Practice 1. Spring Framework

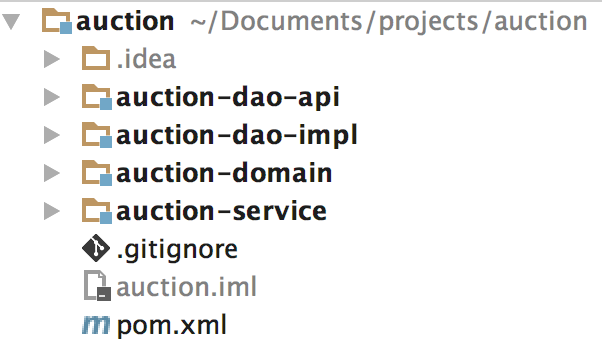
Repository with source code: <https://github.com/gsvlad/skillsupAuction.git>

Branches with step-by-step implementation:

1. **start** - this is a good place to start;
2. **maven-structure** - maven modules skeleton;
3. **domain** - domain classes and main API;
4. **service** - implementation of service and DAO;
5. **main-class** - module with main class that can run application;
6. **war** - added module with Spring MVC and module for war package;
7. **boot-xml** - added module with Spring Boot;
8. **boot-no-xml** - switching to no-configuration (no xml files);

# Create maven structure

* **auction** - parent pom for all other modules;
* **auction-domain** - module with domain classes;
* **auction-service** - module with business logic of the application;
* **auction-dao-api** - API of interaction with persistence layer;
* **auction-dao-impl** - one concrete implementation of that API. If we want we could create different modules for working with SQL DB, file system, NoSQL DB;



# 2. Create domain objects

In auction-domain module create these classes:

## Item

* title: String
* description: String

## User

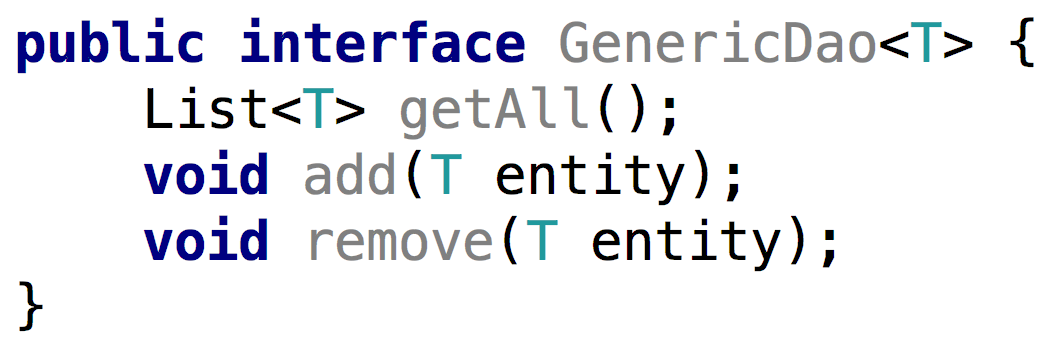
* login: String
* firstName: String
* lastName: String

## Lot

* item: Item
* owner: User
* startPrice: BigDecimal
* datePlaced: Date
* dateEnd: Date
* buyer: User
* currentPrice: BigDecimal

# 3. Create DAO API

Usually all DAO have same methods, so let’s create common Interface for them:



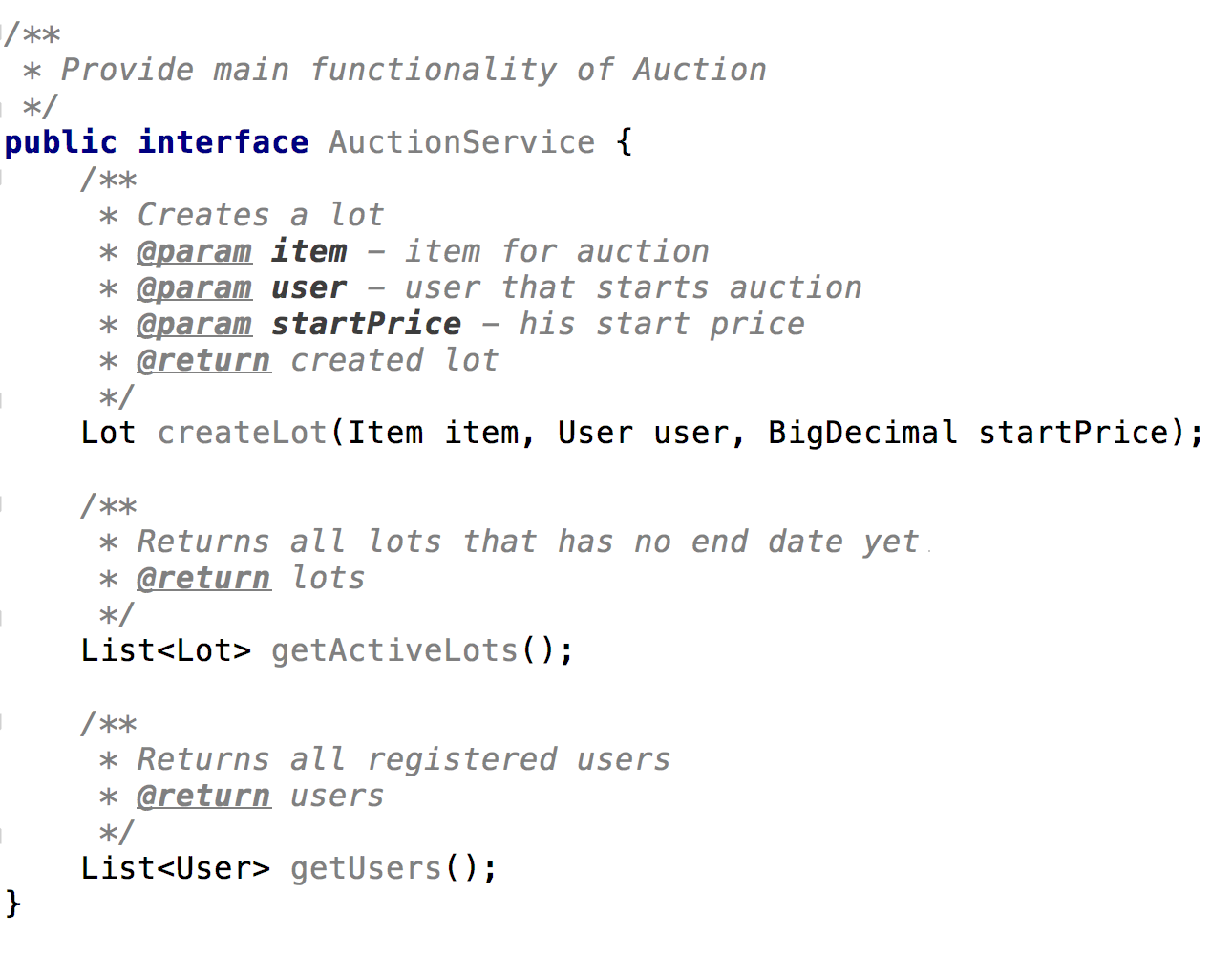
Put it to auction-dao-api module.

Also create more concrete interfaces that extend GenericDao:

* UserDao
* ItemDao
* LotDao

# 4. Create API for service

Let’s start with something that is very simple:



Add this interface to **auction-service**.

Implement this service (**AuctionServiceImpl**) in the same module. Use created DAO interfaces to get and save domain objects.

On what modules **auction-service** should have dependencies?

# 5. Create DAO implementation

Implement interfaces UserDao, ItemDao, LotDao in auction-dao-impl module.

As we don’t use DB yet - use just simple lists to store domain objects.

Add UserDaoImpl::generateSomeUsers() method where you will add few users to the list. This method should be called after bean initialization.

# 6. Add Spring xml configurations

Add **dao-beans.xml** with ItemDaoImpl, UserDaoImpl, LotDaoImpl beans to **auction-dao-impl**.

Add **service-beans.xml** with AuctionServiceImpl to **auction-service**.

Import **dao-beans.xml** in it. Should we have a dependency on auction-dao-impl to do that?

# 7. Add module with Main class

In order to run up Spring context and to run application create **auction-main** module with **Main** class.

User **ClassPathXmlApplicationContext** class to create context from **service-beans.xml**

Try to get **AuctionService** from context and call it’s methods.

# 

# 8. Replace xml configuration with annotations

Wherever it is possible remove xml files and put annotation @Component and @Service on beans.

In **auction-main** module create xml or java configuration and do **component-scan**.