**1. Настройка разворачивания приложения на томкате**

***1.1. Через плагин tomcat7-maven-plugin***

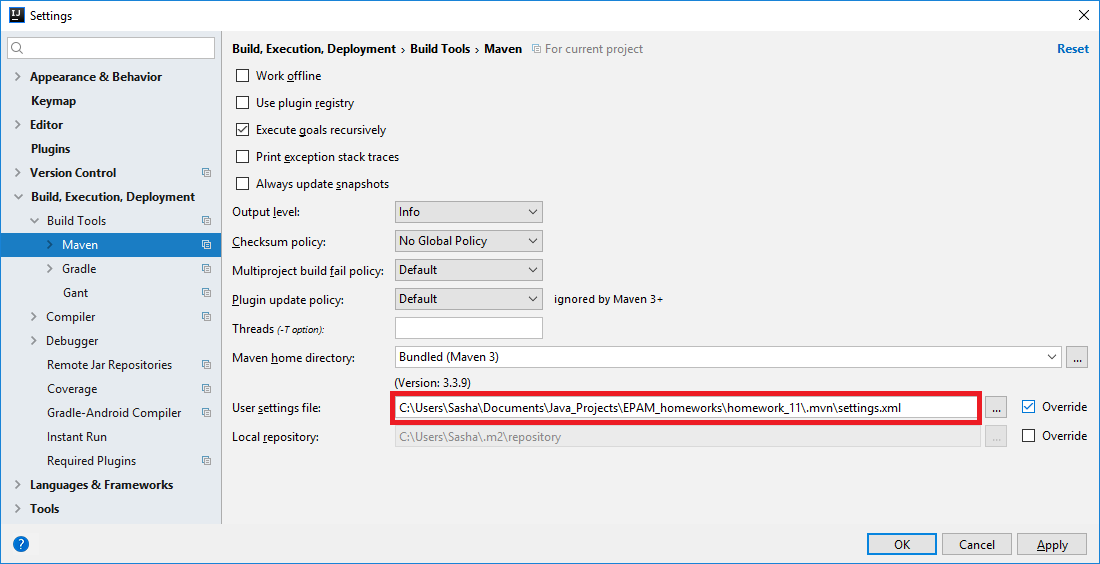
Выполняется командой mvn clean package tomcat7*:run*

Или так *mvn tomcat7:redeploy*

Предварительно надо настроить файлы

А. tomcat-users.xml по адресу C:\Program Files\apache-tomcat-9.0.8\conf

Б. settings.xml по адресу C:\Users\Sasha\.m2



В. pom.xml в проекте – добавить туда плагин:

<**plugin**>  
 <**groupId**>org.apache.tomcat.maven</**groupId**>  
 <**artifactId**>tomcat7-maven-plugin</**artifactId**>  
 <**version**>2.2</**version**>  
 <**configuration**>  
 <**url**>http://localhost:8080/manager/text</**url**>  
 <**server**>apache-tomcat-9.0.8</**server**>*<!-- id-шник из settings.xml -->*

<**update**>true</**update**>  
 <**path**>/${project.build.finalName}</**path**>  
 </**configuration**>  
</**plugin**>

Более подробная информация – в статьях

D:\Programming\Java\05\_Articles\Maven\Работа с Maven + Tomcat

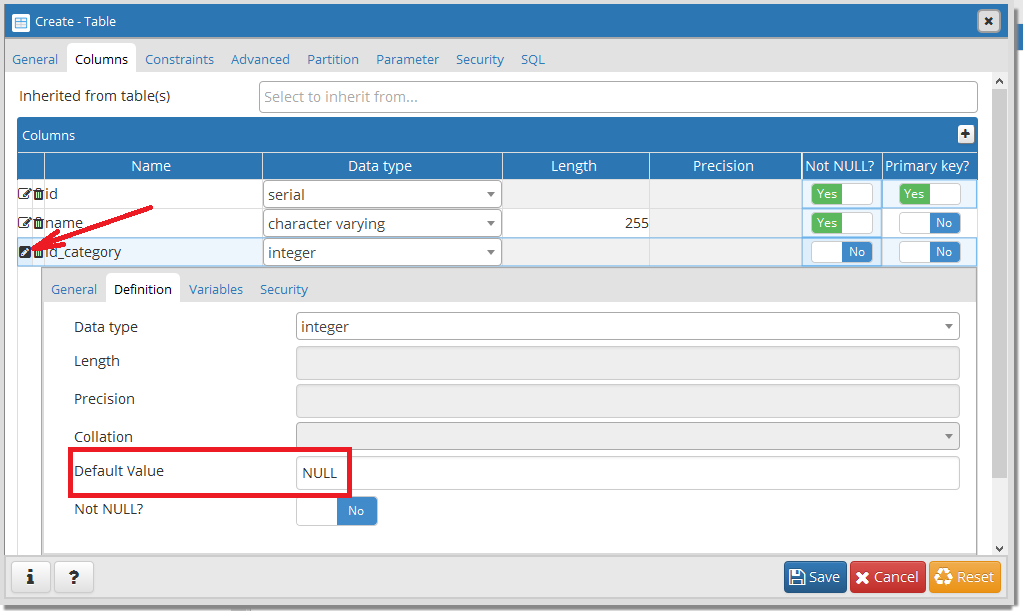
D:\Programming\Java\05\_Articles\Maven\Развертывание веб-приложения с помощью плагина tomcat7-maven-plugin

***1.2 Через плагин cargo-maven2-plugin***

В проекте используется зависимость javax.el-api version 3.0.0. А плагин tomcat7 использует ту же зависимость но version 2.2. Поэтому использование плагина tomcat7 для старта этого приложения представляет из себя нецелесообразно сложную задачу. Вместо этого, можно использовать плагин cargo-maven2-plugin. Для старта приложения - выполнить команду cargo:run

**2. Создание базы данных в Postgres**

2.1. Default values



**3. Разворачивание БД при старте приложения с помощью Liquibase**

На текущий момент надо перед деплоем на томкате выполнить команду

mvn liquibase:update

**4. Использование Lombok**

Кроме зависимости Maven, надо также подключить плагин Lombok, который будет говорить компилятору что не стоит ругаться на отсутствующие геттеры и сеттеры.

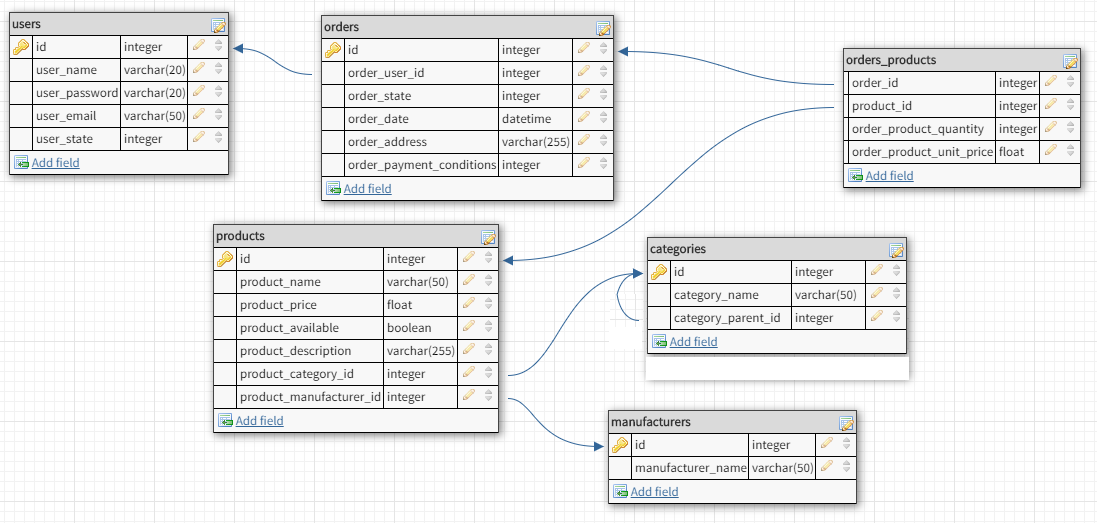
File → Settings → Plugins

Там этого плагина не будет, поэтому надо будет выбрать команду “***Search in repositories***”

**4. Названия свойств типа ${username}**

Надо обязательно задавать уникальные имена. Я для настройки подключения к базе данных задал имя свойства ${username}. Неожиданно оказалось, что такое свойство уже есть – имя пользователя пк. Мне, ессно, выдает ошибку – user Sasha наплужил с паролем. Я часа два копаюсь и понимаю – правда наплужил, и не только с паролем. В общем потом поменял свойство на ${jdbc.username} и все заработало.

**5. Схема базы данных**



**6. Использованные паттерны**

6.1. Паттерн шаблонный метод – классы AbstractMapper и конкретные реализации мапперов.

6.2. Паттерн синглтон – все сервисные классы.

6.3.

**7. Разные тонкие ньюансы**

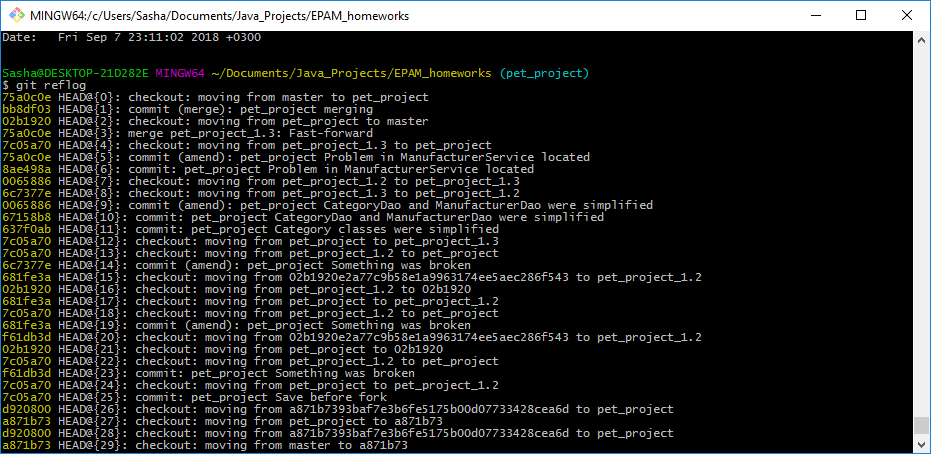
7.1. Классы из пакета model надо делать Serializable. Возможно нам понадобиться сохранять состояние сессий.

7.2. Response can’t be committed twice. If we called response.sendRedirect, we can’t call sendRedirect on the servlet, to which we were redirected. So, before calling response.sendRedirect, it could be useful to check condition response.isCommitted

7.3. If Filter is working for some servlet. For example, for path “/order”. And if filter made redirection, servlet for the path “/order” will work in any case (As far, as I understand). But, we can’t commit response in that servlet, because it was already committed in filter. So, we should check condition response.isCommitted before calling response.sendRedirect.

7.4 Была такая сложно уловимая проблема. Я пытаюсь залогироваться и сайт просто виснет. Без ошибок, как в дедлоке. Оказалось что при логировании инициировались статические объекты типа Dao. Последний такой объект не мог получить Connection и все зависало.

7.5 Если откатиться на безымянный коммит и начать работать с него, то мы создадим таким образом безымянную ветку. Если потом перейти на обычную именованную ветку, то безымянная ветка вообще видна не будет. У меня из-за этого возникла неприятная ситуация. Я с безымянной ветки переключился на master и выполнил merge, как я думал последних изменений из ветки pet\_project. А оказалось, что последние изменения были в безымянной ветки, о которой я даже не подозревал на тот момент. В результате я пытаюсь найти свои изменения, а их нет ни в master ни в pet\_project. Более того, безымянную ветку тоже не видно. Чуть сердце не стало. Проблему решает команда git reflog.



Там показана полная история переходов между ветками. Я просто нашел переход с последнего коммита в безымянной ветки, перешел на этот коммит с помощью git checkout #коммита, а потом на этом коммите создал новую ветку и слил ее туда куда надо было.

7.6. Что означает запрос

**SELECT \* FROM manufacturers**

**WHERE manufacturers.id IN (SELECT product\_manufacturer\_id FROM products WHERE product\_category\_id = ?)**

**ORDER BY manufacturer\_name**

Запрос означает что надо выбрать все строки из таблицы manufacturers, в которых значение столбца manufacturers.id соответствует одному из значений, полученному по запросу **(SELECT product\_manufacturer\_id FROM products WHERE product\_category\_id = ?)**

7.7. В проекте использовались разные подходы к организации классов модели и классов DTO.

7.7.1. При обновлении каких-то товаров, обновление списка товаров из сессии выполняется сразу же тем же сервлетом, что и обновлял товары. При чем это выполняется сервисом ProductService.

А при обновлении информации о заказах, обновление списка заказов выполняется тем же сервлетом, но при этом дергаются по очереди два метода из OrderService – update и getAll.

Что можно улучшить – с товарами сделать так же, как и с заказами. Но для этого мне надо сохранить в сессии настройки предыдущего использованного фильтра.

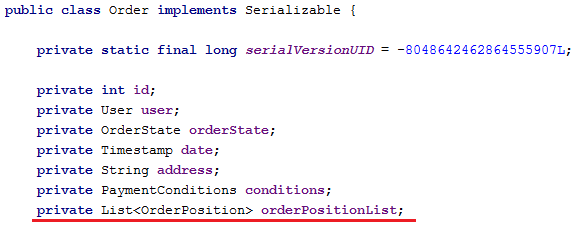
7.7.2 В класса ProductViewDto содержится только поле categoryId и не содержится поле categoryName. Поэтому при отображении информации о товаре, приходится перебирать список всех категорий и находить категорию с нужным id.

В классе OrderViewDto, напротив, содержится информация о id пользователя и о его имени. Отображать проще. Но сама модель неудобна.

Тут надо обдумать как это улучшить. Возможно надо использовать ссылки на целые объекты (типа User, Category).

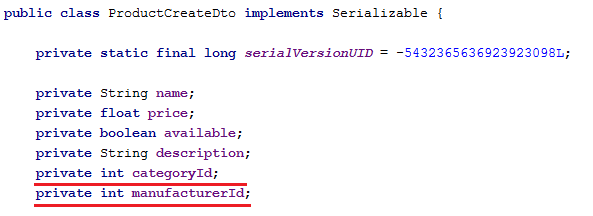
**MODEL, DTO**

1. Списки вложенных объектов содержат в себе только классы Order, OrderCreateDto и OrderViewDto. Они содержат в себе списки объектов OrderPosition.

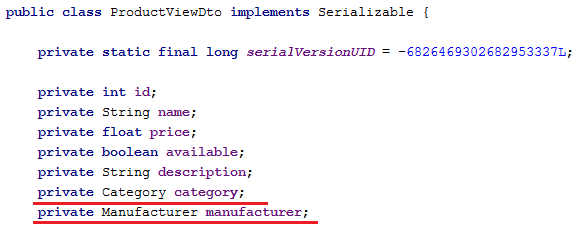


1. Объекты Category и Manufacturer не содержат в себе вложенных списков товаров так как в целом не предполагается вытаскивать товары просто под какую-то категорию или производителя. Поиск товаров (как при покупке, так и при администрировании) в любом случае осуществляется по многопараметрическому критерию и результат поиска нельзя соотнести ни с какой-то категорией, ни с каким-то производителем.
2. В объектах CreateDto для отображения связей с другими сущностями используются идентификаторы этих сущностей.

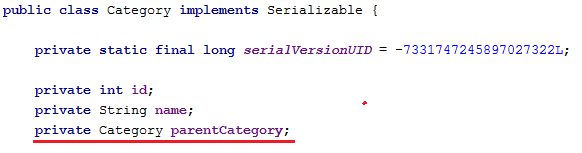
Например:



А в объектах ViewDto, используются уже не идентификаторы сущностей, а сами сущности:



1. Объекты CreateDto не содержат идентификаторов своих собственных сущностей. Они не известны до моменты вставки в базу данных.
2. Объекты Category, CategoryViewDto содержат ссылку на объекты родительских категорий.



1. Объекты OrderPosition не содержат ссылку на целый объект Product, а только на идентификатор и название объекта Product. Так сделано потому что мы фактически не используем остальные поля из Product. Цену мы использовать не можем потому что цена хранится прямо в объекте OrderPosition (если покупатель сделал заказ, нельзя менять цену). Точно по той же логике нельзя использовать и остальные поля товара. Для покупателя в заказе товар должен оставаться в том состоянии, в котором покупатель делал заказ.