FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
PHILOSOPHISCHE FAKULTÄT UND FACHBEREICH THEOLOGIE

The ABC of APIs

Introduction to REST APIs and the MEAN Stack

https://github.com/martialblog/restful-api-presentation

What is an API?

An Application Program Interface (API) is a set of routines, protocols, and tools for building software applications.

- Systemcalls
- POSIX
- Programming Libraries (Java Database Connectivity API)
- Web APIs (REST, SOAP)
    - REST is an architecture style for designing (networked) applications

A RESTful Call Example

```
~/ curl -i -X GET https://api.github.com/users/martialblog

HTTP/1.1 200 OK
Server: GitHub.com
Date: Mon, 13 Dec 2016 13:37:00 GMT
Content-Type: application/json; charset=utf-8
{
  "login": "martialblog",
  "id": 7090372,
  "repos_url": "https://api.github.com/users/martialblog/repos",
  "name": "Markus Opolka",
  "blog": "https://www.martialblog.de",
  "location": "Germany",
  "email": "markus@martialblog.de",
  "public_repos": 21,
  "followers": 3,
}
```

The RESTful API

The **REST** (Representational State Transfer) API principles[1] are:

- Client-Server
- Code on Demand
- Stateless Interactions
- Cacheable
- Layered System
- Uniform Interface

In real-life usually implemented with **HTTP**

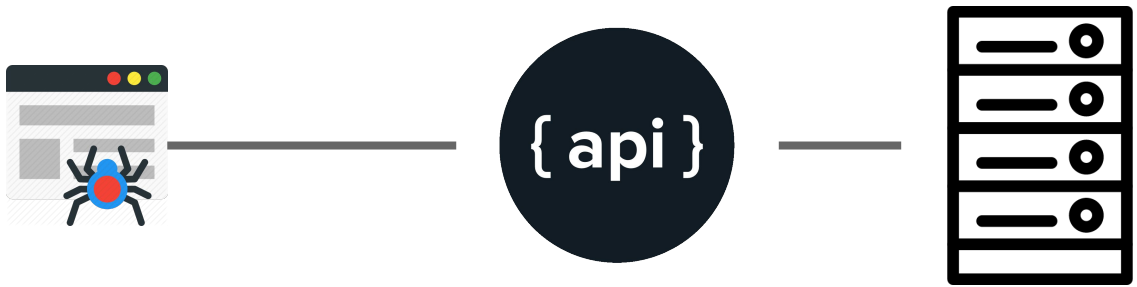
A Short HTTP Primer

- HTTP is an asymmetric request-response client-server protocol
- HTTP defines methods for desired action on the identified resource (URL)
  - GET
  - POST
  - PUT/PATCH
  - DELETE
- In REST terminology called: CRUD (create, read, update, delete)
- Client receives Status Codes for these action
  - 200 OK, 201 Created, 404 Not Found

```
~/ netcat 127.0.0.1 8000
GET /index.html HTTP/1.1

HTTP/1.1 200 OK
Server: SimpleHTTP/0.6 Python/3.5.2
Content-type: text/html
```

Why use REST APIs?

- Simple, unified communication
- Decoupling of server and client
- Reformulate Applications as a Server-Client Architecture
  - Graphical User Interfaces (Angular)
  - Providing Internal Services/Functions (http://nlptools.atrilla.net/)
  - Application Example: Web Crawlers (https://lateral.io/)



How to REST

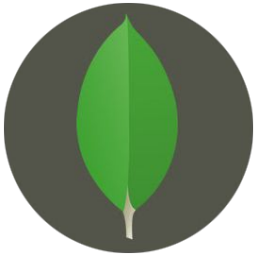
Many programming languages offer API Frameworks:

```python
#PYTHON
from flask import request

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        do_the_login()
    else:
        show_the_login_form()
```

```javascript
//NodeJS

app.route('/book')
  .get(function (req, res) {
    res.send('Get a random book')
  })
  .post(function (req, res) {
    res.send('Add a book')
  })
  .put(function (req, res) {
    res.send('Update the book')
  })
```

The MEAN Stack

NodeJS



MongoDB

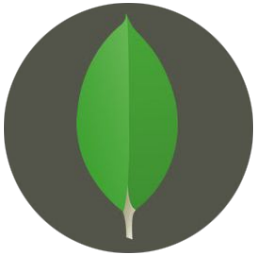
ExpressJS

Angular

JavaScript: *one Language to rule them all*

The MEAN Stack

NodeJS



MongoDB

ExpressJS

Angular

The MEAN Stack - NodejS

Server side JavaScript environment

- JavaScript and consistent models across
- Asynchronous I/O
- Single threaded but highly scalable
- Large number of modules available

The MEAN Stack - MongoDB

Schemaless NoSQL database

- Saves data in binary JSON format
- Overall better (writing) performance

However: Not easy way to join tables and other “Relational DB” features are not available

The MEAN Stack - ExpressJS

Lightweight web applications framework

- Simple to create robust APIs
- Routing and templating
- Everything you need from a modern Webserver


```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
  res.send('Hello World!')
})
```

The MEAN Stack - Angular

JavaScript frontend framework developed by Google

- Modern framework with lots of support
- For web and cross-platform desktop applications

Examples:

https://www.madewithangular.com

Literature

[1] Fieldings, Roy (2000): *Architectural Styles and the Design of Network-based Software Architectures.* [Link].

[2] Grinberg Miguel (2013): *Designing a RESTful API with Python and Flask*. [Link].

Markus Opolka

https://github.com/martialblog

markus.opolka@fau.de