Internet Applications Design and Implementation

2015 - 2016 - 1st edition

MIEI - Integrated Master in Computer Science and Informatics
Specialization block

João Costa Seco (joao.seco@fct.unl.pt)
Jácome Cunha (jacome@fct.unl.pt)

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Internet Applications Design and Implementation

2015 - 2016 - 1st edition

(1 - Introduction and Overview)

MIEI - Integrated Master in Computer Science and Informatics
Specialization block

João Costa Seco (joao.seco@fct.unl.pt)
Jácome Cunha (jacome@fct.unl.pt)

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Internet Applications Design and Implementation

Most **Internet applications**, both **web** and **mobile**, depend on **resources provided by one or more online services**, which may or may not be **aware of all of its clients**.

Clients range from **simple web pages**, **reactive** and collaborative **single page applications**, mobile **native applications**, to **service providers that orchestrate several other services** to provide a compound result.

These challenges are specific of Internet Applications and should be attained using **specific methods, tools and techniques**.

Internet Applications Design and Implementation

This course focuses on the **principles and concepts on the development of Internet applications**.

The syllabus follows an approach based on the fundamentals of software development based on **web and service oriented architectural patterns**, **advanced modularity mechanisms**, **data persistency abstractions**, **good development practices**, **performance concerns**, and **validation techniques**.

Course lectures are accompanied by a **series of practical assignments** and a **final project development** using frameworks, languages, and programming tools for Internet Applications that ensure the safety and compliance of the solution with relation to a specification

Goals: To Know

- the essential aspects of architectural patterns for inversion of control and software architectures specific for Internet Applications.
- the principles of developing web applications and single page web applications.
- the mechanisms of specifying and implementing web service orchestrations.
- the internal structure of an Internet browser and its client applications.
- the principles of data-centric and user-centric development of Internet applications.
- the main data abstraction mechanisms used in Internet applications.
- the major performance pitfalls of Internet applications and their workarounds.
- the main specification and implementation mechanisms for security properties in Internet Applications.

Goals: To Do

- use development frameworks that implement architectural styles for Internet applications.
- specify and build web and cloud applications to support thin, flat, and native clients.
- specify and build client applications for web and cloud applications with reactive and rich behaviour.
- implement authentication mechanisms and specify the core security rules of an Internet Application
- specify and efficiently use abstraction data layers such as Object Relational Mappings in Internet applications.
- design and deploy Internet Applications that are efficient and maintainable.

Course Logistics

- Lectures (Wed 12h-14h, bring a snack!)
- Labs (starting only next week)
- Schedule:

	2ª	3ª	4ª	5ª	6ª	Sábado
8:00 9:00						
9:00 10:00						
10:00 11:00				**CIAI** p.1 Ed 2: Lab 121/Ed.II		
11:00 12:00						
12:00 13:00			**CIAI** t.1 Ed 2: 127/Ed.II	**CIAI** p.2 Ed 2: Lab 121/Ed.II		
13:00 14:00						
14:00 15:00						
15:00 16:00						
16:00 17:00						

Course Logistics

- Lectures (Wed 12h-14h, bring a snack!)
- Labs (starting only next week)
- Office Hours (by appointment):
  - João Costa Seco: Fri 10h-11h (P2/13)
  - Jácome Cunha: Tue 10h-11h (P3/2)

Evaluation

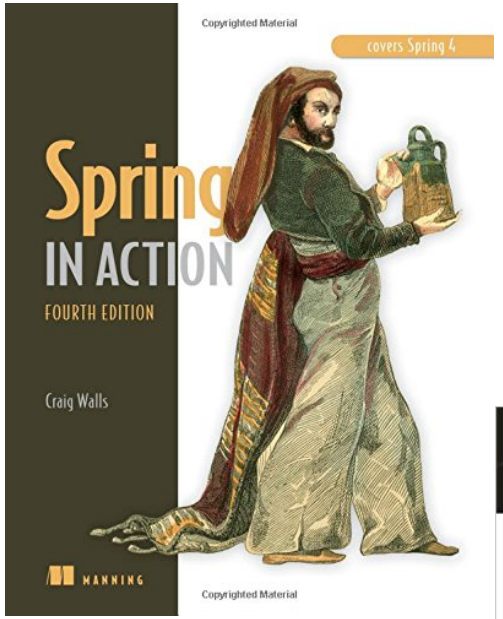
- Written evaluation component (50%)
  - Midterm (Oct 28)
  - Final Test (Dec 9)
- Laboratory work component (50%)
  - TDD development of a sample Internet Application
  - 3 separate deliveries: week 6, 9, and 13
  - a deployed bitbucket project with tags for all delivered versions

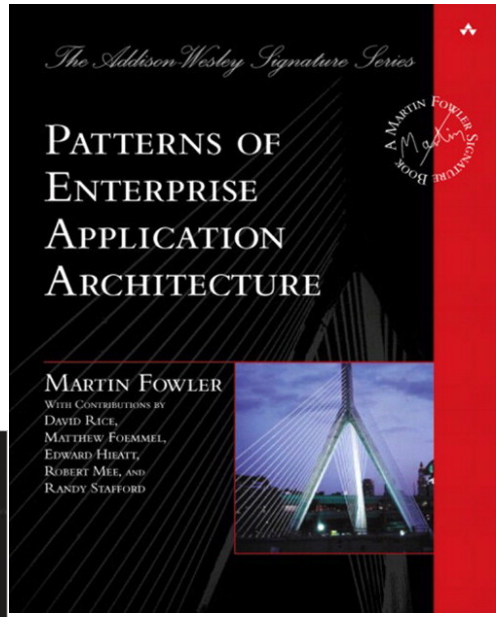
    If you don't know how to operate with git, go learn TODAY!
  - work is evaluated for functionality, performance, method, etc.
  - teams of 3 members (number of commits maybe used)
  - delivery includes a written report and a presentation with demo (20m)

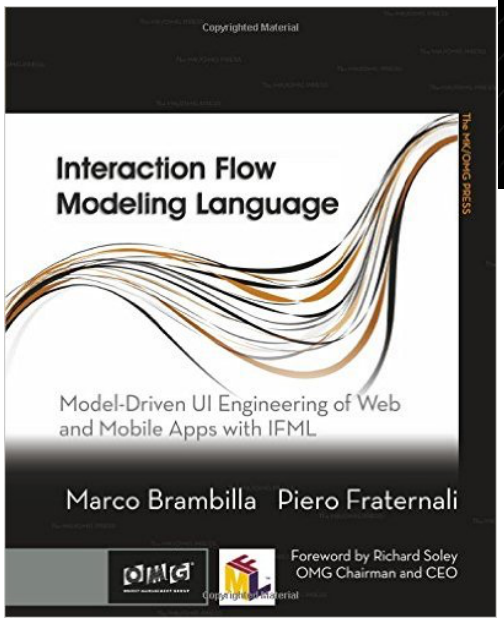
Lecture Plan (draft)

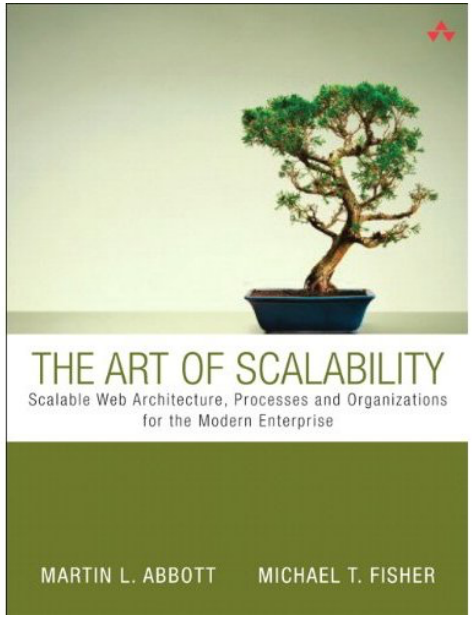
Week	Lecture		Lab
1	Introduction and overview. Software architectures for Int. Apps		-
2	Web development frameworks (Java Spring 101 + REST)		Spring101
3	Specification of internet apps (IFML)		Spring102
4	Client applications (JSP, Ajax, AngularJS, WebSockets)		+HTML/JS
5	Data abstractions (JDBC, JPA, ORM, …) - part I		+JDBC
6	Data abstractions - part II	#project1	+Hibernate
7	Process languages and Service-Oriented architectures		+WebFlow
8	Midterm		…
9	Language based tools for web applications	#project2	
10	Authentication and security models		+
11	Specification and implementation of security policies		+
12	Single page apps, reactive apps (frameworks and languages)		+
13	Performance and scalability		#projectfinal
14	Final test		

Bibliography









learn play download interact

tutorial handbook samples language spec

Bibliography

- Marco Brambilla and Piero Fraternali. *Interaction Flow Modeling Language – Model-Driven UI Engineering of Web and Mobile Apps with IFML*. Morgan Kaufmann. 2014
- Martin L. Abbott and Michael T. Fischer, *The Art of Scalability: Scalable Web Architecture, Processes and Organizations for the Modern Enterprise*, Addison-Wesley Professional. 2009
- Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley. 2002
- Craig Walls. *Spring in Action (4th edition).* Manning. 2014

https://spring.io http://www.typescriptlang.org


applications
web
service
mobile
techniques
development
Internet
services
mechanisms
orchestrate
patterns
data
modularity
clients
MVC
Hibernate
native
abstractions
methods
collaborative
online
practices
persistency
result
reactive
performance
challenges
concerns
good
Spring
REST
integration
fundamentals
Applications
compound
Components
JPA
validation
Clients
software
resources
architectural
page
pages
providers
single
specific
advanced
tools
oriented
TDD
Aspects
Web
Beans
Model
Service

“I find your lack of faith disturbing.”

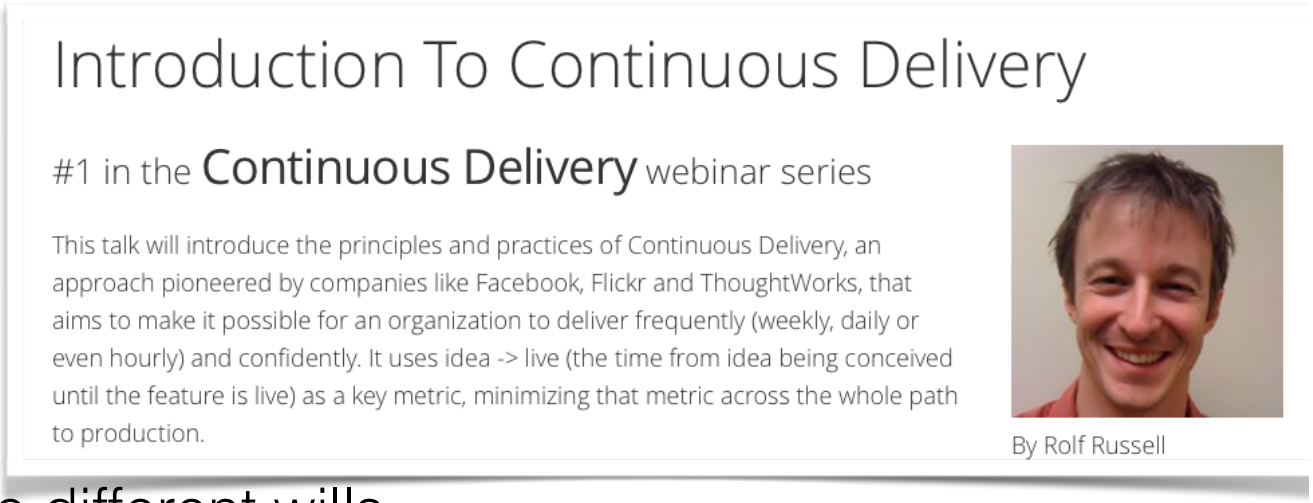
– Darth Vader

Internet Applications

- Everything is (inter)connected, developing for interconnection requires special skills and methods
- Apps can be:
  - Standalone / Desktop
  - Web, accessible through browsers
  - Mobile (web/native) (w/ offline capabilities)
  - Compound and orchestrated results
  - Mash-up interfaces
    - e.g. google maps + rentals, friends, … http://mashable.com/2009/10/08/top-mashups/

Skills for Internet Applications

- Special skills because
  - software evolves fast



  - data and code have different wills

  - software is “open” and connected (the world changes)
  - hence, it must be highly maintainable
  - scalable, secure, reliable… apps have many many users…

Internet Applications

- Everything is (inter)connected, developing for interconnection requires special skills and methods
- Apps can be:
  - Standalone / Desktop (no network, native GUIs)
    - Word
    - Powerpoint
    - …

Internet Applications

- Everything is (inter)connected, developing for interconnection requires special skills and methods
- Apps can be:
  - Standalone / Desktop
  - Web, accessible through browsers
    - facebook.com
    - amazon.com
    - netflix.com
    - clip.unl.pt
    - …

Internet Applications

- Everything is (inter)connected, developing for interconnection requires special skills and methods
- Apps can be:
  - Standalone / Desktop
  - Web, accessible through b
  - Mobile web

Internet Applications

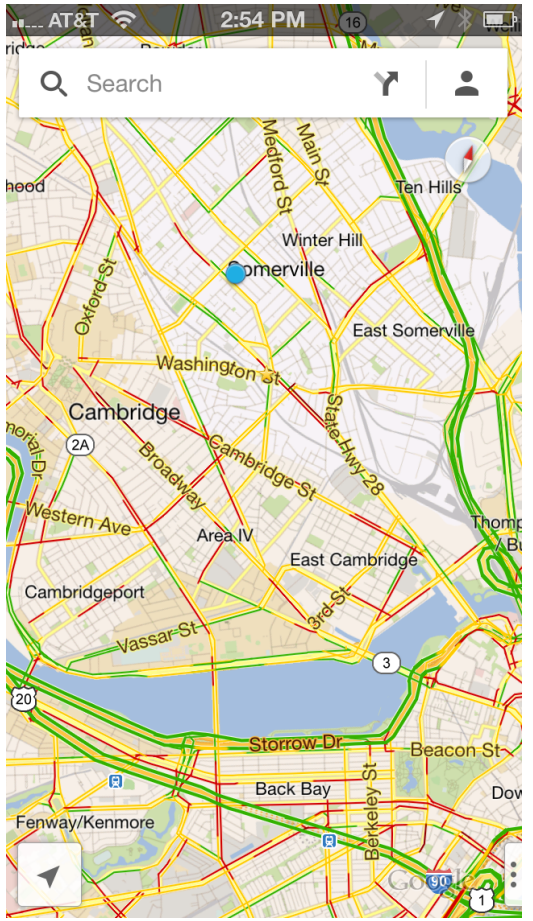
- Everything is (inter)connected, developing for interconnection requires special skills and methods
- Apps can be:
  - Standalone / Desktop
  - Web, accessible through br
  - Mobile native

Internet Applications

- Everything is (inter)connected, developing for interconnection requires special skills and methods
- Apps can be:
  - Standalone / Desktop
  - Web, accessible through browsers
  - Mobile (web/native) (w/ offline capabilities)

Internet Applications

- Everything is (inter)connected, developing for interconnection requires special skills and methods
- Apps can be:
  - Standalone / Desktop
  - Web, accessible through browsers
  - Mobile (web/native) (w/ offline capabilities)
  - Compound and orchestrated results

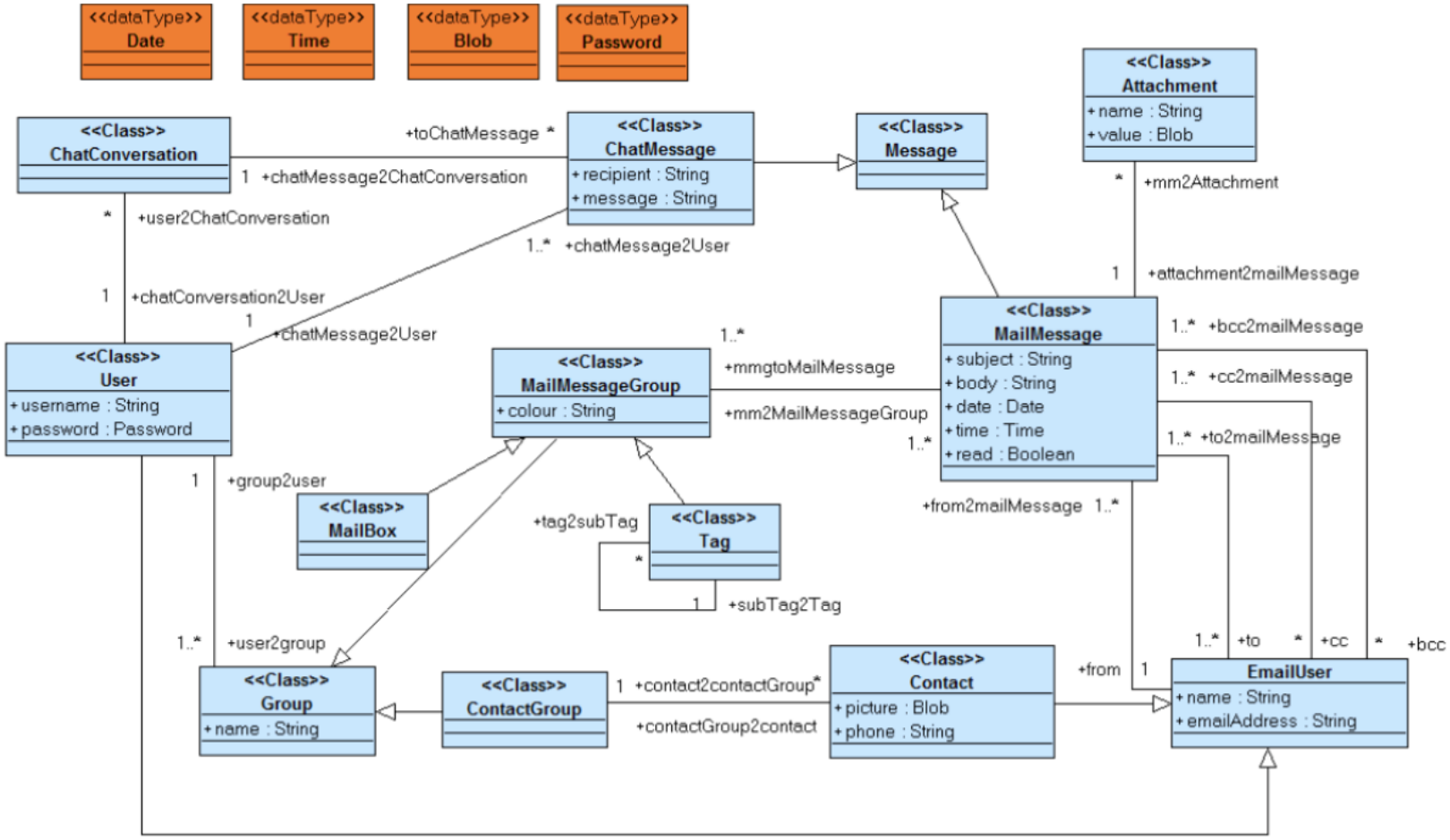


Development method

- Specification (SE tools and Tests)
- Development (Tools, languages and frameworks)
- Validation (TDD)

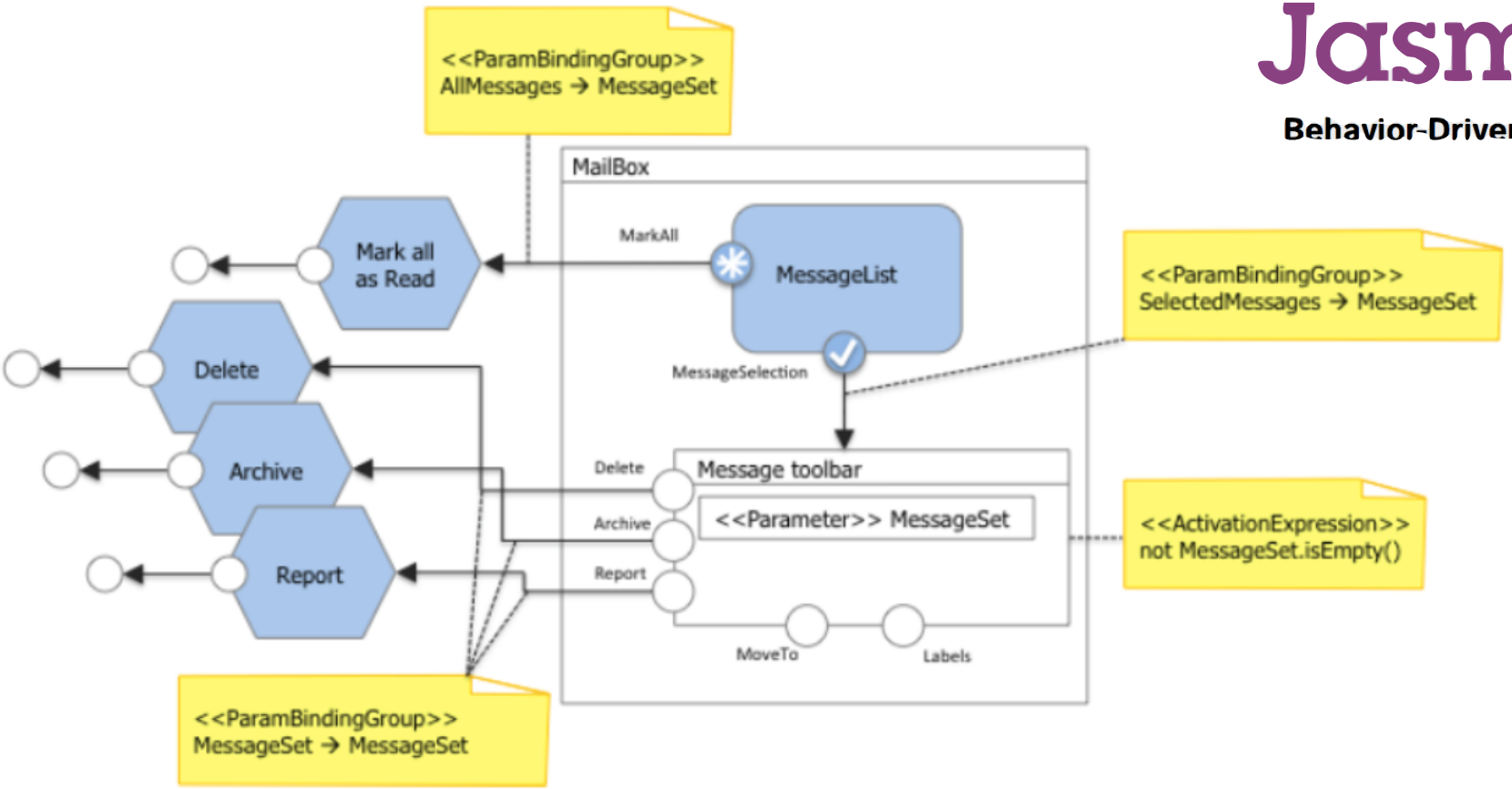
Development method - Specification

- Functional / Data oriented specification - UML
- Test Driven Development - write your tests first



Development method - Specification

- User centered specification - IFML
- Write interface flow testing first



Jasmine

Behavior-Driven JavaScript

Development method - Tools and languages

- Python (+Django)
  - Easy-to-learn programming language
  - Easy-to-use data structures
  - Available libraries

Development method - Tools and languages

- Ruby (+Rails)
- Very flexible programming language
  - Everything is “re-programmable”
- Rails is a versatile tool
  - Support for scaffolding, migrating code and data, and TDD
- Convention over configuration (even more)
- Big community
  - Big pool of top-of-the-line gems

Development method - Tools and languages

- Java + J2EE / Spring / Play
- Most used programming language
  - Well know
  - Easy to start web development
- Typed language
- Huge amount of ready-to-use libraries (Beans)
- Industrial grade efficient implementations

HIBERNATE

spring

play

Web Frameworks (examples)

- Ruby on Rails
  - Based on Ruby (dynamically typed language)
  - Implements the MVC architectural pattern
  - Pattern components assembled by conventions on folders, filenames, and language identifiers
- Java Spring
  - Based on Java (statically typed and dynamically assembled)
  - Beans framework (MVC is just a module - webmvc)
  - Patterns are assembled programatically, or by XML configuration files

Frameworks

- provide re-usable code (as libraries)
- provides controlled and managed resources
  - e.g. transactions, log, security
- introduce abstraction layers
  - to hide complexity of concrete execution scenarios (hardware/software configuration)
  - Sometimes by scaffolding code
- support test driven development and code evolution
- work by explicit configuration or supported by conventions

Frameworks

- Web Frameworks
- Data manipulation Frameworks
- Resource Oriented Frameworks (REST)
- Process Oriented Frameworks
- Client-side frameworks

Frameworks

- Examples of Web Frameworks:
  - Ruby on Rails, Django and Python, Spring and Java, Cake and PHP, nodeJS, Meteor, Revel and Go
- Examples of Client Frameworks: AngularJS, Meteor
- Examples of Languages: LINKS, UrWeb
- Examples of Data Frameworks: LINQ, Hibernate, …
- Examples of Web Service Frameworks: WS-BPEL

“May the Force be with you.”

– Yoda

Internet Applications Design and Implementation

2015 - 2016 - 1st edition

(2 - Web Development Frameworks)

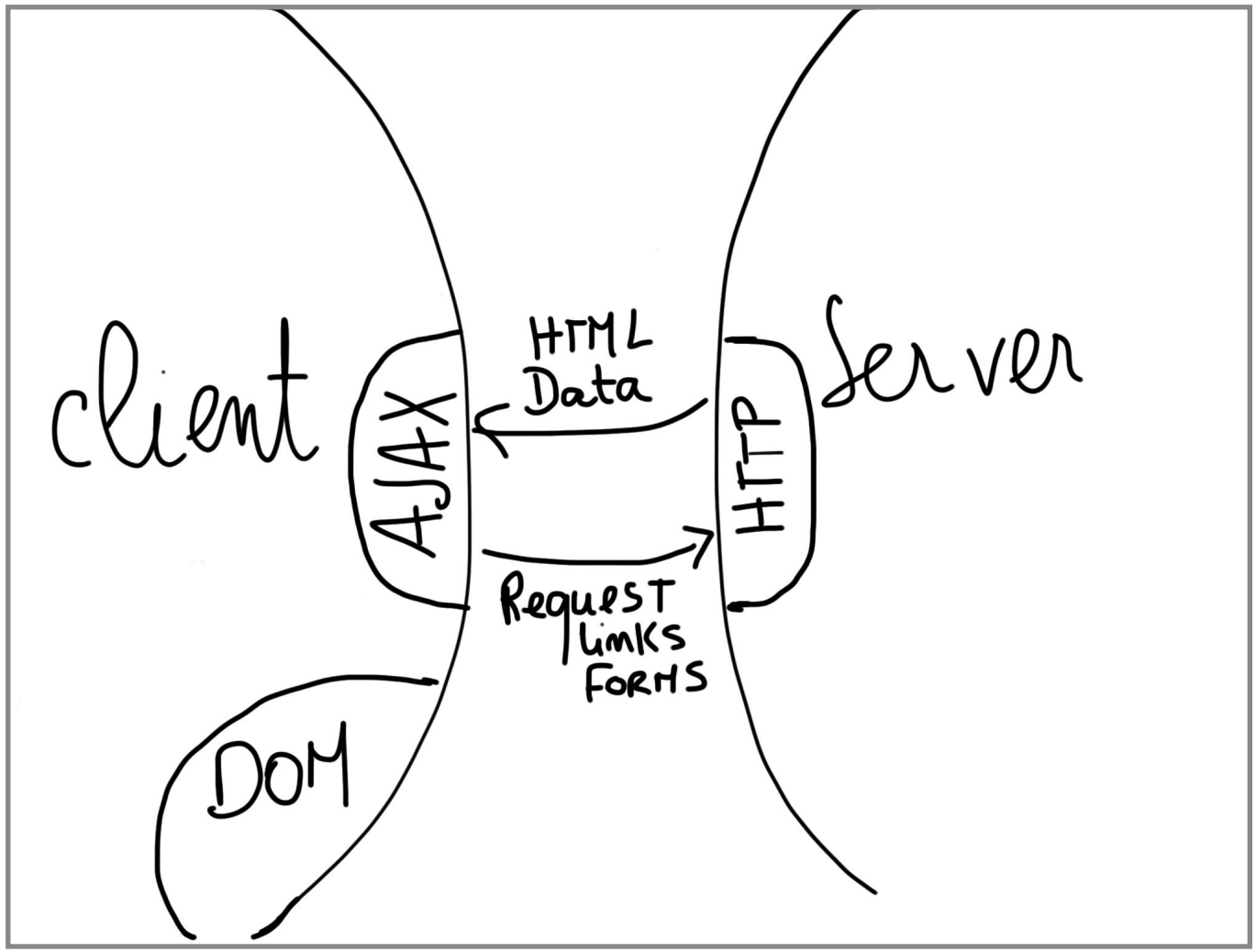
MIEI - Integrated Master in Computer Science and Informatics
Specialization block

João Costa Seco (joao.seco@fct.unl.pt)
Jácome Cunha (jacome@fct.unl.pt)

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

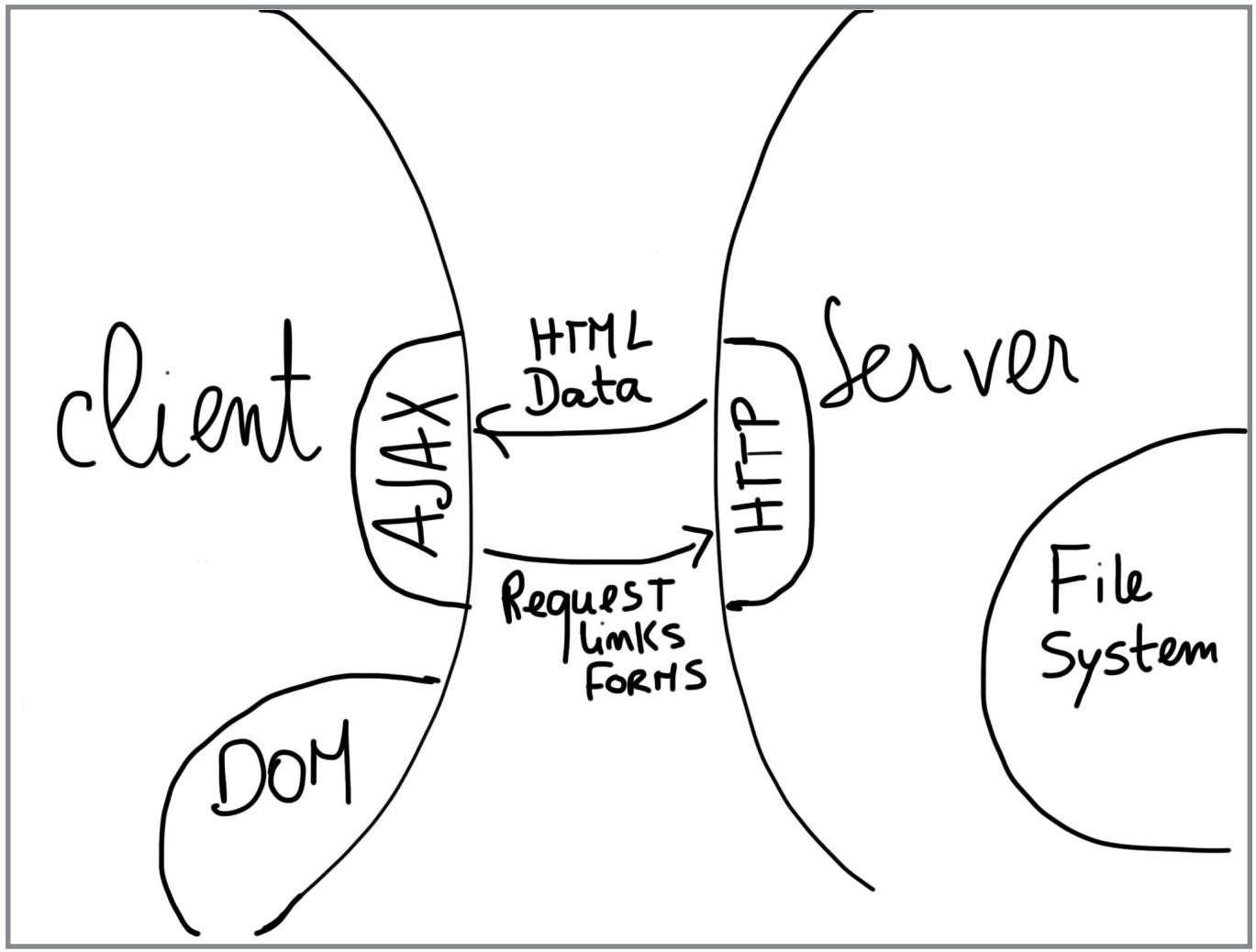
(Logical) Architecture of Web applications

- Interconnection based on the HTTP protocol
  - Method, path, arguments, (a)synchrony, multi-typed response



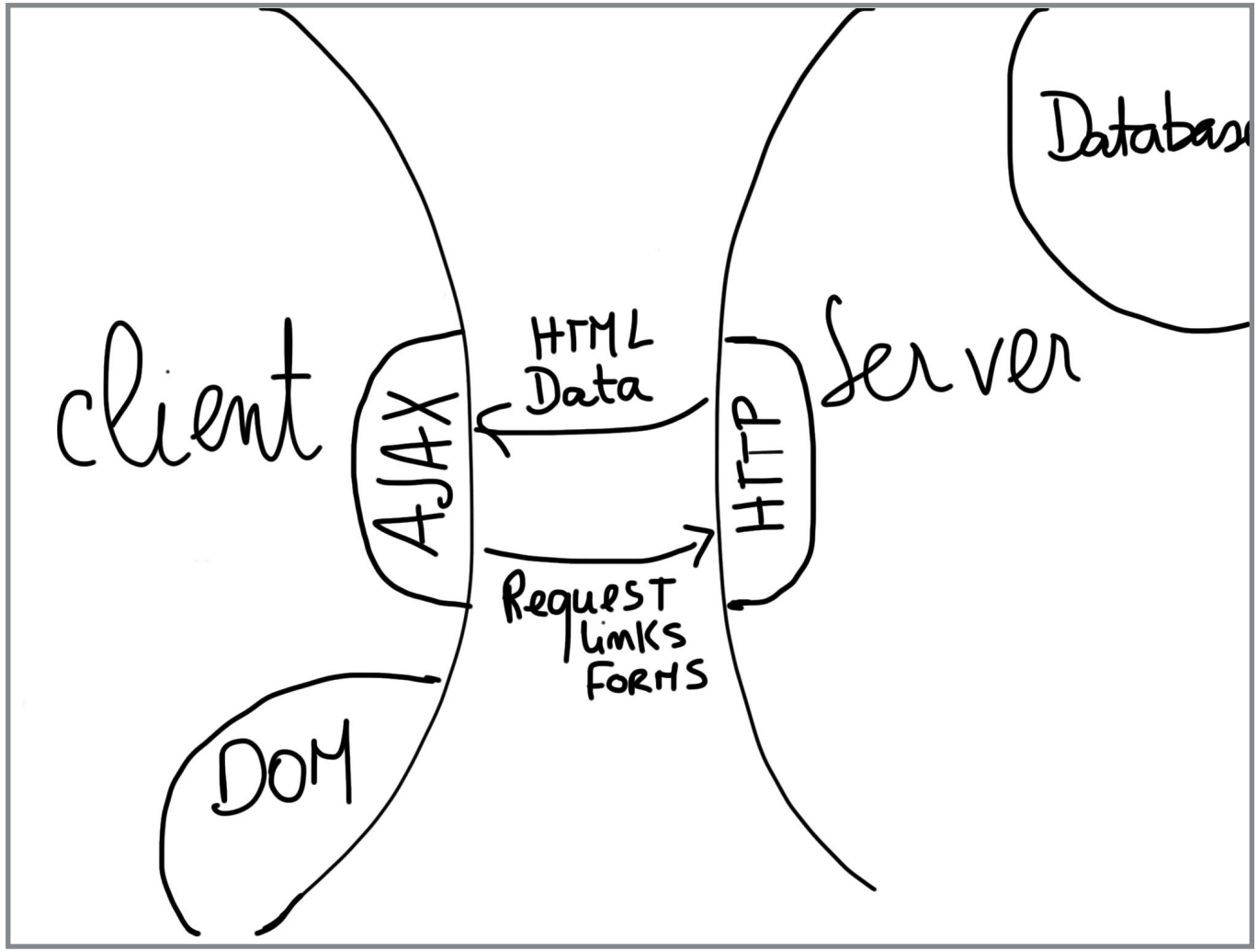
Architecture of Web applications

- Static HTML pages stored in the file system



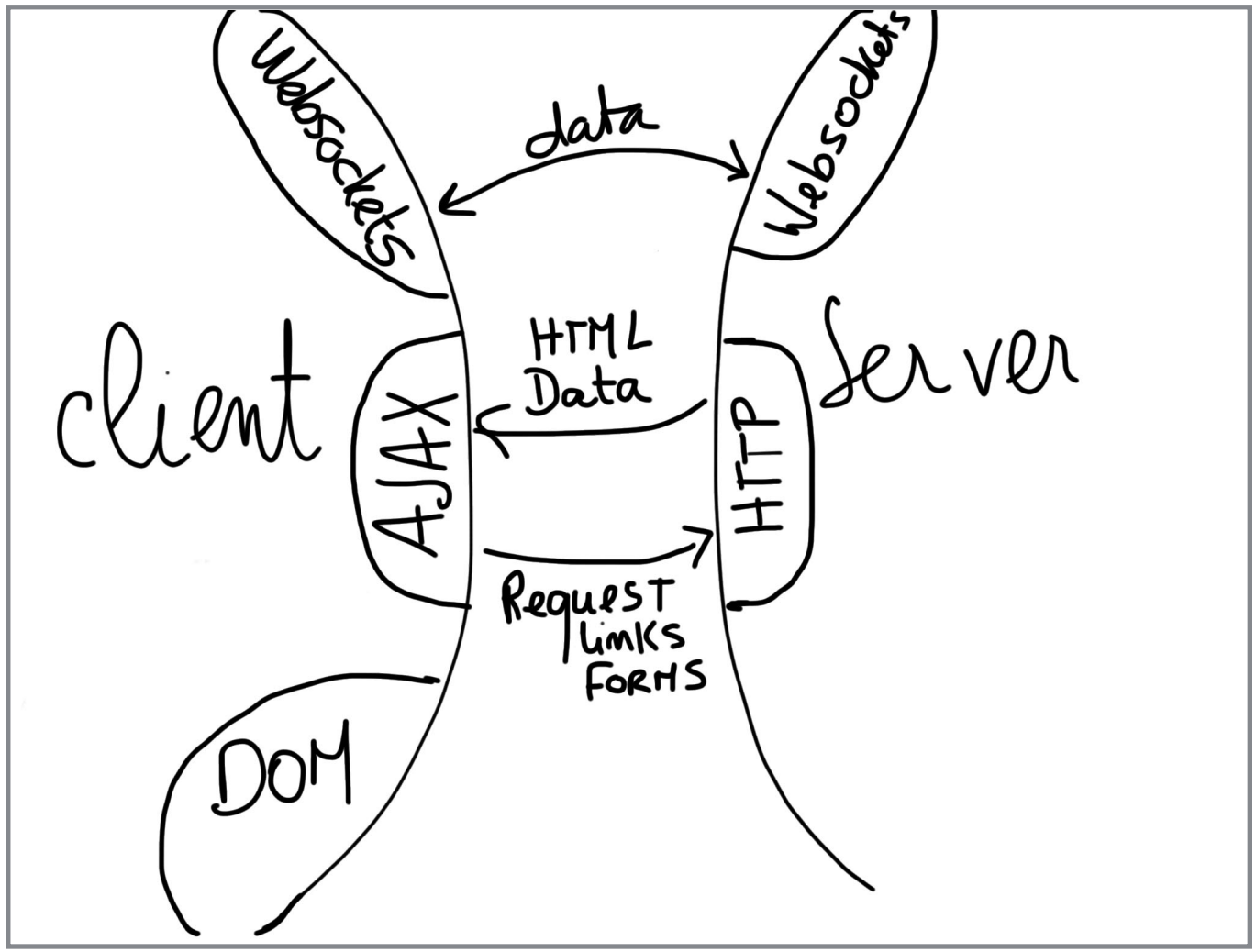
Architecture of Web applications

- Dynamic HTML pages based on data stored in some database.



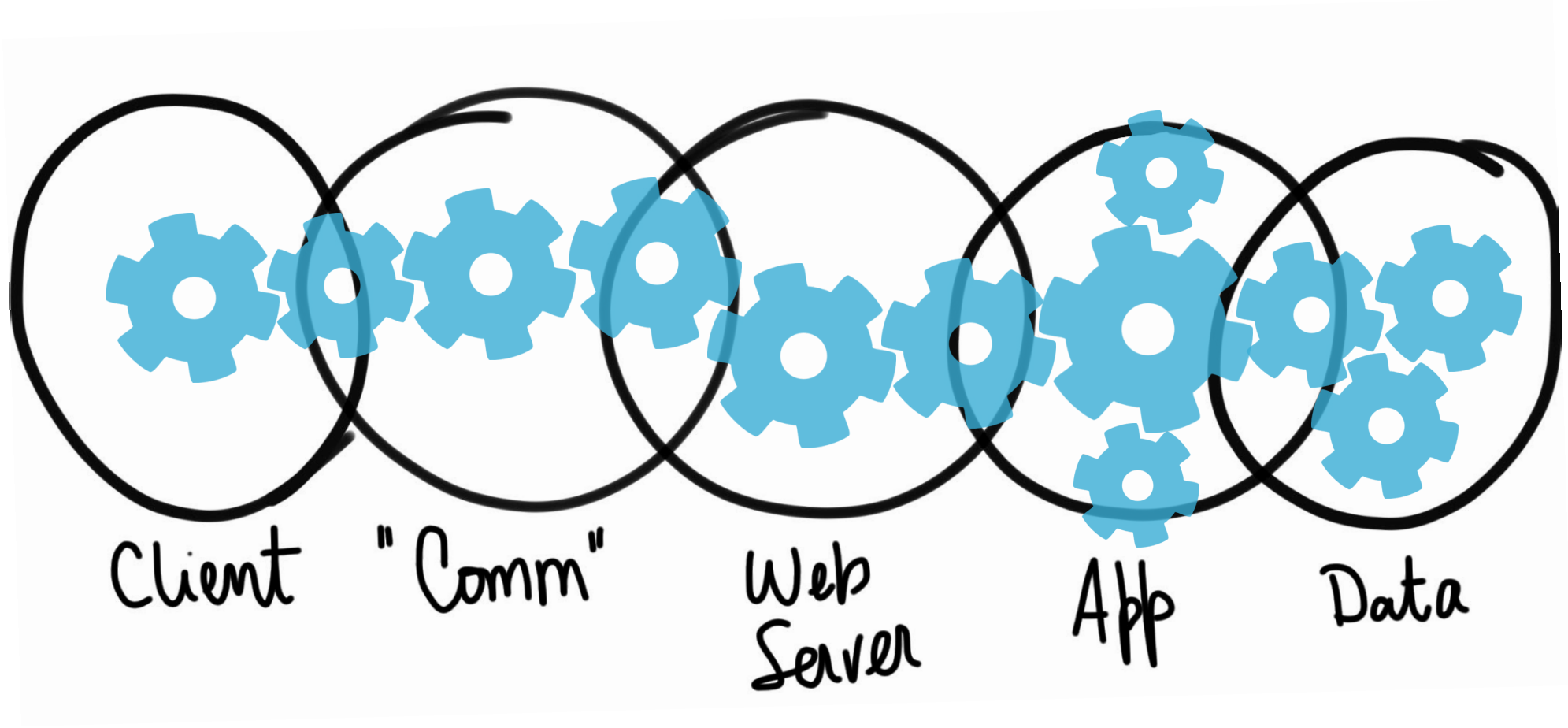
Architecture of Web applications

- Dynamic HTML pages with permanent connection with the server to exchange data.



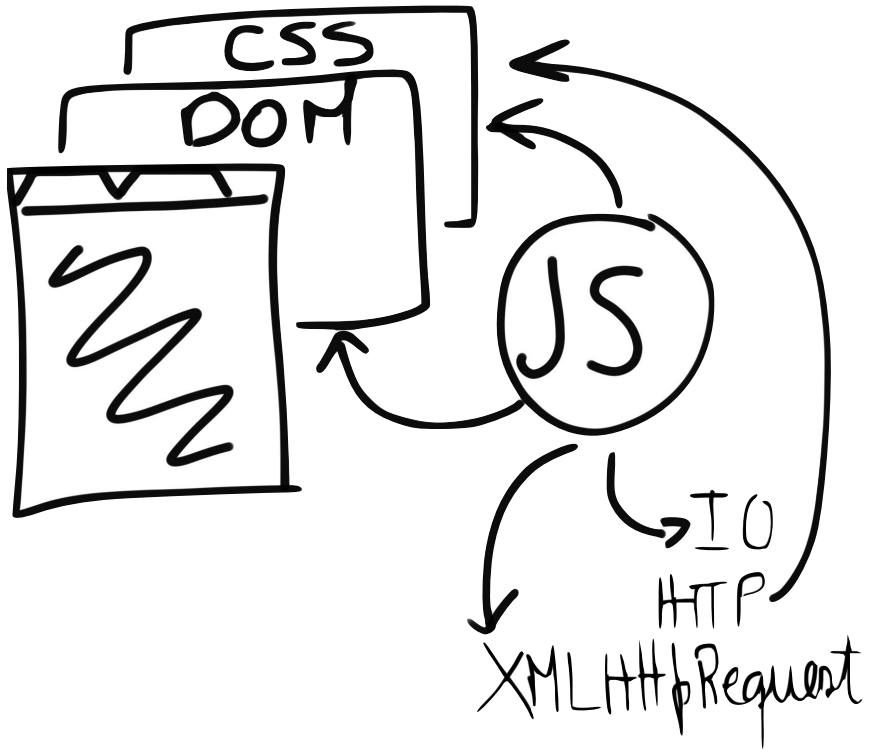
Architecture of Web applications - The Big Picture

- A web application is made of code deployed to the independent and different physical components.



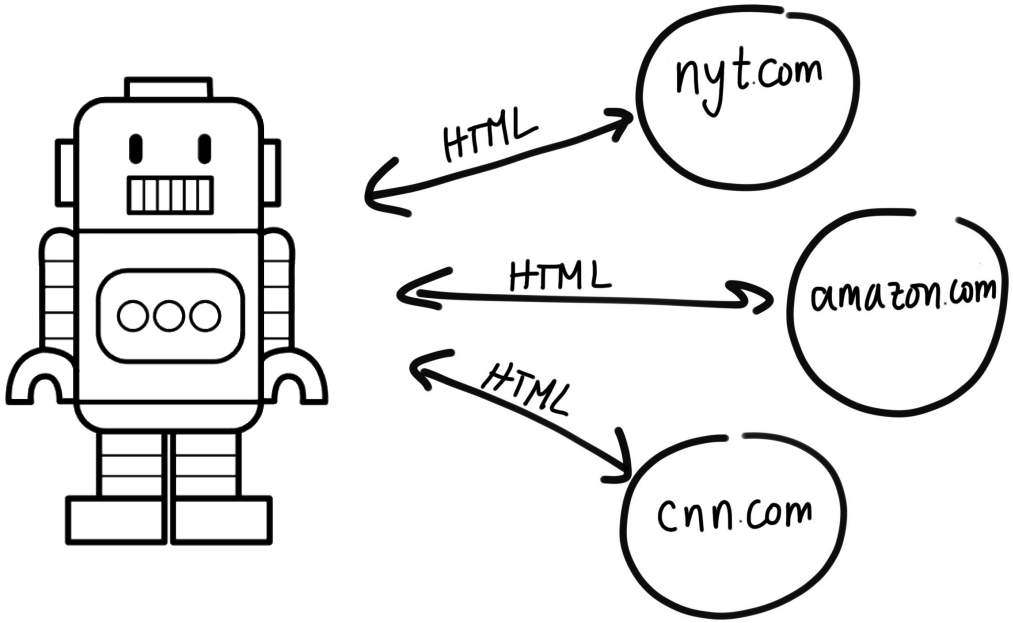
Web client architecture

- Browser (HTML5)
  - HTML (structure and semantics)
  - CSS (style and UI behaviour)
  - JS + AJAX,
    Socket interfaces (behaviour)
  - DOM
    (the supporting data structure)
  - UI Events & callbacks
    (mechanism for dynamic structuring of behaviour)



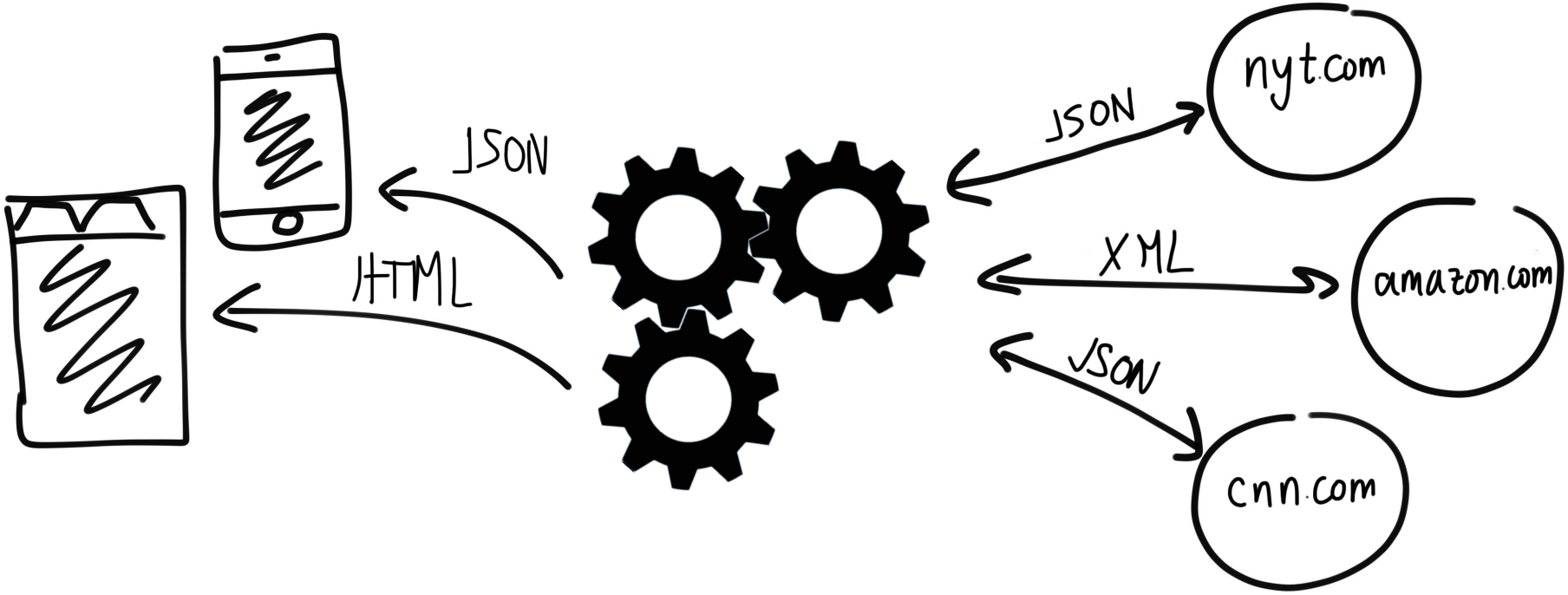
Web client architecture

- Other kind of web clients (asking for HTML)
  - simulate web requests and sessions with web-servers
  - parse/crawl the results to extract information
  - should be built with web-services instead (if possible).



Service based web client architecture

- Provide web content based on data providing services
  - Data can be exchanged in “machine-friendly” formats. (e.g. JSON, XML)
  - Combined and refurbished in HTML or data formats.
  - Even reused inside the same application, in the same server.



Interconnection layer (low-level support)

- HTTP/1.1 (*Hypertext Transfer Protocol*) Protocol
  - Method: GET, HEAD, POST, PUT, DELETE (3 more)
  - Arguments (query string and body)
  - Multi-typed message body
  - Cookies
  - Return codes: 1XX, 2XX, 3XX, 4XX, 5XX
- Websockets (standard RFC6455 2011 - ws:// wss://)
  - supported by most browsers and web servers to allow two way data communication between client and servers.
- HTTP/2 approved May 2015.
  - Faster, compressed, and cyphered transmission of data

Web server / App server architecture

- Web servers handle HTTP requests, map URLs to local files, execute local scripts (e.g. CGI, PHP), or locally bound code (e.g. Servlets).
- App Servers modularly manage bound code, and associated resources (e.g. sessions, context, connections).
  - One web server, many applications.
  - Allows the assembly of components of applications (controllers, views, models)

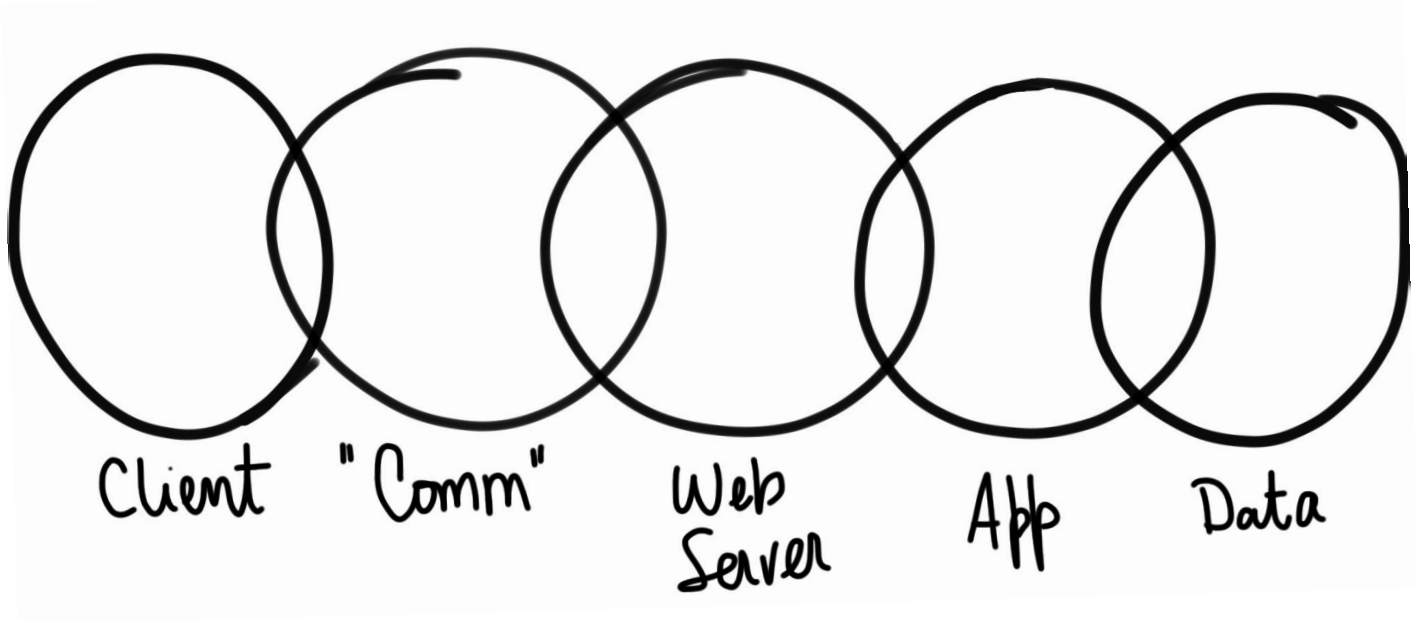
Web architectures, patterns and styles

- Increase the level of abstraction, reusability, and maintainability
  - Software Architectures
  - Architectural and design patterns
  - Architectural styles

- Software frameworks implement some, and complement with libraries and tools.
- User-defined pieces are required to specify the “core” logic, and configure general purpose code.

Web architectures, patterns and styles

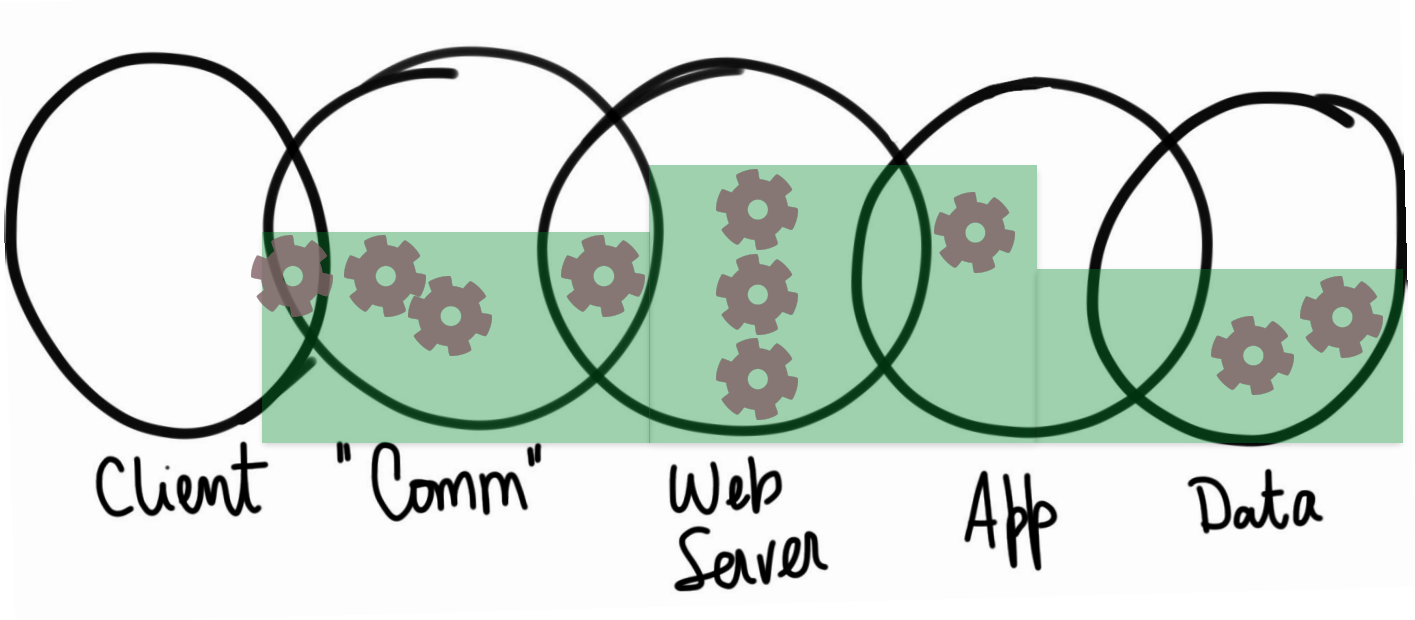
- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



Web architectures, patterns and styles

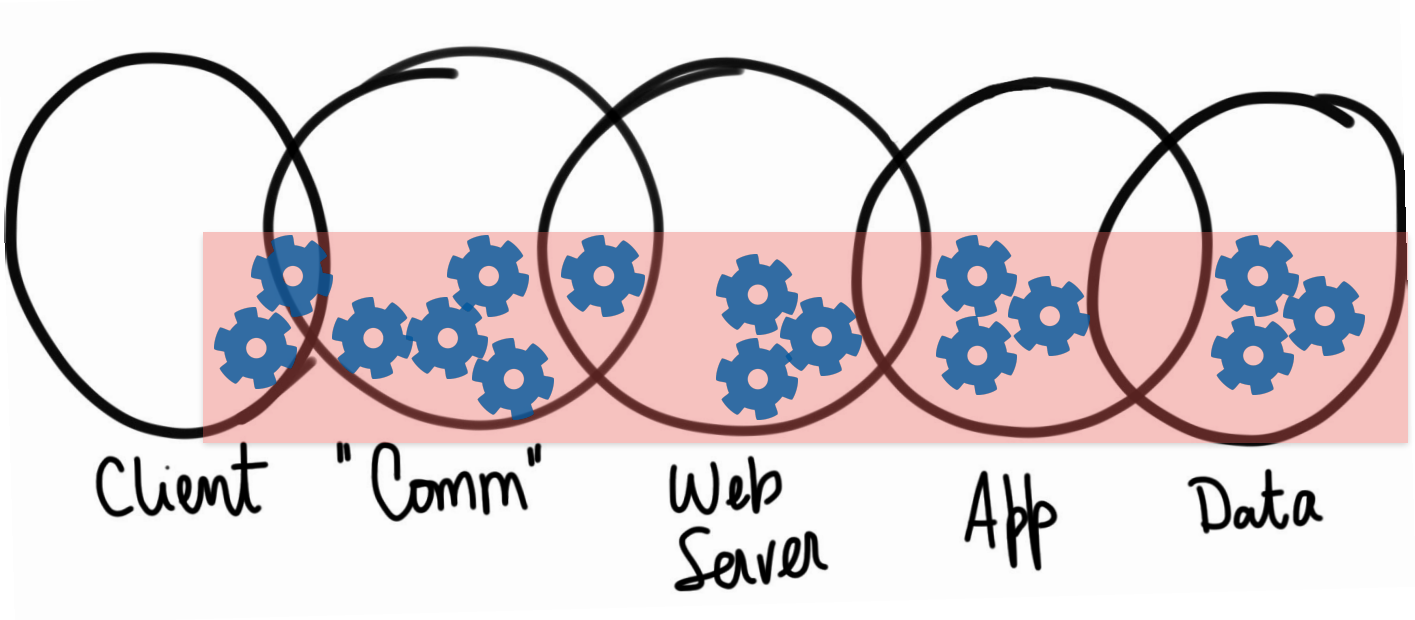
- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)

express + node + mongoDB



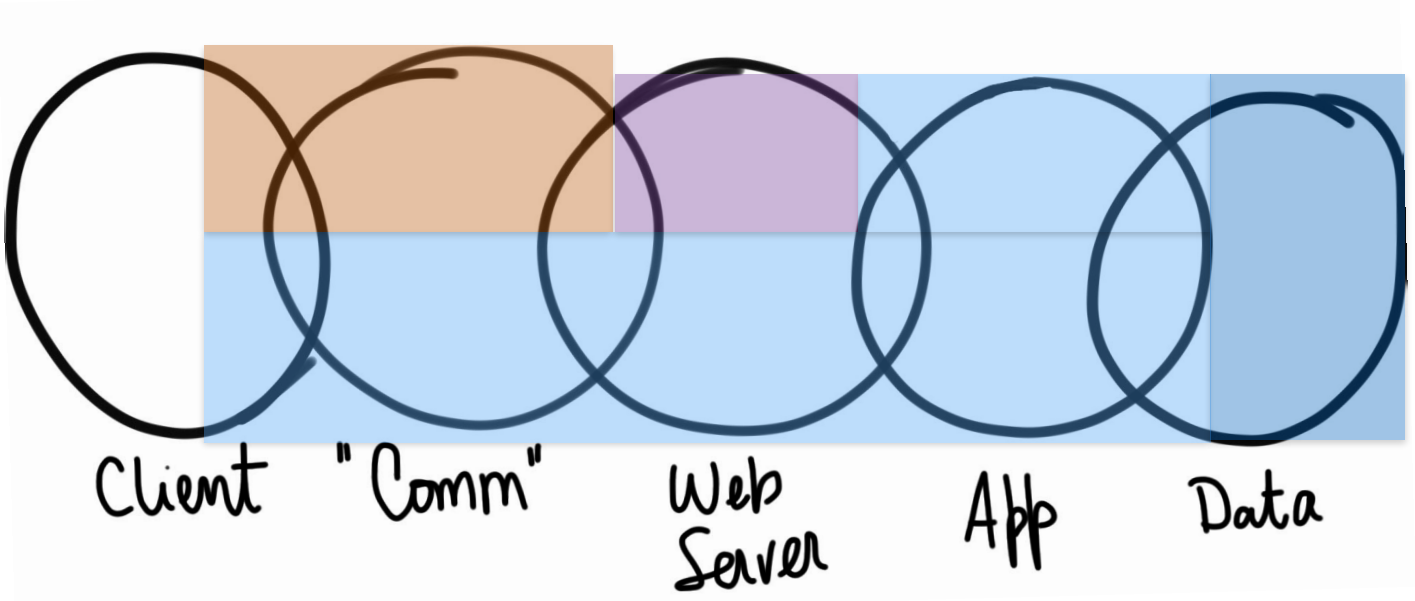
Web architectures, patterns and styles

- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



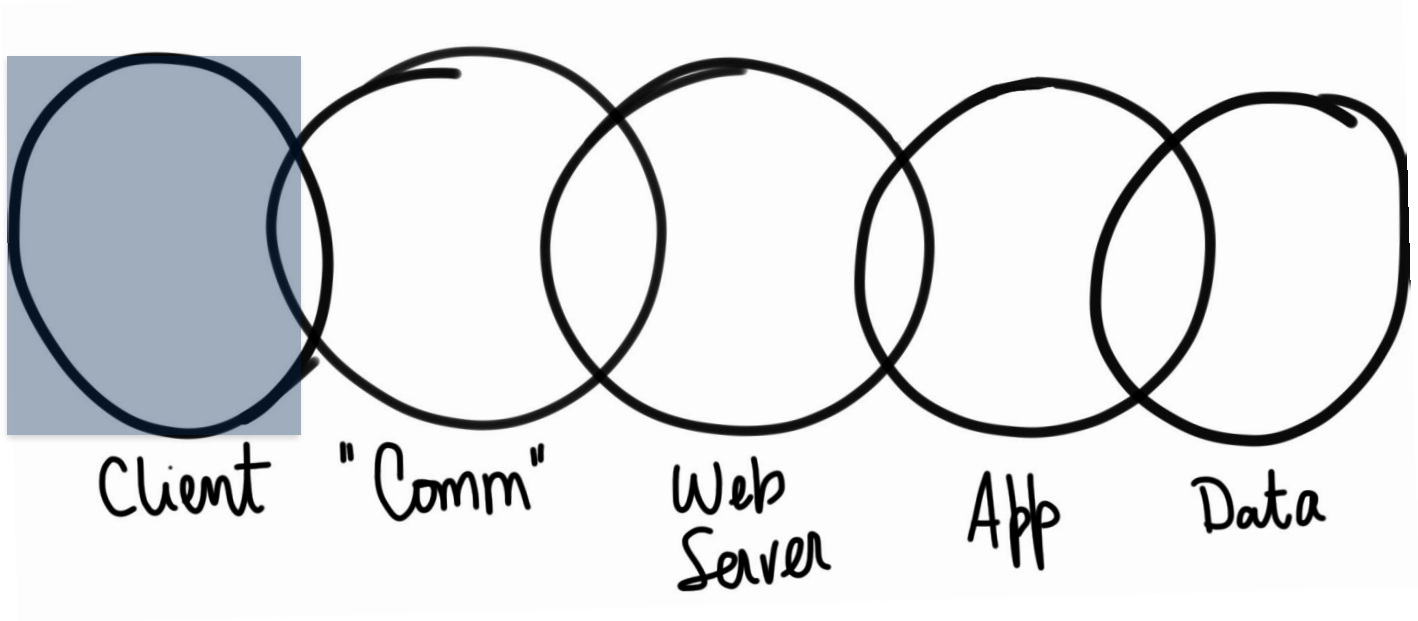
Web architectures, patterns and styles

- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



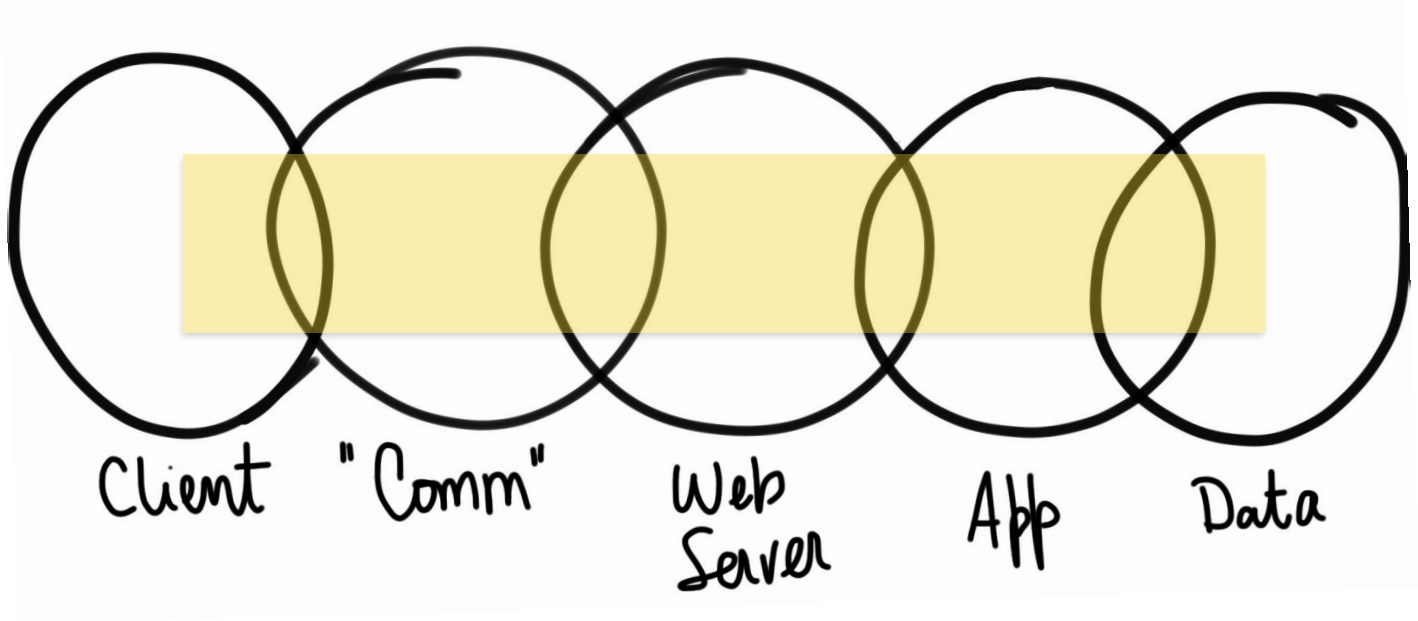
Web architectures, patterns and styles

- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



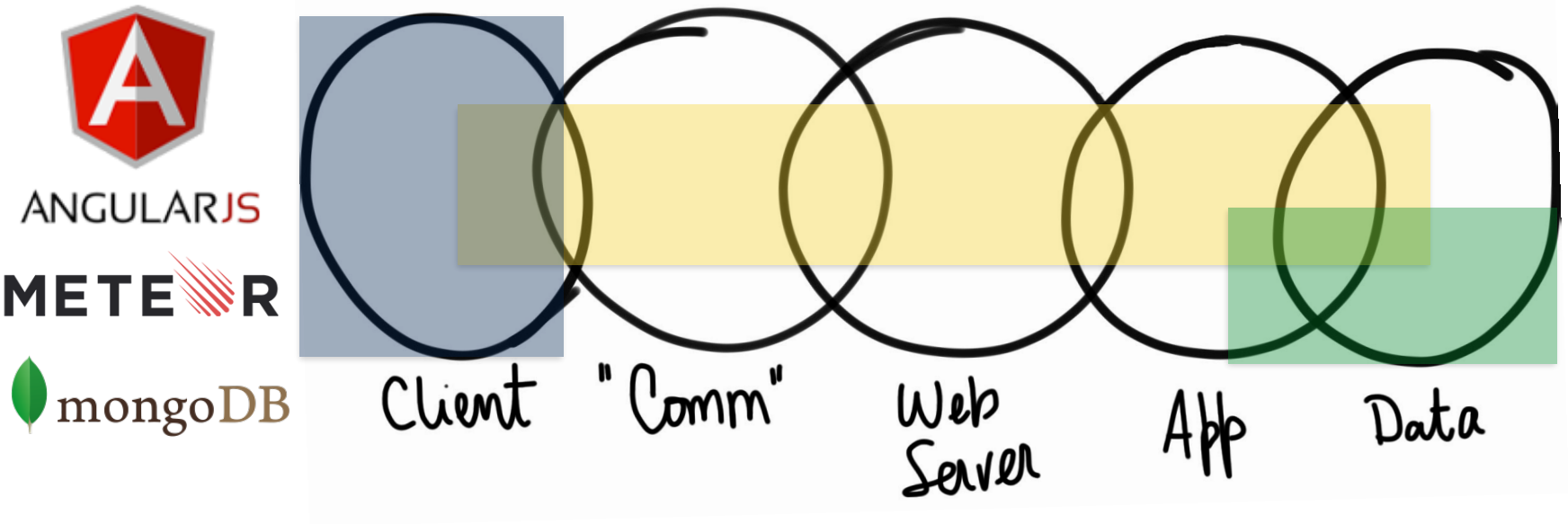
Web architectures, patterns and styles

- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



Web architectures, patterns and styles

- Most common web applications follow the MVC architectural pattern.
  - Model layer - isolate the representation of persistent data and its operations, validations and conditions
  - Controller - contains the core application logic implementing the application interface (e.g. ad-hoc URL mapping, REST convention)
  - View - defines the way in which responses are formed (e.g. HTML, JSON)



Web architectures, patterns, and styles

- Web services are defined over protocol HTTP

- **SOAP** (*Simple Object Access Protocol*)
  - Operation based protocol (on HTTP) to implement web services (XML as format)
- **REST** (*Representational State Transfer*)
  - Resource based architectural style to implement web services over HTTP (or another connection protocol)

Summary - Web Frameworks

- Web Frameworks are “languages” that carry libraries and abstractions that get compiled to run on the “web virtual machine”.