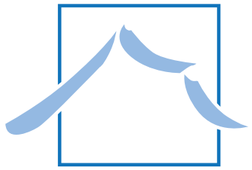
Client-Server Communication

iOS Praktikum WS14/15

Session 08

“Tell me and I will forget.
Show me and I will remember.
Involve me and I will understand.
Step back and I will act.”

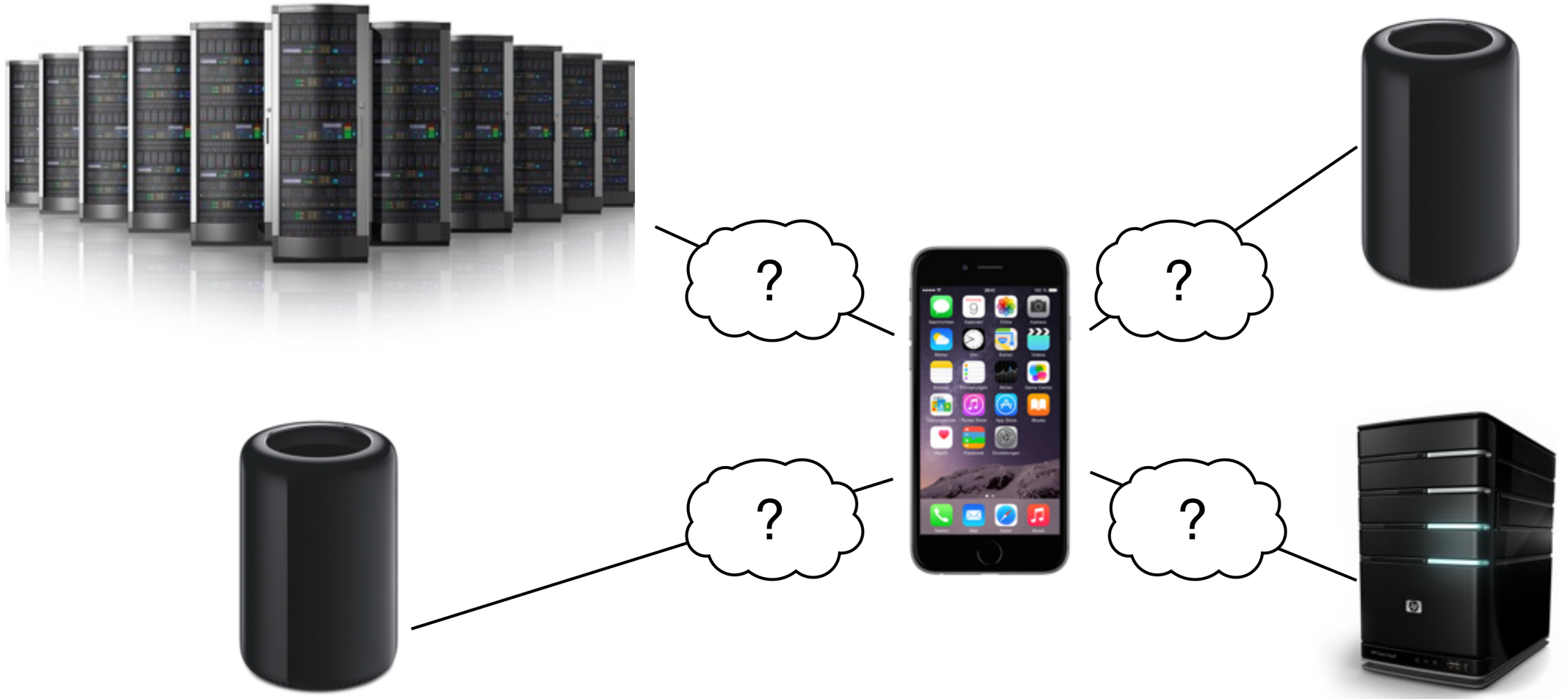
Copyright, Content & Referrals and Links

Technische Universität München   Lehrstuhl für Angewandte Softwaretechnik

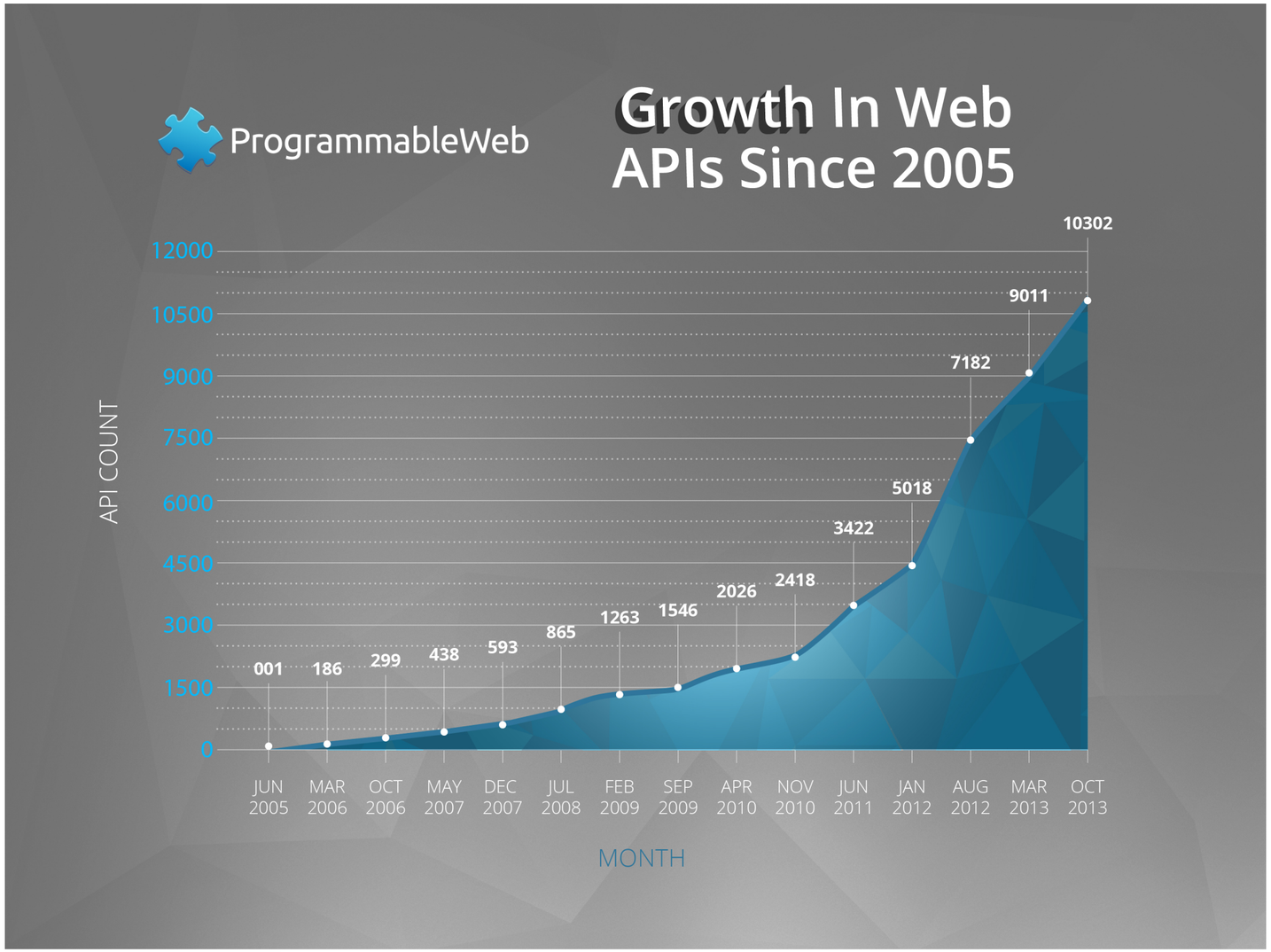
Outline

- **Motivation**
- Basic Terms
- NSURLSession
- REST
- AFNetworking
- Final Exercise

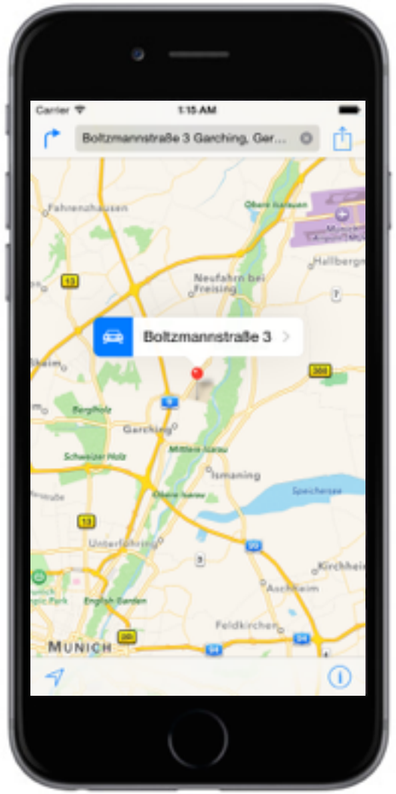
Motivation

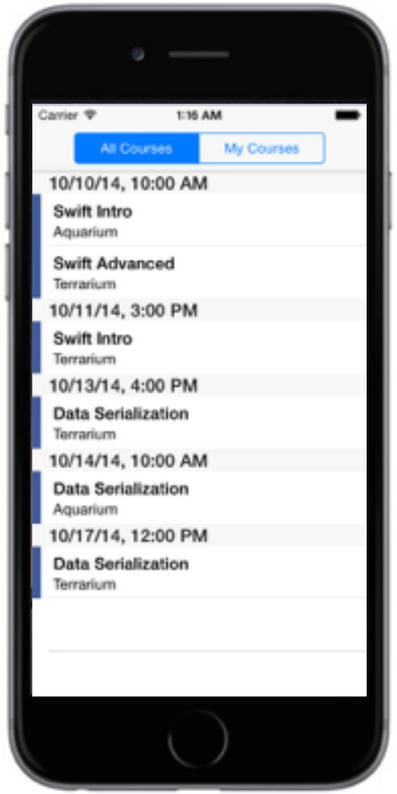


Motivation

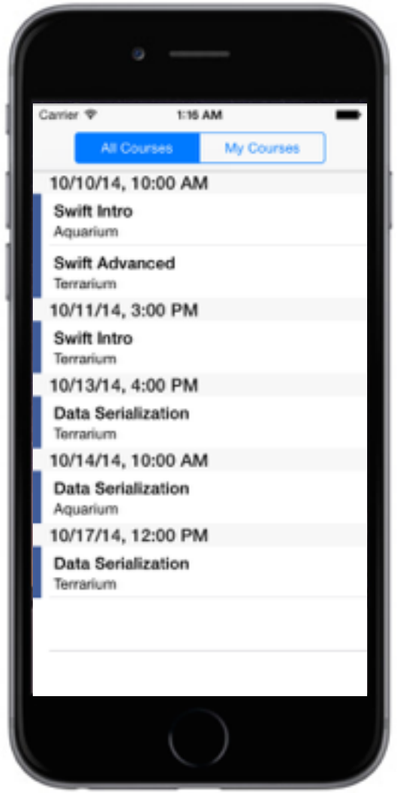

ProgrammableWeb
Growth In Web APIs Since 2005
API COUNT
12000
10500
9000
7500
6000
4500
3000
1500
0
001
186
299
438
593
865
1263
1546
2026
2418
3422
5018
7182
9011
10302
JUN 2005
MAR 2006
OCT 2006
MAY 2007
DEC 2007
JUL 2008
FEB 2009
SEP 2009
APR 2010
NOV 2010
JUN 2011
JAN 2012
AUG 2012
MAR 2013
OCT 2013
MONTH

Example Applications





Basic Workflow



Request

Response

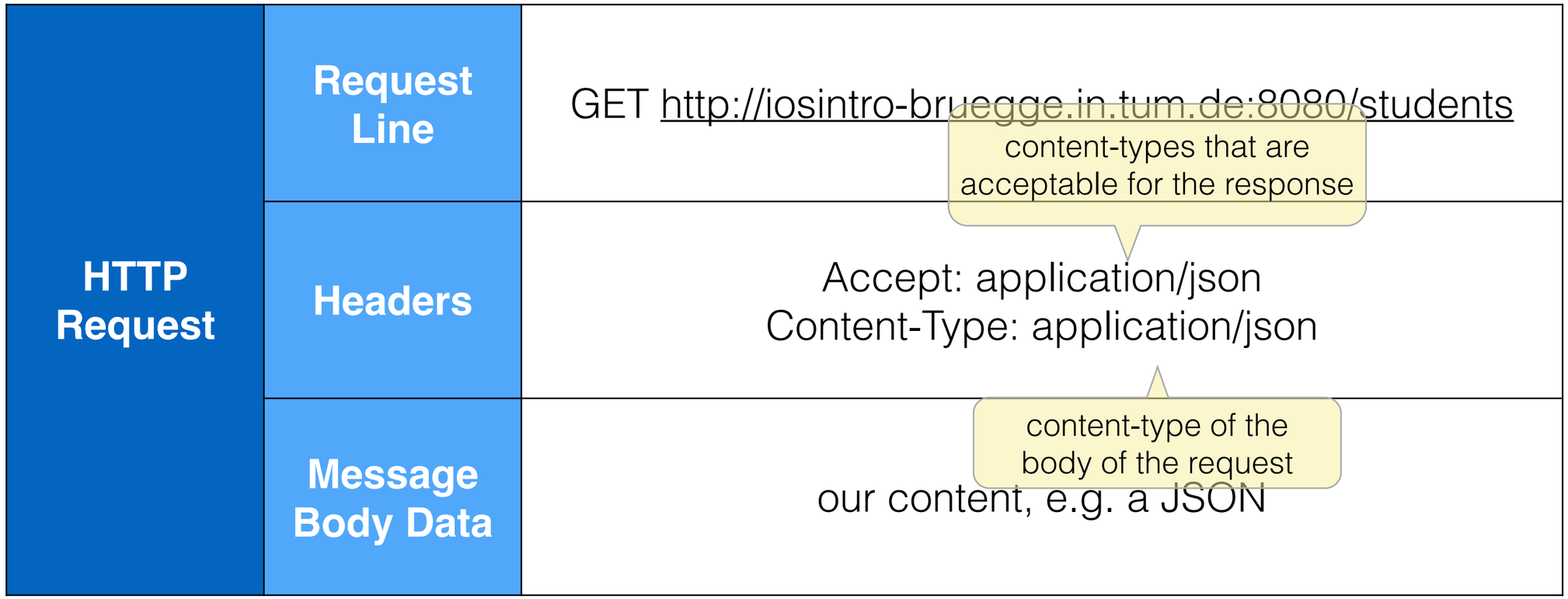
Outline

- Motivation
- **Basic Terms**
- NSURLSession
- REST
- AFNetworking
- Final Exercise

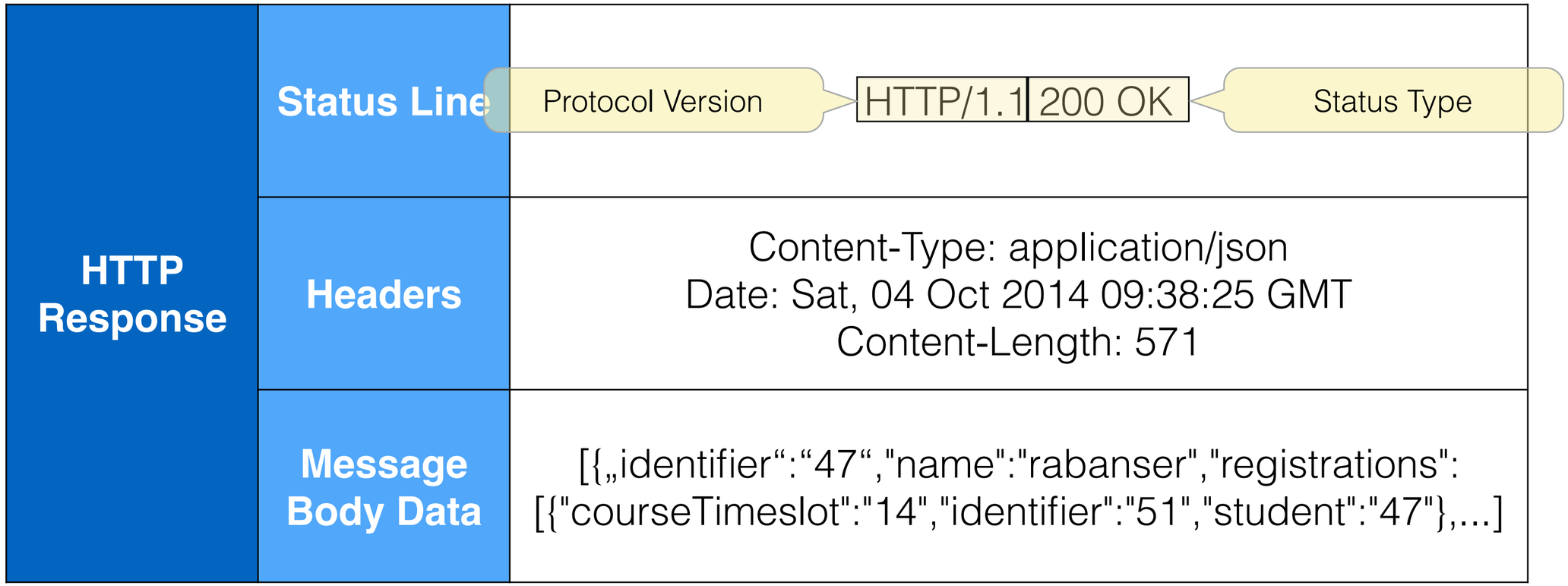
Basic Terms

- **Request**: sent by the client to the server when he needs its services
- **Response**: sent by the server after receiving a request
- API
  - Application Programming Interface
  - Defines inputs and outputs of a software component -> Server

Request


HTTP Request
Request Line
GET http://iosintro-bruegge.in.tum.de:8080/students
content-types that are acceptable for the response
Headers
Accept: application/json
Content-Type: application/json
content-type of the body of the request
Message Body Data
our content, e.g. a JSON

Response


HTTP Response
Status Line
Protocol Version
HTTP/1.1
200 OK
Status Type
Headers
Content-Type: application/json
Date: Sat, 04 Oct 2014 09:38:25 GMT
Content-Length: 571
Message Body Data
[{„identifier“:“47“,"name":"rabanser","registrations":
[{"courseTimeslot":"14","identifier":"51","student":"47"},...]

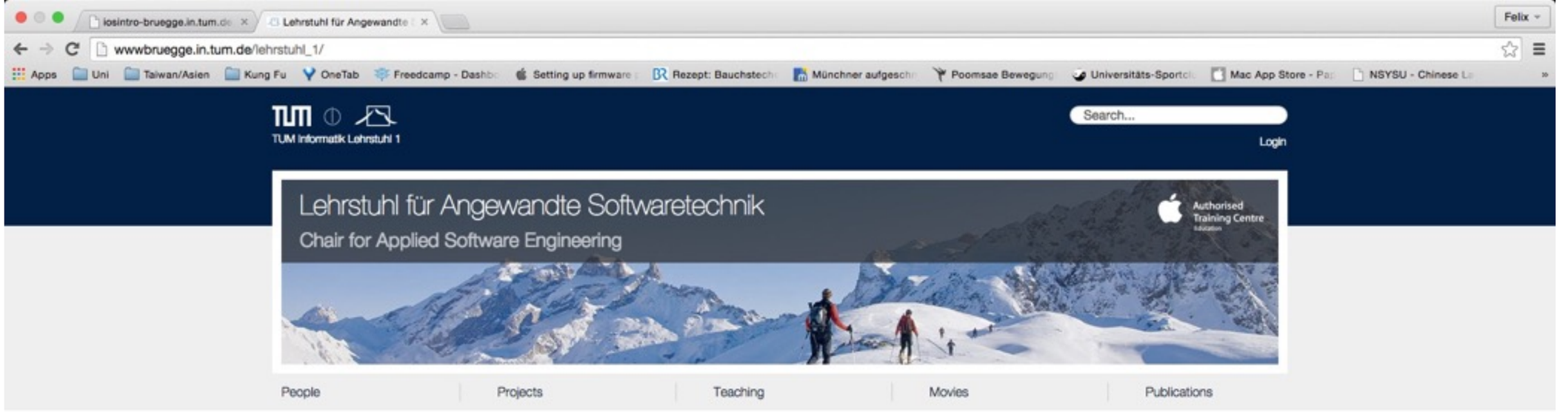
A typical URL

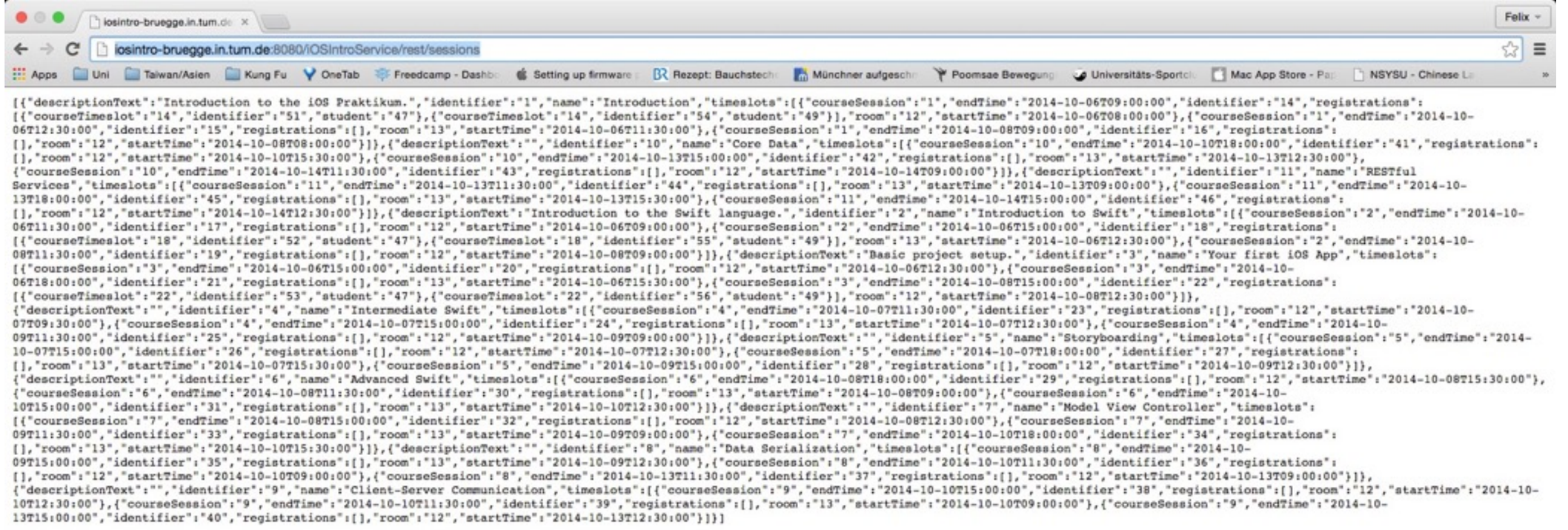
- http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students?username=user&password=pw
- Scheme: *http*
- Domain: *iosintro-bruegge.in.tum.de*
- Port: *8080*
- Path: *iOSIntroService/rest/students*
- Endpoint of our API: *students*
- Query String: *username=user&password=pw* with the values *user* and *pw* for the key *username* and *password*
- scheme://domain:port/path?query_string#fragment_id

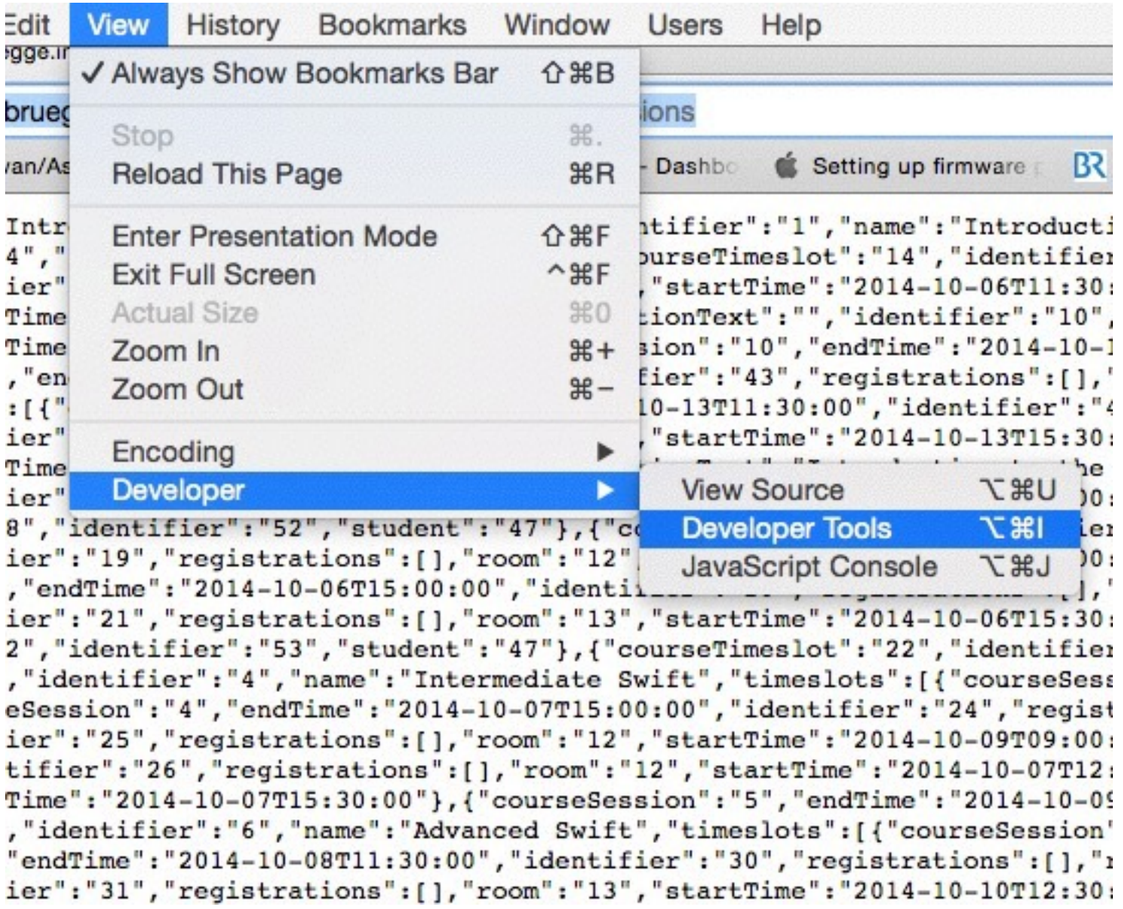
Let’s try this in the browser

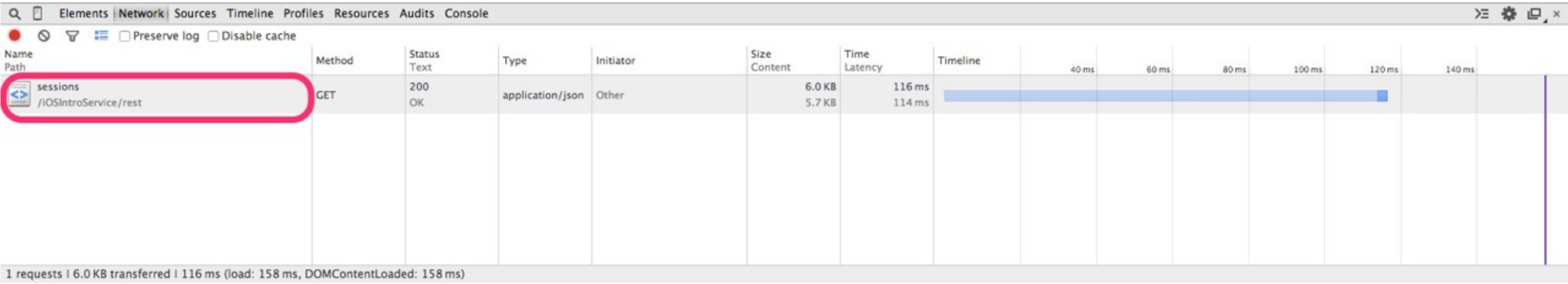
Preferably Chrome 😊

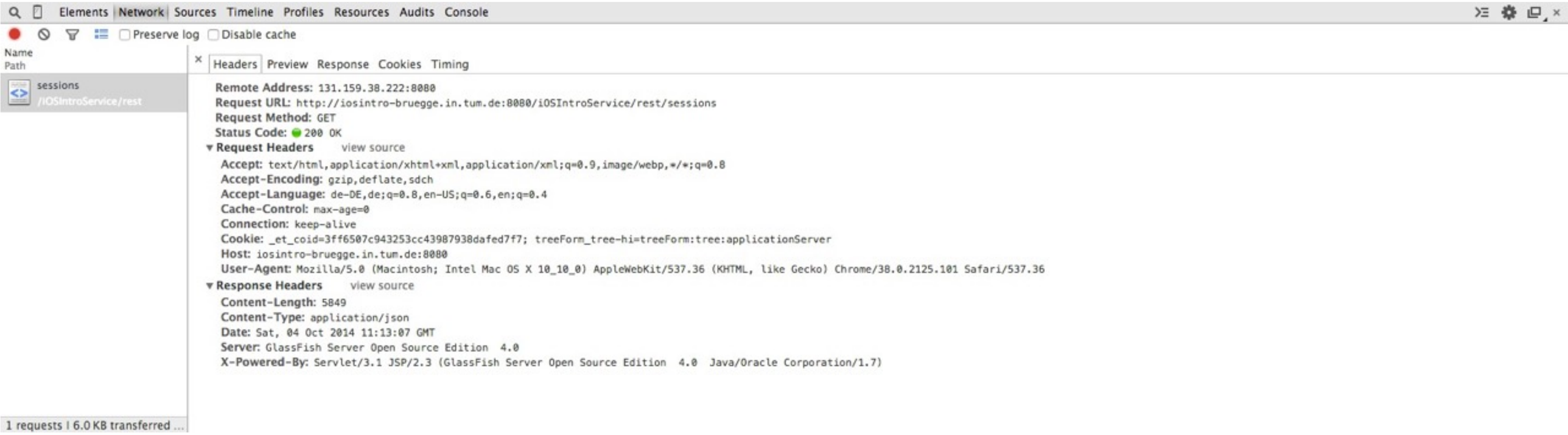
http://wwwbruegge.in.tum.de/lehrstuhl_1/
http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students


iosintro-bruegge.in.tum.de
Lehrstuhl für Angewandte
Felix
wwwbruegge.in.tum.de/lehrstuhl_1/
Apps
Uni
Taiwan/Asien
Kung Fu
OneTab
Freedcamp - Dashbc
Setting up firmware
Rezept: Bauchstech
Münchner aufgeschr
Poomsae Bewegung
Universitäts-Sportcl
Mac App Store - Par
NSYSU - Chinese La
TUM Informatik Lehrstuhl 1
Search...
Login
Lehrstuhl für Angewandte Softwaretechnik
Chair for Applied Software Engineering
Authorised Training Centre
People
Projects
Teaching
Movies
Publications
Best Paper Award in MSR Conference
Posted by Emitzá Guzmán Ortega on 23.06.2014 in Blog
MSR 2014
The paper "Sentiment Analysis of Commit Comments in GitHub: An Empirical Study" by Emitza Guzman, David Azócar and Yang Li won the best paper award in the challenge category of the Mining Software Repositories Conference (MSR 2014). The conference was colocated with the International Conference on Software Engineering (ICSE 2014) in Hyderabad India. In the paper the authors studied the relationship between sentiments expressed in commit comments and other factors such as project programming language, time and day of the week, team geographical distribution and project approval. The study was done with GitHub data provided by the mining challenge.
Gesucht: Coaches für das iOS Praktikum (WS14/15)
Posted by Lukas Alperowitz on 14.06.2014 in Blog
Für das iOS Praktikum im WS14/15 suchen wir Coaches, die bei der Betreuung der Projektteams helfen und/oder projektübergreifende Aufgaben (z.B. Continuous Delivery, Code Quality) übernehmen. Coaches tragen die Verantwortung für ein Team und verbessern Ihre persönlichen Projektmanagement Fähigkeiten in einem agilen Projekt mit einem realen Kunden und echten Lieferungen. Interessierte Student(inn)en können sich für das Seminar oder Praktikum Advanced Project Management bewerben.
Weitere Infos und Details zur Bewerbung sind auf dieser Webseite zusammengefasst. Bei Fragen wendet euch an Lukas Alperowitz.
Mission Statement
Developing and managing software is always on the move. Continuously growing complexity and shortened development cycles require high flexibility, new ideas and the courage to challenge traditional approaches. This is what we aim in our research projects and teaching courses. Together with industrial and research partners, we develop, evaluate and implement new methods and solutions to support practitioners in dealing with software engineering today's challenges.
Teaching and training students in topics around software engineering is our main concern. We offer lectures, seminars and practical courses in real projects and with real customers, following a learning by doing approach. Our students can prepare themselves for professional life, gain much theoretical and practical knowledge and have a lot of fun.
Research Topics


iosintro-bruegge.in.tum.de
Felix
iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/sessions
Apps
Uni
Taiwan/Asien
Kung Fu
OneTab
Freedcamp - Dashb
Setting up firmware
Rezept: Bauchstech
Münchner aufgesch
Poomsae Bewegung
Universitäts-Sportcl
Mac App Store - Pap
NSYSU - Chinese La
[{"descriptionText":"Introduction to the iOS Praktikum.","identifier":"1","name":"Introduction","timeslots":[{"courseSession":"1","endTime":"2014-10-06T09:00:00","identifier":"14","registrations":
[{"courseTimeslot":"14","identifier":"51","student":"47"},{"courseTimeslot":"14","identifier":"54","student":"49"}],"room":"12","startTime":"2014-10-06T08:00:00"},{"courseSession":"1","endTime":"2014-10-
06T12:30:00","identifier":"15","registrations":[],"room":"13","startTime":"2014-10-06T11:30:00"},{"courseSession":"1","endTime":"2014-10-08T09:00:00","identifier":"16","registrations":
[],"room":"12","startTime":"2014-10-08T08:00:00"}]},{"descriptionText":"","identifier":"10","name":"Core Data","timeslots":[{"courseSession":"10","endTime":"2014-10-10T18:00:00","identifier":"41","registrations":
[],"room":"12","startTime":"2014-10-10T15:30:00"},{"courseSession":"10","endTime":"2014-10-13T15:00:00","identifier":"42","registrations":[],"room":"13","startTime":"2014-10-13T12:30:00"},
{"courseSession":"10","endTime":"2014-10-14T11:30:00","identifier":"43","registrations":[],"room":"12","startTime":"2014-10-14T09:00:00"}]},{"descriptionText":"","identifier":"11","name":"RESTful
Services","timeslots":[{"courseSession":"11","endTime":"2014-10-13T11:30:00","identifier":"44","registrations":[],"room":"13","startTime":"2014-10-13T09:00:00"},{"courseSession":"11","endTime":"2014-10-
13T18:00:00","identifier":"45","registrations":[],"room":"13","startTime":"2014-10-13T15:30:00"},{"courseSession":"11","endTime":"2014-10-14T15:00:00","identifier":"46","registrations":
[],"room":"12","startTime":"2014-10-14T12:30:00"}]},{"descriptionText":"Introduction to the Swift language.","identifier":"2","name":"Introduction to Swift","timeslots":[{"courseSession":"2","endTime":"2014-10-
06T11:30:00","identifier":"17","registrations":[],"room":"12","startTime":"2014-10-06T09:00:00"},{"courseSession":"2","endTime":"2014-10-06T15:00:00","identifier":"18","registrations":
[{"courseTimeslot":"18","identifier":"52","student":"47"},{"courseTimeslot":"18","identifier":"55","student":"49"}],"room":"13","startTime":"2014-10-06T12:30:00"},{"courseSession":"2","endTime":"2014-10-
08T11:30:00","identifier":"19","registrations":[],"room":"12","startTime":"2014-10-08T09:00:00"}]},{"descriptionText":"Basic project setup.","identifier":"3","name":"Your first iOS App","timeslots":
[{"courseSession":"3","endTime":"2014-10-06T15:00:00","identifier":"20","registrations":[],"room":"12","startTime":"2014-10-06T12:30:00"},{"courseSession":"3","endTime":"2014-10-
06T18:00:00","identifier":"21","registrations":[],"room":"13","startTime":"2014-10-06T15:30:00"},{"courseSession":"3","endTime":"2014-10-08T15:00:00","identifier":"22","registrations":
[{"courseTimeslot":"22","identifier":"53","student":"47"},{"courseTimeslot":"22","identifier":"56","student":"49"}],"room":"12","startTime":"2014-10-08T12:30:00"}]},
{"descriptionText":"","identifier":"4","name":"Intermediate Swift","timeslots":[{"courseSession":"4","endTime":"2014-10-07T11:30:00","identifier":"23","registrations":[],"room":"12","startTime":"2014-10-
07T09:30:00"},{"courseSession":"4","endTime":"2014-10-07T15:00:00","identifier":"24","registrations":[],"room":"13","startTime":"2014-10-07T12:30:00"},{"courseSession":"4","endTime":"2014-10-
09T11:30:00","identifier":"25","registrations":[],"room":"12","startTime":"2014-10-09T09:00:00"}]},{"descriptionText":"","identifier":"5","name":"Storyboarding","timeslots":[{"courseSession":"5","endTime":"2014-
10-07T15:00:00","identifier":"26","registrations":[],"room":"12","startTime":"2014-10-07T12:30:00"},{"courseSession":"5","endTime":"2014-10-07T18:00:00","identifier":"27","registrations":
[],"room":"13","startTime":"2014-10-07T15:30:00"},{"courseSession":"5","endTime":"2014-10-09T15:00:00","identifier":"28","registrations":[],"room":"12","startTime":"2014-10-09T12:30:00"}]},
{"descriptionText":"","identifier":"6","name":"Advanced Swift","timeslots":[{"courseSession":"6","endTime":"2014-10-08T18:00:00","identifier":"29","registrations":[],"room":"12","startTime":"2014-10-08T15:30:00"},
{"courseSession":"6","endTime":"2014-10-08T11:30:00","identifier":"30","registrations":[],"room":"13","startTime":"2014-10-08T09:00:00"},{"courseSession":"6","endTime":"2014-10-
10T15:00:00","identifier":"31","registrations":[],"room":"13","startTime":"2014-10-10T12:30:00"}]},{"descriptionText":"","identifier":"7","name":"Model View Controller","timeslots":
[{"courseSession":"7","endTime":"2014-10-08T15:00:00","identifier":"32","registrations":[],"room":"12","startTime":"2014-10-08T12:30:00"},{"courseSession":"7","endTime":"2014-10-
09T11:30:00","identifier":"33","registrations":[],"room":"13","startTime":"2014-10-09T09:00:00"},{"courseSession":"7","endTime":"2014-10-10T18:00:00","identifier":"34","registrations":
[],"room":"13","startTime":"2014-10-10T15:30:00"}]},{"descriptionText":"","identifier":"8","name":"Data Serialization","timeslots":[{"courseSession":"8","endTime":"2014-10-
09T15:00:00","identifier":"35","registrations":[],"room":"13","startTime":"2014-10-09T12:30:00"},{"courseSession":"8","endTime":"2014-10-10T11:30:00","identifier":"36","registrations":
[],"room":"12","startTime":"2014-10-10T09:00:00"},{"courseSession":"8","endTime":"2014-10-13T11:30:00","identifier":"37","registrations":[],"room":"12","startTime":"2014-10-13T09:00:00"}]},
{"descriptionText":"","identifier":"9","name":"Client-Server Communication","timeslots":[{"courseSession":"9","endTime":"2014-10-10T15:00:00","identifier":"38","registrations":[],"room":"12","startTime":"2014-10-
10T12:30:00"},{"courseSession":"9","endTime":"2014-10-10T11:30:00","identifier":"39","registrations":[],"room":"13","startTime":"2014-10-10T09:00:00"},{"courseSession":"9","endTime":"2014-10-
13T15:00:00","identifier":"40","registrations":[],"room":"12","startTime":"2014-10-13T12:30:00"}]}]


Edit
View
History
Bookmarks
Window
Users
Help
Always Show Bookmarks Bar ⇧⌘B
Stop ⌘.
Reload This Page ⌘R
Enter Presentation Mode ⇧⌘F
Exit Full Screen ^⌘F
Actual Size ⌘0
Zoom In ⌘+
Zoom Out ⌘−
Encoding
Developer
View Source ⌥⌘U
Developer Tools ⌥⌘I
JavaScript Console ⌥⌘J
Setting up firmware
{"identifier":"1","name":"Introducti
urseTimeslot":"14","identifier
,"startTime":"2014-10-06T11:30:
tionText":"","identifier":"10",
sion":"10","endTime":"2014-10-1
fier":"43","registrations":[],"
10-13T11:30:00","identifier":"
,"startTime":"2014-10-13T15:30:
8","identifier":"52","student":"47"},{"c
ier":"19","registrations":[],"room":"12"
,"endTime":"2014-10-06T15:00:00","identi
ier":"21","registrations":[],"room":"13","startTime":"2014-10-06T15:30:
2","identifier":"53","student":"47"},{"courseTimeslot":"22","identifier
,"identifier":"4","name":"Intermediate Swift","timeslots":[{"courseSess
eSession":"4","endTime":"2014-10-07T15:00:00","identifier":"24","regist
ier":"25","registrations":[],"room":"12","startTime":"2014-10-09T09:00:
tifier":"26","registrations":[],"room":"12","startTime":"2014-10-07T12:
Time":"2014-10-07T15:30:00"},{"courseSession":"5","endTime":"2014-10-0
,"identifier":"6","name":"Advanced Swift","timeslots":[{"courseSession"
"endTime":"2014-10-08T11:30:00","identifier":"30","registrations":[],"r
ier":"31","registrations":[],"room":"13","startTime":"2014-10-10T12:30:

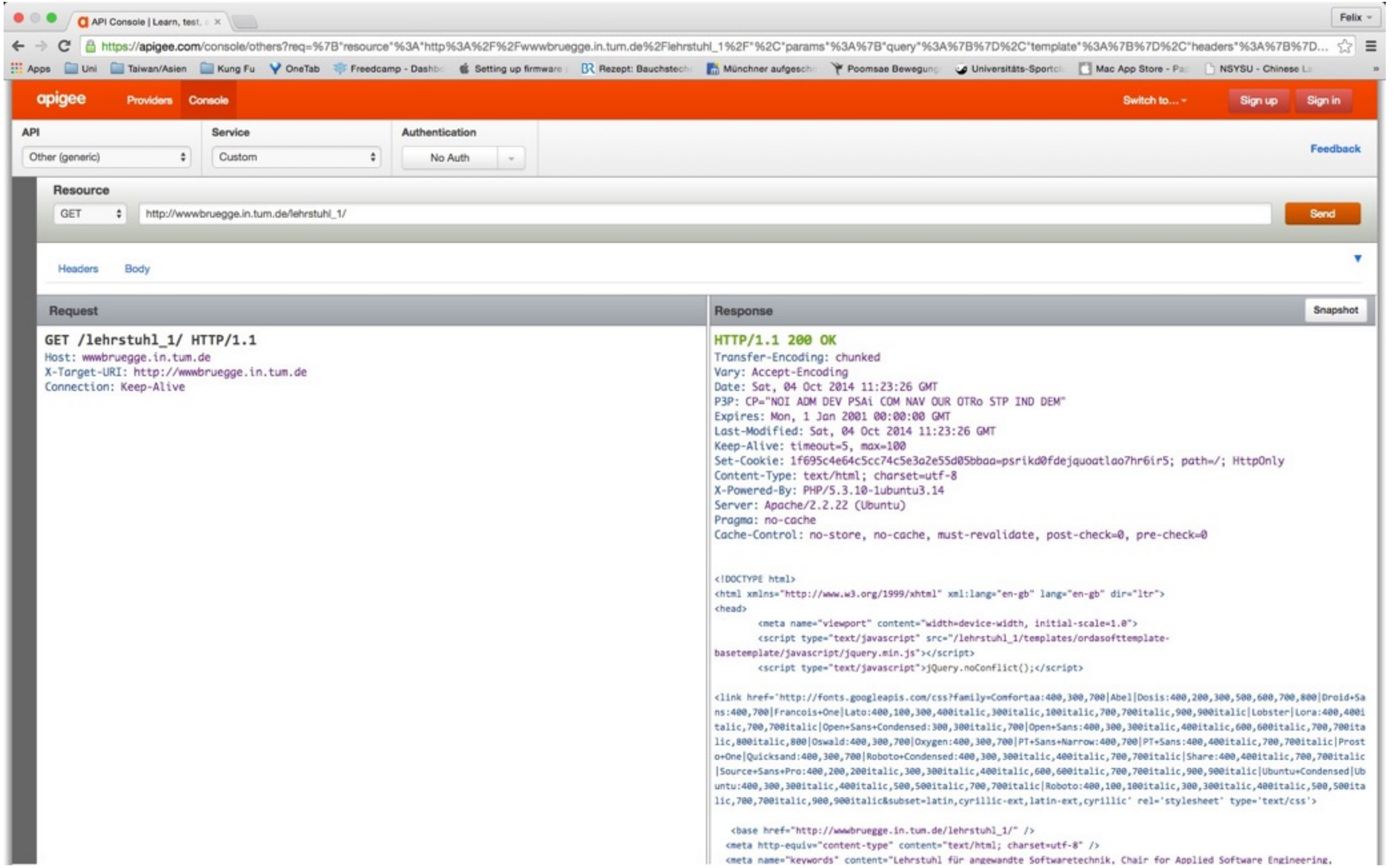

Elements Network Sources Timeline Profiles Resources Audits Console
Preserve log Disable cache
Name Path
Method
Status Text
Type
Initiator
Size Content
Time Latency
Timeline
40 ms 60 ms 80 ms 100 ms 120 ms 140 ms
sessions /iOSIntroService/rest
GET
200 OK
application/json
Other
6.0 KB 5.7 KB
116 ms 114 ms
1 requests | 6.0 KB transferred | 116 ms (load: 158 ms, DOMContentLoaded: 158 ms)

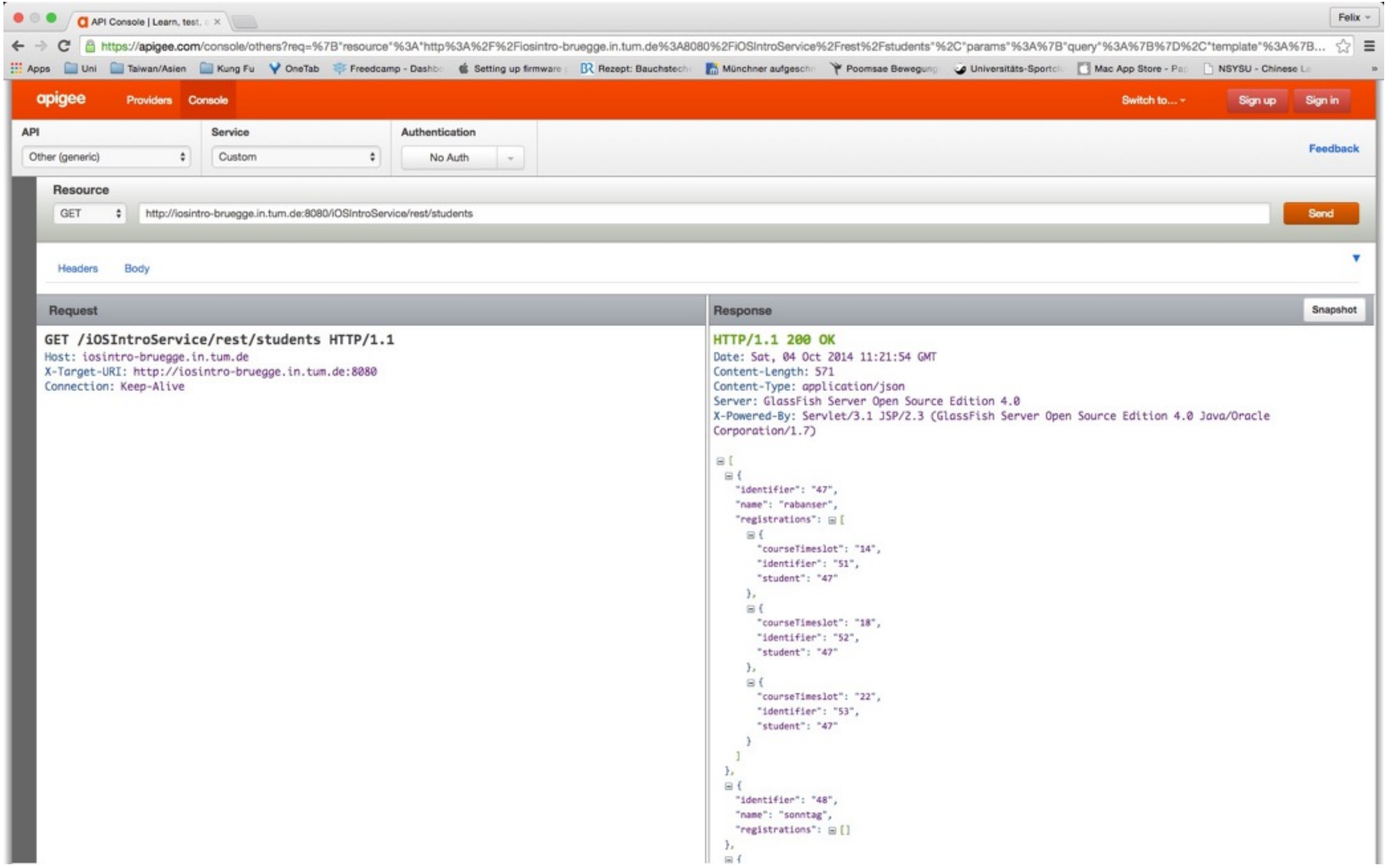

Elements Network Sources Timeline Profiles Resources Audits Console
Preserve log Disable cache
Name
Path
sessions
/iOSIntroService/rest
Headers Preview Response Cookies Timing
Remote Address: 131.159.38.222:8080
Request URL: http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/sessions
Request Method: GET
Status Code: 200 OK
Request Headers view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
Cache-Control: max-age=0
Connection: keep-alive
Cookie: _et_coid=3ff6507c943253cc43987938dafed7f7; treeForm_tree-hi=treeForm:tree:applicationServer
Host: iosintro-bruegge.in.tum.de:8080
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.101 Safari/537.36
Response Headers view source
Content-Length: 5849
Content-Type: application/json
Date: Sat, 04 Oct 2014 11:13:07 GMT
Server: GlassFish Server Open Source Edition 4.0
X-Powered-By: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.0 Java/Oracle Corporation/1.7)
1 requests | 6.0 KB transferred ...

An online REST-console (click on the image)

Execute these commands

GET http://wwwbruegge.in.tum.de/lehrstuhl_1/
GET http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students


API Console | Learn, test.
Felix
https://apigee.com/console/others?req=%7B"resource"%3A"http%3A%2F%2Fwwwbruegge.in.tum.de%2Flehrstuhl_1%2F"%2C"params"%3A%7B"query"%3A%7B%7D%2C"template"%3A%7B%7D%2C"headers"%3A%7B%7D...
Apps
Uni
Taiwan/Asien
Kung Fu
OneTab
Freedcamp - Dashbo
Setting up firmware
Rezept: Bauchstech
Münchner aufgesch
Poomsae Bewegung
Universitäts-Sportcl
Mac App Store - Pa
NSYSU - Chinese La
apigee
Providers
Console
Switch to...
Sign up
Sign in
API
Other (generic)
Service
Custom
Authentication
No Auth
Feedback
Resource
GET
http://wwwbruegge.in.tum.de/lehrstuhl_1/
Send
Headers
Body
Request
GET /lehrstuhl_1/ HTTP/1.1
Host: wwwbruegge.in.tum.de
X-Target-URI: http://wwwbruegge.in.tum.de
Connection: Keep-Alive
Response
Snapshot
HTTP/1.1 200 OK
Transfer-Encoding: chunked
Vary: Accept-Encoding
Date: Sat, 04 Oct 2014 11:23:26 GMT
P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"
Expires: Mon, 1 Jan 2001 00:00:00 GMT
Last-Modified: Sat, 04 Oct 2014 11:23:26 GMT
Keep-Alive: timeout=5, max=100
Set-Cookie: 1f695c4e64c5cc74c5e3a2e55d05bbaa=psrikd0fdejquoatlao7hr6ir5; path=/; HttpOnly
Content-Type: text/html; charset=utf-8
X-Powered-By: PHP/5.3.10-1ubuntu3.14
Server: Apache/2.2.22 (Ubuntu)
Pragma: no-cache
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-gb" lang="en-gb" dir="ltr">
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<script type="text/javascript" src="/lehrstuhl_1/templates/ordasofttemplate-basetemplate/javascript/jquery.min.js"></script>
<script type="text/javascript">jQuery.noConflict();</script>
<link href='http://fonts.googleapis.com/css?family=Comfortaa:400,300,700|Abel|Dosis:400,200,300,500,600,700,800|Droid+Sans:400,700|Francois+One|Lato:400,100,300,400italic,300italic,100italic,700,700italic,900,900italic|Lobster|Lora:400,400italic,700,700italic|Open+Sans+Condensed:300,300italic,700|Open+Sans:400,300,300italic,400italic,600,600italic,700,700italic,800italic,800|Oswald:400,300,700|Oxygen:400,300,700|PT+Sans+Narrow:400,700|PT+Sans:400,400italic,700,700italic|Prosto+One|Quicksand:400,300,700|Roboto+Condensed:400,300,300italic,400italic,700,700italic|Share:400,400italic,700,700italic|Source+Sans+Pro:400,200,200italic,300,300italic,400italic,600,600italic,700,700italic,900,900italic|Ubuntu+Condensed|Ubuntu:400,300,300italic,400italic,500,500italic,700,700italic|Roboto:400,100,100italic,300,300italic,400italic,500,500italic,700,700italic,900,900italic&subset=latin,cyrillic-ext,latin-ext,cyrillic' rel='stylesheet' type='text/css'>
<base href="http://wwwbruegge.in.tum.de/lehrstuhl_1/" />
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<meta name="keywords" content="Lehrstuhl für angewandte Softwaretechnik, Chair for Applied Software Engineering,


API Console | Learn, test
https://apigee.com/console/others?req=%7B"resource"%3A"http%3A%2F%2Fiosintro-bruegge.in.tum.de%3A8080%2FiOSIntroService%2Frest%2Fstudents"%2C"params"%3A%7B"query"%3A%7B%7D%2C"template"%3A%7B...
apigee
Providers
Console
Switch to...
Sign up
Sign in
API
Other (generic)
Service
Custom
Authentication
No Auth
Feedback
Resource
GET
http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students
Send
Headers
Body
Request
GET /iOSIntroService/rest/students HTTP/1.1
Host: iosintro-bruegge.in.tum.de
X-Target-URI: http://iosintro-bruegge.in.tum.de:8080
Connection: Keep-Alive
Response
Snapshot
HTTP/1.1 200 OK
Date: Sat, 04 Oct 2014 11:21:54 GMT
Content-Length: 571
Content-Type: application/json
Server: GlassFish Server Open Source Edition 4.0
X-Powered-By: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.0 Java/Oracle Corporation/1.7)
{
"identifier": "47",
"name": "rabanser",
"registrations": [
{
"courseTimeslot": "14",
"identifier": "51",
"student": "47"
},
{
"courseTimeslot": "18",
"identifier": "52",
"student": "47"
},
{
"courseTimeslot": "22",
"identifier": "53",
"student": "47"
}
]
},
{
"identifier": "48",
"name": "sonntag",
"registrations": []
},

But how to do this in iOS?

NSURLSession

AFNetworking

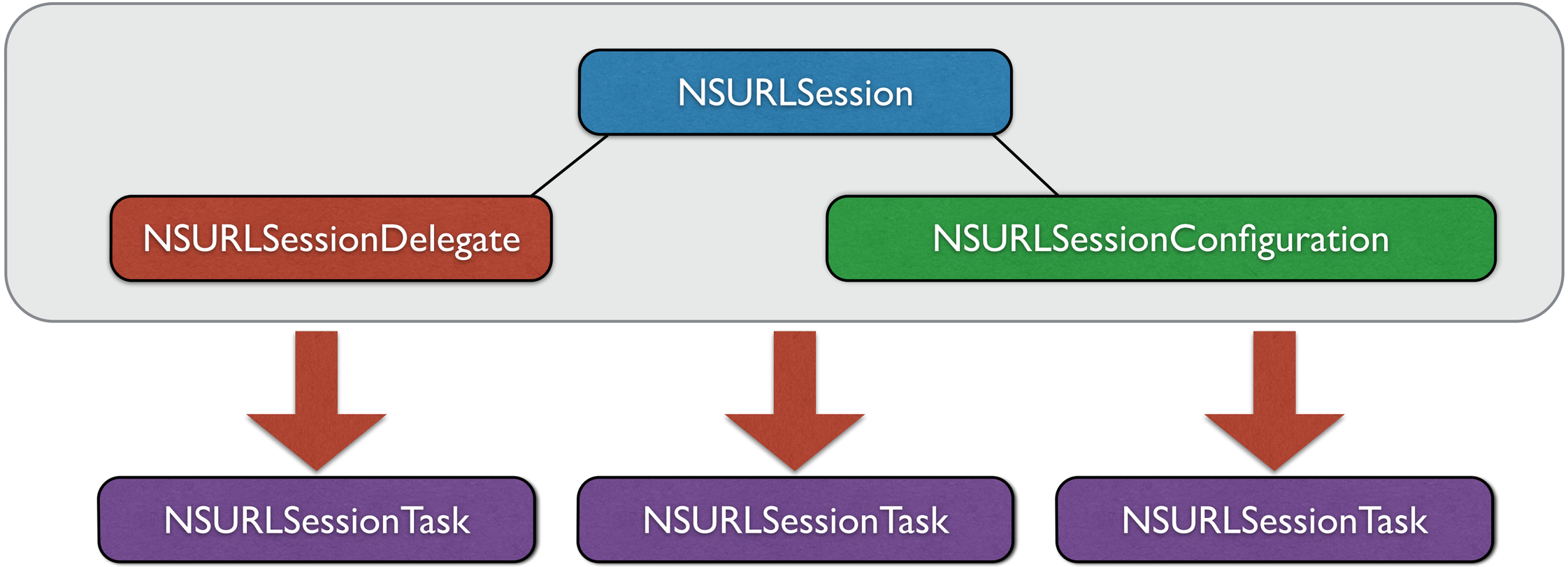
Outline

- Motivation
- Basic Terms
- **NSURLSession**
- REST
- AFNetworking
- Final Exercise

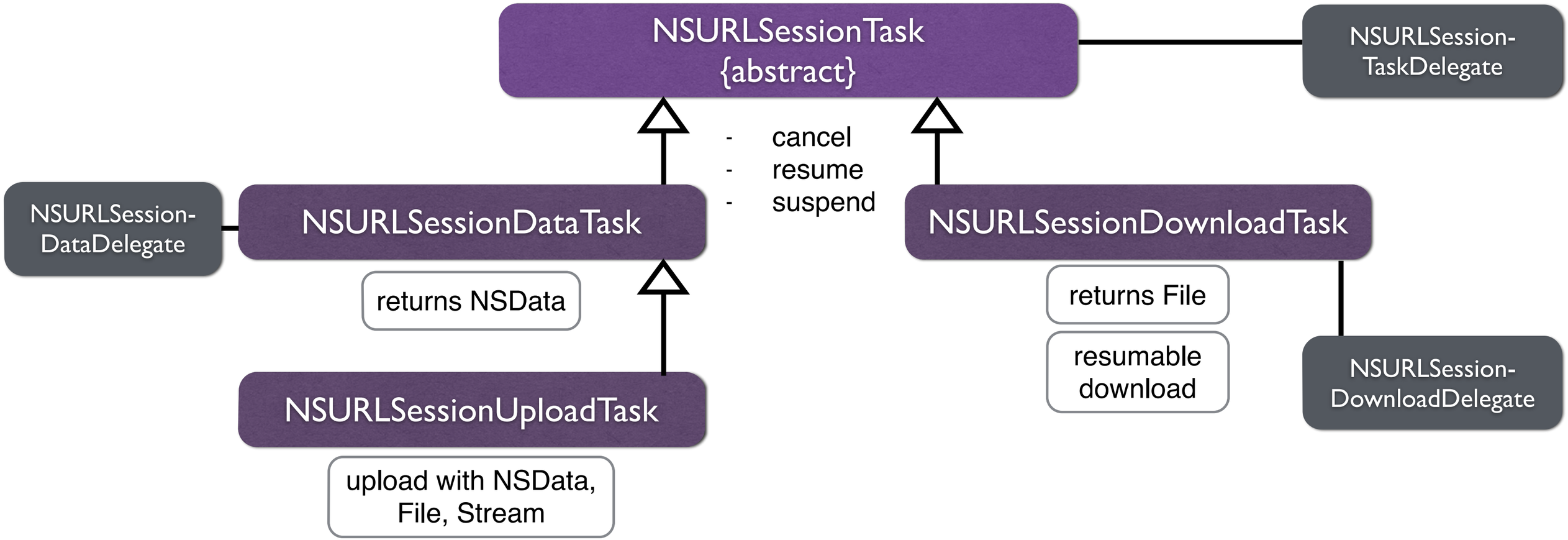
NSURLSession

Let’s see what Apple got!

NSURLSession

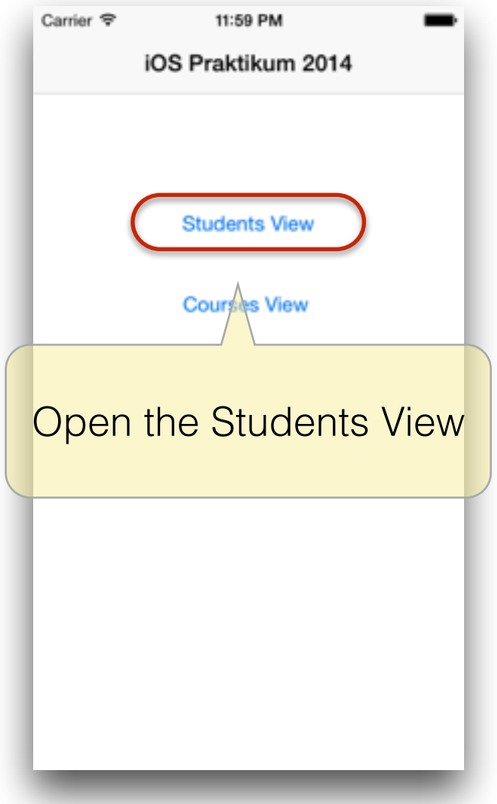


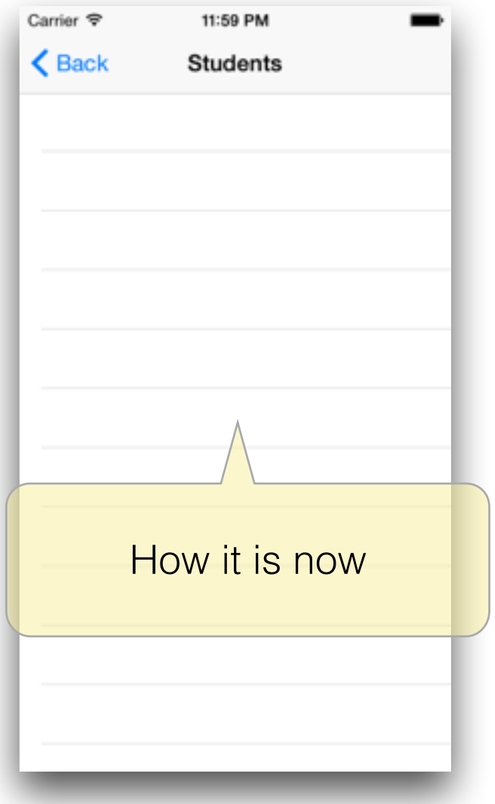
NSURLSession

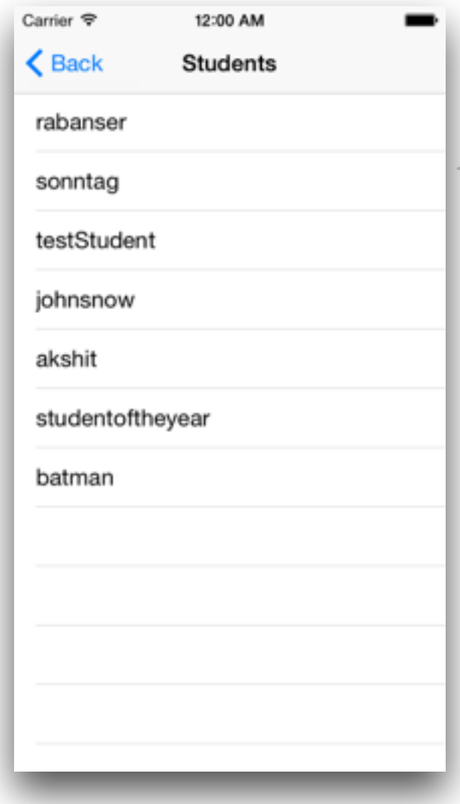


Example 1.1: Student List

Implement getting the student List with NSURLSession







How it should be like

Login Credentials

Username: *batman*
Password: *catwoman*

Example 1.1: Student List

Implement getting the student List with NSURLSession

- *updateStudents* in the *DataManager* needs your work (try to understand the method declaration)
- we already implemented further failure and success handling
  → just call the failure and success closures provided as parameters in the function at the right time (error checking!)
- URL to call:
  http://iosintro-bruegge.in.tum.de:8080/IntroService/rest/students
- **Hint**: use the *studentsFromJSONData* of the *serializationHelper* to get the students of the NSData

Example 1.1: Student List

Implement getting the student List with NSURLSession: Setup

1) Create `NSURLSessionConfiguration`

```
let configuration = NSURLSessionConfiguration.defaultSessionConfiguration()
configuration.HTTPAdditionalHeaders = ["Accept": "application/json", „Content-Type": "application/json"]
```

Example 1.1: Student List

Implement getting the student List with NSURLSession: Setup

1) Create NSURLSessionConfiguration

```
let configuration = NSURLSessionConfiguration.defaultSessionConfiguration()
configuration.HTTPAdditionalHeaders = ["Accept": "application/json", „Content-Type": "application/json"]
```

2) Create NSURLSession

```
let session = NSURLSession(configuration: configuration, delegate: nil,
                           delegateQueue: NSOperationQueue.mainQueue())
```

queue for delegate calls and completion handlers

- **NSOperationQueue**: regulates concurrent execution of operations
- **mainQueue()** returns the operation queue associated with the main thread
- **Golden Rule of iOS Programming: Never Block The Main Thread**

Example 1.1: Student List

Implement getting the student List with NSURLSession: Setup

1) Create NSURLSessionConfiguration

```
let configuration = NSURLSessionConfiguration.defaultSessionConfiguration()
configuration.HTTPAdditionalHeaders = ["Accept": "application/json", „Content-Type": "application/json"]
```

2) Create NSURLSession

```
let session = NSURLSession(configuration: configuration, delegate: nil, delegateQueue:
NSOperationQueue.mainQueue())
```

3) Create NSURL

```
let url = NSURL(string: self.baseUrlString + "students")
```

Example 1.1: Student List

Implement getting the student List with NSURLSession

create the NSURLSessionDataTask

```
let task = session.dataTaskWithURL(url,
   completionHandler: { (data: NSData!, response: NSURLResponse!, error: NSError!) -> Void in
       if (error != nil) {
          // error handling
       } else {
          // success handling
       }
   }
)

task.resume()
```

the response data

the error object

Start the task

Example 1.1: Student List

Implement getting the student List with NSURLSession

```
func updateStudents(success: (students:[Student]) -> Void, failure: (error : NSError!) -> Void) {

    … … …
    let task = session.dataTaskWithURL(url, completionHandler: { (data: NSData!,
                                        response: NSURLResponse!, error: NSError!) -> Void in









}
```

Example 1.1: Student List

Implement getting the student List with NSURLSession

```
func updateStudents(success: (students:[Student]) -> Void, failure: (error : NSError!) -> Void) {

    … … …
    let task = session.dataTaskWithURL(url, completionHandler: { (data: NSData!,
                                        response: NSURLResponse!, error: NSError!) -> Void in

        if (error != nil) {
            failure(error: error)
        }

}
```

Call failure closure with error

Example 1.1: Student List

Implement getting the student List with NSURLSession

```
func updateStudents(success: (students:[Student]) -> Void, failure: (error : NSError!) -> Void) {

    … … …
    let task = session.dataTaskWithURL(url, completionHandler: { (data: NSData!,
                                        response: NSURLResponse!, error: NSError!) -> Void in

        if (error != nil) {
            failure(error: error)

        } else {
            let students = self.serializationHelper.studentsFromJSONData(data)
            success(students: students)
        }

    })


}
```

Call failure closure with error

Call success closure with students array

De-Serialization to get array of students

Example 1.1: Student List

Implement getting the student List with NSURLSession

```
func updateStudents(success: (students:[Student]) -> Void, failure: (error : NSError!) -> Void) {

    … … …
    let task = session.dataTaskWithURL(url, completionHandler: { (data: NSData!,
                                        response: NSURLResponse!, error: NSError!) -> Void in

        if (error != nil) {
            failure(error: error)

        } else {
            let students = self.serializationHelper.studentsFromJSONData(data)
            success(students: students)
        }

    })

    task.resume()
}
```

Call failure closure with error

De-Serialization to get array of students

Call success closure with students array

Start the task

Outline

- Motivation
- Basic Terms
- NSURLSession
- **REST**
- AFNetworking
- Final Exercise

REST

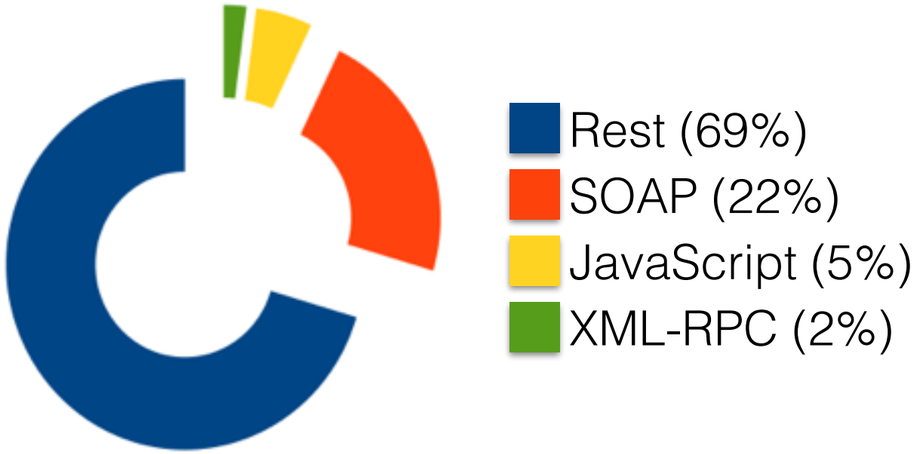
A short glimpse

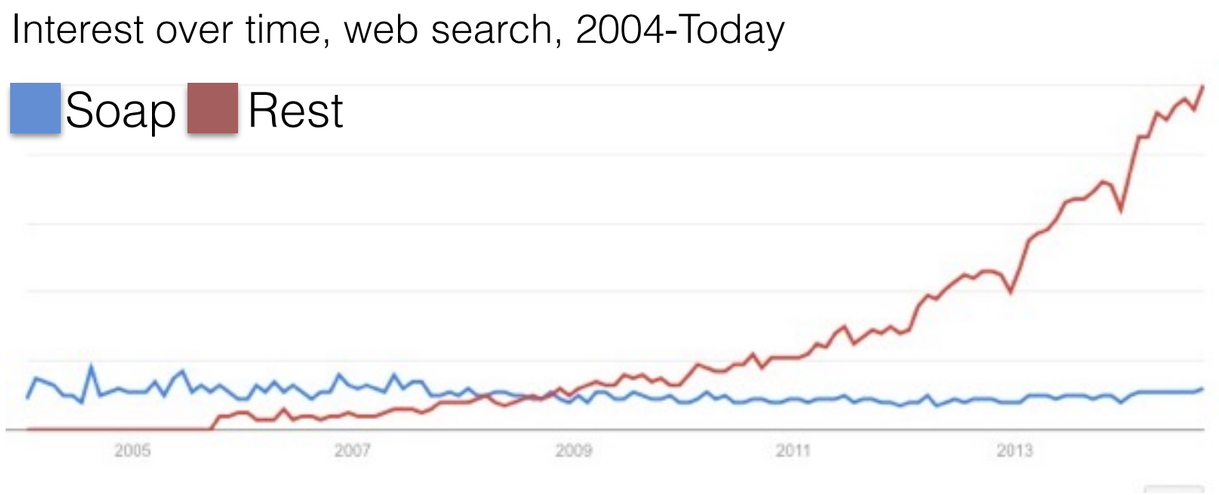
REST: Representational State Transfer

- Architectural style for distributed hypermedia systems (like the internet)
- REST provides a set of architectural constraints
  - **Client-Server**: separates user interface from data storage
  - **Stateless Communication**: each request from client to server must contain all of the information necessary to understand the request
  - **Cacheable**: data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable
  - **Uniform interface**: implementations are decoupled from the service they provide by their interface
  - **Layered System**: allows hierarchical layers

More in the RESTful Services Talk

Why REST?


Rest (69%)
SOAP (22%)
JavaScript (5%)
XML-RPC (2%)


Interest over time, web search, 2004-Today
Soap
Rest
2005
2007
2009
2011
2013

REST Methods

REST command (HTTP method)	Description	Operation
POST	Create or add new entries	**C**reate
GET	Read, retrieve, search, or view existing entries	**R**ead
PUT or PATCH	Update or edit existing entries	**U**pdate
DELETE	Delete/deactivate existing entries	**D**elete

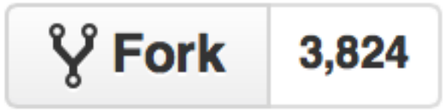
Question:
What kind of command did we just use?

Outline

- Motivation
- Basic Terms
- NSURLSession
- REST
- **AFNetworking**
- Final Exercise

“AFNetworking is about getting what you want”
- Matt Thompson, author of AFNetworking

One of the most widely used open source library



Serves as *foundation* for dozens of open source libraries

Built on top of NSURLSession

- NSURLSession Compatibility
- Serialization
- Security
- Reachability
- UIKit Extensions

AFURLSessionManager and its subclass **AFHTTPSessionManager** manage a NSURLSession

- NSURLSession Compatibility
- **Serialization**
- Security
- Reachability
- UIKit Extensions

AFURLRequestSerialization decodes data and **AFURLResponseSerialization** encodes parameters

- NSURLSession Compatibility
- Serialization
- **Security**
- Reachability
- UIKit Extensions

AFSecurityPolicy evaluates server trust against pinned X.509 certificates and public keys

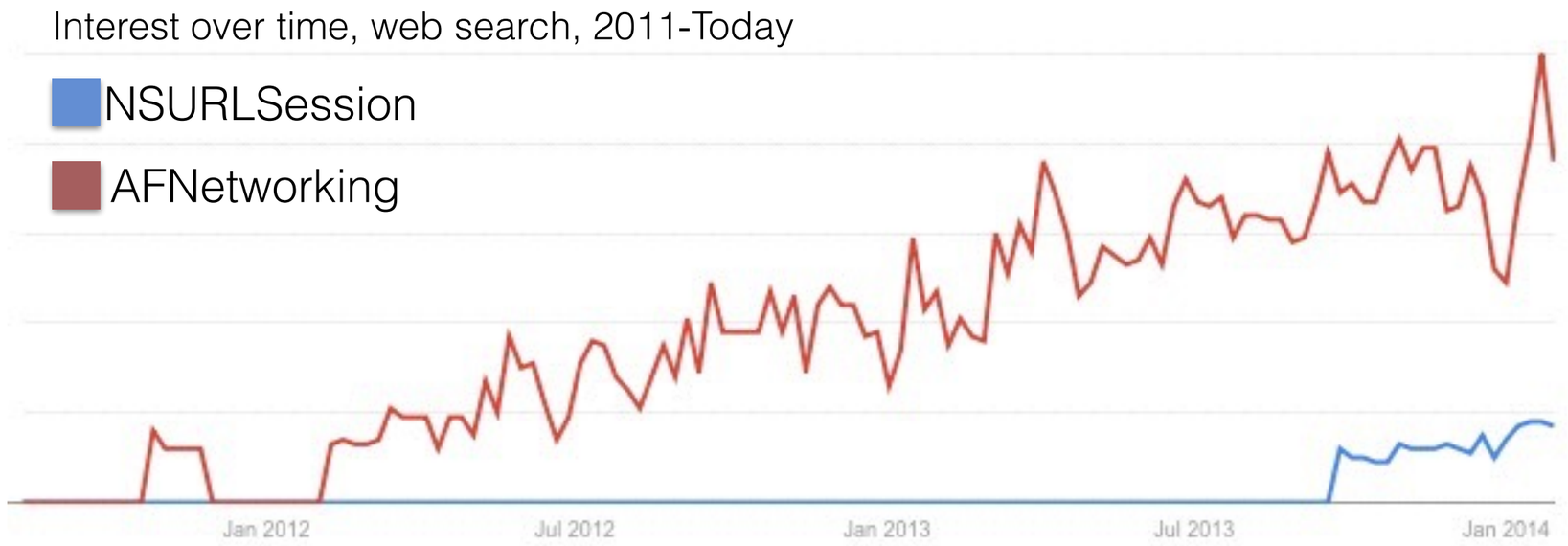
- NSURLSession Compatibility
- Serialization
- Security
- **Reachability**
- UIKit Extensions

AFNetworkReachabilityManager monitors the reachability of domains, and addresses for both WWAN and WiFi network interfaces

- NSURLSession Compatibility
- Serialization
- Security
- Reachability
- **UIKit Extensions**

UIKit+AFNetworking provides several extensions for UIKit classes

NSURLSession vs. AFNetworking

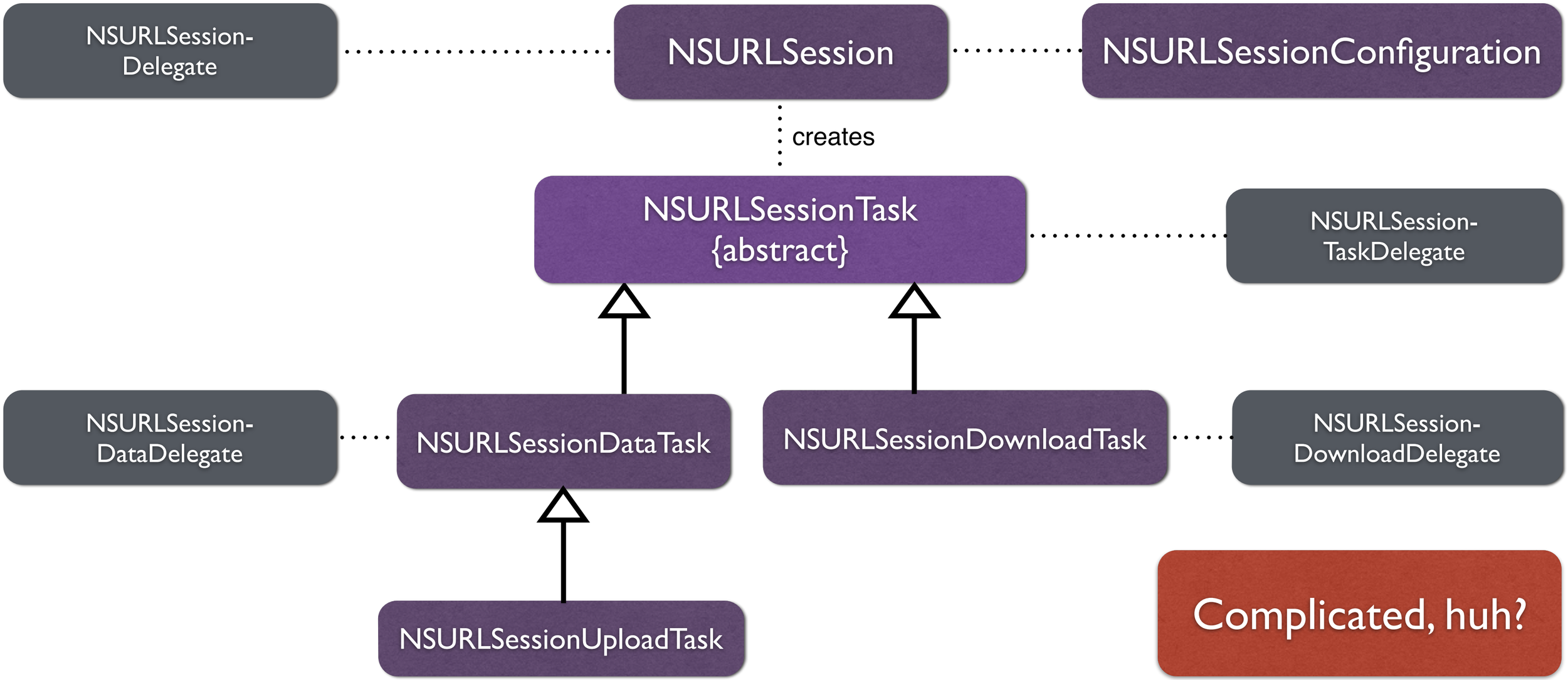


Recall:

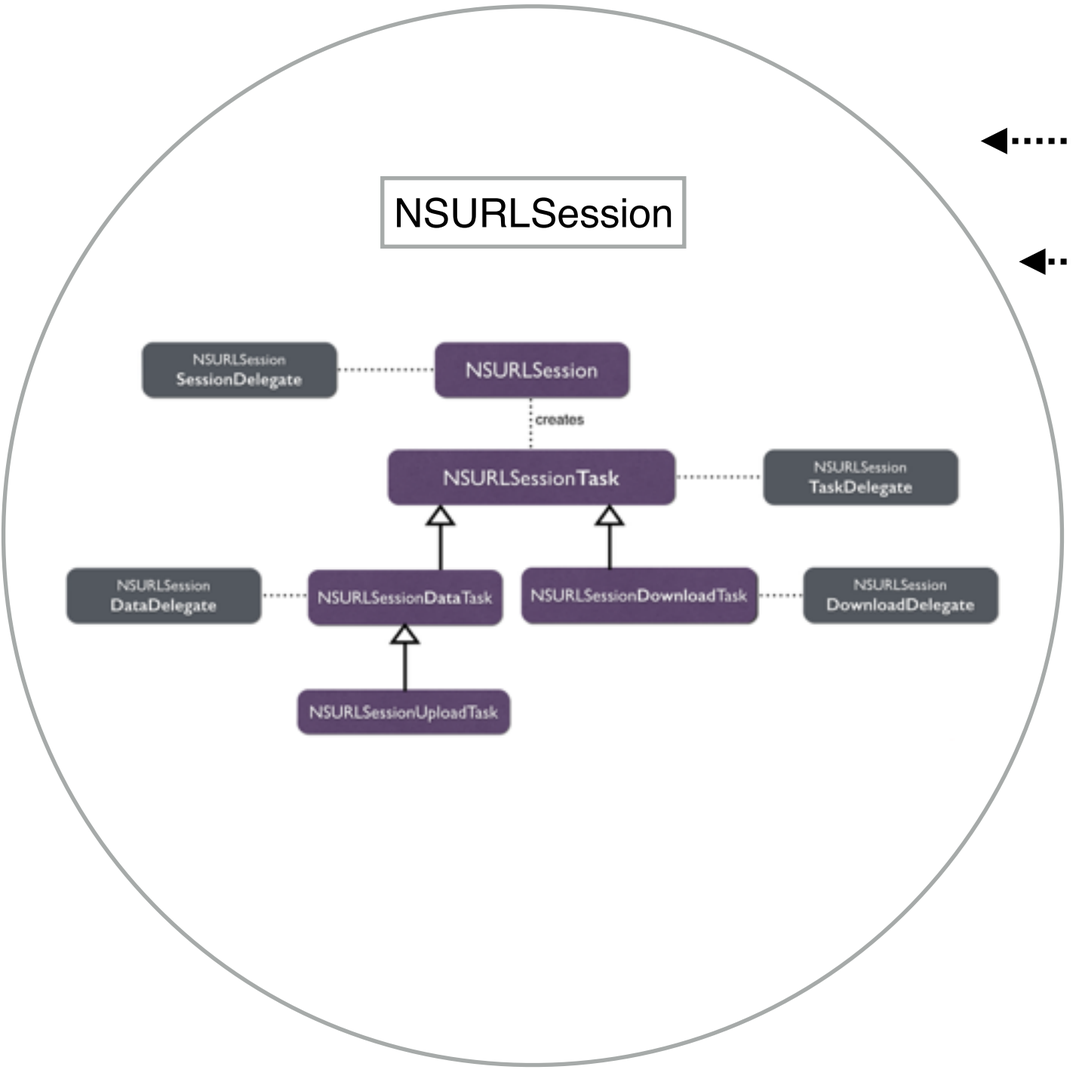
AFNetworking is built on top of NSURLSession

Recap

NSURLSession



So, what does AFNetworking do for us?



implement delegates

creates the tasks

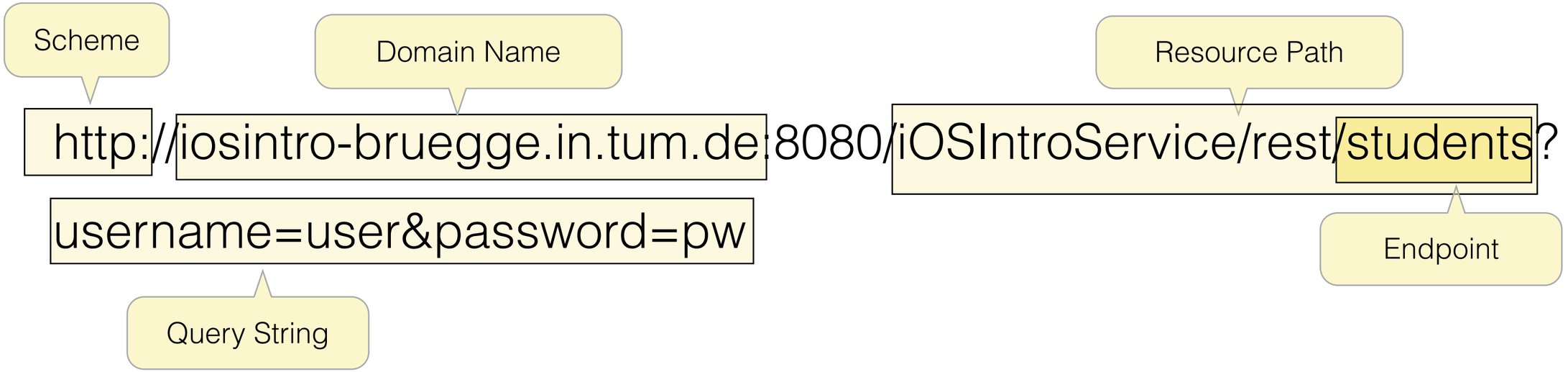
AFURLSessionManager

AFHTTPSessionManager

convenience methods for making HTTP requests

Hands On

Recap: A typical URL



Resource Path: path at the server

Endpoint: API calling point to a service

Query String: includes fields added to a base URL

Configurations

Base Resource:

```
let baseUrlString = "http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/"
```

AFNetworking Initialization parts :

```
let baseUrl = NSURL(string: baseUrlString)
manager = AFHTTPSessionManager(baseURL: baseUrl)
```

Create session manager with baseURL

Configurations

Base Resource:

```
let baseUrlString = "http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/"
```

AFNetworking Initialization parts :

```
let baseUrl = NSURL(string: baseUrlString)
manager = AFHTTPSessionManager(baseURL: baseUrl)
```

Create session manager with baseURL

```
manager.requestSerializer = AFJSONRequestSerializer()
```

Set JSON Serializer for Requests

```
manager.responseSerializer = AFJSONResponseSerializer()
```

Set JSON Serializer for Responses

Configurations

Base Resource:

```
let baseUrlString = "http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/"
```

AFNetworking Initialization parts :

```
let baseUrl = NSURL(string: baseUrlString)
manager = AFHTTPSessionManager(baseURL: baseUrl)
```

Create session manager with baseURL

```
manager.requestSerializer = AFJSONRequestSerializer()
```

Set JSON Serializer for Requests

```
manager.responseSerializer = AFJSONResponseSerializer()
```

Set JSON Serializer for Responses

There is still one more thing missing?

Recap: Request

HTTP Request	**Request Line**	GET http://iosintro-bruegge.in.tum.de:8080/students
	Headers	Accept: application/json Content-Type: application/json
	Message Body Data	our content, e.g. a JSON

We need to set Header Fields to make a request

Configurations

Base Resource:

```
let baseUrlString = "http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/"
```

AFNetworking Initialization parts :

```
let baseUrl = NSURL(string: baseUrlString)
manager = AFHTTPSessionManager(baseURL: baseUrl)

manager.requestSerializer = AFJSONRequestSerializer()

manager.responseSerializer = AFJSONResponseSerializer()

manager.requestSerializer.setValue("application/json", forHTTPHeaderField: “Accept")
manager.requestSerializer.setValue("application/json", forHTTPHeaderField: "Content-Type")
```

Create session manager with baseURL

Set JSON Serializer for Requests

Set JSON Serializer for Responses

Set Header Fields “Accept” and “Content-type” to JSON

Task : Configurations

- Reopen RegistrationApp project
- Find *init()* function in *DataManager.swift*
  - remove this line: `manager = AFHTTPSessionManager()`
  - add the following lines of code

```
let baseUrl = NSURL(string: baseUrlString)
manager = AFHTTPSessionManager(baseURL: baseUrl)

manager.requestSerializer = AFJSONRequestSerializer()

manager.responseSerializer = AFJSONResponseSerializer()

manager.requestSerializer.setValue("application/json", forHTTPHeaderField: “Accept")
manager.requestSerializer.setValue("application/json", forHTTPHeaderField: "Content-Type")
```

Let’s see how we do a GET in AFNetworking

GET Example in AFNetworking

To Fetch list of superheroes with **AFNetworking**

```
http://iosintro-bruegge.in.tum.de:8080/
  iOSIntroService/rest/superheroes
```

GET Example in AFNetworking

To Fetch list of superheroes with **AFNetworking**

REST command

```
GET http://iosintro-bruegge.in.tum.de:8080/
    iOSIntroService/rest/superheroes
```

What is the Endpoint here?

GET Example in AFNetworking

To Fetch list of superheroes with **AFNetworking**

REST command

```
GET http://iosintro-bruegge.in.tum.de:8080/
    iOSIntroService/rest/superheroes
```

Endpoint

What is the Endpoint here?

GET Example in AFNetworking

convenience method for REST command

Endpoint

Additional *Query String* Parameters

```
manager.GET("superheroes", parameters: nil,
```

Success closure

```
    success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in



    },
```

Failure closure

```
    failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in


    }
)
```

GET Example in AFNetworking

convenience method for REST command

Endpoint

Additional *Query String* Parameters

```
manager.GET("superheroes", parameters: nil,
    success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
```

Success closure

Looks Familiar?

```
    },
    failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in
```

Failure closure

```
    }
)
```

GET Example in AFNetworking

convenience method for REST command

Endpoint

Additional *Query String* Parameters

```
manager.GET("superheroes", parameters: nil,
    success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
    },
    failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in
    }
)
```

Success closure

Data Task Class from NSURLSession

Failure closure

GET Example in AFNetworking

convenience method for REST command

Endpoint

Additional *Query String* Parameters

Response Object

```
manager.GET("superheroes", parameters: nil,
    success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
```

Success closure

Data Task Class from NSURLSession

```
    },
    failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in
```

Failure closure

```
    }
)
```

GET Example Response for SuperHeroes

```
[
  {
    "id":"1",
    "name":"Batman",
    "hobbies": [
      “Save the World”,
      “Buy rose for CatWoman”,
      “No Relaxing”
    ]
  },
  {
    "id":"2",
    "name":"SpiderMan",
    "hobbies": [
      “Save the World”,
      “Buy rose for Mary Jane”,
      “Climb Walls”
    ]
  }
]
```

One element

Array of Elements

GET Example Response for SuperHeroes

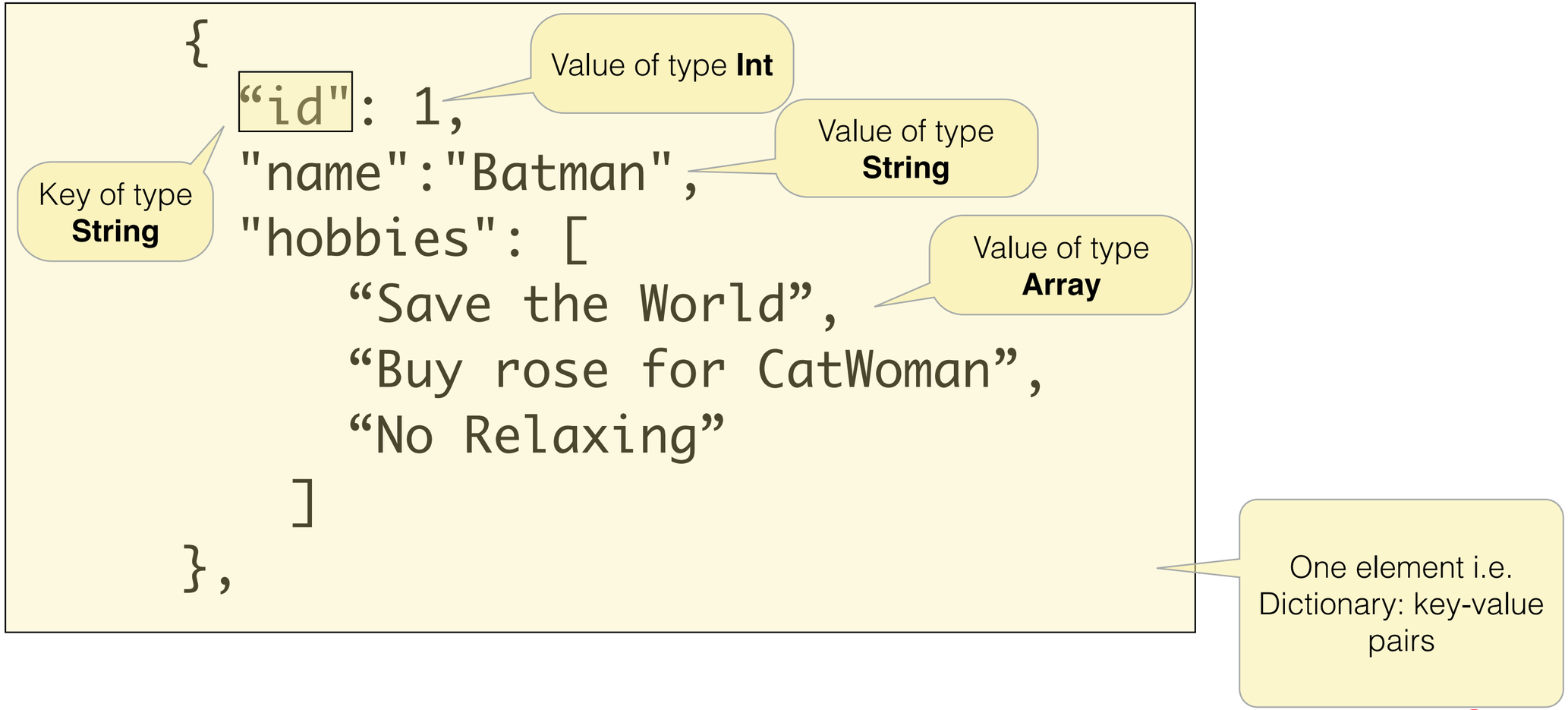
```
{
    “id": 1,
    "name":"Batman",
    "hobbies": [
        “Save the World”,
        “Buy rose for CatWoman”,
        “No Relaxing”
    ]
},
```

Value of type **Int**

Key of type **String**

One element i.e. Dictionary: key-value pairs

GET Example Response for SuperHeroes



GET Example Response for SuperHeroes

```
Dictionary< String, ? >
```

What would be the type for *value*?

```
{
    “id": 1,
    "name":"Batman",
    "hobbies": [
        “Save the World”,
        “Buy rose for CatWoman”,
        “No Relaxing”
    ]
},
```

Value of type **Int**

Value of type **String**

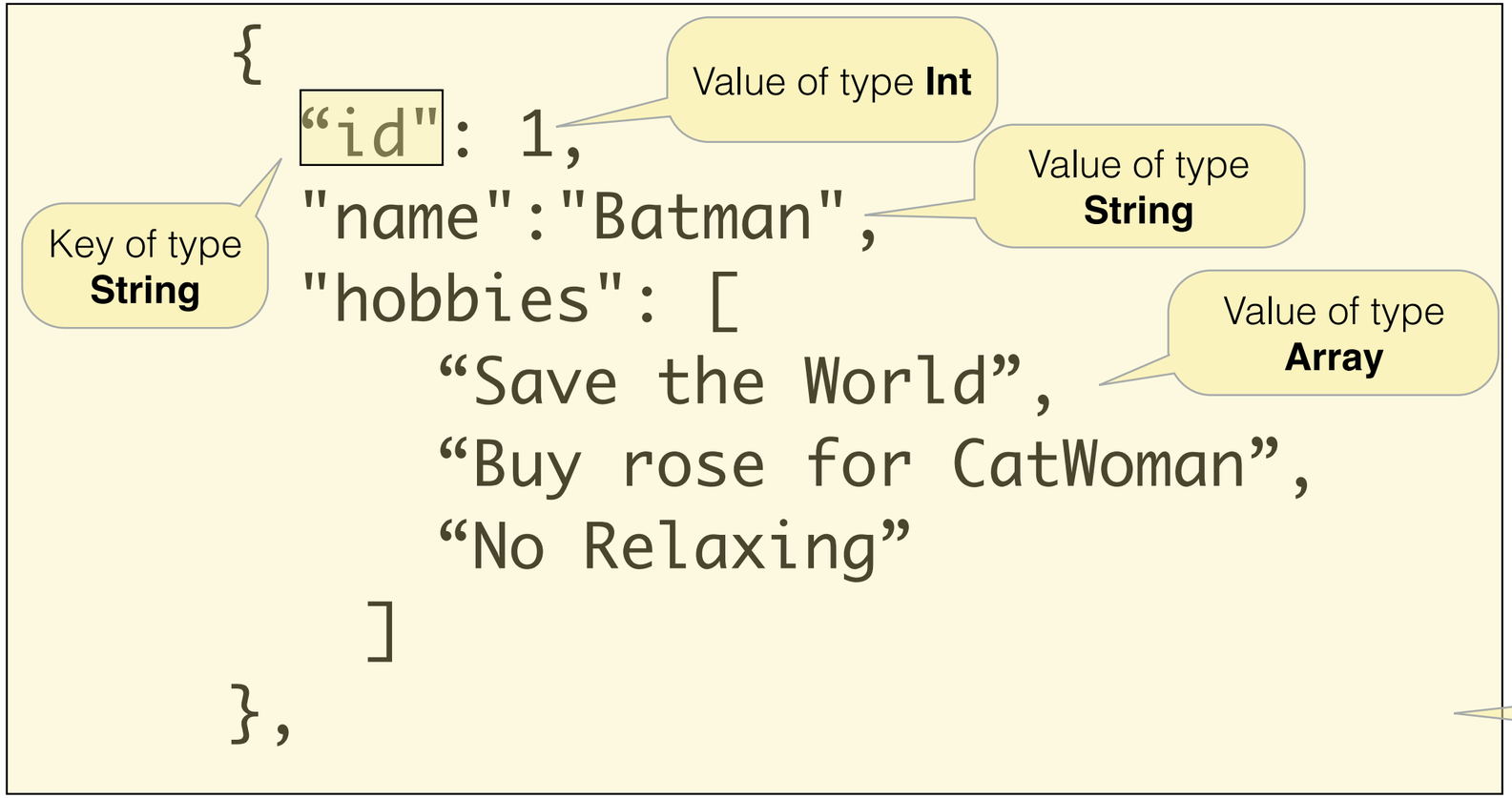
Key of type **String**

Value of type **Array**

One element i.e. Dictionary: key-value pairs

GET Example Response for SuperHeroes

```
Dictionary<String, AnyObject>
```



GET Example Response for SuperHeroes

```
[
  {
    "id":"1",
    "name":"Batman",
    "hobbies": [
      “Save the World”,
      “Buy rose for CatWoman”,
      “No Relaxing”
    ]
  },
  {
    "id":"2",
    "name":"SpiderMan",
    "hobbies": [
      “Save the World”,
      “Buy rose for Mary Jane”,
      “Climb Walls”
    ]
  }
```

Array of Dictionaries

```
[Dictionary<String, AnyObject>]
```

GET Example in AFNetworking

REST command

Endpoint

Additional *Query String* Parameters

Response Object

```
manager.GET("superheroes", parameters: nil,
    success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
```

Success closure

```
    },
    failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in
```

Failure closure

```
    }
)
```

GET Example in AFNetworking

```
success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in

    let superheroesDict = responseObject as [Dictionary<String, AnyObject>]
```

Casting Response

Array of Dictionaries

GET Example in AFNetworking

```
success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
    let superheroesDict = responseObject as [Dictionary<String, AnyObject>]
    let superheroes = self.serializationHelper.superheroesFromDict(superheroesDict)
```

Array of Dictionaries

Casting Response

Array of SuperHeroes

Serialization Helper Class

De-Serialization to get array of Superheroes

GET Example in AFNetworking

```
success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
        let superheroesDict = responseObject as [Dictionary<String, AnyObject>]
        let superheroes = self.serializationHelper.superheroesFromDict(superheroesDict)
        println(superheroes)
},
failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in
        println(error)
}
```

Casting Response

De-Serialization to get array of Superheroes

GET Example in AFNetworking

REST command

Endpoint

Additional *Query String* Parameters

```
manager.GET("superheroes", parameters: nil,
    success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
        let superheroesDict = responseObject as [Dictionary<String, AnyObject>]
        let superheroes = self.serializationHelper.superheroesFromDict(superheroesDict)
        println(superheroes)
    },
    failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in
        println(error)
    }
)
```

Success closure

Failure closure

GET
Exercise

Exercise 1.2: GET Student List

Implement Previous Exercise of Retrieving list of students with **AFNetworking**

- Open RegistrationApp Xcode Project
- Go to *updateStudents* function in the *DataManager.swift*
- And comment out *Exercise 1.1 solution*
- In this exercise, we will implement GET functionality using AFNetworking
- Build and run again

Exercise 1.2: GET Student List

Implement Previous Exercise of Retrieving list of students with
AFNetworking

`GET` `http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students`

Login Credentials

Username: *batman*
Password: *catwoman*

Exercise 1.2: GET Student List

Implement Previous Exercise of Retrieving list of students with **AFNetworking**

`GET` `http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students`

- Go to *updateStudents* function in the *DataManager.swift*

```
func updateStudents(success: (students:[Student]) -> Void, failure: (error : NSError!) -> Void) {
```

Functions as Parameters

Exercise 1.2: GET Student List

Implement Previous Exercise of Retrieving list of students with **AFNetworking**

`GET` `http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students`

Steps->

- Cast *ResponseObject* (**Hint**: Print the response to console and try to infer its type)
- For Deserialization, use *studentsFromArrayOfDictionary* function from *serializationHelper* to get the array of students
- Call Success and Failure closures of *updateStudents* function at the right time

Solution 1.2: GET Student List

REST command

Endpoint

```
manager.GET("students", parameters: nil,
    success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
        let studentsDict = responseObject as [Dictionary<String, AnyObject>]
        let studentsArray = self.serializationHelper.studentsFromArrayOfDictionary(studentsDict)
        success(students: studentsArray)
    },
    failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in
        failure(error: error)
    }
)
```

Casting Response

De-Serialization to get array of Students

Call Success Closure with Students array

Call Failure Closure with error

Solution 1.2: GET Student List

```
func updateStudents(success: (students:[Student]) -> Void, failure: (error : NSError!) -> Void) {
    manager.GET("students", parameters: nil,
        success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in

            let studentsDict = responseObject as [Dictionary<String, AnyObject>]

            let studentsArray = self.serializationHelper.studentsFromArrayOfDictionary(studentsDict)

            success(students: studentsArray)
        },
        failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in

            failure(error: error)

        }
    )
}
```

That was way more easy, right?

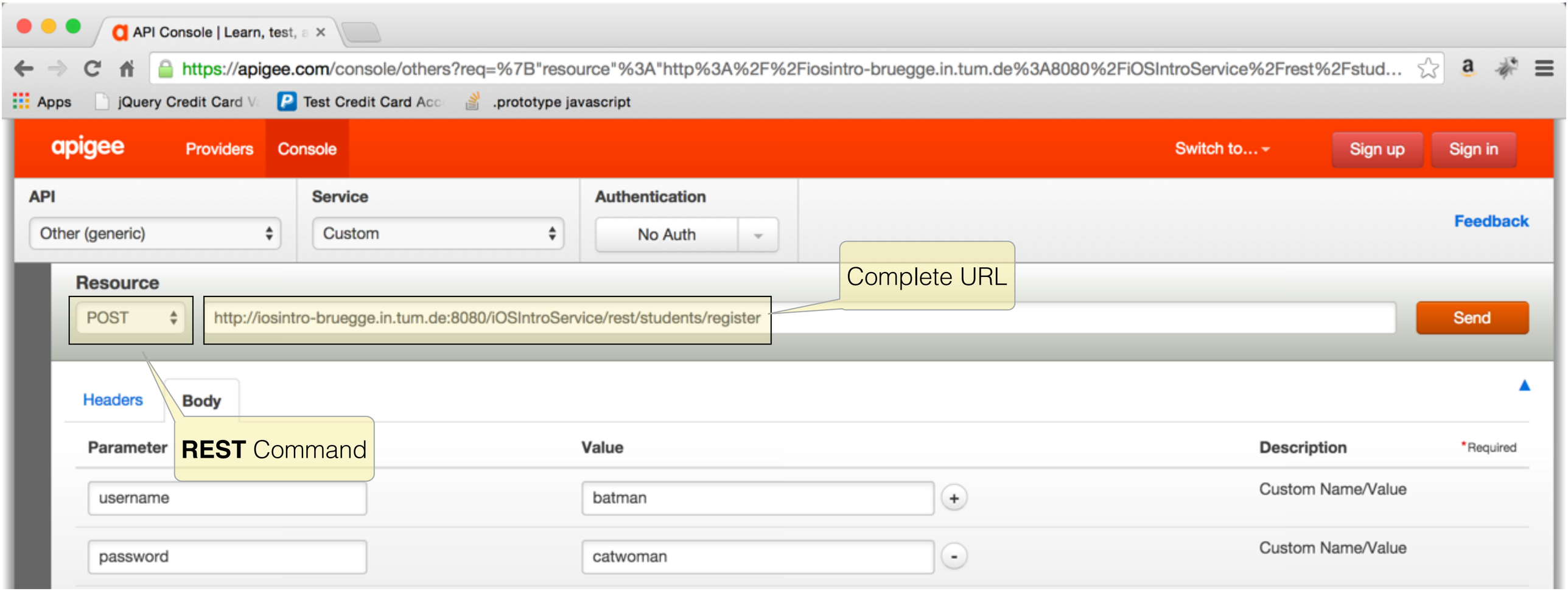
Let’s do a POST now 😊

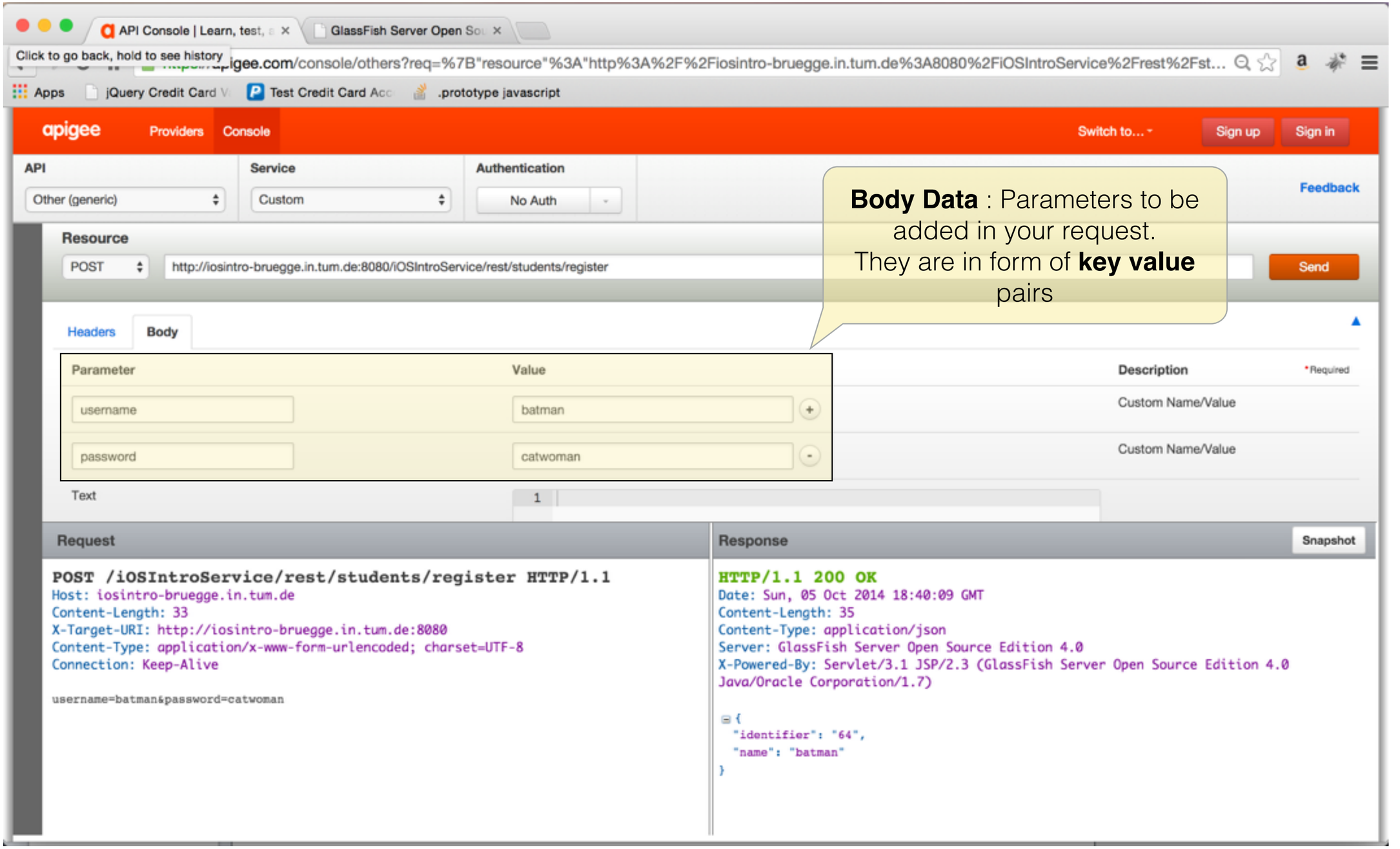
An online REST-console (click on the image)

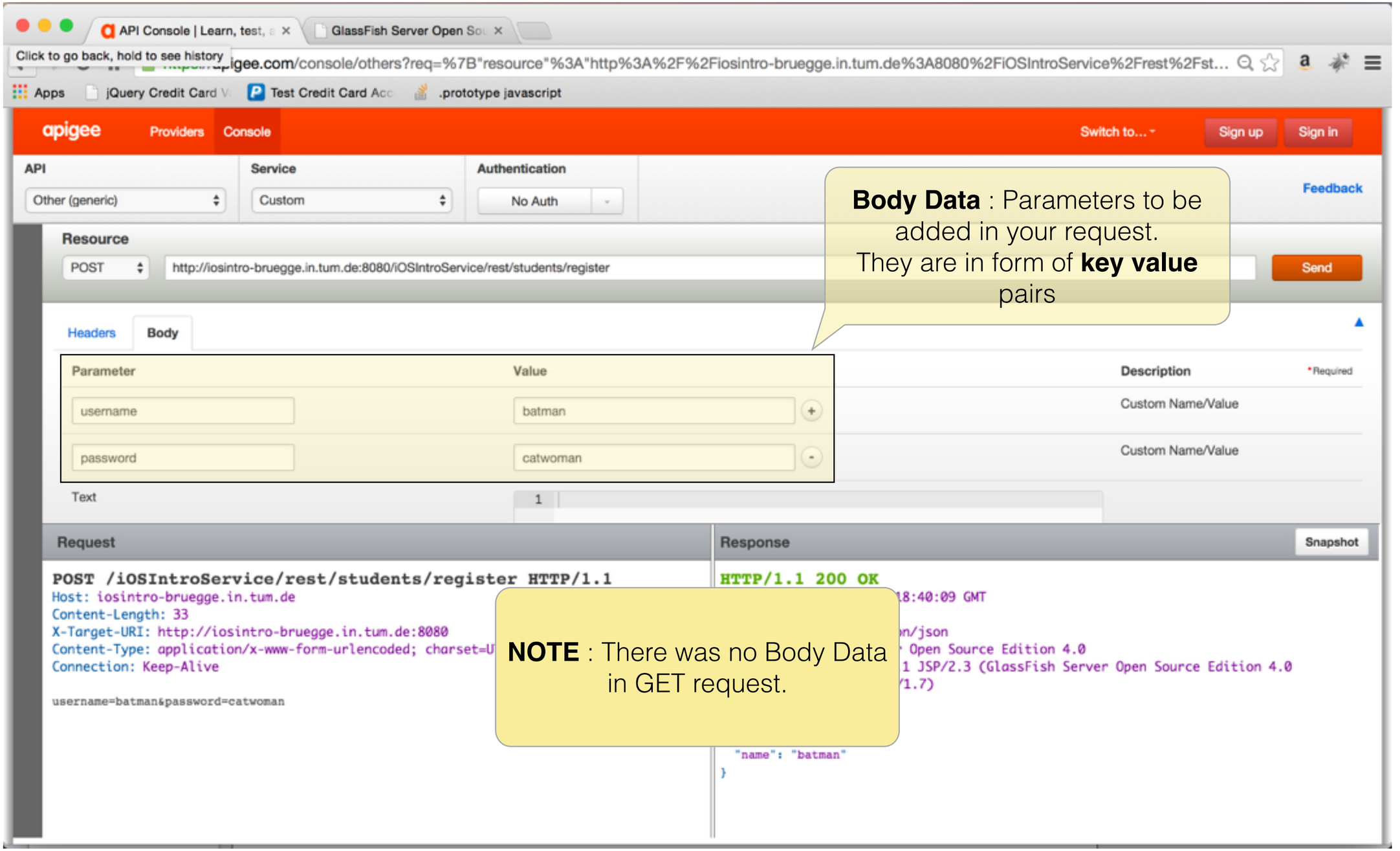
Execute the command

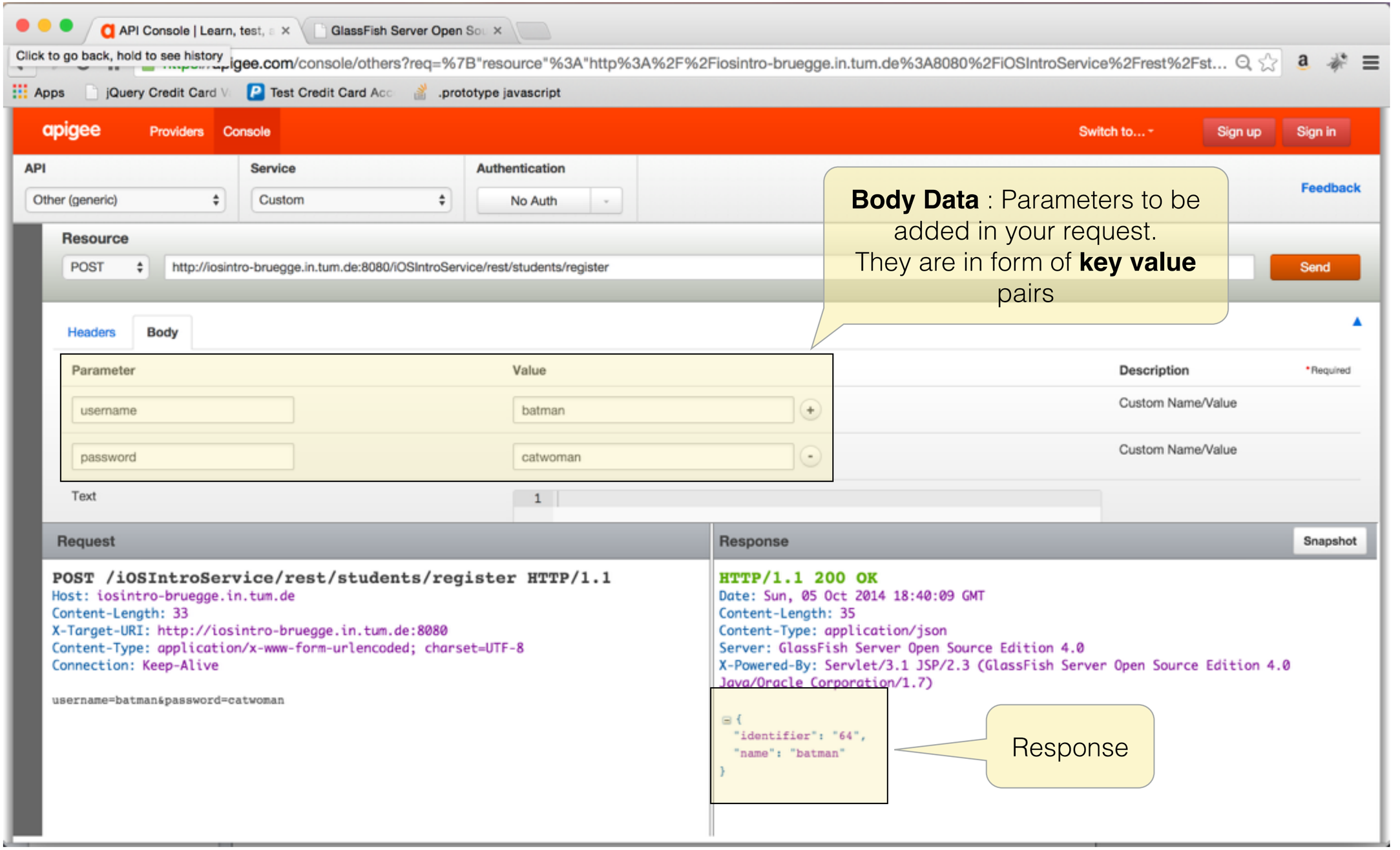
POST http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students/register

POST in Browser




API Console | Learn, test,
GlassFish Server Open So
Click to go back, hold to see history
igee.com/console/others?req=%7B"resource"%3A"http%3A%2F%2Fiosintro-bruegge.in.tum.de%3A8080%2FiOSIntroService%2Frest%2Fst...
Apps
jQuery Credit Card V
Test Credit Card Acc
.prototype javascript
apigee
Providers
Console
Switch to...
Sign up
Sign in
API
Other (generic)
Service
Custom
Authentication
No Auth
Feedback
Resource
POST
http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students/register
Send
Body Data : Parameters to be added in your request. They are in form of key value pairs
Headers
Body
Parameter
Value
Description
*Required
username
batman
Custom Name/Value
password
catwoman
Custom Name/Value
Text
1
Request
POST /iOSIntroService/rest/students/register HTTP/1.1
Host: iosintro-bruegge.in.tum.de
Content-Length: 33
X-Target-URI: http://iosintro-bruegge.in.tum.de:8080
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Connection: Keep-Alive
username=batman&password=catwoman
Response
Snapshot
HTTP/1.1 200 OK
Date: Sun, 05 Oct 2014 18:40:09 GMT
Content-Length: 35
Content-Type: application/json
Server: GlassFish Server Open Source Edition 4.0
X-Powered-By: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.0
Java/Oracle Corporation/1.7)
{
"identifier": "64",
"name": "batman"
}


API Console | Learn, test,
GlassFish Server Open So
Click to go back, hold to see history
Apps
jQuery Credit Card V
Test Credit Card Acc
.prototype javascript
apigee
Providers
Console
Switch to...
Sign up
Sign in
API
Other (generic)
Service
Custom
Authentication
No Auth
Feedback
Resource
POST
http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students/register
Send
Headers
Body
Parameter
Value
Description
*Required
username
batman
Custom Name/Value
password
catwoman
Custom Name/Value
Text
1
Request
POST /iOSIntroService/rest/students/register HTTP/1.1
Host: iosintro-bruegge.in.tum.de
Content-Length: 33
X-Target-URI: http://iosintro-bruegge.in.tum.de:8080
Content-Type: application/x-www-form-urlencoded; charset=U
Connection: Keep-Alive
username=batman&password=catwoman
Response
Snapshot
HTTP/1.1 200 OK
"name": "batman"
}
Body Data : Parameters to be added in your request.
They are in form of key value pairs
NOTE : There was no Body Data in GET request.


API Console | Learn, test,
GlassFish Server Open So
Click to go back, hold to see history
igee.com/console/others?req=%7B"resource"%3A"http%3A%2F%2Fiosintro-bruegge.in.tum.de%3A8080%2FiOSIntroService%2Frest%2Fst...
Apps
jQuery Credit Card V
Test Credit Card Acc
.prototype javascript
apigee
Providers
Console
Switch to...
Sign up
Sign in
API
Other (generic)
Service
Custom
Authentication
No Auth
Feedback
Resource
POST
http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students/register
Send
Headers
Body
Parameter
Value
Description
*Required
username
batman
Custom Name/Value
password
catwoman
Custom Name/Value
Text
1
Request
POST /iOSIntroService/rest/students/register HTTP/1.1
Host: iosintro-bruegge.in.tum.de
Content-Length: 33
X-Target-URI: http://iosintro-bruegge.in.tum.de:8080
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Connection: Keep-Alive
username=batman&password=catwoman
Response
Snapshot
HTTP/1.1 200 OK
Date: Sun, 05 Oct 2014 18:40:09 GMT
Content-Length: 35
Content-Type: application/json
Server: GlassFish Server Open Source Edition 4.0
X-Powered-By: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.0
Java/Oracle Corporation/1.7)
{
"identifier": "64",
"name": "batman"
}
Body Data : Parameters to be added in your request. They are in form of key value pairs
Response

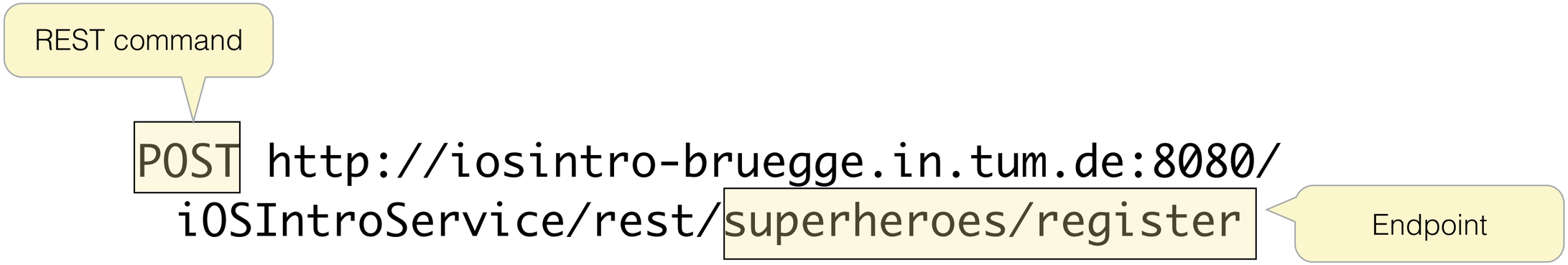
Now, we see how we do a POST in AFNetworking

POST Example in AFNetworking

To add a new superhero with **AFNetworking**

POST Example in AFNetworking

To add a new superhero with **AFNetworking**



POST Example in AFNetworking

REST command

Endpoint

Body Data

In Dictionary form

“username” : Batman
“password” : Catwoman

```
manager.POST(“superheroes/register", parameters: superheroAttributes,
```

Success closure

```
    success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
```

```
    },
```

Failure closure

```
    failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in

        failure(error: error)

    }
)
```

POST Example in AFNetworking

REST command

Endpoint

Body Data

In Dictionary form

“username” : Batman
“password” : Catwoman

```
manager.POST(“superheroes/register", parameters: superheroAttributes,
```

Success closure

```
    success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
```

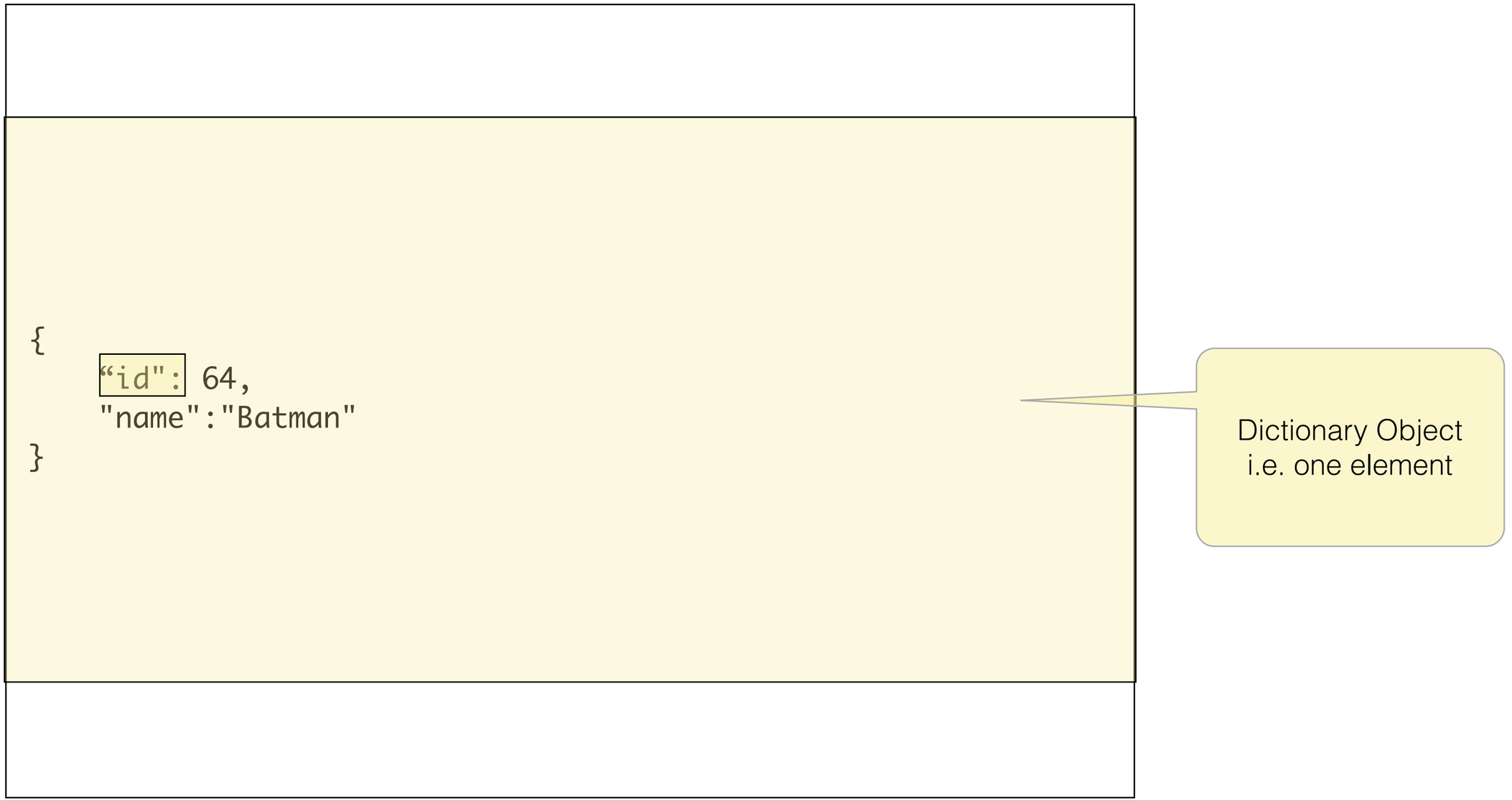
Response Object

```
    },
```

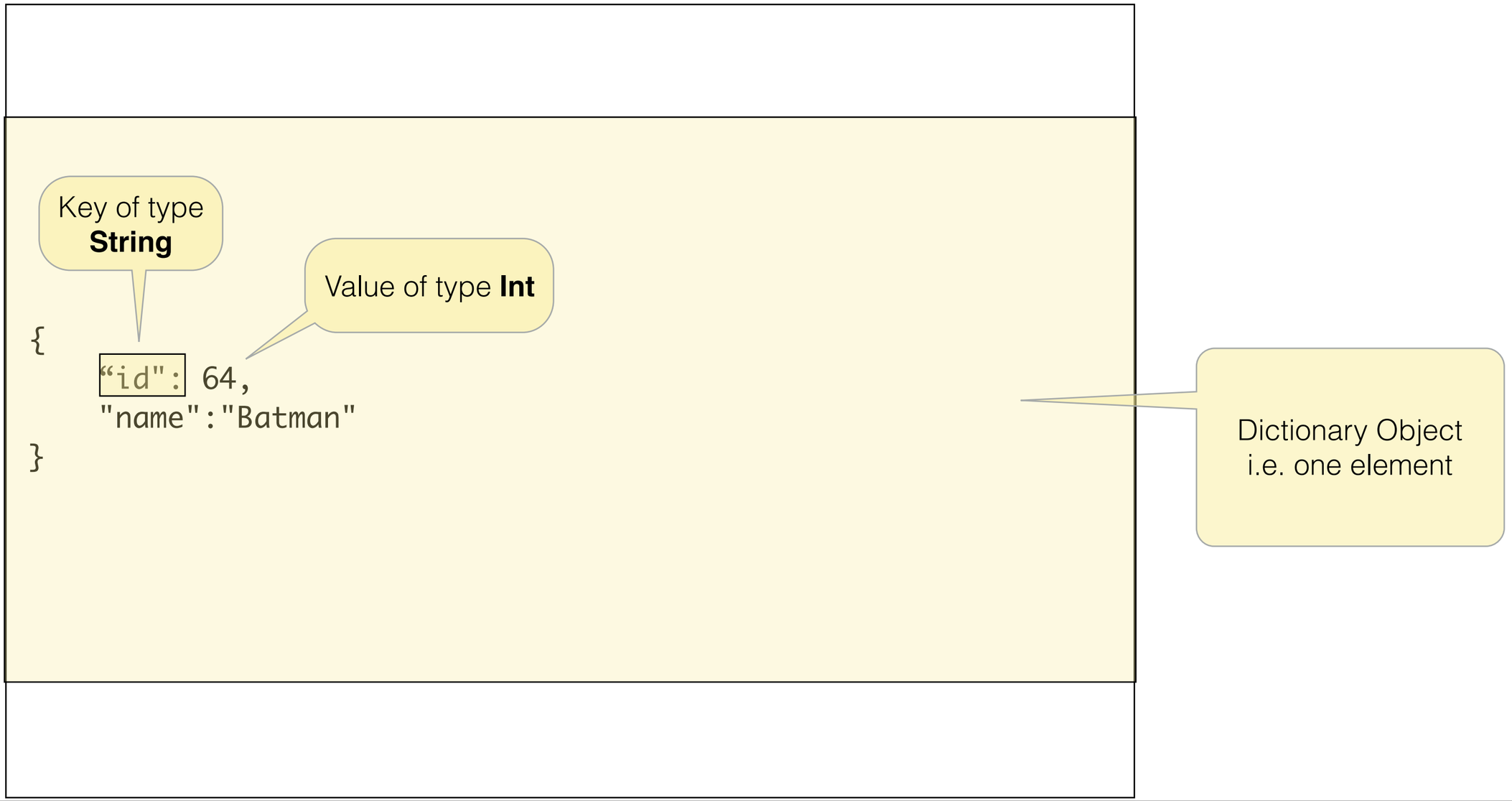
Failure closure

```
    failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in

        failure(error: error)

    }
)
```

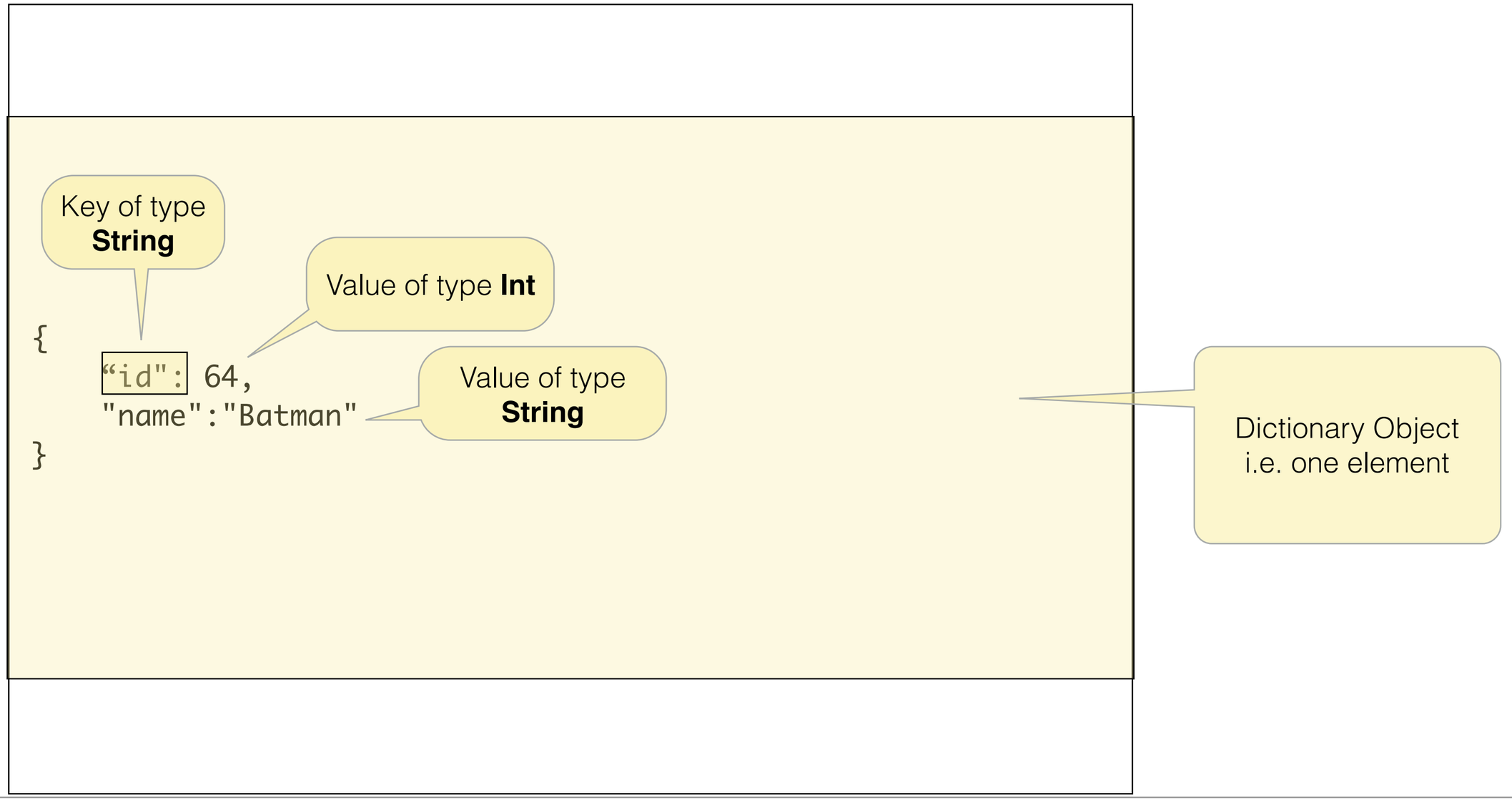
POST Example Response for Superheroes



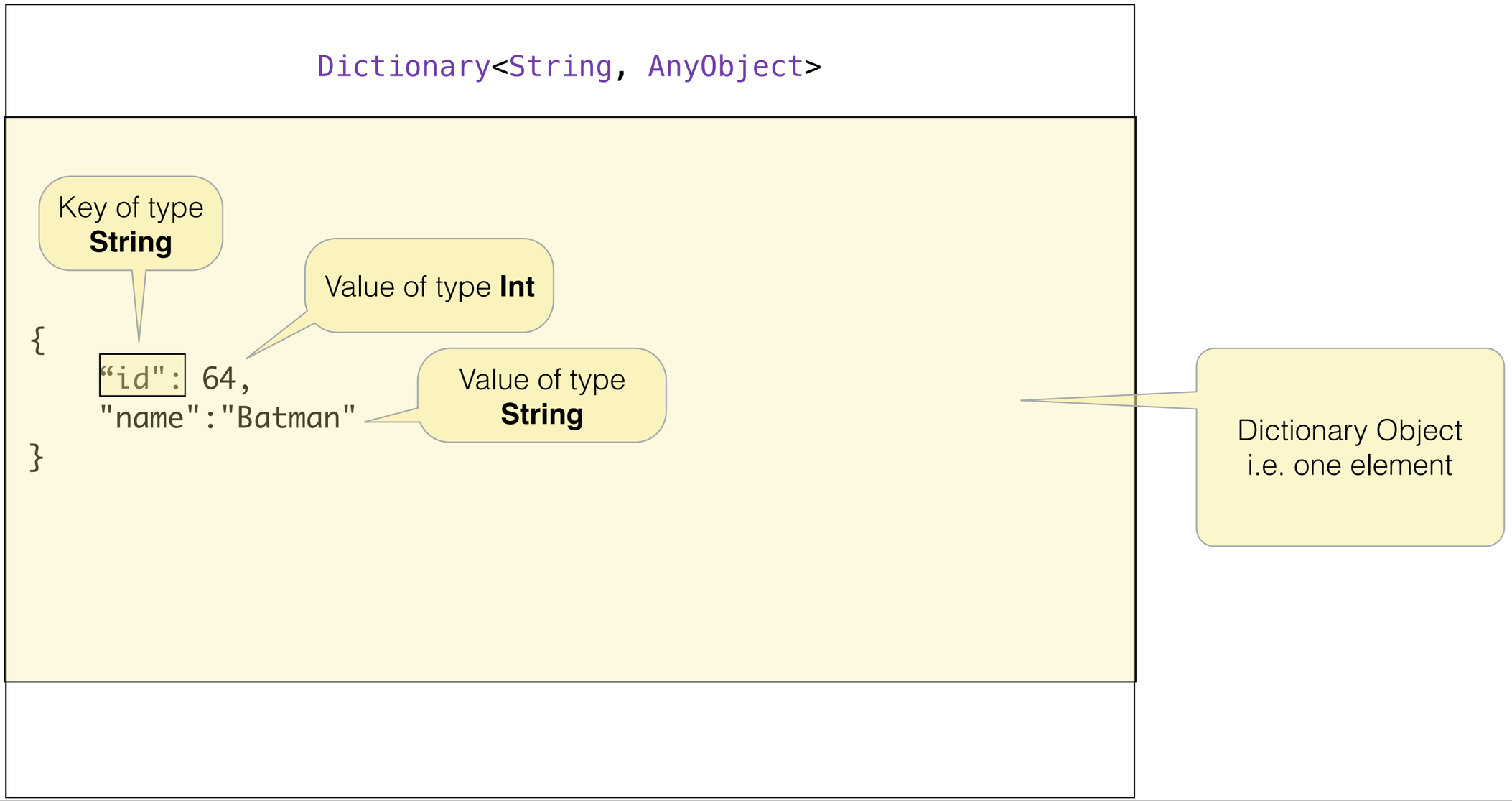
POST Example Response for Superheroes



POST Example Response for Superheroes



POST Example Response for Superheroes



POST Example in AFNetworking

```
success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
    let superheroDict = responseObject as Dictionary<String, AnyObject>
```

Dictionary Object

Casting Response

POST Example in AFNetworking

```
success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
    let superheroDict = responseObject as Dictionary<String, AnyObject>
    self.superhero = self.serializationHelper.superheroFromDict(superheroDict)
```

Dictionary Object

Casting Response

Superhero Object

De-Serialization to get object of Superhero

POST Example in AFNetworking

```
success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
        let superheroDict = responseObject as Dictionary<String, AnyObject>
        self.superhero = self.serializationHelper.superheroFromDict(superheroDict)
        println(self.superhero)
 },
failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in
        println(error)
 }
```

Casting Response

De-Serialization to get object of Superhero

POST Example in AFNetworking

REST command

Endpoint

Body Data

"username" : Batman
"password" : Catwoman

```
manager.POST("superheroes/register", parameters: superheroAttributes,
    success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in
        let superheroDict = responseObject as Dictionary<String, AnyObject>
        self.superhero = self.serializationHelper.superheroFromDict(superheroDict)
        println(self.superhero)
    },
    failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in
        println(error)
    }
)
```

Success closure

Failure closure

Authentication

How can we Authenticate a GET/POST Request?

Authentication

How can we Authenticate a GET/POST Request?

```
let endpoint = “superheroes/register”
             + “?username=\(self.superhero.username)”
             + ”&password=\(self.superhero.password)”
```

Authentication as Query string

(not how you should do it in a real life application)

Authentication

How can we Authenticate a GET/POST Request?

```
let endpoint = “superheroes/register”

manager.requestSerializer.setAuthorizationHeaderFieldWithUsername("batman", password: "catwoman")
```

Authentication parameters included in the Header Field

(Recommended way)

Authentication

Authentication as Query string

```
let endpoint = “superheroes/register?username=\(self.superhero.username)&password=\(self.superhero.password)”

 manager.GET/POST(endpoint, parameters: superheroAttributes,
    success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in




    },
   failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in
            failure(error: error)
   }
 )
```

POST
Exercise

Exercise 2: POST Student

Implement POST to register yourself as student in the intro course
AFNetworking

- Reopen RegistrationApp project
- Build and run it again
- In your login screen you already have a register button
- In this exercise, you will implement this registration functionality

Exercise 1.2: GET Student List

Implement Previous Exercise of Retrieving list of students with **AFNetworking**

`GET` `http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students`

- Go to *registerStudent* in the *DataManager.swift*

```
func registerStudent(studentAttributes: Dictionary<String, String>,
                     success: (student:Student!) -> Void,
                     failure: (error : NSError!) -> Void) {
```

Functions as Parameters

Exercise 2: POST Student

Implement POST to register yourself as student in the intro course
AFNetworking

`POST` `http://iosintro-bruegge.in.tum.de:8080/iOSIntroService/rest/students/register`

- **NO Authentication** required for this exercise
- Remember **Body Data** should be a dictionary :check *studentAttributes?*

Steps->

- Cast *ResponseObject* (**Hint**: Print the response to console and try to infer its type)
- Use *studentFromDictionary* function from *serializationHelper* to get the student attributes and store the result in *self.student*
- Call Success closure and failure at the right time

Solution 2: POST Student

REST command

Endpoint

Body Data

```
manager.POST("students/register", parameters: studentAttributes,
    success: { (operation:NSURLSessionDataTask!, responseObject:AnyObject!) -> Void in

        let studentDict = responseObject as Dictionary<String, AnyObject>

        self.student = self.serializationHelper.studentFromDictionary(studentDict)

        success(student:self.student)

    },
    failure: { (operation:NSURLSessionDataTask!, error:NSError!) -> Void in

        failure(error: error)

    }
)
```

Casting Response

Call Success Closure with self.student

De-Serialization to get Student

Call Failure Closure with error

Thats how you do GET & POST in AFNetworking!

Easy? 😊

Summary

- There are different ways how you communicate with a server.
- Apple Foundation provides *NSURLSession*, which is new preferred method for communication.
- Alternate to that, we have a widely used 3rd party library, AFNetworking which is more easy to use for complex tasks.

References

- AFNetworking Documentation
- AFNetworking Integrate Swift
- NSURLSession Documentation
- Programmable Web
- Google Trends
- REST on Wikipedia

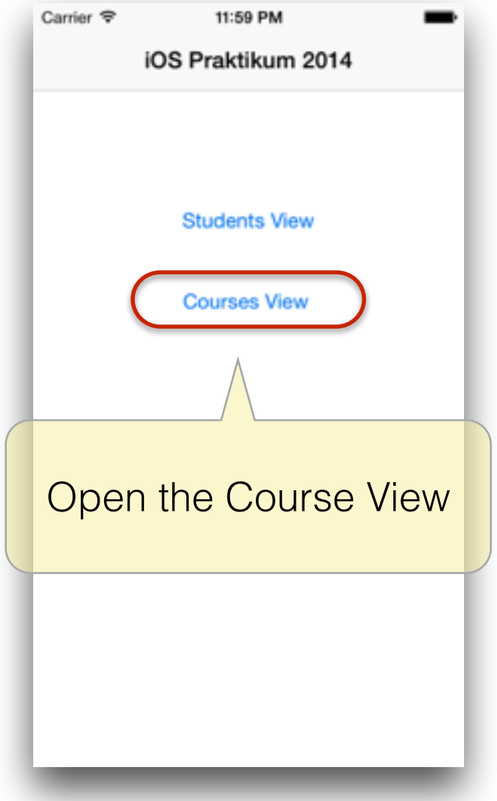
Final Exercise
GET

Task 1: GET list of Course Sessions

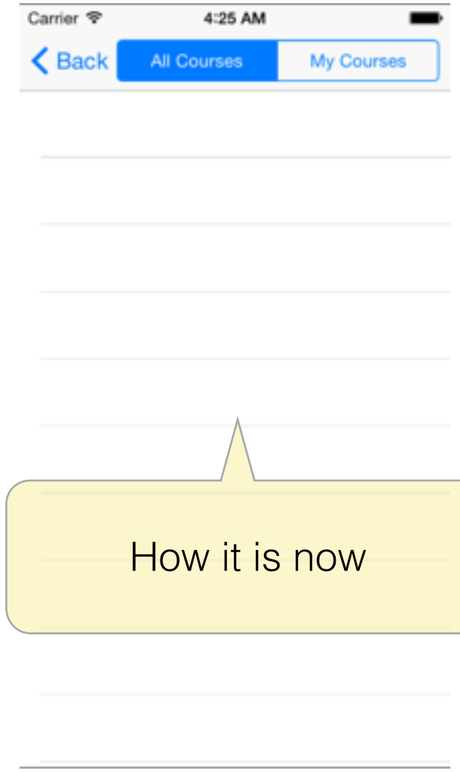
- Find the function *updateAllAvailableCourses()* in DataManager
- **GET** /sessions
- **Params**: nil
- Cast *ResponseObject* (**Hint**: Print the response to console and try to infer its type)
- Response: Use De-serialization function *coursesFromArray*
- Store the response in *self.availableCourseTimeslots*
- Finally call success and failure closures at right time

Easy 😊

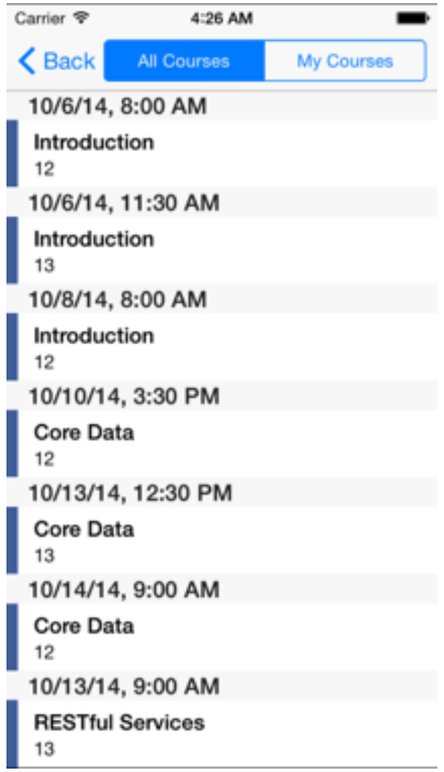
Final Exercise
GET



Open the Course View



How it is now



How it should be like

Login Credentials

Username: *batman*
Password: *catwoman*

Final Exercise
POST

Task 2: POST Register for a Course Session

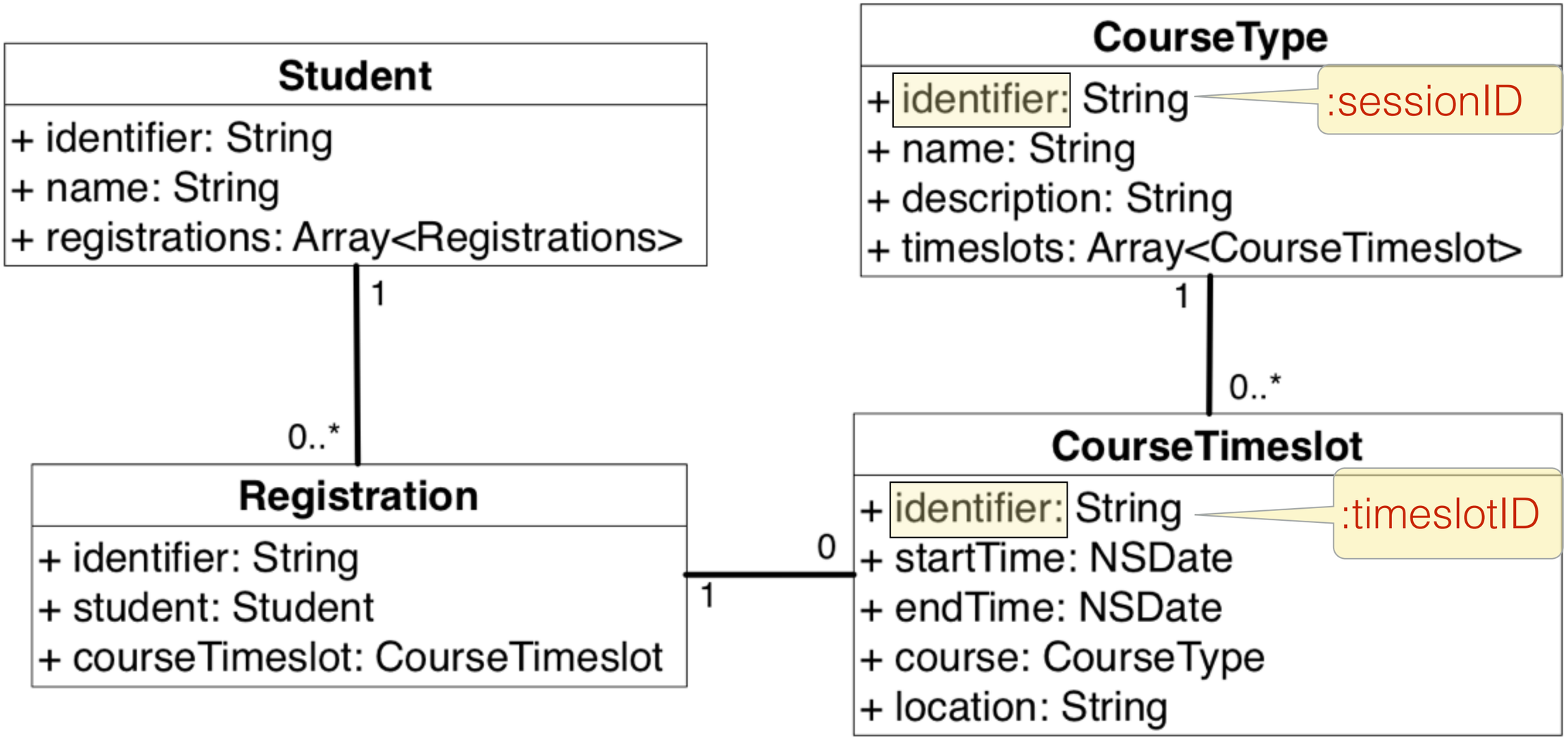
- Find the function *postNewRegistration()* in DataManager
- **POST** /sessions/:sessionID/timeslots/:timeslotID/registrations?username=testStudent&password=testPassword
- **Remember** Authentication as Query String
- **Body Data**: Use serialization function *registrationToJSON* from *serializationHelper*
- Cast *ResponseObject* (**Hint**: Print the response to console and try to infer its type)
- **Response**: Use deserialization function *addRegistrationFromDictionary from serializationHelper*
- Finally call success and failure closures at right time

Hint: Use *registration* parameter already passed in the *postNewRegistration* function

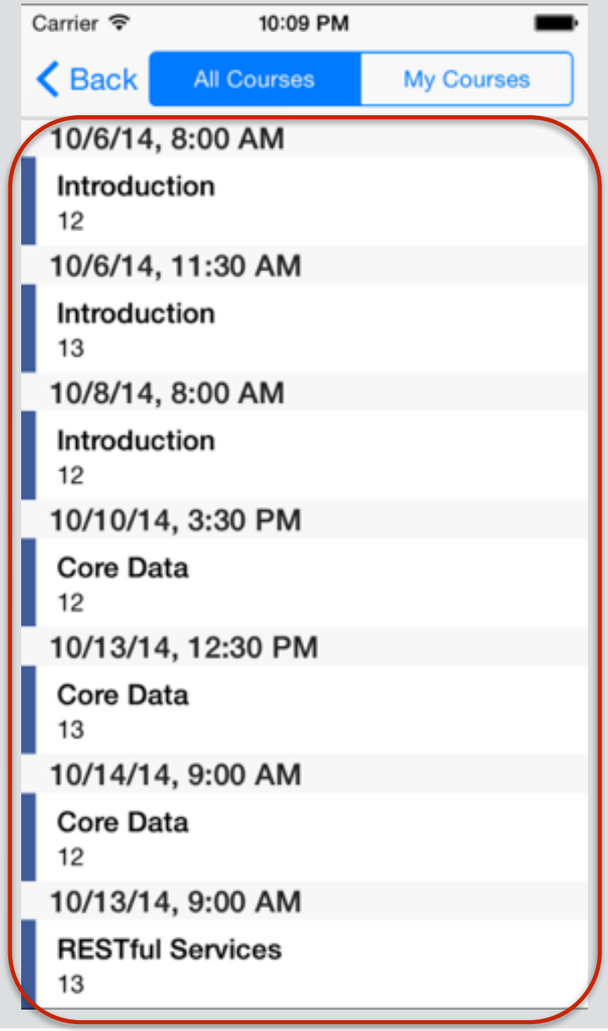
Difficult 😊

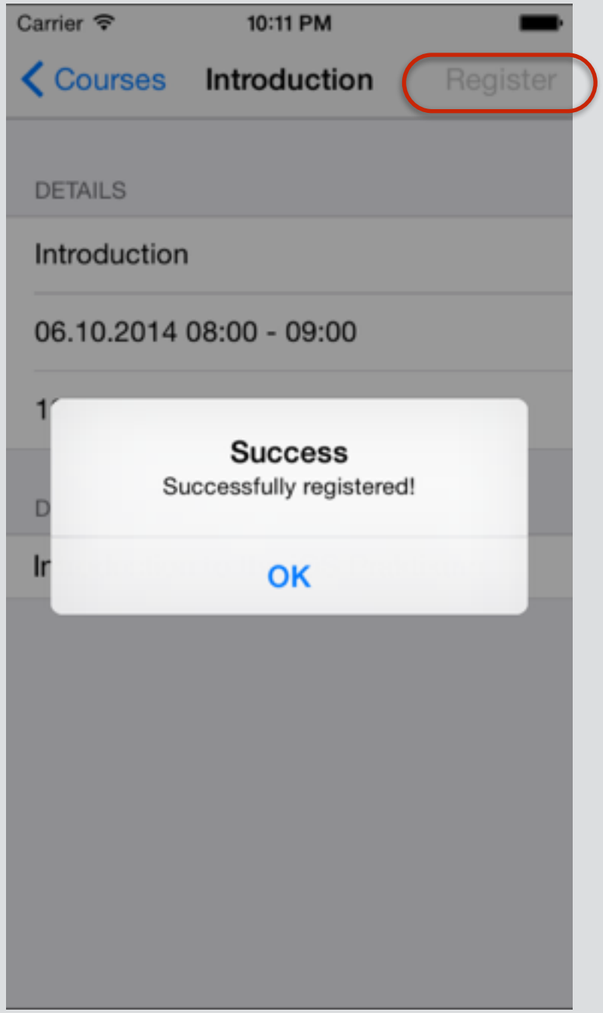
Final Exercise

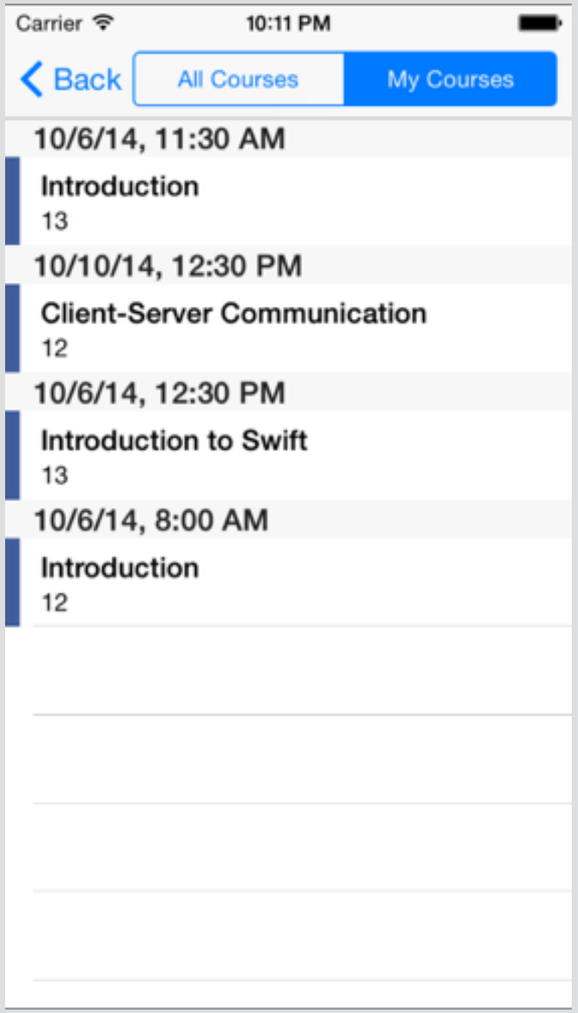
UML



Final Exercise
POST







Thanks for listening 😄
Have fun in the iOS Praktikum!