**Git**

Write the following in the comments before merging to master branch:

[] Code

[] Test

[] Review

Git files should have src, .gitignore and pom.xml

If the pom.xmpl isn’t detecting the Junit dependency click on project file -> maven -> update project…

C:\Users\Admin\.m2\repository shows all the external files ever downloaded by maven.

Collection is an interface.

accountMap.values().stream().filter(

Transactions- Either All happens or nothing happens. Rolling back to original state. E.g. when a person is trying to transfer money to another person. If one server dies in the middle of the stages the money will go back.

JPA is the API which works with the persistence layer.

Bean is a managed java object.

POJO- plain old java object

CDI API

**Steps to converting to EE:**

Copy everything in movie-sample api pom.xml from packaging to your pom.xml

Persistence.xml is configuration for JPA.

Copy WEB-INF

META-INF

Copy import.sqp

Com.qa.domain

Com.qa.service

Com.qa.util

Com.qa.persistenece.repository

Com.qa.REST

Create, Delete, Update should be transactional whereas Read shouldn’t because it doesn’t need to roll back to anything. @Transaction(Required) is anything which makes changes, e.g. adding data to the DB.

@Inject means its creating without using the ‘new‘ keyword.

We can’t create an instance of an interface so we use @Inject to create an object of the class and since there is only one class which implements the interface it will automatically create an object of that one class.

Entity manager is what we use to communicate with our database table and goes under the persistence layer (in the repository). Used to communicate with the domain in the persistence layer as well as the actual database.

JPQL is what is used to make queries.

@Entity – means it’s going to represents a table.

We focus on loosely coupled because if you make a small change in a class where there is an issue in it, it wouldn’t affect other classes.

SOLID principle:

Single – each class has one reason to change

Cohesion means that all methods within a class aim to achieve one goal and make sense together.

We want to aim for high cohesion and low coupling (loosely coupled).

***@Inject***

**private** AccountRepository repo;

You can then specify what object is should be created from by using @Default or @Alternative.

CDI manages the life cycle of a bean automatically.

CDI Scopes – we will only use ApplicationScoped and RequesteScoped. You would apply the annotation at the top of the class.

Create new maven project.

Set up pom.xml and other configuration files.

Create META-INF file which contains persetence.xml

Create the folder in Src main webapp WEB-INF application-…. & and beans.

Create packages:

Com.qa.util

Com.qa.persitence.domain

Com.qa.persitence.repository

Com.qa.service

Com.qa.rest

Create new class in domain called “projectname” (noun)

@Size(min = 6, max = 6)

Private String accountNumber

By using @Inject you let the CDI manage the life cycle of the bean.