Mobile Application Development

Higher Diploma in Science in Computer Science

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

http://elearning.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

Application Programmer Interfaces

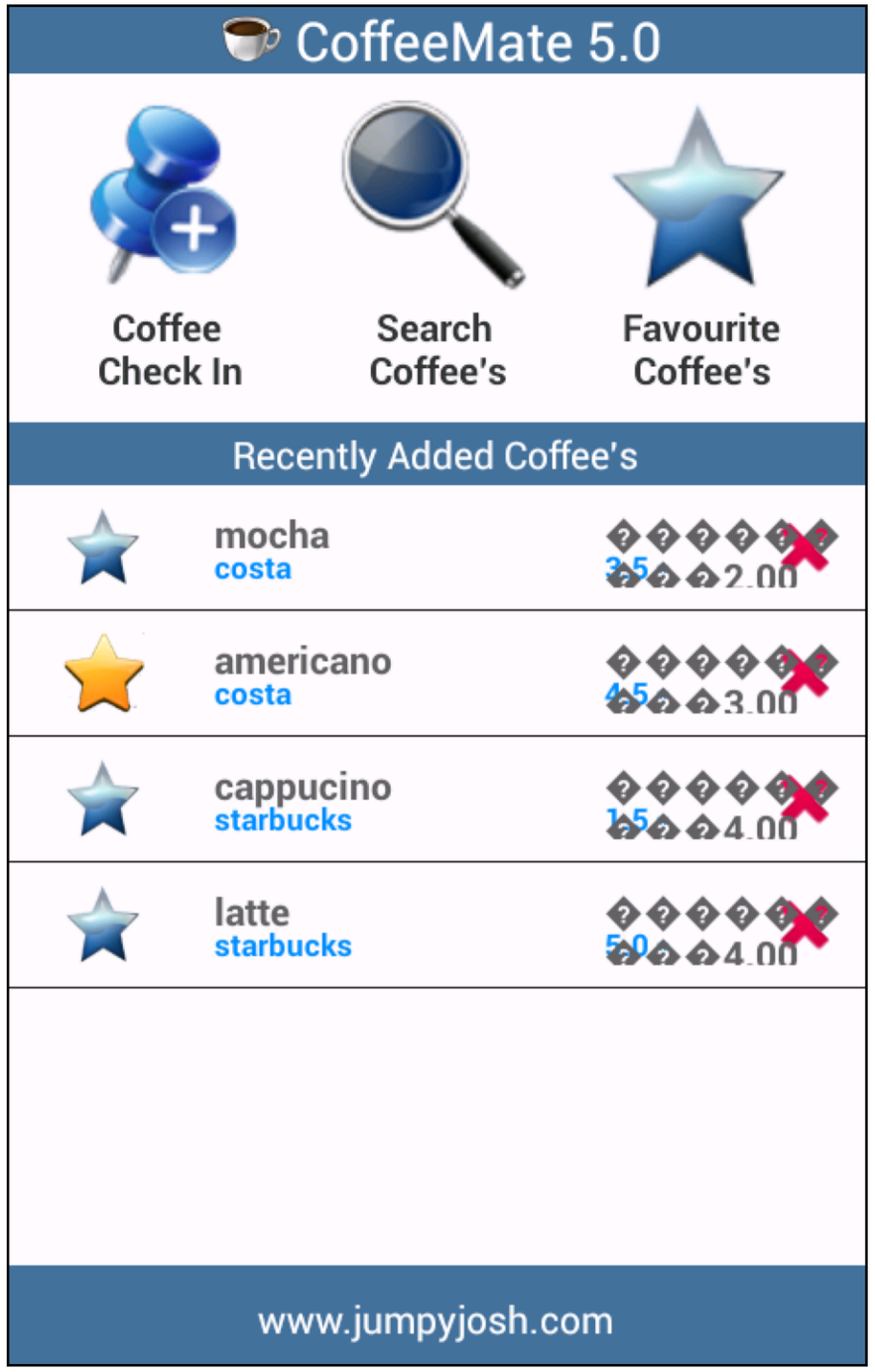
API Introduction

- User Experience vs Developer Experience
- Common Information Formats
- coffeemate-service V1

User Experience

http://en.wikipedia.org/wiki/User_experience

User experience (**UX**) involves a person's behaviors, attitudes, and emotions about using a particular product, system or service. User experience includes the practical, experiential, affective, meaningful and valuable aspects of human-computer interaction and product ownership. Additionally, it includes a person's perceptions of system aspects such as utility, ease of use and efficiency. User experience may be considered subjective in nature to the degree that it is about individual perception and thought with respect to the system. User experience is dynamic as it is constantly modified over time due to changing usage circumstances and changes to individual systems as well as the wider usage context in which they can be found.



Developer Experience

http://uxmag.com/articles/effective-developer-experience

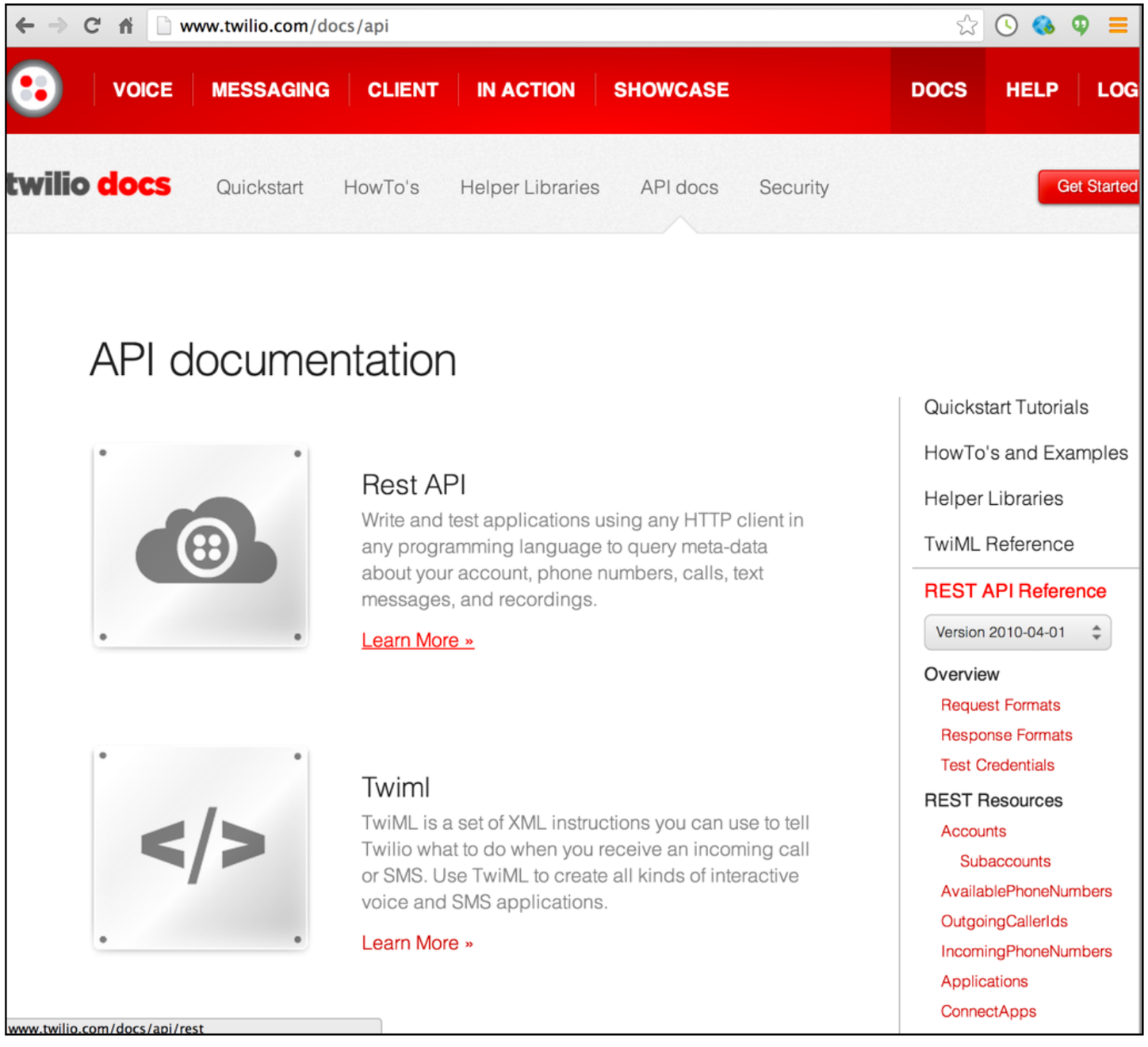
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6225984

---

http://blog.pusher.com/getting-the-developer-experience-right/

"For all the abundance of APIs, there are multiple ways of getting a user up to speed with your product. The experience of learning how to use an API is very similar to that of using a consumer facing website or application. Indeed your conversion rate is likely to be higher if you remove as much friction as possible.

There has been a huge amount of literature produced to make those kind of consumer user experiences better, but only limited information about what the ingredients of a highly successful onboarding experience for an API product are. The principles are generally very similar...



http://www.twilio.com/docs/api

Common Formats

- Comma Separated Values (CSV)
- Name/Value Pairs
- YAML
- XML
- JSON

CSV

http://en.wikipedia.org/wiki/Comma-separated_values

A **comma-separated values** (**CSV**) (also sometimes called *character-separated values,* because the separator character does not have to be a comma) file stores tabular data (numbers and text) in plain-text form. Plain text means that the file is a sequence of characters, with no data that has to be interpreted instead, as binary numbers. A CSV file consists of any number of records, separated by line breaks of some kind; each record consists of fields, separated by some other character or string, most commonly a literal comma or tab. Usually, all records have an identical sequence of fields.

```
“mocha”, “costa”, 2.0,  3.5,  0
“americano”, “costa”, 3.0, 4.5, 1
“cappucino”, “starbucks”, 4.0, 1.5, 0
```

coffees.csv

Name/Value Pairs

http://en.wikipedia.org/wiki/Attribute-value_pair

A **name–value pair**, **key–value pair**, **field–value pair** or **attribute–value pair** is a fundamental data representation in computing systems and applications. Designers often desire an open-ended data structure that allows for future extension without modifying existing code or data. In such situations, all or part of the data model may be expressed as a collection of tuples <*attribute name*, *value*>; each element is an attribute–value pair. Depending on the particular application and the implementation chosen by programmers, attribute names may or may not be unique.

```
db.url=jdbc:cloudbees://pacemaker
db.driver=com.mysql.jdbc.Driver
db.user=pacemaker
db.pass=pacemaker
jpa.ddl=create
```

application .conf

```
name="mocha"
shop="costa"
rating=3.5
price=2.0
favourite=0
id=1
```

coffees.conf

YAML

http://en.wikipedia.org/wiki/YAML

YAML (/ˈjæməl/, rhymes with *camel*) is a human-readable data serialization format that takes concepts from programming languages such as C, Perl, and Python, and ideas from XML and the data format of electronic mail (RFC 2822). YAML was first proposed by Clark Evans in 2001,[1] who designed it together with Ingy döt Net[2] and Oren Ben-Kiki.[2] It is available for several programming languages.

YAML is a recursive acronym for "YAML Ain't Markup Language". Early in its development, *YAML* was said to mean "Yet Another Markup Language",[3] but it was then reinterpreted (backronyming the original acronym) to distinguish its purpose as data-oriented, rather than document markup.

```
Coffee(c1):
    name       : mocha
    shop       : costa
    price      : 2.0
    rating     : 3.5
    favourite  : 0

Coffee(c2):
    name       : americano
    shop       : costa
    price      : 3.0
    rating     : 4.5
    favourite  : 1


Coffee(c3):
    name       : cappucino
    shop       : starbucks
    price      : 4.0
    rating     : 1.5
    favourite  : 0
```

data.yaml

XML

http://en.wikipedia.org/wiki/XML

Extensible Markup Language (**XML**) is a markup language that defines a set of rules for encoding documents in aformat that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification[3] produced by theW3C, and several other related specifications,[4] all free open standards.[5]

The design goals of XML emphasize simplicity, generality, and usability over the Internet.[6] It is a textual data format with strong support via Unicode for the languages of the world. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services.

Many application programming interfaces (APIs) have been developed to aid software developers with processing XML data, and several schema systems exist to aid in the definition of XML-based languages.

```xml
<?xml version="1.0"
encoding="UTF-8"?>

<coffee objname="c1">
    <name> mocha </name>
    <shop> costa </shop>
    <price> 2.0 </price>
    <rating> 3.5</rating>
    <favourite> 0 </favourite>
</coffee>

<coffee objname="c1">
    <name> americano </name>
    <shop> costa </shop>
    <price> 3.0 </price>
    <rating> 4.5 </rating>
    <favourite> 1 </favourite>
</coffee>

<coffee objname="c1">
    <name> cappucino </name>
    <shop> starbucks </shop>
    <price> 4.0 </price>
    <rating> 1.5 </rating>
    <favourite> 0 </favourite>
</coffee>
```

data.xml

JSON

http://en.wikipedia.org/wiki/JSON

JSON (/ˈdʒeɪsɒn/ ***jay-sawn***, /ˈdʒeɪsən/ ***jay-sun***), or **JavaScript Object Notation**, is a text-based open standard designed for human-readable data interchange. Derived from the JavaScript scripting language, JSON is a language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, JSON is language-independent, with parsers available for many languages.

The JSON format was originally specified by Douglas Crockford, and is described in RFC 4627. The official Internet media type for JSON is `application/json`. The JSON filename extension is `.json`.

The JSON format is often used for serializing and transmitting structured data over a network connection. It is used primarily to transmit data between a server and web application, serving as an alternative to XML.

```
{
  "name":"mocha",
  "shop":"costa",
  "rating":3.5,
  "price":2.0,
  "favourite":0,
  "id":1
},
{
  "name":"americano",
  "shop":"costa",
  "rating":4.5,
  "price":3.0,
  "favourite":1,
  "id":2
},
{
  "name":"cappuccino lite",
  "shop":"starbucks",
  "rating":1.5,
  "price":4.0,
  "favourite":1,
  "id":3
}
```

coffemate-service - Lab05,06,07

- coffemate-service is a play project without a user interface
    - ie. there is no User Experience (UX)
- However, it does have a Developer API (Application Programmer Interface)
    - we might call it a Developer Experience (DX)

coffemate-service API

- Routes (to be implemented in labs 05 and 06)

```
GET     /api/coffees                                CoffeeServiceAPI.coffees
GET     /api/coffees/{id}                           CoffeeServiceAPI.coffee
POST    /api/coffees                                CoffeeServiceAPI.createCoffee
PUT     /api/coffees/{id}                           CoffeeServiceAPI.updateCoffee
DELETE  /api/coffees/{id}                           CoffeeServiceAPI.deleteCoffee
```

coffeemate-service API

- Instead of calling ‘render’ - which will build a new HTML view from a set of templates

- ->Call ‘renderJSON’, which will return a Json encoded representation of some object(s).

```
public class CoffeeServiceAPI extends Controller
{
  public static void coffee (Long id)
  {
   Coffee coffee = Coffee.findById(id);
   renderJSON (JsonParsers.coffee2Json(coffee));
  }

 ...
}
```

JsonParsers

- Utilty class to turn specific objects into Json strings

  or

- Json strings into Java Objects

```
public class JsonParsers
{
  static Gson gson = new Gson();

  public static Coffee json2Coffee(String json)
  {
    return gson.fromJson(json, Coffee.class);
  }

  public static String coffee2Json(Object obj)
  {
    return gson.toJson(obj);
  }
}
```


localhost:9000/api/coffee
localhost:9000/api/coffee/1
{"favourite":0,"id":1,"name":"one","price":2.0,"rating":3.5,"shop":"here"}