Git :

<https://github.com/>

* Pas encore fait le helloWorld

Projet Machine à café :

<http://simcap.github.io/coffeemachine/>

<https://translate.google.fr/translate?hl=fr&sl=en&u=http://simcap.github.io/coffeemachine/&prev=search> : site en français

* Mockito : <http://blog.soat.fr/2013/07/mockito-ou-comment-faciliter-lecriture-de-tests-unitaires/>
* Junit def : <https://fr.wikipedia.org/wiki/JUnit>
* Junit : <http://blog.soat.fr/2014/02/du-bon-usage-de-junit-12/>

<http://blog.soat.fr/2014/02/du-bon-usage-de-junit-22/>

* + 1er test avec Junit : <http://www.junit.fr/2011/11/20/tutoriel-eclipse-junit-mon-premier-test-automatique/>

Applet HelloWorld : A faire !!

Exemple de codage : <http://entraide-sr.org/pages/documents/documents_ISN/COURS-ISN-semaine_4_machine-a-cafe.pdf>

Spring : <http://www.tutorialspoint.com/spring/index.htm>

JPA :

Persistance des données : c’est le fait de stocker des données d’une application de manière… persistante ! Autrement dit, de les sauvegarder afin qu’elles ne disparaissent pas lorsque le programme se termine. Rien de bien compliqué, en somme.

En Java,  on évoque couramment un système permettant **la sauvegarde des données contenues dans des objets**. En réalité, vous connaissez donc déjà tous un moyen de persistance : le stockage dans une base de données relationnelle via JDBC.

\*l'objectif du pattern DAO (Data Access Object) tient en une phrase : il permet de faire la distinction entre les données auxquelles vous souhaitez accéder, et la manière dont elles sont stockées. Permet de faire le lien entre la couche métier et la couche persistante, ceci afin de centraliser les mécanismes de mapping entre notre système de stockage et nos objets Java\*

* C’est quoi JPA ? : Littéralement « Java Persistence API », il s’agit d’un standard faisant partie intégrante de la plate-forme Java EE, une spécification qui définit un ensemble de règles permettant la gestion de la correspondance entre des objets Java et une base de données, ou autrement formulé la gestion de la persistance.

Ce mécanisme qui gère la correspondance entre des objets d’une application et les tables d’une base de données se nomme ORM, pour « Object-Relational Mapping ».

Lorsqu’on parle de JPA, il s’agit uniquement d’une API, c’est-à-dire une description d'un comportement à suivre, en l'occurrence pour respecter un standard en place.

Les interfaces de JPA décrivent comment respecter le standard, mais nous devons utiliser une implémentation pour en tirer parti ! Voilà donc ce que sont Hibernate, EclipseLink & consorts : des implémentations du standard JPA. En d’autres termes JPA constitue la théorie, et ces frameworks en sont la pratique.

* Pourquoi elle a été mise en place ? : Il permet de s’assurer que la correspondance entre les tables et les objets est correctement réalisée, et ce pour chaque action effectuée : lecture de données, mises à jour, insertions, suppressions… il permet d’écrire une requête SQL spécifique, à chaque fois il faudrait récupérer son résultat et éventuellement en extraire les données pour créer ou mettre à jour les attributs d’un bean, à chaque fois…
* C’est quoi une entité ?
  + l’EJB Entity. C’est lui qui définit quelles données doivent être sauvegardées, et c’est à travers lui qu’est effectuée la correspondance entre un objet et une table d’une base de données. Ce sont des objets ressemblant très fortement à des JavaBeans, dans lesquels des annotations définissent des correspondances entre les objets et leurs attributs d'un côté, et les tables relationnelles de la base de données et leurs champs de l'autre (on parle alors de mapping relationnel/objet).
  + EJB Session : Il s'agit simplement d'un objet qui donne accès aux services & méthodes qu'il contient. Il existe deux types d’EJB Session : ceux qui sont Stateless, et ceux qui sont Stateful.
* comment on la déclare ? conteneur EJB, qui permet de gérer entièrement le cycle de vie des EJB qu’il contient.
* C’est quoi un EntityManager ? un système qui permet de définir comment les données qu’on veut sauvegarder doivent être sauvegardées !

Les méthodes permettant d’établir une connexion avec la base de données et de gérer la persistance se trouvent dans un objet particulier nommé EntityManager, ou gestionnaire d'entités. Il s’agit, là encore, d’un objet dont le cycle de vie est géré par le conteneur.

***C'est une classe qui est chargée de mettre en musique les correspondances définies dans les entités, et qui réalise donc toutes les opérations CRUD (Create, Read, Update, Delete) sur la base de données***.

L'EntityManager est donc l'objet qui va permettre au développeur de manipuler ses objets Java devenus des entités et ainsi lui permettre de les persister

* C’est quoi un JPQL ? : Le langage JPQL est un langage de requête dont la grammaire est définie par la spécification J.P.A. Il est très proche du langage SQL dont il s'inspire fortement mais offre une approche objet.

// SQL Query

SELECT \* FROM STUDENT

// JPQL Query

SELECT s FROM Student s

<http://igm.univ-mlv.fr/~dr/XPOSE2007/acollign_ORM-JPA/jpa-tech2.html>

Spring :

* C’est quoi un Bean ? : un JavaBean désigne tout simplement un composant réutilisable. un bean est un simple objet Java. Il est paramétrable, persistant (conserve l’état du bean et peut être restauré), paramétrable de manière dynamique.
* Comment on le déclare ? :
  + doit être une classe publique
  + doit avoir au moins un constructeur par défaut, public et sans paramètres.
  + peut implémenter l'interface Serializable, il devient ainsi persistant et son état peut être sauvegardé
  + ne doit pas avoir de champs publics
  + peut définir des propriétés (des champs non publics), qui doivent être accessibles via des méthodes publiques getter et setter, suivant des règles de nommage.
* Les differents types de beans ? : EJB sessions, EJB Entity, EJB message (les clients ne s’adressent pas aux composants …)
* Inversion de dépendances : L’inversion de contrôle (inversion of control, IoC) est un patron d'architecture commun à tous les frameworks (ou cadre de développement et d'exécution). Il fonctionne selon le principe que le flot d'exécution d'un logiciel n'est plus sous le contrôle direct de l'application elle-même mais du framework ou de la couche logicielle sous-jacente.

L’inversion de contrôle est un terme générique. Selon le problème, il existe différentes formes, ou représentation d'IoC, le plus connu étant l'injection de dépendances (dependency injection) qui est un patron de conception permettant, en programmation orientée objet, de découpler les dépendances entre objets.

* L’AOP (Aspect oriented programming) : L'AOP est un type de programmation qui propose de séparer le code technique du code métier d'une application pour des préoccupations transversales qu'elles soient techniques ou architecturales. Complète la POO.