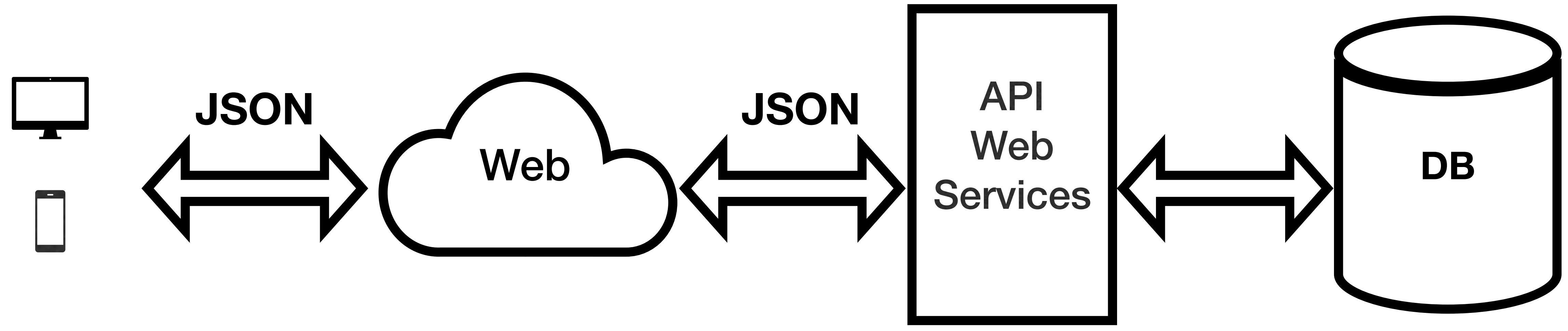
Building APIs with GraphQL in Python

Raj Sunderraman
Professor
Department of Computer Science
Georgia State University

Modern Web/Mobile Application Architecture



Front End
(UI Client)

HTML5
Javascript
jQuery

Angular
React

iOS/Swift
Android/Java

Middleware
(Server)

REST
GraphQL

Python
Java
node.js
Ruby
Golang

Back End
(Server)

MySQL
Postgres
Oracle
MongoDB
neo4J
...

Classroom List: Atlanta Campus

Bldg	Room	Cap	Layout	Media	Restriction	Type	Dept
ADHOLD	106	56	Fixed/Tiered Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,		G	
ADHOLD	107	56	Fixed/Tiered Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,		G	
ADHOLD	12	55	Fixed/Tiered Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,		G	
ADHOLD	2	56	Fixed/Tiered Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,		G	
ADHOLD	202	56	Fixed/Tiered Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,		G	
ADHOLD	203	30	Tablet Armchairs	IWS, LAC, VP, DC, BR, DVD, WT,		G	
ADHOLD	204	45	Chairs & Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,	2	G	
ADHOLD	205	30	Tablet Armchairs	IWS, LAC, VP, DC, BR, DVD, WT,		G	
ADHOLD	206	60	Fixed/Tiered Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,		G	
ADHOLD	212	60	Fixed/Tiered Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,		G	
ADHOLD	213	45	Chairs & Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,	2	G	
ADHOLD	214	60	Fixed/Tiered Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,		G	
ADHOLD	223	60	Fixed/Tiered Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,		G	
ADHOLD	224	45	Chairs & Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,	2	G	
ADHOLD	225	60	Fixed/Tiered Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,		G	
ADHOLD	229	56	Fixed/Tiered Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,		G	
ADHOLD	230	30	Tablet Armchairs	IWS, LAC, VP, DC, BR, DVD, WT,		G	
ADHOLD	231	45	Chairs & Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,	2	G	
ADHOLD	233	60	Fixed/Tiered Tables	IWS, LAC, VP, DC, BR, DVD, WT, NP,		G	

Building Codes

ADHOLD - Adherhold Learning Center
ARTS - Art & Humanities Bldg
CLSO - Classroom South Bldg
COE - College of Education
KELL - Kell Hall
LANGDL - Langdale Hall
LIBSO - Library South
PSC - Petit Science Building
SPARKS - Sparks Hall
URBAN - Urban Life Bldg

Media Codes

BR - Blu-Ray Player
CB - Chalk Board
DC - Document Camera
DVD - DVD Capabilities through IWS or Player
IWS - Instructor Workstation (Network)
LAC - Laptop to AUX Connectivity
LC - Lecture Capture/Video Confrencing Camera and Mic
M - Map
MWB - Mobile White Board
NP - Network Port w/ charging station at each seat
PS - Plasma Screen
SWS - Student Workstation
TP - Transparency Projector
TV - TV for DVD/VCR
VCR - VHS Capabilities through Player or Combo
VP - Video Projector (Digital)
WB - White Board
WP - Wireless presentation system
WT - Wall Talker

Restriction Codes

1 - After pick-off COAS
2 - After pick-off COB
3 - After pick-off COE
! - OCS
R - Restricted to Academics only
S - Swipe Card Access

https://bitbucket.org/rajbucket28/graphql-classrooms

Look at the following directories:

sql
  contains MySQL table definitions

load-data
  contains csv files
  Python programs to read these files and create MySQL insert statements

http://tinman.cs.gsu.edu/~raj/rooms/static

REST Web Services

REpresentational State Transfer
(Roy Fielding, 2000, PhD Dissertation UC Irvine)

- **Resources (objects in the backend)**

endpoints:

- **URL pointing to a resource**
- **Request Verbs (GET, POST, PUT, DELETE)**
- **Request Headers (type of data - JSON, Authorization Tokens)**
- **Request Body (Data associated with POST request)**

- **Response Body (Data requested from server)**
- **Response Status Codes (200: no error, 404: error)**

http://localhost:5000/classroom/api/v1.0/rooms/CLSO/400
using GET to retrieve room details
response body would contain `{“bldg”:”CLSO”,”rno”:”400”,”cap”:30,…}`

http://localhost:5000/classroom/api/v1.0/rooms/room
using POST to add a room
request body contains `{“bldg”:”CLSO”,”rno”:”222”,”cap”:40,…}`

REST Web Service

```python
from flask import Flask, jsonify
from flask import abort
from flask import make_response
from flask import request
import mysql.connector as mysql

app = Flask(__name__)

@app.route(‘/classroom/api/v1.0/buildings/',
           methods=['GET'])
def get_buildings():
  db = mysql.connect(
    host="localhost",
    database="raj",
    user="raj",
    passwd="r123",
    auth_plugin='mysql_native_password'
  )
  query = "select bcode,bname from BUILDING "
  cursor = db.cursor()
  cursor.execute(query)
  records = cursor.fetchall()
  bldgs = []
  for record in records:
    bldgs.append({'bldg':record[0],
                  ‘bname':record[1]})
  result = {'buildings': bldgs}
  cursor.close()
  db.close()
  return jsonify(result)
```

```python
@app.errorhandler(404)
def not_found(error):
    return make_response(
      jsonify({'error': 'Not found'}), 404)

if __name__ == '__main__':
    app.run(host='localhost',debug=True)
```

http://flask.palletsprojects.com

Flask
web development,
one drop at a time

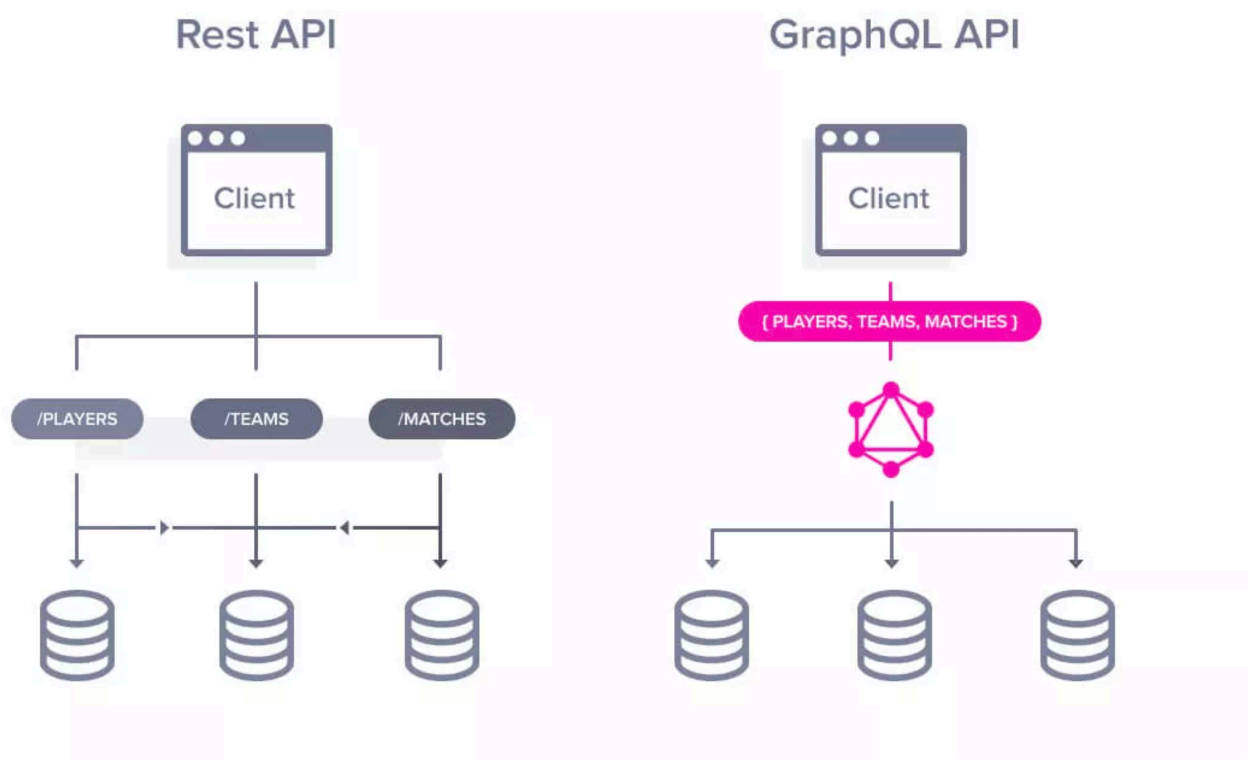
GraphQL

http://howtographql.com

- GraphQL is a new API standard that provides an efficient, powerful, and flexible alternative to REST
- Developed and open-sourced by Facebook in 2015
- Enables *declarative data fetching*
- GraphQL is <u>not</u> a query language for databases; it is a query language for APIs
- Advantages of GraphQL over REST
    - No more over-fetching and under-fetching
    - Rapid product iterations on the front-end
    - Insightful analytics on the back-end
    - Benefits of a Schema and Type-System

GraphQL vs REST

REST	GraphQL
Multiple Endpoints	**Single Endpoint**
`GET /users/1`	`POST /graphql`
`POST /product`	`{room {rno desc}}`
JSON Data	**JSON Data**
Any Server-side Language	**Any Server-side Language**
Any Front-end Framework	**Any Front-end Framework**
Stateless (no session info)	**Stateless (no session info)**
URL Driven	**Query Driven**



GraphQL Types on Server

```
class Room(graphene.ObjectType):
    bldg = graphene.String()
    rnumber = graphene.String()
    cap = graphene.Int()
    layout = graphene.String()
    rtype = graphene.String()
    dept = graphene.String()
    media = graphene.List(Media)
```

```
class Media(graphene.ObjectType):
    mcode = graphene.String()
    description = graphene.String()
```

GraphQL Queries from Client

```
query q1 {room (building: “CLSO”,
                rno: “206”)  {
    cap
    layout
    rtype
    dept
    media {
      code
      description
    }
  }
}
```

```
mutation m1 {
  createRoom (building: “CLSO”, rno: “999”
              cap: 44, layout: “Round Tables”,
              rtype: “G”, dept: “”,
              media: [“IWS”,”LAC”])  {
     ok
     bldg
     rnumber
  }
}

mutation m2 {
  updateRoomCapacity (building: “CLSO”,
                      rno: “999”
                      cap: 55)  {
     ok
     bldg
     rnumber
  }
}
```

GraphQL Server in Python

rooms.py

GraphiQL