UNIVERSITY OF LEICESTER

INFORMATICS

Spring MVC: Design Principles

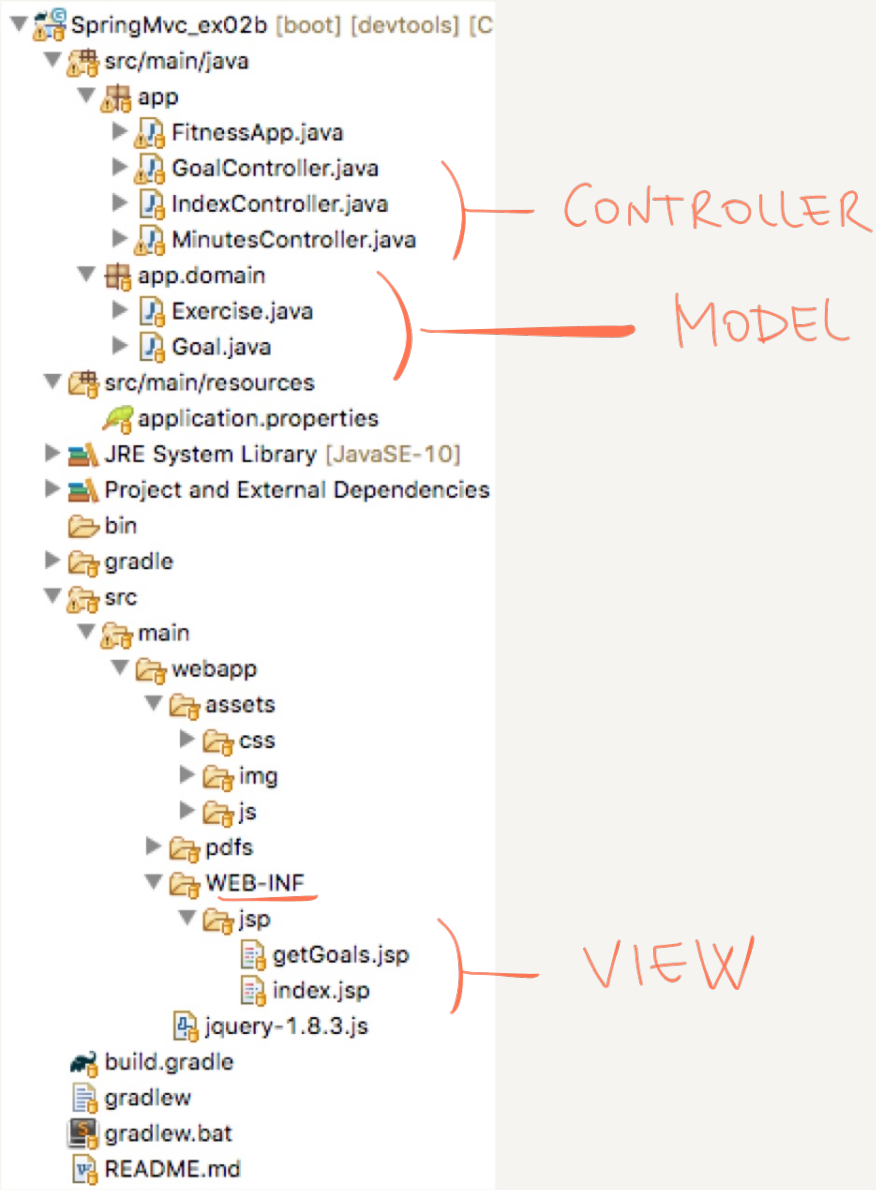
Artur Boronat

SPRING MVC ARCHITECTURE

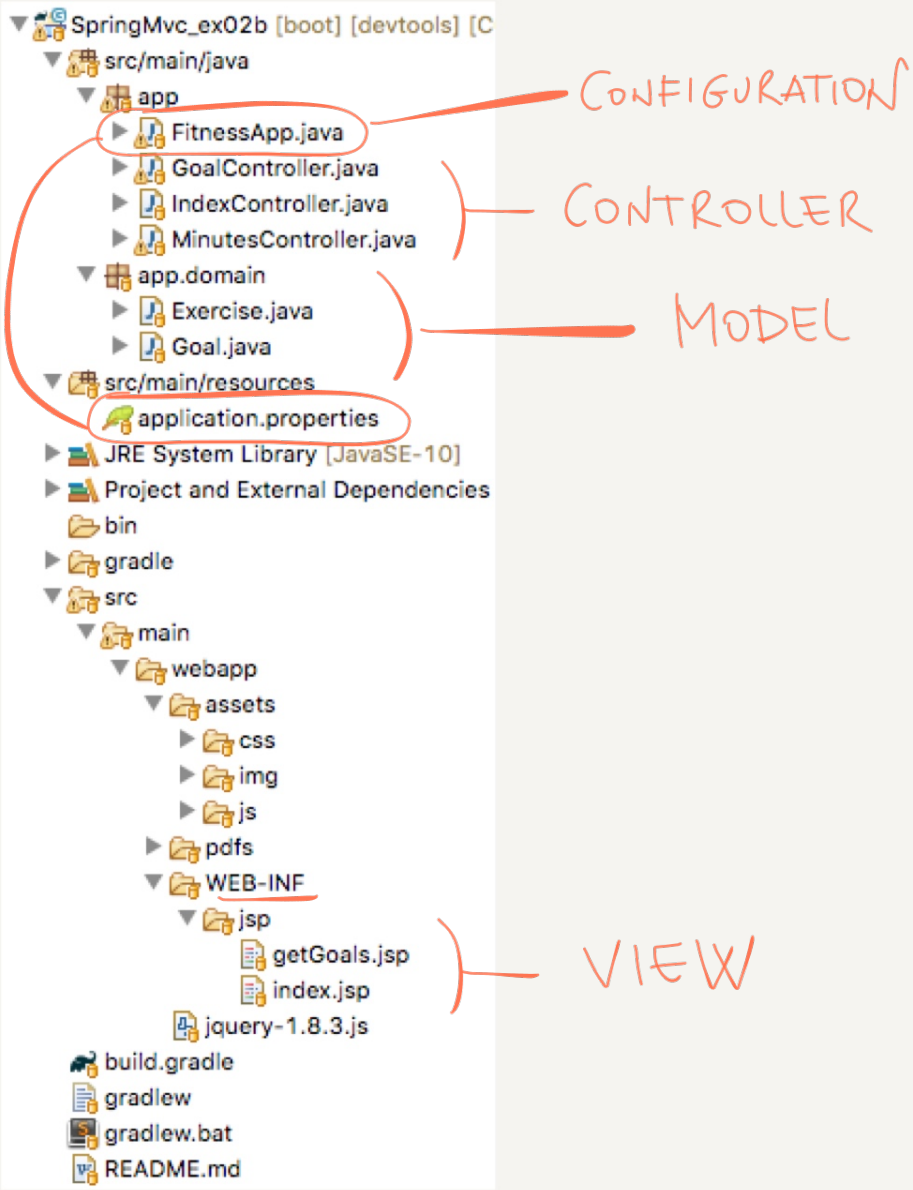
SPRING MVC ARCHITECTURE

SPRING MVC ARCHITECTURE

SPRING MVC ARCHITECTURE



SPRING MVC ARCHITECTURE



SPRING MVC ARCHITECTURE

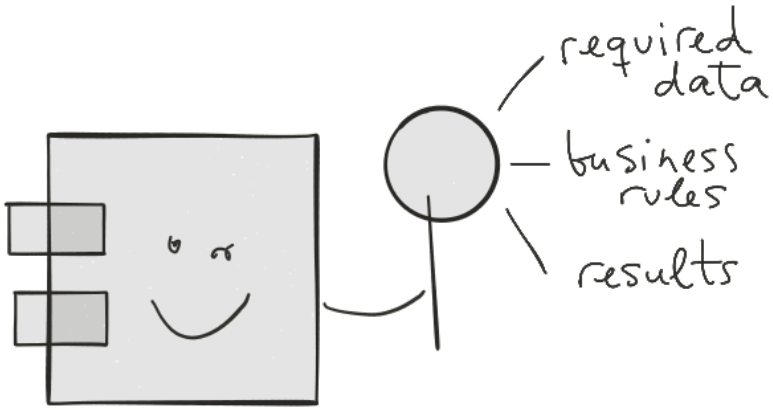
Design Principles

- Guidelines for
  - decomposing our system's required functionality and behaviour into subsystems
  - deciding what information to provide (and what to conceal) in the resulting subsystems
- Design principles
  - Modularity: coupling and cohesion
  - Encapsulation/information hiding

Design principles: Modularity

Modularity: degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components

- A design is modular when each activity of the system is performed by exactly one software unit, and when the inputs and outputs of each software unit are well-defined.
- A software unit is well-defined if its interface accurately and precisely specifies the unit's externally visible behaviour.



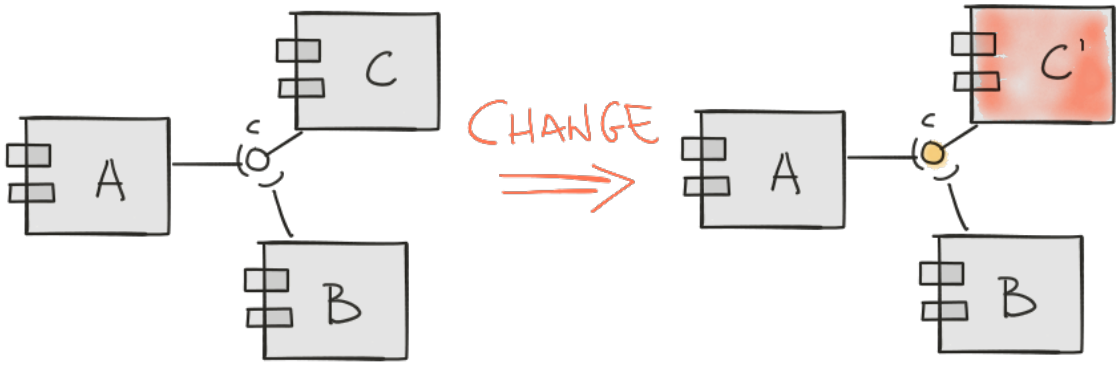
Design principles: Modularity

Separation of concerns

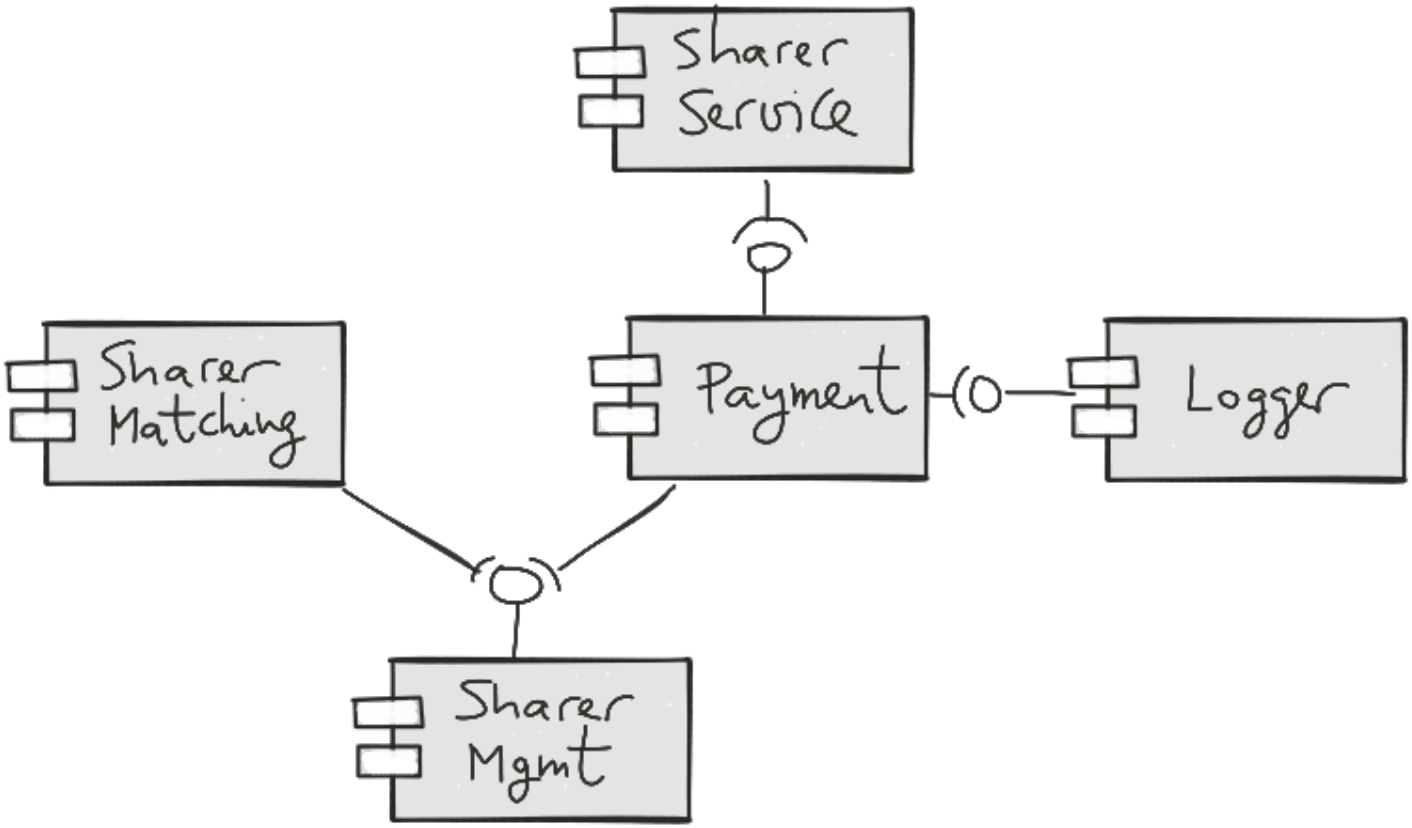
- Keep separate the various unrelated aspects of a system.
- Identify the system's unrelated concerns and encapsulate each in its own subsystem.

Subsystem independence

- Makes it easier to locate faults
- To measure subsystem independence: coupling and cohesion

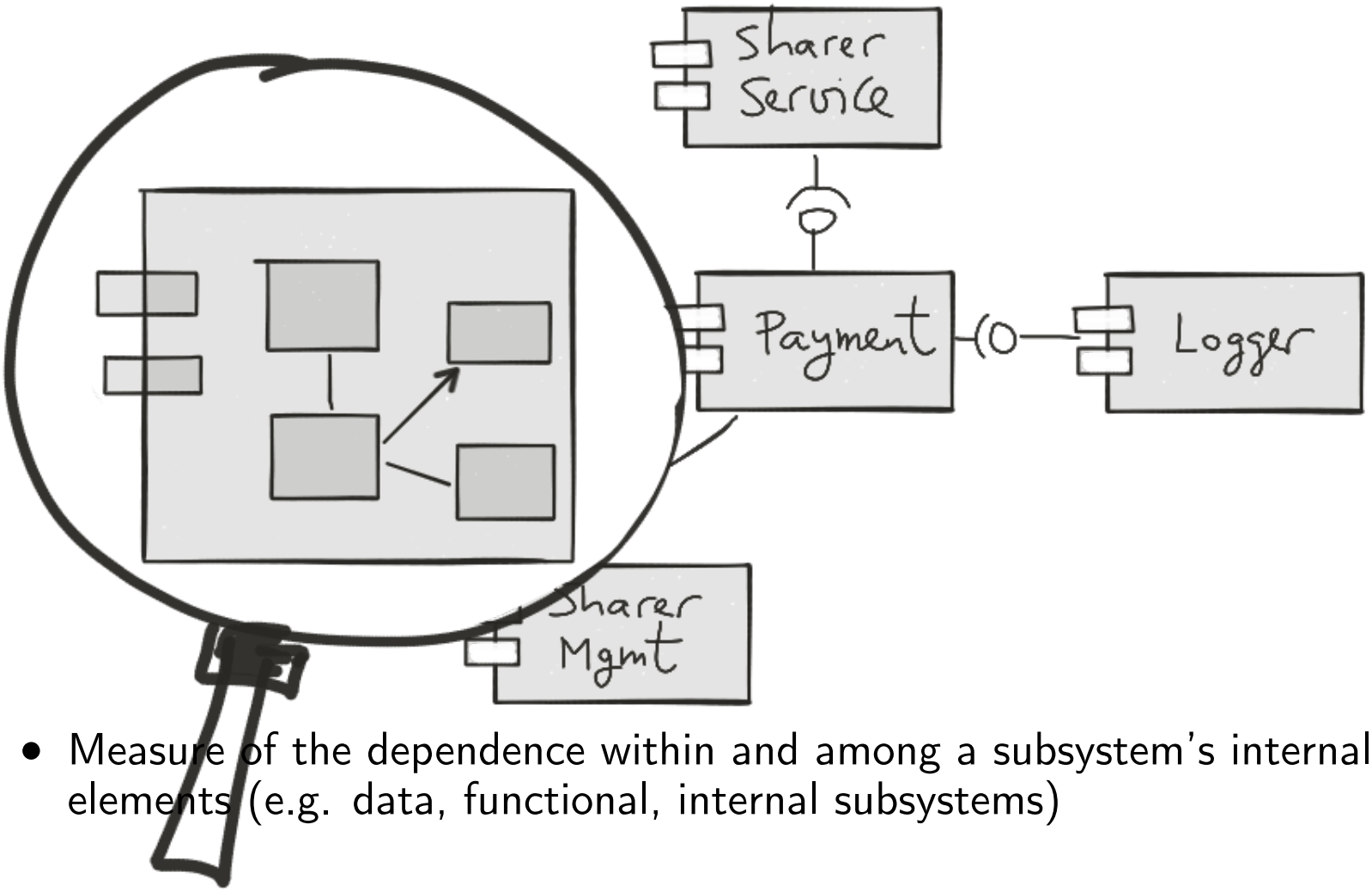


Design Principles: Modularity (Coupling)



- Measure of the dependency between subsystems
  - tight coupling: subsystems depend a great deal on each other;
  - loose coupling: subsystems have some dependence but their interconnections are weak;
  - low coupling: very few interconnections.

Design Principles: Modularity (Cohesion)



- Measure of the dependence within and among a subsystem's internal elements (e.g. data, functional, internal subsystems)

Design Principle: Encapsulation/Information Hiding

- Aim: to make the software system easier to maintain (assuming we are correctly predicting which aspects of the design will change over time) using **well-defined interfaces**
- Each software unit (e.g. a controller) encapsulates a separate design decision (e.g. a functional requirement) that could be changed in the future
- By following this principle, all the software units are
  - highly cohesive: each unit **hides** exactly one design decision
  - loosely coupled: the **interface** to each unit lists the set of access functions that the unit offers

SPRING MVC ARCHITECTURE

CONFIGURATION

CONTROLLER

MODEL

M | V | C

VIEW

SUBSYTEM DECOMPOSITION

SPRING MVC ARCHITECTURE

CONFIGURATION

CONTROLLER

MODEL

VIEW

M | V | C

HORIZONTAL PROTOTYPE