@ReviewHibernate

* ORM (object relational mapping framework)
* It connects your entity classes in Java to tables in the database
* Benefits of Hibernate
  + Abstracts away JDBC. Removes you having to write a lot of boiler plate code
  + Faster development and easier to update
  + Hibernate’s abstraction of JDBC means that you are not tied to a specific database
  + Features like HQL and criteria provide faster more convenient ways of getting information
* States of an object in Hibernate
  + Transient (The object has not been saved)
  + Persisted (The object is saved and connected to a session)
  + Detached (The object was part of a session but is now detached)
* When we use annotations we are using annotations from javax.persistence
* These annotations come from JPA (Java Persistence API)
  + JPA is the Java Standard for ORM
* Annotations
  + @Entity marks class to be saved to database
  + @Table(name=”table\_name”)
  + @Id marks field as primary key
  + @GeneratedValue(strategy = Identity)
  + @Column
  + @JoinColumn (says that this field is actually a foreign key to a record)
  + @OneToMany (this class is a parent to another table)
    - Belongs over some list or set
  + @ManyToOne (this class is a child to another table)
    - Belongs over a single object
  + ManyToMany (this class has a many to many relationship with another class
    - Can be placed on either class
  + @JoinTable (specify the table that holds the foreign keys that connects the two)
    - Joincoulmns {holds the join column connecting the jointtable to the current entity}
    - InverseJoinColumns {holds the foreign key to the other entity}

Interfaces of Hibernate

* Configuration “interface” – Object that is used to store your Hibernate configurations
  + When you first create a configuration object the object will automatically search for a hibernate.cfg.xml to configure itself
* SessionFactory – Heavy duty object that is used to **create sessions**. It is responsible for creating a connection to the database as well as performing the initial mapping.
* Session – smaller objects that are used to interact with the database.
  + You need them to make queries, criteria and transactions
  + Methods for sessions
    - Save()
    - Update()
    - Delete()
    - Get()
* Transaction – Interface used for creating committing and rolling back transactions in hibernate
  + Methods
    - Commit()
    - Rollback()
* Criteria – most OOP approach to make queries of your database
* Query – used to create HQL or SQL queries

Ways to Query in Hibernate

* Criteria API
  + Very OOP approach to make queries
  + You get a criteria from a session object
  + You add Restrictions to the criteria
  + You then execute that criteria
  + Improves interoperability
* HQL
  + Hibernate Query Language
  + Similar to SQL but with a more oop flavor
  + Does not tie you to a specific sql syntax database
  + Improves interoperability
* SQL
  + Just straight up SQL
  + Try to avoid using
  + It ties you to that specific database

Configuring Hibernate

* Hibernate uses the hibernat.cfg.xml
* This file contains basic information
  + Database location
  + Username
  + Password
  + Type of database (MariaDB, Postgres, Oracle)
* Contains the classes that you have mapped with annotations

Eager vs Lazy Loading

* Eager loading is when an object is loaded with all nested objects in it all at once
  + Ex. A director object is loaded with all of its movies when you get the director
  + Make query and get all the information back immediately
* Lazy loading is when an you get a back a **proxy** of an object in the database.
  + The proxy does not have the nested objects within it
  + Ex. A director lazy loaded will not have its movies until you call a method that actually uses those movie objects
  + Lazy loading can help prevent loading nested objects that you do not actually use
  + Lazy loading is also the default in one to many relationships

L1 vs L2 caching

* Level 1 caching
  + A cache associated with each session
  + Each session can have a whole bunch of objects in it
  + Rather than immediately going to the database to search for a record. Hibernate will check the session cache to see if the object is already there
  + This is done for you automatically
* Level 2 caching
  + A cache associated with a SessionFactory
  + You CAN configure Hibernate to check all sessions before making a database query
  + Not done automatically
  + Applies to open sessions
  + Must use some 3rd part software outside of base Hibernate