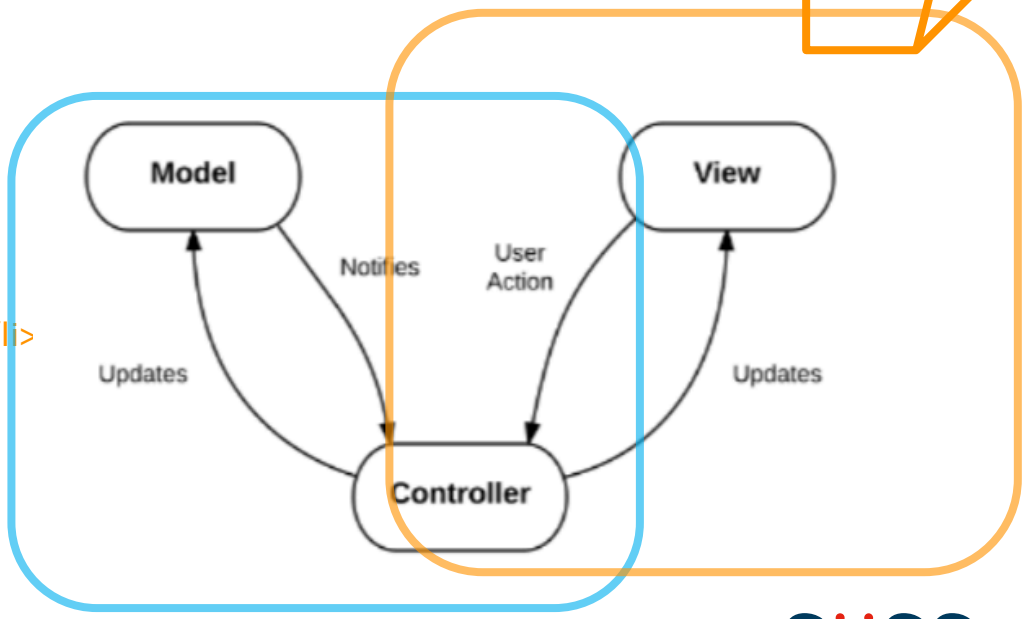
http:/www.server.com/best?team=my_team&show=11




```
@app.route("/best")
def best():
    # Searches the database for teams which are ranked “best”, then displays the
top 11
    …
    return (team_list)
```

templates

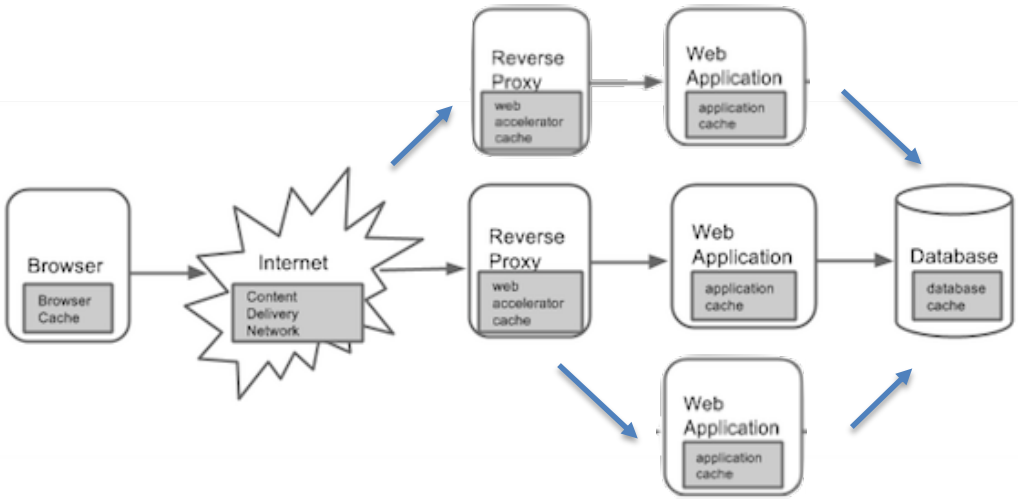
```
<html>
<body>
        <!-- check if variable exists -->
        {% if teams %}
        <ul>
                {% for team in team_list %}
                        <li>{{ team.team_name }}</li>
                {% endfof %}
        </ul>
        {% endif %}
</body>
</html>
```



Selection of Server Side Frameworks

Selection of Server-side Frameworks

- Effort to learn (depends on individual)
- Productivity
  - Framework purpose (why was it created?)
    - RoR - project mgmt., Django – publishing, Java Spring – complexity of J2EE
  - Opinionated (vs non)
  - Batteries included vs BYO (Build your own)
  - Support good coding practice (MVC vs spaghetti code)
- Performance
  - C++ / Python speed
  - Java – memory intensive
- Caching
- Scalability
- Web Security



https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

Chapter 1: Flask Framework Deployment

Topic 1: Set up and Dependencies

- python – programming language
- pip - is a package-management system used to install and manage software packages written in Python.
- virtualenv - https://virtualenv.pypa.io – tool for creating isolated versions/environment for python
- http://flask.pocoo.org/docs/1.0/

```
#app.py
from flask import Flask
app = Flask(__name__)
@app.route("/")
def main():
   return "Hello World!"

if __name__ == "__main__":
   app.run(debug=True)


FLASK_APP=app.py flask run

python app.py
```

python app.py
flask_app=app.py
flask run

SEMINAR OVERVIEW

BACKEND DEVELOPMENT – LEARNING OBJECTIVES

1. Differentiate Static vs Dynamic Sites
2. Understand the different purposes of Client side vs Server side programming
3. Learn about Web Application Programming Frameworks
4. Deploy a Web Application - Flask

TODO BEFORE NEXT SEMINAR

Reminder

- Read Study Unit 4,5
- References
  - https://developer.mozilla.org/en-US/docs/Learn/Server-side
  - http://flask.pocoo.org/docs/1.0/
  - Try out the exercises! (In the Study Unit, Mozilla site)
- Use your Canvas resources
  - Study Guide
  - Discussion Forums
  - Course Textbook / Google

For AJAX to pass frontend parameters to backend

https://api.jquery.com/jquery.ajax/

SUSS
SINGAPORE UNIVERSITY
OF SOCIAL SCIENCES

Thank You.