

Web Applications 2016

Intro

Content

- General Programming principles
- Domain driven design
- The Shop case/model
- Development environment
- NetBeans

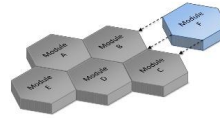
General Principles

- Inappropriate naming
- Comments
- Dead code
- Duplicated code
- Primitive obsession
- Large class
- God class
- Lazy class
- Middle man
- Data clumps
- Data class
- Long method
- Long parameter list
- Switch statements
- Speculative generality
- Oddball solution
- Feature envy
- Refused bequest
- Black sheep
- Contrived complexity
- Divergent change
- Shotgun Surgery

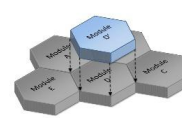
✂telerik Fundamental Principles of OOP

- Inheritance
 - Inherit members from parent class
- Abstraction
 - Define and execute abstract actions
- Encapsulation
 - Hide the internals of a class
- Polymorphism
 - Access a class through its parent interface

Extensibility



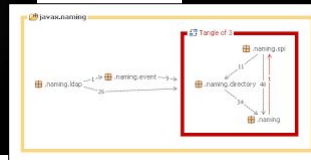
Substitutability



A module is a self-contained unit of code that has a well-defined interface.

SOLID

- Single Responsibility Principle
- Open/Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle



Course is a perfect environment for exercising your SE skills!

- Have fun!

Domain Driven Design



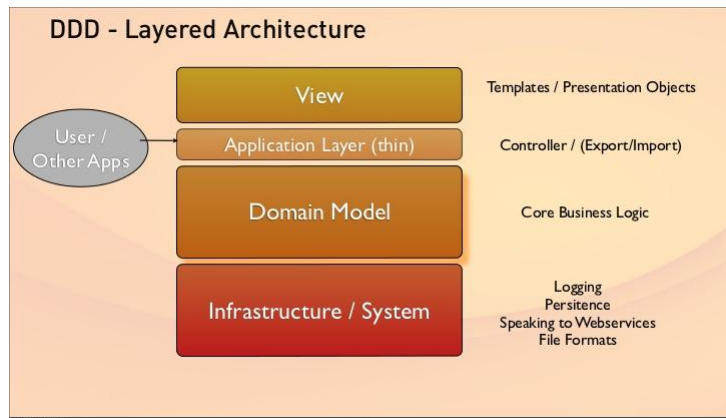
Domain driven design

- The domain model is what matters!
 - Can't "program away" the problem
- [The book](#)

How we surround the model is secondary, many options ...

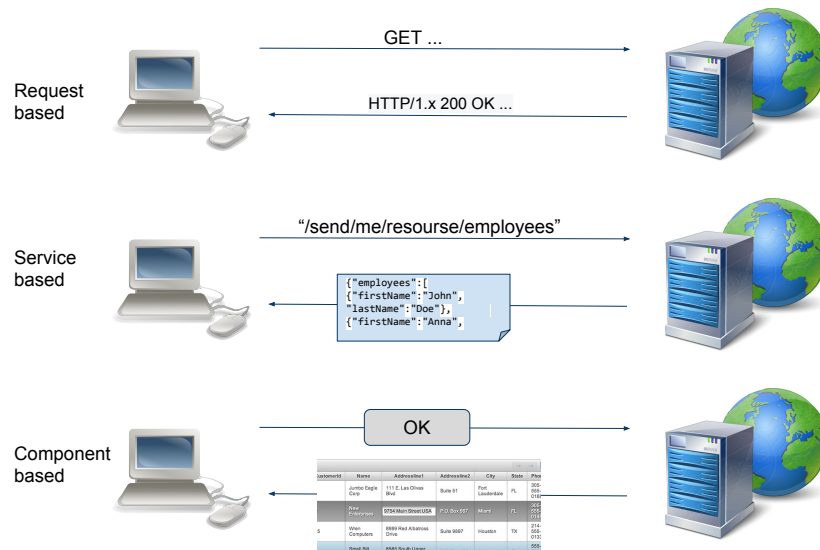
- Many types of services, networking, database, ...
- Many application contexts
 - Standalone
 - Web
 - Mobile (App)

Layered View



Domain model is the core OO-model of the problem

Web Application Approaches



Requests based

- Original approach, close to HTTP, low abstraction level, many programmers like, full control

Service based (REST)

- Application composed of services, more client side processing, latest approach

Component based

- Very high abstraction level (abstract away network, JavaScript, CSS, ...). Emulate desktop programming

The Shop Case



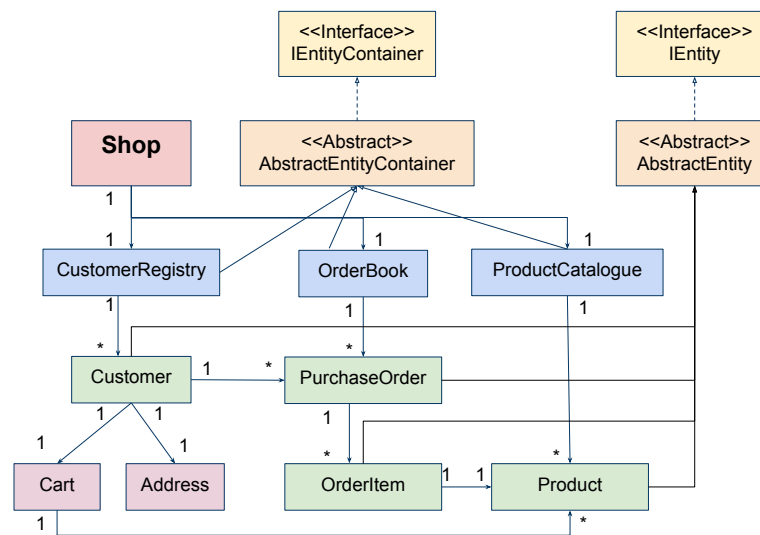
The workshops will focus on the web techniques

- I.e. no real business flow ...
- ... you practise later in your project
- In project have to use imagination for business processes (we're no system analysts)

To simplify the business parts of the workshops there is a simple OO-model of a generic shop

- Used in all workshops

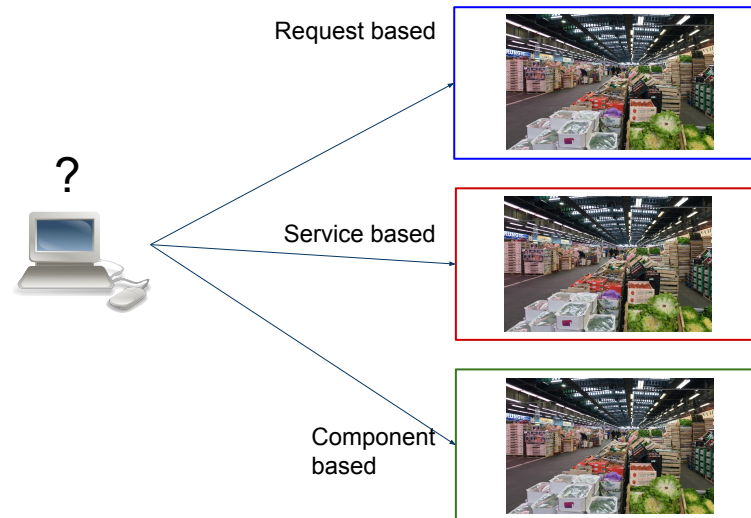
The Shop Model



NOTE: Shop model (and other code) uses the [Lombok library](#) to reduce the need to code setters/getters>equals/hashCode

- Some interfaces are missing

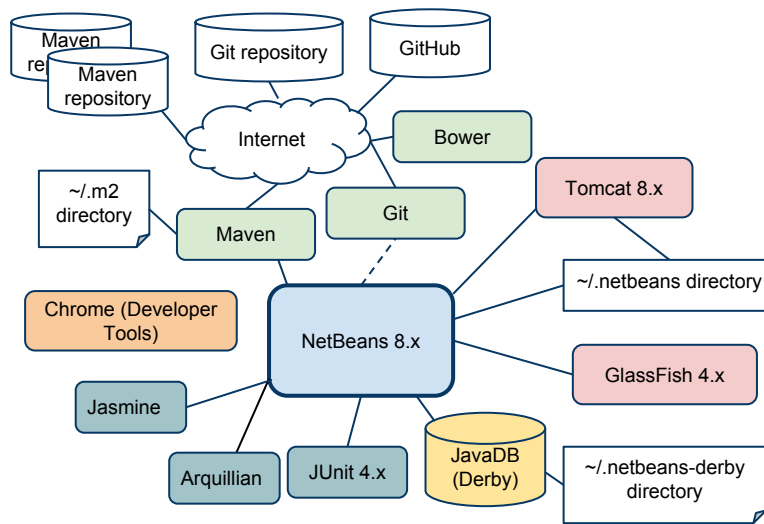
Webify the Shop Model



The web-parts are just a wrapping of the model!

- Clients sees very little difference
- Functionality same

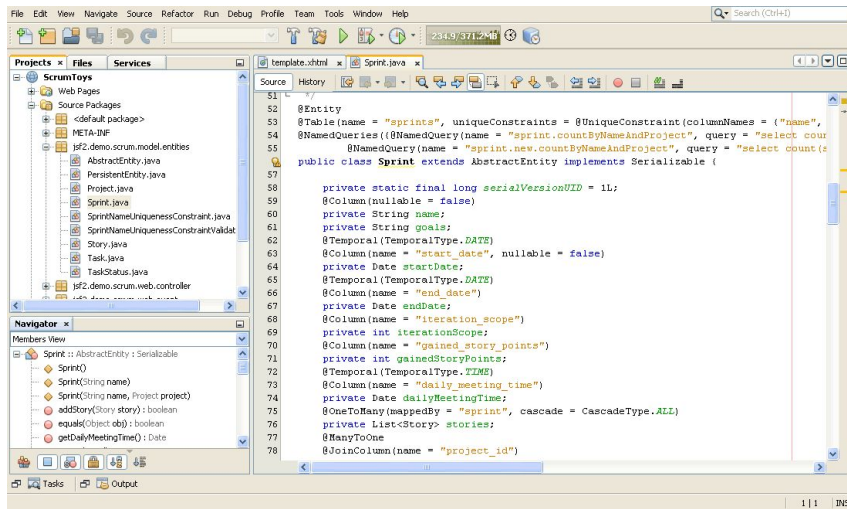
Development Environment



We need quite a few tools

- Mostly basic usage
- Some should be (well)known: Maven, Git
- Client side main tool : Chrome [Developer tools](#) (or similar, FireBug etc.)
- Client side package handling: Bower, ... more to come
- Testing: JUnit, Arquillian, Jasmine.
- Servers: Tomcat and GlassFish

NetBeans



Nice IDE targeted at Java EE, more to come ...

If problems

- Clean and build!!!
- Netbeans caches hard ... Possibly: Remove manually the target-directory (Files tab)
- Clear cache: .cache/netbeans (on my system)
- Fix clean installation. Remove .netbeans and .netbeans-derby