ORM with Hibernate

Krishantha Dinesh
[kdinesh@virtusa.com]

@2013 Nov

2000 West Park Drive
Westborough MA 01581 USA
Phone: 508 389 7300 Fax: 508 366 9901

virtusa®
Accelerating Business Outcomes

objectives

- Not to make a exerts about hibernate at end of the session
- But give a basic usage and understanding about hibernate
- Get in to advance topics about hibernate

virtusa

What is hibernate

- Open source ORM
- Use to mapping to objects to relational database
- It can use with almost all available database
- Originally create in 2001 to help EJB 2.0

virtusa

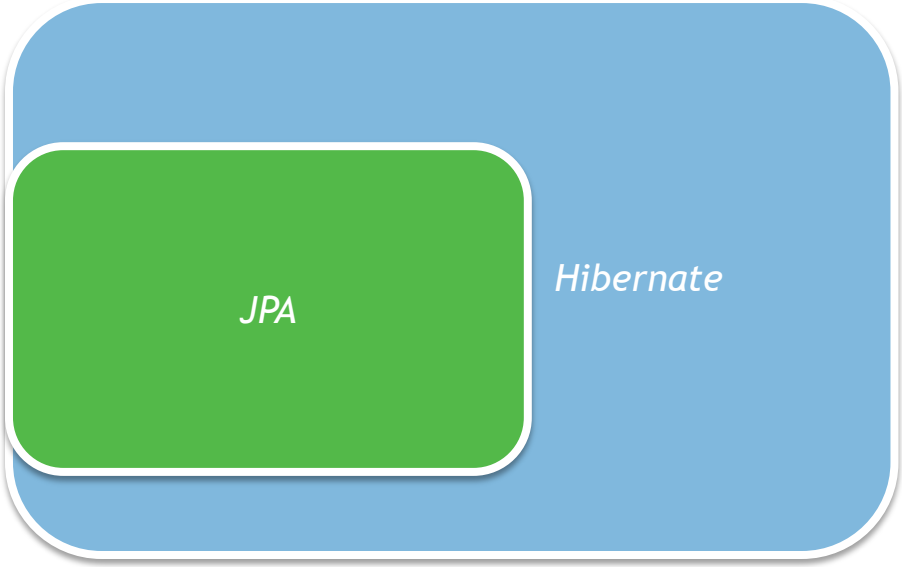
JPA and hibernate

- JPA – it is a specification for java. It is not a framework
- JPA works like a interface means it guide to implement ORM framework
- JPA is act as a abstraction layer of implementation of persistence. Means if develop a persistence using jpa interface it can work with any ORM framework which use JPA

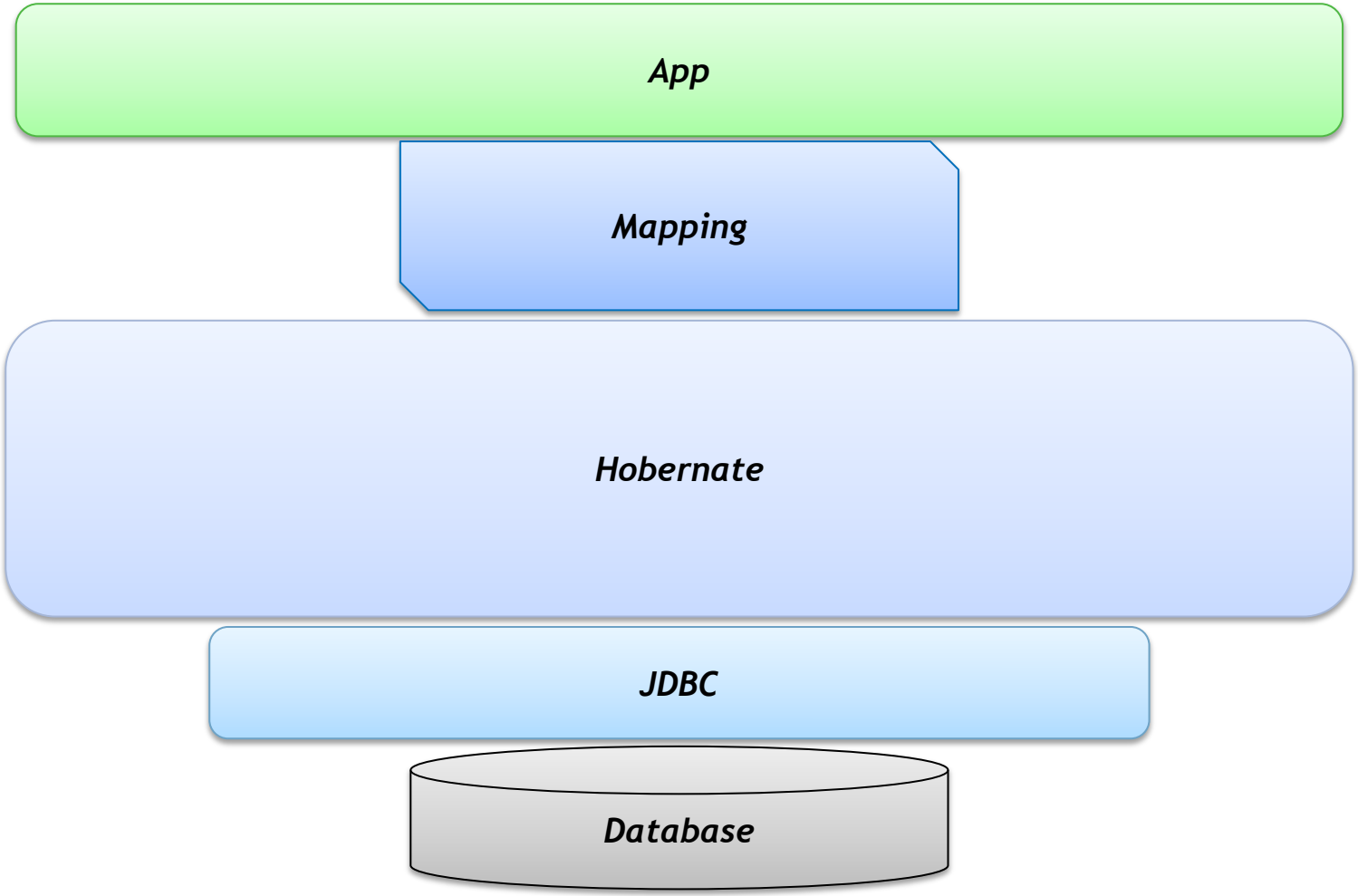
virtusa

What we use JPA or Hibernate

- This course titles as hibernate. So we use hibernate API ☺
- If you consider coverage hibernate has larger power than classic JPA. So we use hibernate

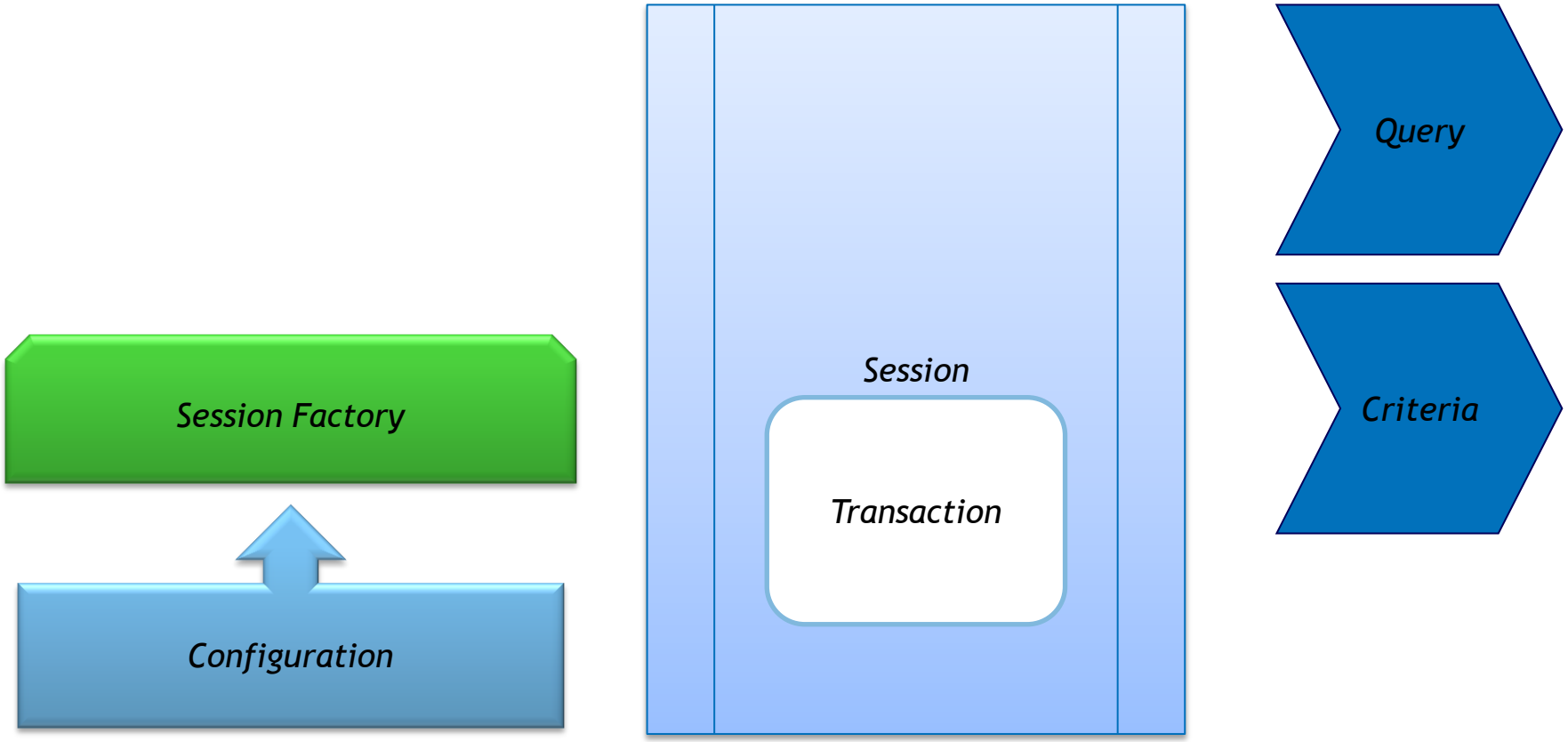


Hibernate overview



virtusa

Hibernate Inside



virtusa

Notes

- Configuration
  - Need create configuration class to read the data base configuration
- Session Factory
  - Need session factory instance to create a session. Usual way is single instance o this and use application wide
- Transaction
  - It is optional but recommended

virtusa

Configure IDE

- You can use eclipse or spring STS
- You need to install hibernate plugin based on your eclipse version.
- If you use spring STS find the relevant eclipse version. (look the zip file name)
  - Eg: spring-tool-suite-3.4.0.RELEASE-e4.3.1-win32-x86_64 means eclipse 4.3.1
- Go to following URL and find relevant update site
  - http://www.jboss.org/tools/download/
- According to above zip file update URL is
  - http://download.jboss.org/jbosstools/updates/stable/kepler/
- Use that URL to install plugin. See next slide
- [if you have problem with install download archive and install manually]

virtusa

www.jboss.org/tools/download/

hibernate tool eclipse

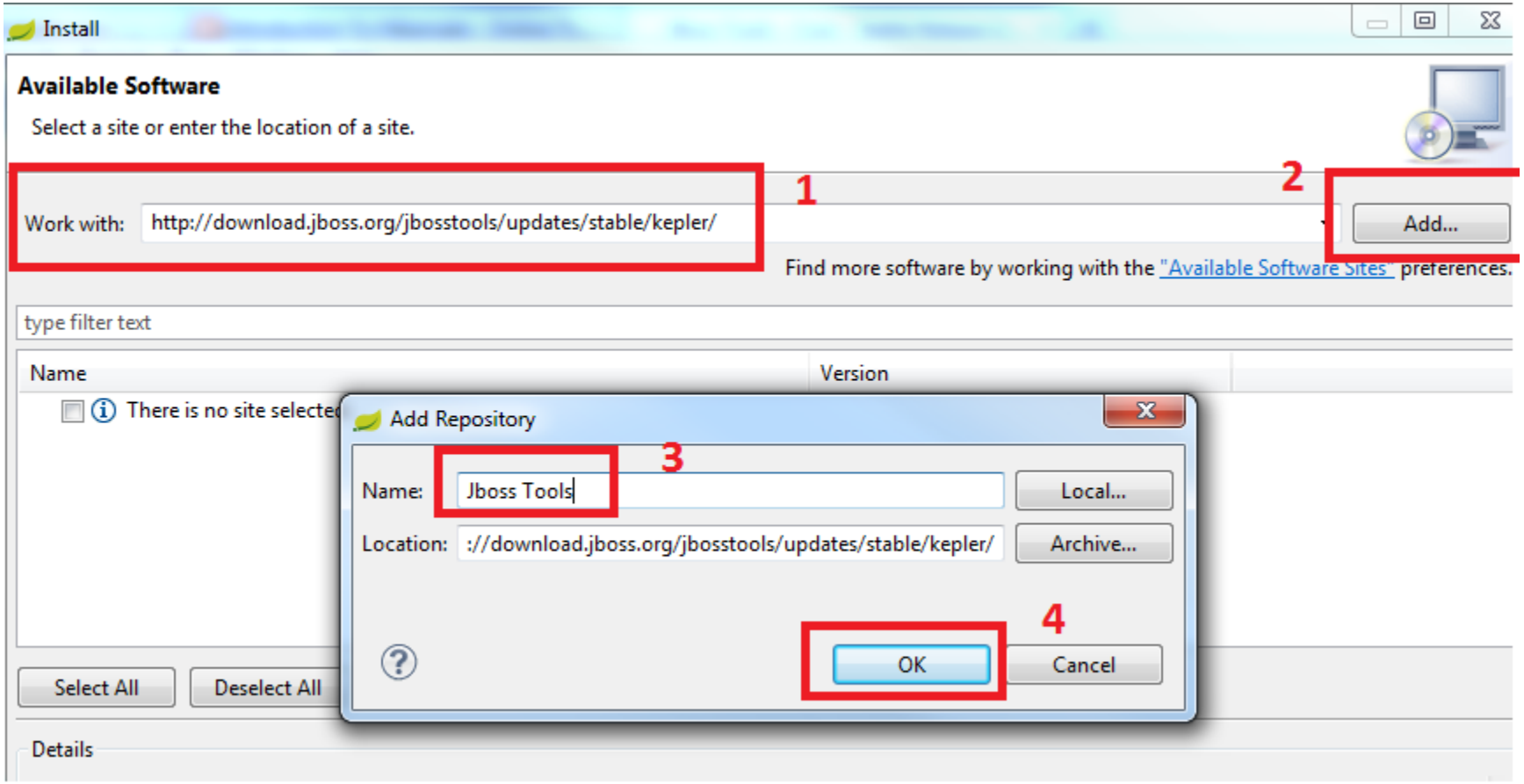
Update Sites

To install via update site, simply right-click the link below from which you'd like to install, copy the link, and paste it into Eclipse's Update or Install Manager. See Installing JBoss Tools for more information.

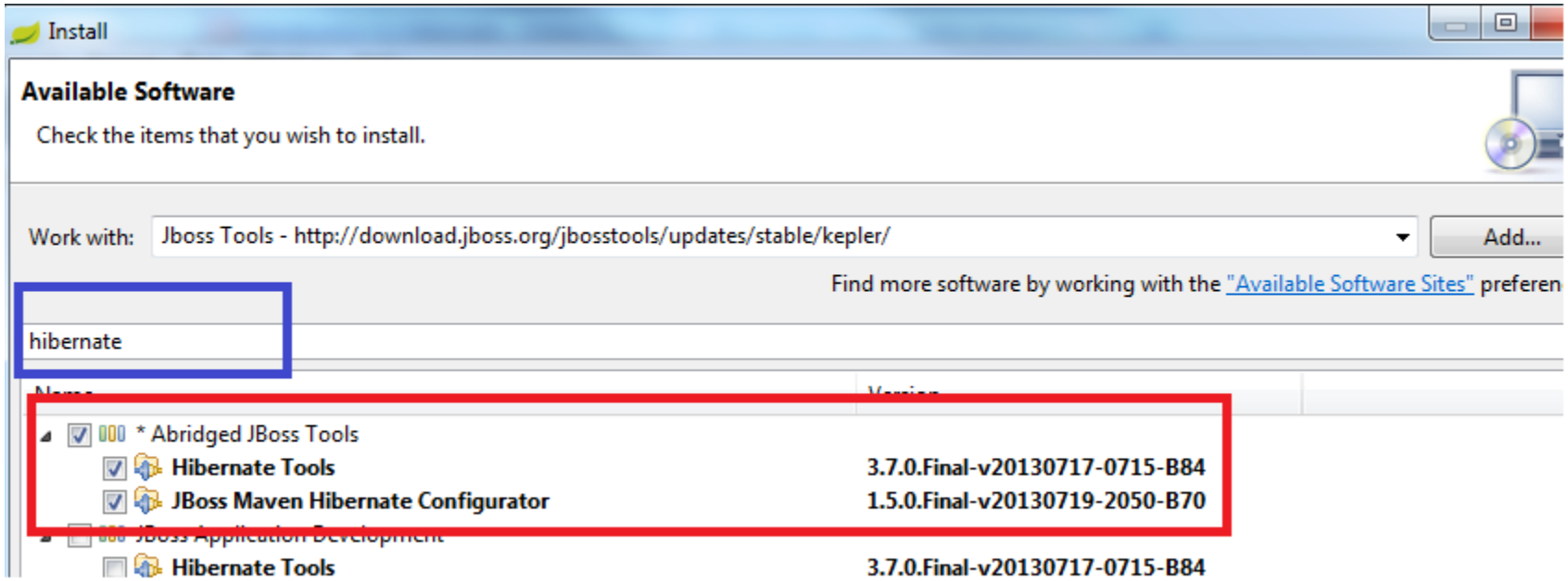
Stable Releases	Development Milestones	Nightly / Integration Update Sites
JBoss **Core** Tools 4.1 :: Eclipse 4.3.0	JBoss **Core** Tools 4.1.1.Beta1 :: Eclipse 4.3.0	JBoss **Core** Tools 4.2 :: Eclipse 4.4.x
JBoss **Core** Tools 4.0 :: Eclipse 4.2.2	JBoss Tools **Integration Stack** 4.1.3.Beta4 :: Eclipse 4.3.0	JBoss **Core** Tools 4.1 :: Eclipse 4.3.x
JBoss **SOA** Tools 3.3 :: Eclipse 3.7.2		JBoss Tools **Integration Stack** 4.1 :: Eclipse 4.3.x
JBoss **Core** Tools 3.3 :: Eclipse 3.7.2		JBoss Tools **Integration Stack** 4.0 :: Eclipse 4.2.2
JBoss Tools 3.2 :: Eclipse 3.6.2		JBoss **SOA** Tools 3.3 :: Eclipse 3.7.2
JBoss Tools 3.1 :: Eclipse 3.5.2		
JBoss Tools 3.0 :: Eclipse 3.4.2		
JBoss Tools 2.1 :: Eclipse 3.3.2		

Downloads

- In eclipse or STS go to help → install new software and paste copied update site URL



- Filter to hibernate and select hibernate tools and maven configurator
- Click Next and again Next
- Accept the agreement and finish



If not install manually

Install

Available Software

Select a site or enter the location of a site.

Work with: type or select a site

1 Add...

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
There is no site selected.	

Select All | Deselect All

Details

Show only the latest versions

Group items by category

Show only software applicabl

Contact all update sites durin

Add Repository

Name: hibernate-tools 3 Local...

Location: jar:file:/C:/Krishantha/developments/java/ide/hibernatetc Archive... 2

4 OK Cancel

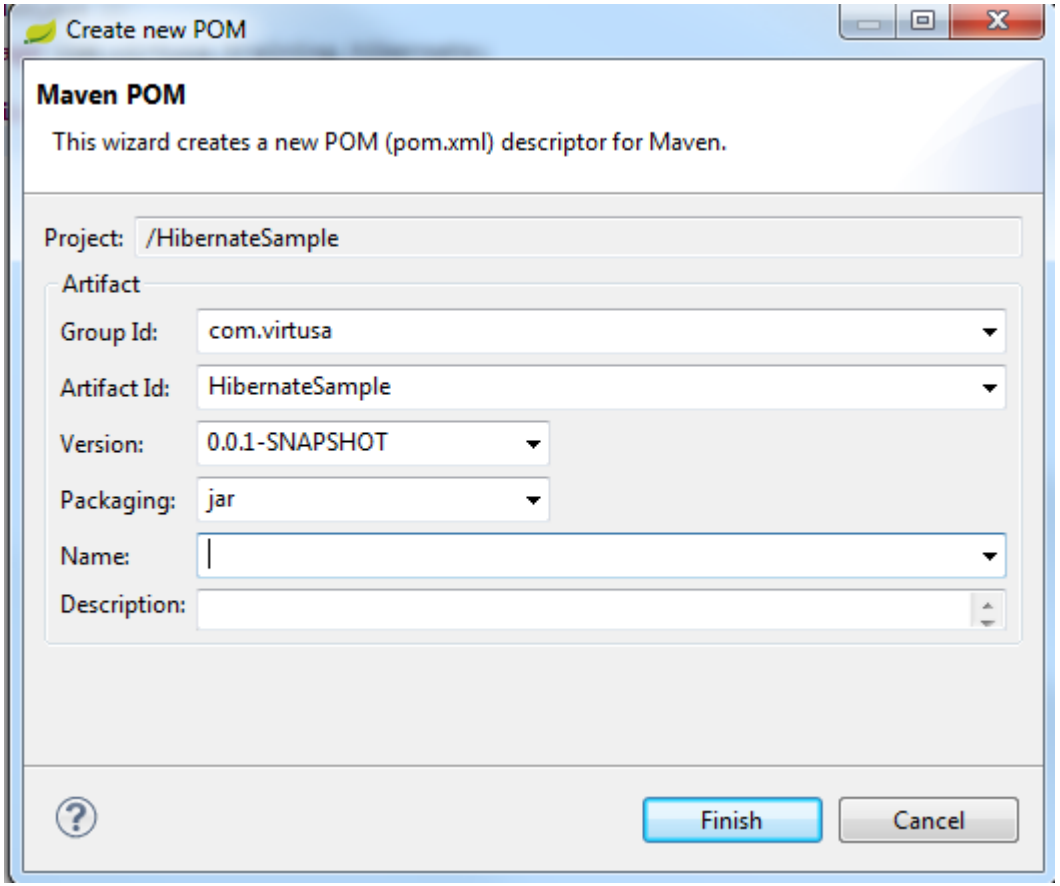
< Back | Next > | Finish | Cancel

virtusa

Start a Project

- Open IDE and create new java project
- File → new → java project
- Project name : HibernateTraining
- Create a new class Application.java
- Package com.virtusa.training.hibernate
- Right click on project → configure → convert to maven

virtusa



Add Hibernate to project

- Open pom.xml file and add following dependency
- It will download hibernate for your project

```
<repositories>
        <repository>
                <id>jboss-public-repository</id>
                <url>https://repository.jboss.org/nexus/content/groups/public/</url>
        </repository>
</repositories>
<dependencies>
        <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-core</artifactId>
                <version>4.2.7.SP1</version>
                <scope>compile</scope>
        </dependency>

        <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
                <version>5.0.8</version>
                <type>jar</type>
                <scope>compile</scope>
        </dependency>

</dependencies>
```

virtusa

Project structure should look like this

- Maven Dependencies
  - hibernate-core-4.2.7.SP1.jar - C:\Users\kdinesh\.m2\reposi
  - antlr-2.7.7.jar - C:\Users\kdinesh\.m2\repository\antlr\antl
  - jboss-logging-3.1.0.GA.jar - C:\Users\kdinesh\.m2\reposit
  - dom4j-1.6.1.jar - C:\Users\kdinesh\.m2\repository\dom4j\
  - javassist-3.18.1-GA.jar - C:\Users\kdinesh\.m2\repository\
  - jboss-transaction-api_1.1_spec-1.0.1.Final.jar - C:\Users\kd
  - hibernate-jpa-2.0-api-1.0.1.Final.jar - C:\Users\kdinesh\.m
  - hibernate-commons-annotations-4.0.2.Final.jar - C:\Users
  - mysql-connector-java-5.0.8.jar - C:\Users\kdinesh\.m2\rep
- bin
- target
- pom.xml

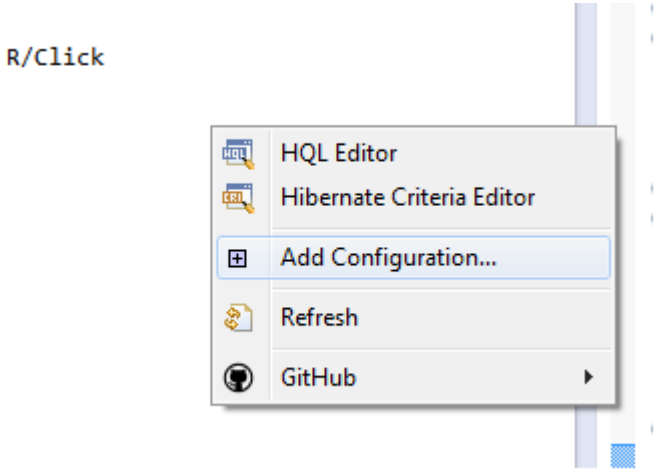
My SQL database

- Using command line tool or work bench create new database on mysql
  - CREATE SCHEMA `hibernate_sample` ;

virtusa

Create Hibernate configuration file (xml)

- Shift your perspective to hibernate perspective.
- You can do it from windows → open perspective → other → hibernate
- Right click on hibernate configuration window and select add Configuration



- Select project under project section
- Database connection : hibernate configured connection
- Under configuration file section : select setup

virtusa

Hibernate Configuration File (cfg.xml)

This wizard creates a new configuration file to use with Hibernate.

Container:	/HibernateSample/src
File name:	hibernate2.cfg.xml
Session factory name:	

Get values from Connection

Database dialect:	MySQL
Driver class:	com.mysql.jdbc.Driver
Connection URL:	jdbc:mysql://localhost:3306
Default Schema:	hibernate_sample
Default Catalog:	
Username:	system
Password:	dbsys

< Back | Next > | Finish | Cancel

Create a session factory

- Create new package as com.virtusa.training.hibernate.util
- Create new class as HibernateUtilities

```
public class HibernateUtilities {

    private static SessionFactory sessionFactory;
    private static ServiceRegistry serviceRegistry; // new from version 4

    static {
        try {
            Configuration configuration = new Configuration().configure();
            serviceRegistry = new ServiceRegistryBuilder().applySettings(
                    configuration.getProperties()).buildServiceRegistry();
            sessionFactory = configuration.buildSessionFactory(serviceRegistry);
        } catch (HibernateException exception) {
            System.out.println("Error on session factory"
                    + exception.getMessage());

        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Application.java

```java
package com.virtusa.training.hibernate;

import org.hibernate.Session;

import com.virtusa.training.hibernate.util.HibernateUtilities;


public class Application {

    public static void main (String[] args){
        Session session = HibernateUtilities.getSessionFactory().openSession();
        session.close();
    }

}
```

virtusa

```
Nov 19, 2013 11:01:50 PM org.hibernate.hql.internal.ast.ASTQueryTranslatorFactory <init>
INFO: HHH000397: Using ASTQueryTranslatorFactory
Nov 19, 2013 11:01:50 PM org.hibernate.internal.SessionFactoryRegistry addSessionFactory
WARN: HHH000277: Could not bind factory to JNDI
org.hibernate.service.jndi.JndiException: Error parsing JNDI name []
        at org.hibernate.service.jndi.internal.JndiServiceImpl.parseName(JndiServiceImpl.java:92)
        at org.hibernate.service.jndi.internal.JndiServiceImpl.bind(JndiServiceImpl.java:108)
        at org.hibernate.internal.SessionFactoryRegistry.addSessionFactory(SessionFactoryRegistry.java:89)
        at org.hibernate.internal.SessionFactoryImpl.<init>(SessionFactoryImpl.java:481)
        at org.hibernate.cfg.Configuration.buildSessionFactory(Configuration.java:1794)
        at com.virtusa.training.hibernate.util.HibernateUtilities.<clinit>(HibernateUtilities.java:19)
        at com.virtusa.training.hibernate.Application.main(Application.java:11)
Caused by: javax.naming.NoInitialContextException: Need to specify class name in environment or system property, or as an applet paramete
        at javax.naming.spi.NamingManager.getInitialContext(Unknown Source)
        at javax.naming.InitialContext.getDefaultInitCtx(Unknown Source)
        at javax.naming.InitialContext.getURLOrDefaultInitCtx(Unknown Source)
        at javax.naming.InitialContext.getNameParser(Unknown Source)
        at org.hibernate.service.jndi.internal.JndiServiceImpl.parseName(JndiServiceImpl.java:86)
        ... 6 more
```

Test the session factory

- Go back to configuration file and remove name=“”

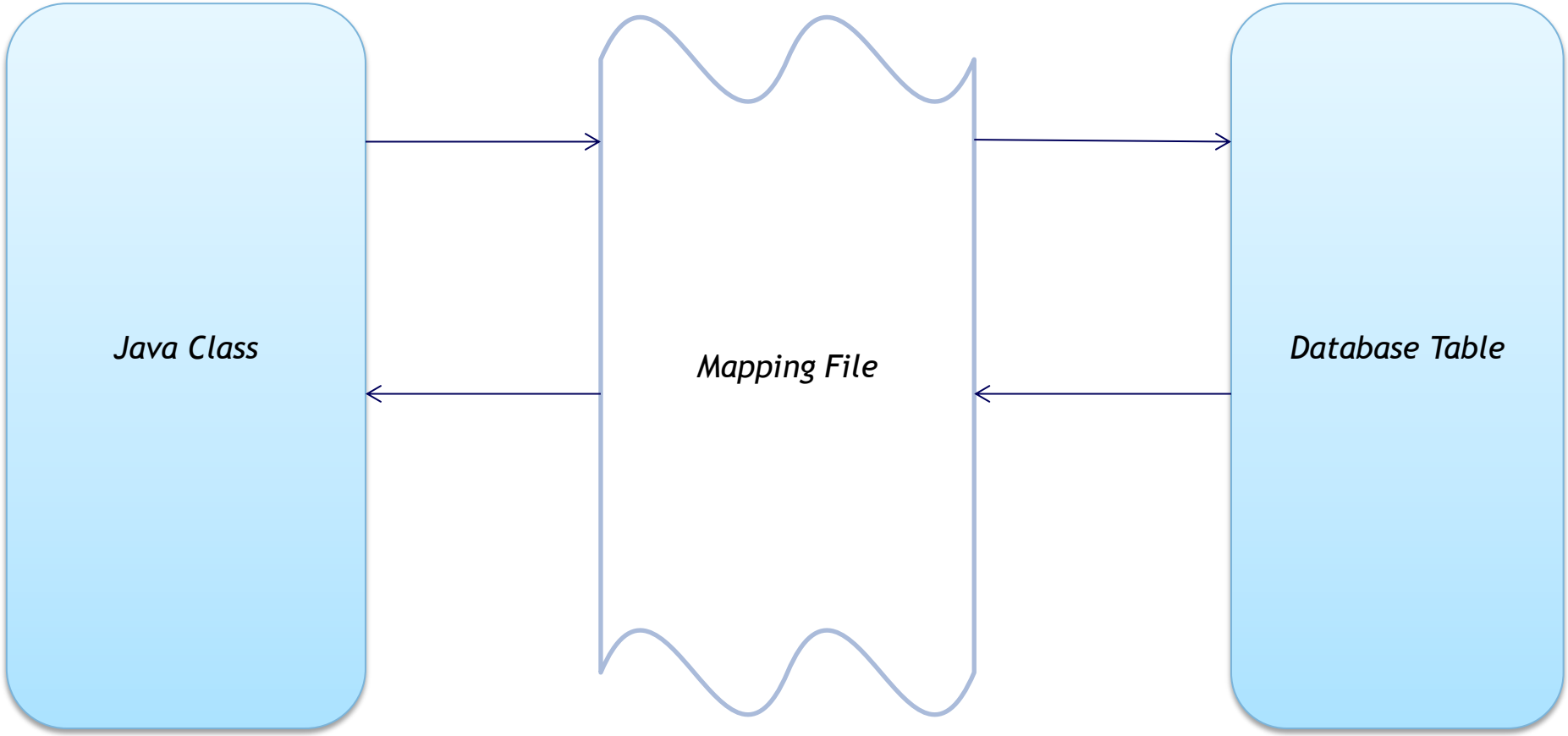
```
                                        "http://hibernate.sourceforge.net/hibernate-c
<hibernate-configuration>
 <session-factory >
  <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
  <property name="hibernate.connection.password">dbsys</property>
  <property name="hibernate.connection.url">jdbc:mysql://localhost:3306</property>
  <property name="hibernate.connection.username">system</property>
  <property name="hibernate.default_schema">hibernate_sample</property>
  <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
  <property name="hibernate.use_sql_comments">true</property>
 </session-factory>
</hibernate-configuration>
```

Worked fine ☺

```
Nov 19, 2013 11:04:03 PM org.hibernate.cfg.Configuration doConfigure
INFO: HHH000041: Configured SessionFactory: null
Nov 19, 2013 11:04:03 PM org.hibernate.service.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
INFO: HHH000402: Using Hibernate built-in connection pool (not for production use!)
Nov 19, 2013 11:04:03 PM org.hibernate.service.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
INFO: HHH000115: Hibernate connection pool size: 20
Nov 19, 2013 11:04:03 PM org.hibernate.service.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
INFO: HHH000006: Autocommit mode: false
Nov 19, 2013 11:04:03 PM org.hibernate.service.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
INFO: HHH000401: using driver [com.mysql.jdbc.Driver] at URL [jdbc:mysql://localhost:3306]
Nov 19, 2013 11:04:03 PM org.hibernate.service.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
INFO: HHH000046: Connection properties: {user=system, password=****}
Nov 19, 2013 11:04:03 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Nov 19, 2013 11:04:03 PM org.hibernate.engine.jdbc.internal.LobCreatorBuilder useContextualLobCreation
INFO: HHH000423: Disabling contextual LOB creation as JDBC driver reported JDBC version [3] less than 4
Nov 19, 2013 11:04:03 PM org.hibernate.engine.transaction.internal.TransactionFactoryInitiator initiateService
INFO: HHH000399: Using default transaction strategy (direct JDBC transactions)
Nov 19, 2013 11:04:03 PM org.hibernate.hql.internal.ast.ASTQueryTranslatorFactory <init>
INFO: HHH000397: Using ASTQueryTranslatorFactory
```

virtusa

Mapping



Mapping can be either XML ot annotation based

How its works

- Mapping can be either xml or annotation based
- We don’t want to do any special thing to map class to object
- To mapping only need is default constructor

virtusa

Data types - primitive

- As we know java data types are not matched with database data type. To overcome this issue hibernate maintain its own data type

Mapping type	Java type	ANSI SQL Type
integer	int or java.lang.Integer	INTEGER
long	long or java.lang.Long	BIGINT
short	short or java.lang.Short	SMALLINT
float	float or java.lang.Float	FLOAT
double	double or java.lang.Double	DOUBLE
big_decimal	java.math.BigDecimal	NUMERIC
character	java.lang.String	CHAR(1)
string	java.lang.String	VARCHAR
byte	byte or java.lang.Byte	TINYINT
boolean	boolean or java.lang.Boolean	BIT
yes/no	boolean or java.lang.Boolean	CHAR(1) ('Y' or 'N')
true/false	boolean or java.lang.Boolean	CHAR(1) ('T' or 'F')

virtusa

Data types - Date

Mapping type	Java type	ANSI SQL Type
date	java.util.Date or java.sql.Date	DATE
time	java.util.Date or java.sql.Time	TIME
timestamp	java.util.Date or java.sql.Timestamp	TIMESTAMP
calendar	java.util.Calendar	TIMESTAMP
calendar_date	java.util.Calendar	DATE

virtusa

Data types - BLOBS

Mapping type	Java type	ANSI SQL Type
binary	byte[]	VARBINARY (or BLOB)
text	java.lang.String	CLOB
serializable	any Java class that implements java.io.Serializable	VARBINARY (or BLOB)
clob	java.sql.Clob	CLOB
blob	java.sql.Blob	BLOB

virtusa

Data types – JDK related

Mapping type	Java type	ANSI SQL Type
class	java.lang.Class	VARCHAR
locale	java.util.Locale	VARCHAR
timezone	java.util.TimeZone	VARCHAR
currency	java.util.Currency	VARCHAR

virtusa

Mapping techniques

- You can either first develop your java classes and then design database.
- you can use database tables and then convert those in to mapping files.

virtusa

Create java class

- Create new class on the project
- Package : com.virtusa.training.hibernate.objects;
- Class name: Employee

```java
public class Employee {
    private int empid;
    private String name;
    private int age;
    private String city;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public int getEmpid() {
        return empid;
    }

    public void setEmpid(int empid) {
        this.empid = empid;
    }
}
```

virtusa

Create table in mysql database using Workbench

- ```
  CREATE TABLE `employee`
  (  `empid` int(11) NOT NULL,
   `Name` varchar(100) NOT NULL,
   `Age` int(11) NOT NULL DEFAULT '18',
   `City` varchar(45) NOT NULL,
   PRIMARY KEY (`empid`))
  ```

virtusa
  ```

Create mapping file

- We can generate mapping file using hibernate perspective
- Right click on project → New → Hibernate XML Mapping file
- Remove packages and add you class to here
- Click finish
- It will create a mapping xml file and you can edit it using UI or source

virtusa

```xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated Nov 20, 2013 1:53:50 PM by Hibernate Tools 3.4.0.CR1 -->
<hibernate-mapping>
    <class name="com.virtusa.training.hibernate.objects.Employee" table="EMPLOYEE">
        <id name="empid" type="int">
            <column name="EMPID" />
            <generator class="assigned" />
        </id>
        <property name="name" type="java.lang.String">
            <column name="NAME" />
        </property>
        <property name="age" type="int">
            <column name="AGE" />
        </property>
        <property name="city" type="java.lang.String">
            <column name="CITY" />
        </property>
    </class>
</hibernate-mapping>
```

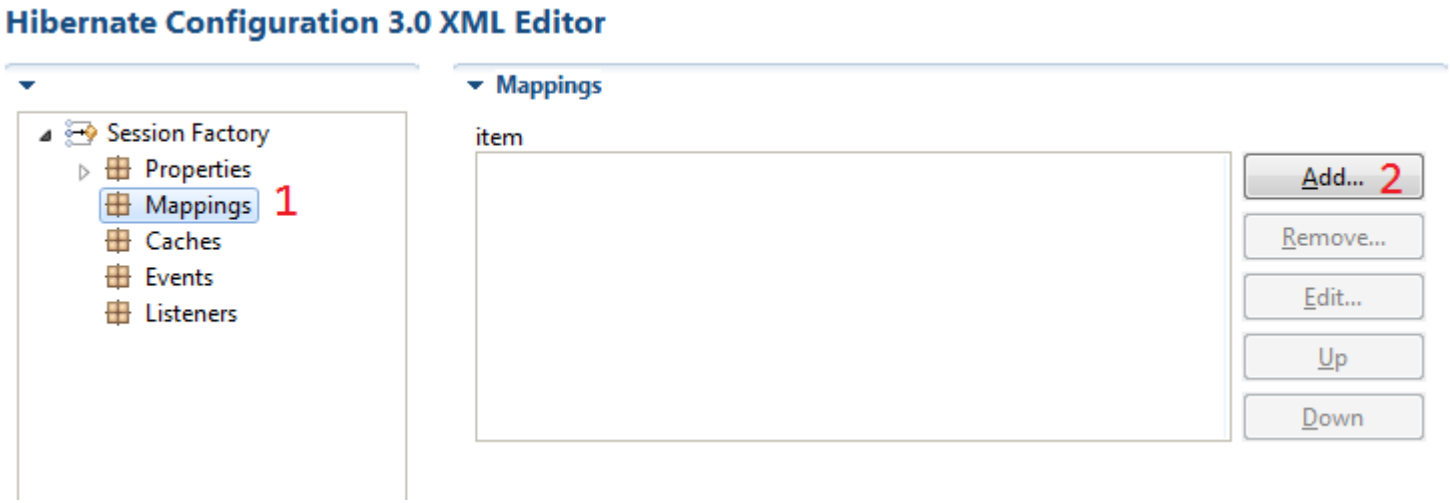
Edit mapping file is possible !!!

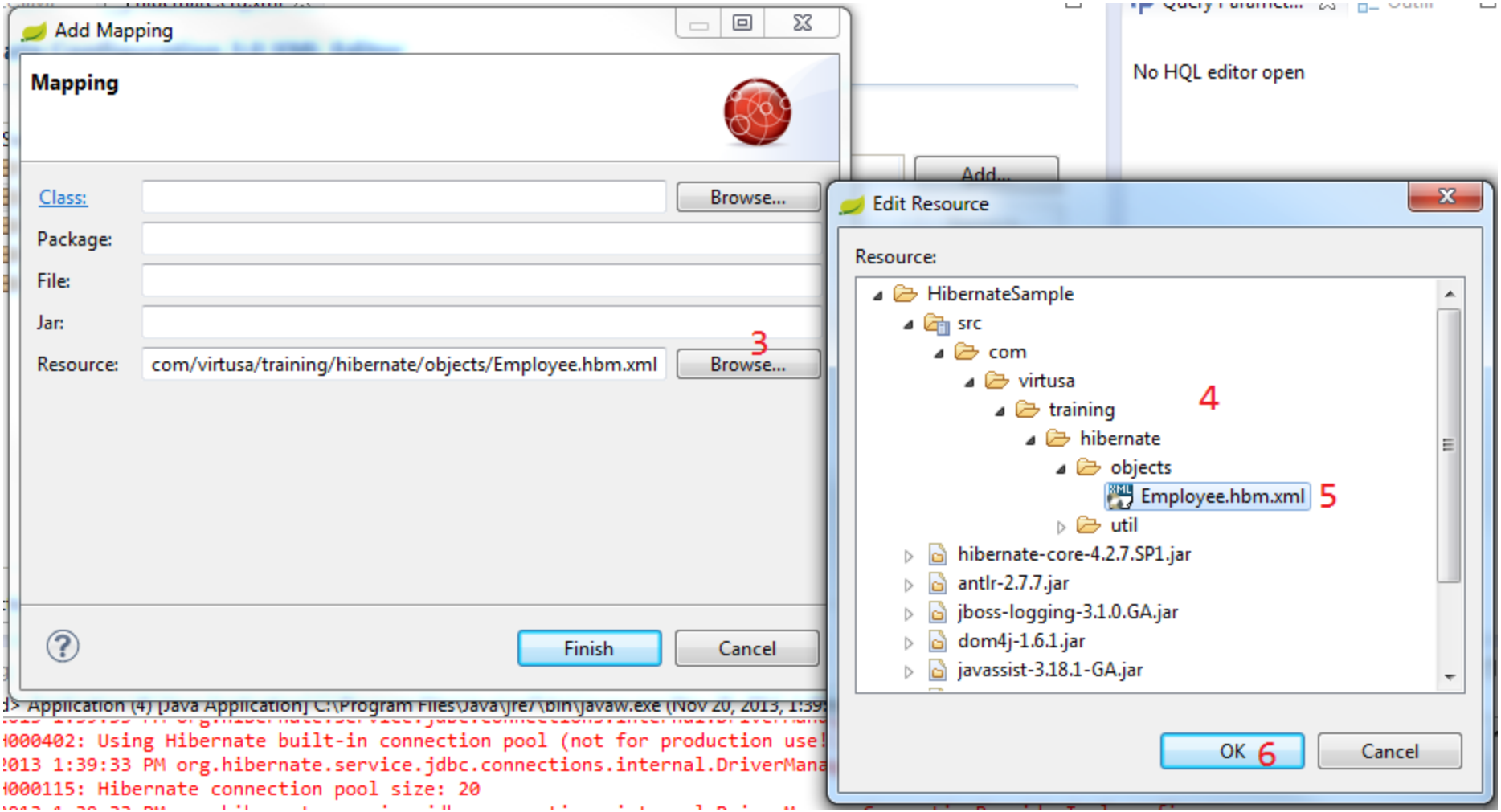
- Lets assume empid is a column which is increment automatically.
- To meet that we can change the generator to “identity”
- Make sure you change the database as well
- `ALTER TABLE `hibernate_sample`.`employee` CHANGE COLUMN `empid` `empid` INT(11) NOT NULL AUTO_INCREMENT ;`

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated Nov 20, 2013 1:53:50 PM by Hibernate Tools 3.4.0.CR1 -->
<hibernate-mapping>
    <class name="com.virtusa.training.hibernate.objects.Employee" table="EMPLOYEE">
        <id name="empid" type="int">
            <column name="EMPID" />
            <generator class="increment" />
        </id>
        <property name="name" type="java.lang.String">
            <column name="NAME" />
        </property>
        <property name="age" type="int">
            <column name="AGE" />
        </property>
        <property name="city" type="java.lang.String">
            <column name="CITY" />
        </property>
    </class>
</hibernate-mapping>
```

Add mapping to configuration

- Go to configuration file and navigate to mapping section. And add mapping file





Step in to new ERA of Data saving

- Make sure remove name section from configuration file
- Edit application.java as file
- Run the program

```
public class Application {

    public static void main(String[] args) {
        Session session = HibernateUtilities.getSessionFactory().openSession();

        session.beginTransaction();
        Employee employee = new Employee();
        employee.setName("Vicky");
        employee.setAge(45);
        employee.setCity("Orian City");
        session.save(employee);
        session.getTransaction().commit();
        session.close();
        HibernateUtilities.getSessionFactory().close();
    }

}
```

Its worked!!!

```
1  SELECT * FROM hibernate_sample.employee;
```

Result Set Filter:

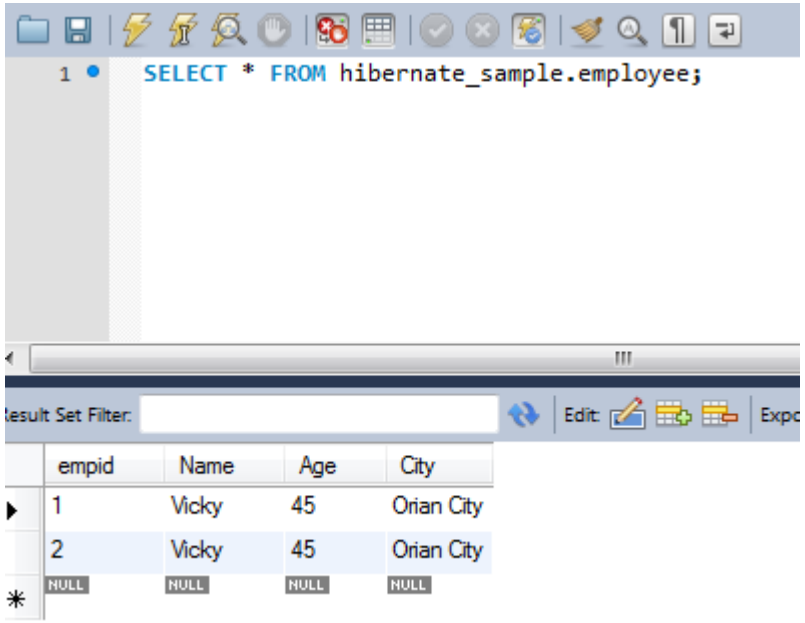
Edit:

empid	Name	Age	City
1	Vicky	45	Orian City
NULL	NULL	NULL	NULL

virtusa

Experiment

- Remove type from mapping file
- <property name="name" ~~type="java.lang.String"> <column name="NAME"~~ />
- Now it will look like
- `<property name="name" />`
- Execute the program and see



Generate tables through hibernate

- This is make life easy as much as DANGEROUS.
- Validate
  - Validates the existing schema with the current entities configuration. When using this mode Hibernate will not do any changes to the schema and will not use the import.sql file.
- Update
  - Hibernate creates an update script trying to update the database structure to the current mapping. Does not read and invoke the SQL statements from import.sql. Useful, but we have to be careful, not all of the updates can be done performed ? for example adding a not null column to a table with existing data.

virtusa

- Create
  - Hibernate will create the database when the Hibernate's SessionFactory is created by the entity manager factory). If a file named import.sql exists in the root of the class path ('/import.sql') Hibernate will execute the SQL statements read from the file after the creation of the database schema. It is important to remember that before Hibernate creates the schema it empties it (delete all tables, constraints, or any other database object that is going to be created in the process of building the schema).
- Create drop
  - Same as 'create' but when the entity manager factory (which holds the SessionFactory) is explicitly closed the schema will be dropped.

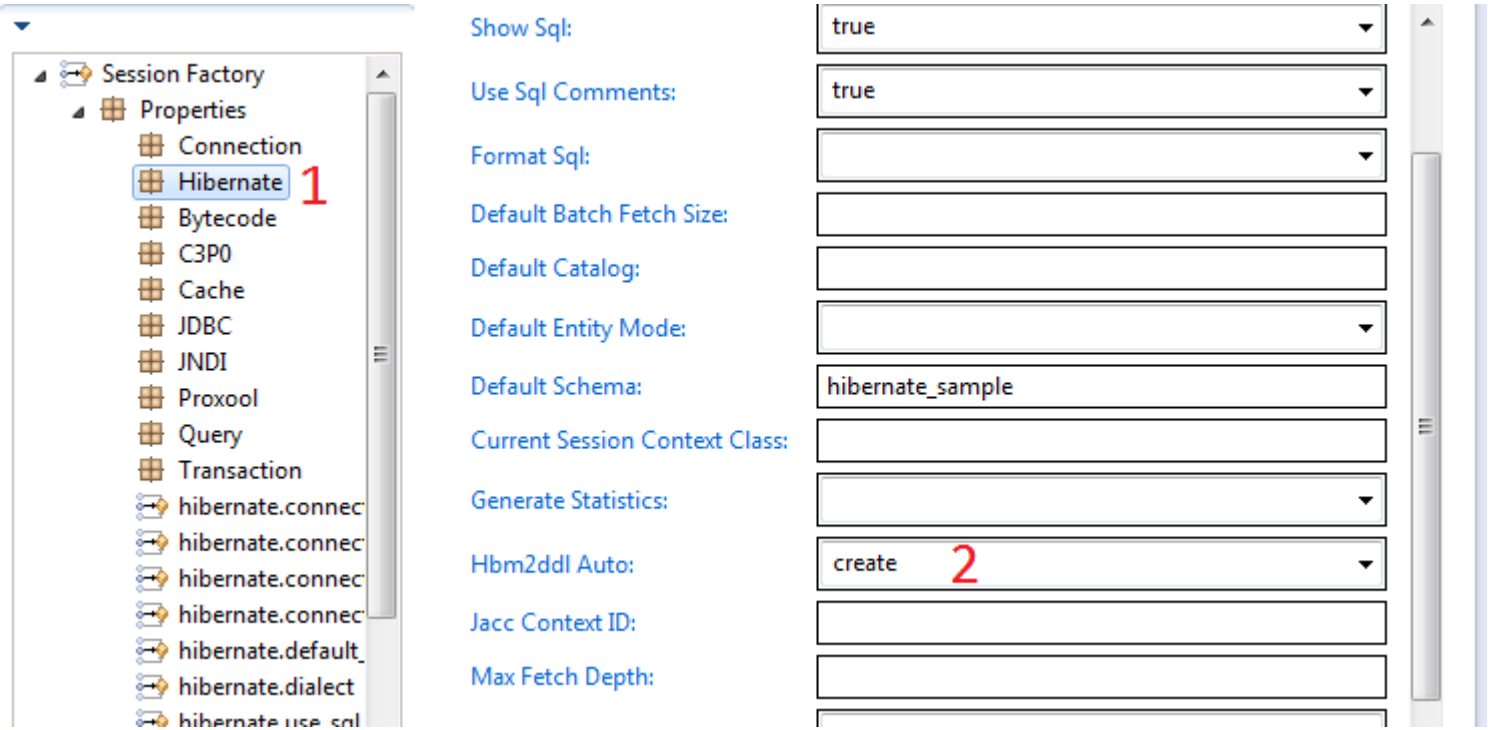
virtusa

All-in-one

Mode	Read import.sql	Alter db structure	comments
Update	NO	YES	
Create	YES	YES	Empty the db before create
Create-drop	YES	YES	Drop when session factory is closed
Validate	NO	NO	

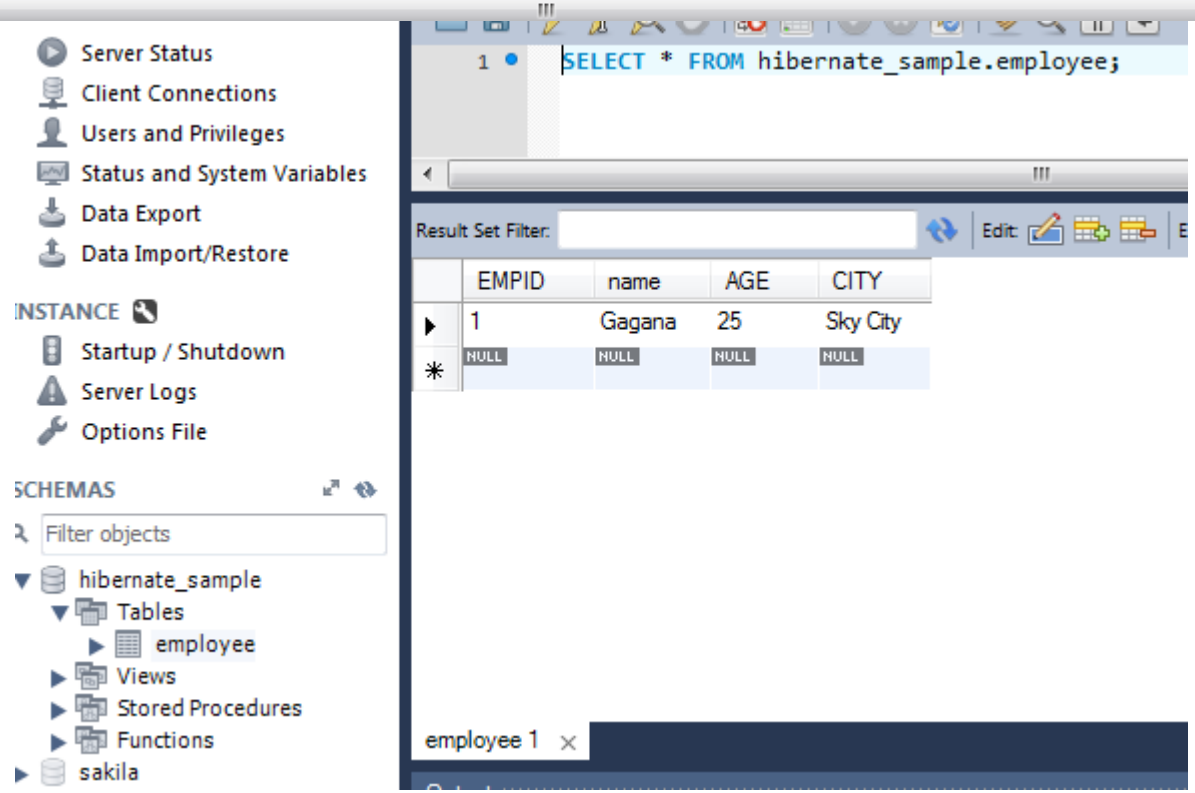
virtusa

- Drop the employee table from database
  - `DROP TABLE `hibernate_sample`.`employee`;`
- Go to configuration UI and change the settings under hibernate section



Execute the program

```
Nov 20, 2013 4:17:55 PM org.hibernate.hql.internal.ast.ASTQueryTranslatorFactory <init>
INFO: HHH000397: Using ASTQueryTranslatorFactory
Nov 20, 2013 4:17:55 PM org.hibernate.tool.hbm2ddl.SchemaExport execute
INFO: HHH000227: Running hbm2ddl schema export
Hibernate: drop table if exists hibernate_sample.EMPLOYEE
Hibernate: create table hibernate_sample.EMPLOYEE (EMPID integer not null, name varchar(255), AGE integer, CITY varchar(255), primary key (
Nov 20, 2013 4:17:56 PM org.hibernate.tool.hbm2ddl.SchemaExport execute
INFO: HHH000230: Schema export complete
Hibernate: select max(EMPID) from hibernate_sample.EMPLOYEE
Hibernate: /* insert com.virtusa.training.hibernate.objects.Employee */ insert into hibernate_sample.EMPLOYEE (name, AGE, CITY, EMPID) valu
```



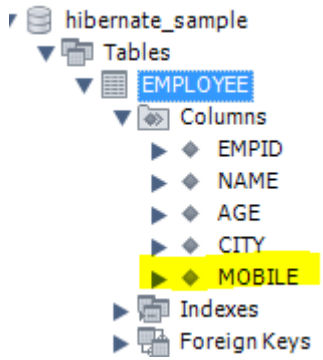
```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0/
                                         "http://hibernate.sourceforge.net/hibernate-co
<hibernate-configuration>
 <session-factory >
  <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
  <property name="hibernate.connection.password">dbsys</property>
  <property name="hibernate.connection.url">jdbc:mysql://localhost:3306</property>
  <property name="hibernate.connection.username">system</property>
  <property name="hibernate.default_schema">hibernate_sample</property>
  <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
  <property name="hibernate.use_sql_comments">true</property>
  <property name="hibernate.show_sql">true</property>
  <property name="hibernate.hbm2ddl.auto">create</property>
  <mapping resource="com/virtusa/training/hibernate/objects/Employee.hbm.xml"/>
 </session-factory>
</hibernate-configuration>
```

How update is work

- Change the class and mapping file accordingly

```java
public class Employee {
    private int empid;
    private String name;
    private int age;
    private String city;
    private String mobile;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
```

```xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated Nov 20, 2013 5:07:52 PM by Hibernate Tools 3.4.0.CR1 -->
<hibernate-mapping>
    <class name="com.virtusa.training.hibernate.objects.Employee" table="EMPLOYEE">
        <id name="empid" type="int">
            <column name="EMPID" />
            <generator class="identity" />
        </id>
        <property name="name" type="java.lang.String">
            <column name="NAME" />
        </property>
        <property name="age" type="int">
            <column name="AGE" />
        </property>
        <property name="city" type="java.lang.String">
            <column name="CITY" />
        </property>
          <property name="mobile" type="java.lang.String">
            <column name="MOBILE" />
        </property>

    </class>
</hibernate-mapping>
```



Load data from database

- Change the Application.java as follows

```
public static void main(String[] args) {
    //saving data
    saveData();

}

static void saveData() {
    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();
    Employee employee = new Employee();
    employee.setName("Yamuna");
    employee.setAge(25);
    employee.setCity("Sky City");
    session.save(employee);
    session.getTransaction().commit();
    session.close();
    HibernateUtilities.getSessionFactory().close();
}
```

- Implement new method to load data.

```
static void loadData() {
    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();
    Employee employee = (Employee) session.get(Employee.class, 1);
    System.out.println(employee.getName() + "-" + employee.getMobile());
    session.getTransaction().commit();
    session.close();
    HibernateUtilities.getSessionFactory().close();
}
```

- You can use LOAD and GET methods for load data
- If you use GET for non exists key it will return null. If use load it will return exception

virtusa

Auto update

- Hibernate have capability of implicit update over get

```
    static void loadData() {
        Session session = HibernateUtilities.getSessionFactory().openSession();
        session.beginTransaction();
        Employee employee = (Employee) session.get(Employee.class, 2);
        System.out.println(employee.getName() + "-" + employee.getAge());
        employee.setAge(employee.getAge()+5);
        session.getTransaction().commit();
        session.close();

    }
}
```

Error Log | Hibernate Query Result | Hibernate Dynamic SQL Preview | Console

```
<terminated> Application (4) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Nov 21, 2013, 2:33:56 P
Nov 21, 2013 2:33:57 PM org.hibernate.tool.hbm2ddl.TableMetadata <init>
INFO: HHH000126: Indexes: [primary]
Nov 21, 2013 2:33:57 PM org.hibernate.tool.hbm2ddl.SchemaUpdate execute
INFO: HHH000232: Schema update complete
Hibernate: /* insert com.virtusa.training.hibernate.objects.Employee */ insert into h
Hibernate: /* load com.virtusa.training.hibernate.objects.Employee */ select employee
Gagana-25
Hibernate: /* update com.virtusa.training.hibernate.objects.Employee */ update hiberr
Hibernate: /* load com.virtusa.training.hibernate.objects.Employee */ select employee
Gagana-30
```

Working with Relationship ☺

virtusa

Value type vs Entities

- Any data will fall under either value type or entity type
- Eye and head… can eye exist with out head… of course yes.. But no purpose. If data does not have mean without its context its call value type

virtusa

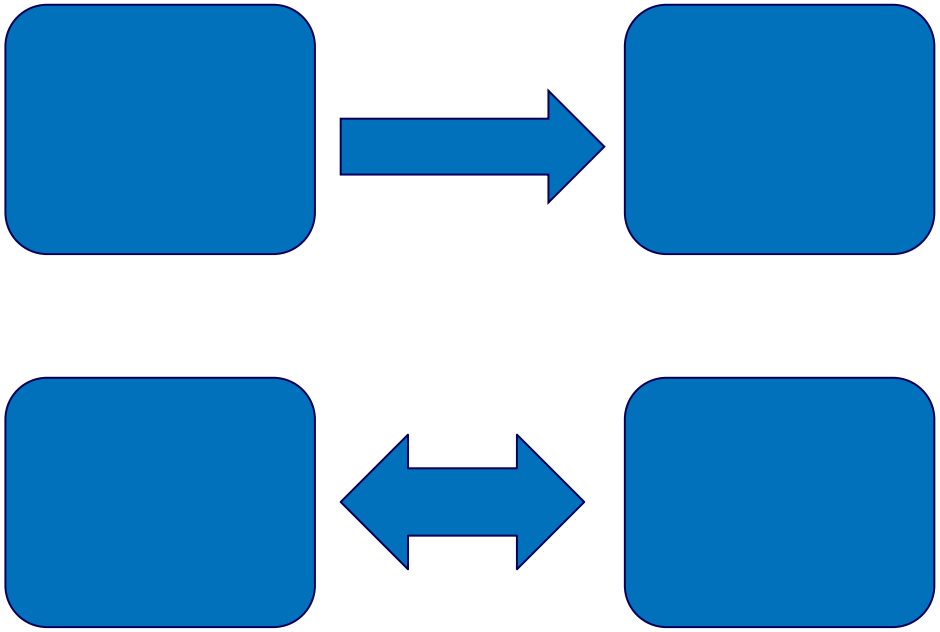
Entity type

- Girl has a car. But car can exist without girl. Mean no requirement of context to have a mean for car
- Entity mean data which can stand it own can be reference outside of context with other.

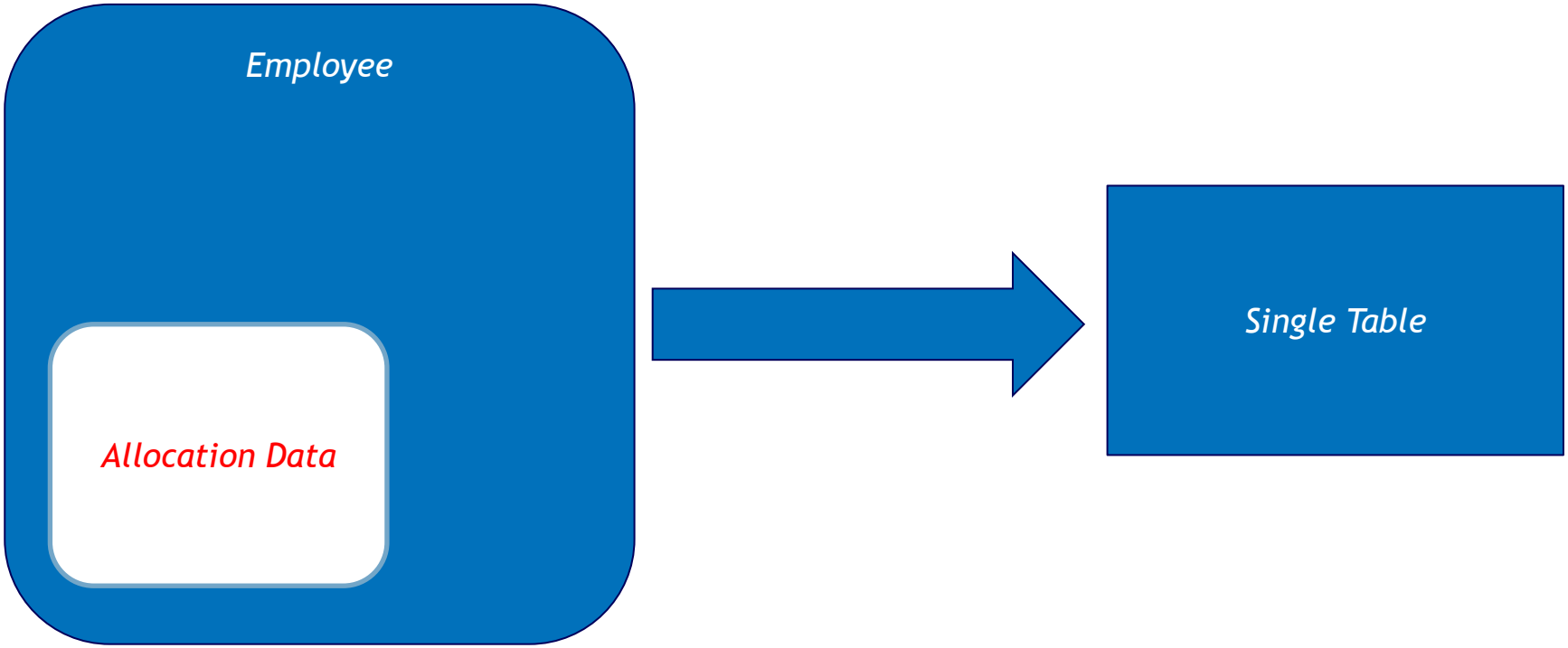
virtusa

Directionality

- Relationship can be uni directional or bi-directional.
- Uni-directional has reference to one peace of data. But that data not reference back.
- Bi-directional mean one data refer to other and its reference back



Component



- Component type just a java composition relationship. Compose one object using other objects which treat as value type

Convert project to composition

```java
public class Employee {
    private int empid;
    private String name;
    private int age;

    private AllocationData allocationData=new AllocationData();

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getEmpid() {
        return empid;
    }

    public void setEmpid(int empid) {
        this.empid = empid;
    }

    public AllocationData getAllocationData() {
        return allocationData;
    }

    public void setAllocationData(AllocationData allocationData) {
        this.allocationData = allocationData;
    }
}
```

```java
public class AllocationData {
    private String city;
    private String mobile;
    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }
    public String getMobile() {
        return mobile;
    }

    public void setMobile(String mobile) {
        this.mobile = mobile;
    }
}
```

virtusa

@ Application.java

```
Employee employee = new Employee();
employee.setName("Tharaka");
employee.setAge(25);
employee.getAllocationData().setCity("Sky City");
employee.getAllocationData().setMobile("07145XXXXX");
session.save(employee);
session.getTransaction().commit();
session.close();

}
```

Change the mapping file

- UI does not work well for this mapping. Therefore it is easy to change mapping file manually.
- And execute the program. You can see both classes save to one table

```
<component name="AllocationData">
    <property name="city" type="java.lang.String">
        <column name="CITY" />
    </property>
    <property name="mobile" type="java.lang.String">
        <column name="MOBILE" />
    </property>
</component>
```

virtusa

Value type collection

- When your application has value type reference list you can use it as collection in hibernate

```
private int empid;
private String name;
private int age;

private AllocationData allocationData=new AllocationData();
Set<AllocationHistory> allocationHistory = new HashSet<>();

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public int getAge() {
```

virtusa

Create constructor

```java
package com.virtusa.training.hibernate.objects;

import java.util.Date;

public class AllocationHistory {
    private Date allocationDate;
    private String projectCode;

    public AllocationHistory(){

    }

    public AllocationHistory(Date allocationDate, String projectCode) {
        super();
        this.allocationDate = allocationDate;
        this.projectCode=projectCode;
    }

    public String getProjectCode() {
        return projectCode;
    }
    public void setProjectCode(String projectCode) {
        this.projectCode = projectCode;
    }
    public Date getAllocationDate() {
        return allocationDate;
    }
    public void setAllocationDate(Date allocationDate) {
        this.allocationDate = allocationDate;
    }

}
```

Hash code

- Since we do not have id column there should be a way to hibernate to know uniqueness.
- r/click on class → source → generate hash code and equals

```
@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result
            + ((allocationDate == null) ? 0 : allocationDate.hashCode());
    result = prime * result
            + ((projectCode == null) ? 0 : projectCode.hashCode());
    return result;
}

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    AllocationHistory other = (AllocationHistory) obj;
    if (allocationDate == null) {
        if (other.allocationDate != null)
            return false;
```

virtusa

- @Employee class create getters and setters for allocation history
- Go to application class and add allocation history
- Refer Next page

virtusa

```java
    static void saveData() {
        Session session = HibernateUtilities.getSessionFactory().openSession();
        session.beginTransaction();
        Employee employee = new Employee();
        employee.getAllocationHistory().add(new AllocationHistory(new Date(), "Project 1"));
        employee.setName("Nikman");
        employee.setAge(25);
        employee.getAllocationData().setCity("This City");
        employee.getAllocationData().setMobile("0714XXXX78");
        employee.getAllocationHistory().add(new AllocationHistory(new Date(), "Project 2"));
        session.save(employee);
        session.getTransaction().commit();
        session.close();

    }
    static void loadData() {
        Session session = HibernateUtilities.getSessionFactory().openSession();

        session.beginTransaction();
        Employee employee = (Employee) session.get(Employee.class, 2);
        employee.getAllocationHistory().add(new AllocationHistory(new Date(), "Project 3"));

        System.out.println(employee.getName() + "-" + employee.getAge());
        employee.setAge(employee.getAge() + 5);
        employee.getAllocationHistory().add(new AllocationHistory(new Date(), "Project 4"));

        for (AllocationHistory allocationHistory : employee.getAllocationHistory()) {
            System.out.println(allocationHistory.getAllocationDate() + "#"
                    + allocationHistory.getProjectCode());
        }

        session.getTransaction().commit();
        session.close();

    }
}
```

Finalize the mapping

- @ mapping class

```
</component>
<set name="allocation" table="EMPLOYEE_HISTORY">
    <key column="ID" />
    <composite-element
        class="com.virtusa.training.hibernate.objects.AllocationHistory">
        <property name="allocationDate" type="date" column="ALLOCATION_DATE" />
        <property name="projectCode" type="string" column="PROJECT_CODE" />
    </composite-element>
</set>
```

```
public class AllocationHistory {
    private Date allocationDate;
    private String projectCode;

    public AllocationHistory(){

    }

    public AllocationHistory(Date allocationDate,
        super();
        this.allocationDate = allocationDate;
```

```
public class Employee {
    private int empid;
    private String name;
    private int age;

    private AllocationData allocationData=new AllocationData();
    private Set<AllocationHistory> allocation = new HashSet<>();

    public String getName() {
        return name;
    }
```

Convert program to List instead of set

- It is just converting set to list and hashset to ArrayList
- But need to do small change on mapping file

```
<list name="allocation" table="EMPLOYEE_HISTORY">
<key column="ID" />
<list-index column="ROWID"/>
    <composite-element
        class="com.virtusa.training.hibernate.objects.AllocationHistory">
        <property name="allocationDate" type="date" column="ALLOCATION_DATE" />
        <property name="projectCode" type="string" column="PROJECT_CODE" />
    </composite-element>
</list>
```

Convert program to Collection

- @ mapping use idbag. It bag is a container which can hold things. They can have duplicate but don't have an order

```java
import java.util.ArrayList;
import java.util.Collection;

public class Employee {
    private int empid;
    private String name;
    private int age;

    private AllocationData allocationData=new AllocationData();
    private Collection<AllocationHistory> allocation = new ArrayList<>();

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
```

```java
        </property>
    </component>
    <idbag name="allocation" table="EMPLOYEE_HISTORY">
        <collection-id type="int" column="id">
            <generator class="increment"></generator>
        </collection-id>
        <key column="ID" />
        <composite-element
            class="com.virtusa.training.hibernate.objects.AllocationHistory">
            <property name="allocationDate" type="date" column="ALLOCATION_DATE" />
            <property name="projectCode" type="string" column="PROJECT_CODE" />
        </composite-element>
    </idbag>
</class>
```

Mapping Entity type relationships

```java
public class AllocationHistory {
    private Date allocationDate;
    private String projectCode;

    private int employeeid;
    private Employee employee;

    public AllocationHistory() {

    }

    public AllocationHistory(Date allocationDate, String projectCode) {
        super();
        this.allocationDate = allocationDate;
        this.projectCode = projectCode;
    }

    public String getProjectCode() {
        return projectCode;
    }

    public void setProjectCode(String projectCode) {
        this.projectCode = projectCode;
    }

    public Date getAllocationDate() {
        return allocationDate;
    }

    public void setAllocationDate(Date allocationDate) {
        this.allocationDate = allocationDate;
    }

    public int getEmployeeid() {
        return employeeid;
    }
```

- Developers really missing this. In order to work relationship In both way we must have this method

```
import java.util.ArrayList;

public class Employee {
    private int empid;
    private String name;
    private int age;

    private AllocationData allocationData=new AllocationData();
    private List<AllocationHistory> allocation = new ArrayList<>();

    public void addAllocationHistory(AllocationHistory allocationHistory){
        allocationHistory.setEmployee(this);
        allocation.add(allocationHistory);

    }

    public String getName() {
        return name;
    }
```

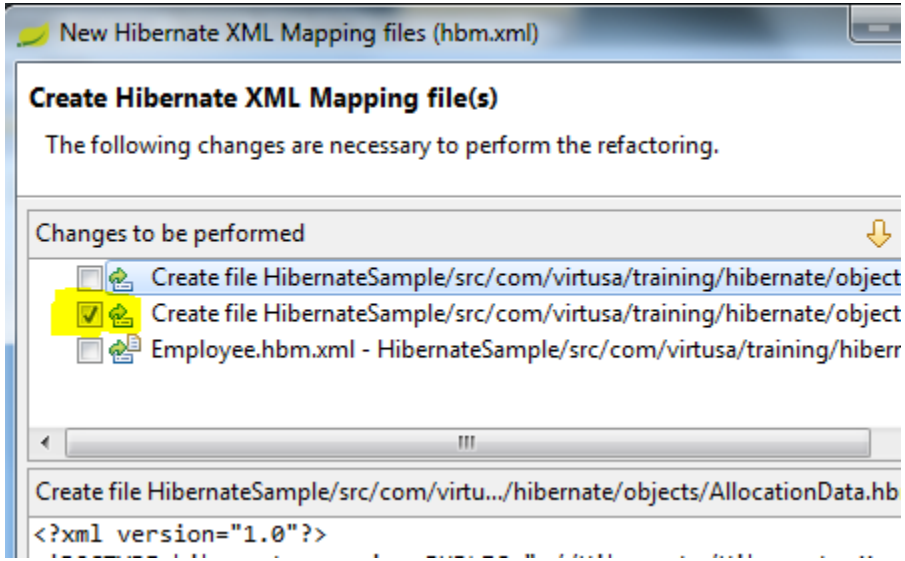
Create other reference at Employee class

```
public void addAllocationHistory(AllocationHistory allocationHistory){
    allocationHistory.setEmployee(this);
    allocation.add(allocationHistory);

}
```

- On application change `employee.getAllocation().add(new AllocationHistory(new Date(), "Project 2"));` as follows

- `employee.addAllocationHistory(new AllocationHistory(new Date(),"Project 2"));`

Create mapping file to allocation history



```xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated Nov 25, 2013 11:28:12 PM by Hibernate Tools 3.4.0.CR1 -->
<hibernate-mapping>
    <class name="com.virtusa.training.hibernate.objects.AllocationHistory" table="ALLOCATIONHISTORY">
        <id name="employeeid" type="int">
            <column name="EMPLOYEEID" />
            <generator class="increment" />
        </id>
        <property name="allocationDate" type="java.util.Date">
            <column name="ALLOCATIONDATE" />
        </property>
        <property name="projectCode" type="java.lang.String">
            <column name="PROJECTCODE" />
        </property>
        <many-to-one name="employee" class="com.virtusa.training.hibernate.objects.Employee" not-null="true">
            <column name="EMPLOYEE" />
        </many-to-one>
    </class>
</hibernate-mapping>
```

Change the employee mapping file

- Need to add inverse =true in order to tell hibernate that this is bi-directional relationship use inverse key word for that

```
<list name="allocation" table="EMPLOYEE_HISTORY" inverse="true">
    <key column="empid" />
    <list-index column="ROWID"/>
    <one-to-many class="com.virtusa.training.hibernate.objects.AllocationHistory"/>
</list>
```

Configure to save together

- Now application has two entities need to work together.
- Instead of managing separately we can use cascade to “save update”
- Also update the mapping section of config file and execute the program

```
        </property>
    </component>
    <list name="allocation" table="EMPLOYEE_HISTORY" inverse="true" cascade="save-update">
        <key column="empid" />
        <list-index column="ROWID"/>
        <one-to-many class="com.virtusa.training.hibernate.objects.AllocationHistory"/>
    </list>
</class>
```

Result Set Filter: | Edit: | Export/Import: | Wrap Cell Content:

EMPLOYEEID	ALLOCATIONDATE	PROJECTCODE	EMPLOYEE	empid	ROWID
1	2013-11-26 00:54:56	Project 1	2	NULL	
2	2013-11-26 00:54:56	Project 2	NULL	NULL	
3	2013-11-26 00:54:56	Project 3	1	NULL	
4	2013-11-26 00:54:56	Project 4	1	NULL	
NULL	NULL	NULL	NULL	NULL	NULL

virtusa

How to fix it ?

- @ Employee mapping file

```
</component>
<list name="allocation" table="EMPLOYEE_HISTORY" inverse="true" cascade="save-update">
    <key column="empid" />
    <list-index column="ROWID"/>
    <one-to-many class="com.virtusa.training.hibernate.objects.AllocationHistory"/>
</list>
```

- @ Allocation history mapping file

```
    </property>
    <many-to-one name="employee" class="com.virtusa.training.hibernate.objects.Employee" not-null="true">
        <column name="EMPLOYEE" />
    </many-to-one>
</class>
```

Change in to…

```
        </property>
    </component>
    <list name="allocation" table="EMPLOYEE_HISTORY" inverse="true" cascade="save-update">
        <key column="EMPID" />
        <list-index column="ROWID"/>
        <one-to-many class="com.virtusa.training.hibernate.objects.AllocationHistory"/>
    </list>
</class>
```

```
        <column name="PROJECTCODE" />
    </property>
    <many-to-one name="employee" class="com.virtusa.training.hibernate.objects.Employee" not-null="true">
        <column name="EMPID" />
    </many-to-one>
</class>
</hibernate-mapping>
```

Result Set Filter: | Edit: | Export/Import: | W

EMPLOYEEID	ALLOCATIONDATE	PROJECTCODE	EMPID	ROWID
1	2013-11-26 01:15:18	Project 3	1	0
2	2013-11-26 01:15:18	Project 4	1	1
NULL	NULL	NULL	NULL	NULL

virtusa

One to one mapping

- Make Allocation Data as entity
- Add following variable and getter-setter for those
  - `private int id;`
  - `private Employee employee;`

Change the Employee class as

```
public Employee(){
    setAllocationData(new AllocationData());
}

public void setAllocationData(AllocationData allocationData) {
    this.allocationData = allocationData;
    allocationData.setEmployee(this);
}
```

Change mapping

- Generate mapping for allocation data
- Edit it as follows

```
<hibernate-mapping>
    <class name="com.virtusa.training.hibernate.objects.AllocationData"
        table="ALLOCATIONDATA">
        <id name="id" type="int">
            <column name="ID" />
            <generator class="foreign">
                <param name="property">employee</param>
            </generator>
        </id>
        <property name="city" type="java.lang.String">
            <column name="CITY" />
        </property>
        <property name="mobile" type="java.lang.String">
            <column name="MOBILE" />
        </property>
        <one-to-one name="employee"
            class="com.virtusa.training.hibernate.objects.Employee" constrained="true" />
    </class>
</hibernate-mapping>
```

to depend with forign key

to validate as one object

Edit the Employee Mapping

- You can remove component section and add one-to-one
- Add to mapping and execute

```
<hibernate-mapping>
    <class name="com.virtusa.training.hibernate.objects.Employee"
        table="EMPLOYEE">
        <id name="empid" type="int">
            <column name="EMPID" />
            <generator class="identity" />
        </id>
        <property name="name" type="java.lang.String">
            <column name="NAME" />
        </property>
        <property name="age" type="int">
            <column name="AGE" />
        </property>
        <!-- <component name="AllocationData">
            <property name="city" type="java.lang.String">
                <column name="CITY" />
            </property>
            <property name="mobile" type="java.lang.String">
                <column name="MOBILE" />
            </property>
        </component> -->
        <one-to-one name="AllocationData" class="com.virtusa.training.hibernate.objects.AllocationData" cascade="save-update"/>

        <list name="allocation" table="EMPLOYEE_HISTORY" inverse="true" cascade="save-update">
            <key column="EMPID" />
            <list-index column="ROWID"/>
            <one-to-many class="com.virtusa.training.hibernate.objects.AllocationHistory"/>
        </list>
    </class>
</hibernate-mapping>
```

How can use real foreign key

- Change one to one mapping of employee to many-to-many as follows
  - ```
    <many-to-one name="AllocationData"
    class="com.virtusa.training.hibernate.objects.AllocationData"
    column="ALLOCATION_ID" cascade="save-update" unique="true"/>
    ```

- With set unique=true we can create one to one mapping over many to one as foreign table must have unique key and change allocation data mapping as follows

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated Nov 26, 2013 4:50:30 PM by Hibernate Tools 3.4.0.CR1 -->
<hibernate-mapping>
    <class name="com.virtusa.training.hibernate.objects.AllocationData"
        table="ALLOCATIONDATA">
        <id name="id" type="int">
            <column name="ID" />
            <generator class="increment" />
        </id>
        <property name="city" type="java.lang.String">
            <column name="CITY" />
        </property>
        <property name="mobile" type="java.lang.String">
            <column name="MOBILE" />
        </property>
        <one-to-one name="employee"
            class="com.virtusa.training.hibernate.objects.Employee"  constrained="true"
                property-ref="allocationData" />
    </class>
</hibernate-mapping>
```
```

```java
public class Employee {
    private int empid;

    private int age;
    private String name;

    private AllocationData allocationData ;//= new AllocationData();
    private List<AllocationHistory> allocationHistory = new ArrayList<>();

    public void addAllocationHistory(AllocationHistory allocationHistory) {
        this.allocationHistory.add(allocationHistory);
        allocationHistory.setEmployee(this);

    }

    public void setAllocationData(AllocationData allocationData) {
        this.allocationData = allocationData;
        allocationData.setEmployee(this);

    }
```

Join Tables

- Create new class as Alerts and mapping file

```java
public class Alerts {
    private int id;
    private String message;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getMessage() {
        return message;
    }
    public void setMessage(String message) {
        this.message = message;
    }

    public Alerts(String message) {
        super();
        this.message = message;
    }

    public Alerts() {

    }
}
```

```hibernate-mapping
<hibernate-mapping>
    <class name="com.virtusa.training.hibernate.objects.Alerts" table="ALERTS">
        <id name="id" type="int">
            <column name="ID" />
            <generator class="increment" />
        </id>
        <property name="message" type="java.lang.String">
            <column name="MESSAGE" />
        </property>
    </class>
</hibernate-mapping>
```

- Add alert to employee class as
  - `private Alerts alerts;`

```
public Alerts getAlerts() {
return alerts;
}

public void setAlerts(Alerts alerts) {
this.alerts = alerts;
}
```

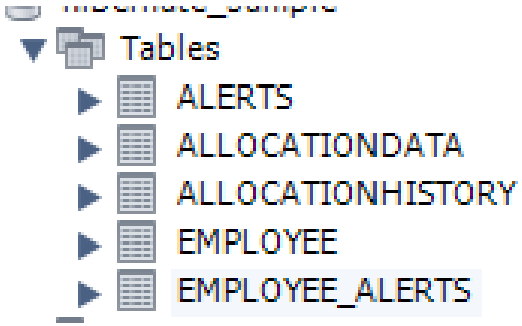
Modify application.java

```
static void saveData() {
    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();
    Employee employee = new Employee();
    employee.setName("Nikman");
    employee.setAge(25);
    employee.getAllocationData().setCity("This City");
    employee.getAllocationData().setMobile("0714XXXX78");

    employee.setAlerts(new Alerts("Congratzzz"));

    session.save(employee);
    session.getTransaction().commit();
    session.close();

}
```

Modify Employee mapping

- Do the mapping file changes and execute the program

```
        class= com.virtusa.training.hibernate.objects.AllocationHi
</list>
<join table="EMPLOYEE_ALERTS" optional="true">
    <key column="EMPID" />
    <many-to-one name="alerts" column="ALERT_ID" not-null="true"
        unique="true" cascade="save-update" />
</join>
```



Many to Many

- Change program to have SET<alerts> and generate getters and setters
- `private  Set<Alerts> alerts = new HashSet<>();`
- Change the appication as
- `employee.getAlerts().add(new Alerts("Congratzzz"));`
- `employee.getAlerts().add(new Alerts("Ready ?"));`
- `employee.getAlerts().add(new Alerts("DId it"));`

virtusa

Change employee mapping

- Do following change and execute program

```
<!-- <join table="EMPLOYEE_ALERTS" optional="true">
    <key column="EMPID" />
    <many-to-one name="alerts" column="ALERT_ID" not-null="true"
        unique="true" cascade="save-update" />
</join> -->
<set name="alerts" table="EMPLOYEE_ALERTS" cascade="save-update">
<key column="EMPID"/>
<many-to-many class="com.virtusa.training.hibernate.objects.Alerts" column="ALERT_ID"/>
</set>
```

ID	MESSAGE
1	Did it
2	Congratzzz
3	Ready ?
NULL	NULL

EMPID	ALERT_ID
1	1
1	2
1	3
NULL	NULL

Querying

Modes

- HQL / JPA QL
- Criteria API
- Native SQL

virtusa

HQL

- Almost like SQL
- If you familiar with SQL it is easy to understand HQL
- It support most features of SQL
- Weakness of HQL is it is difficult to dynamically Construct

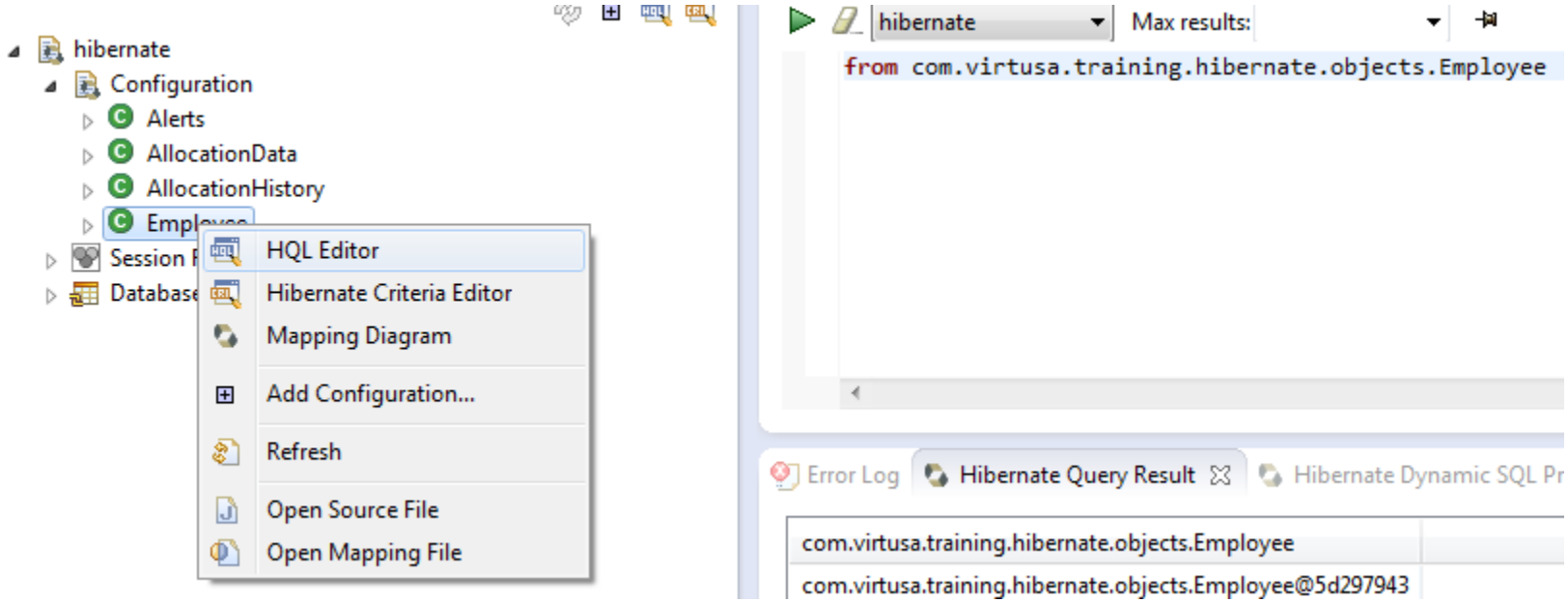
virtusa

Pre-work

- Change program as follows (change object classes accordingly)

```
static void saveData() {
    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();
    Employee saman = new Employee(1, "saman", 22, new AllocationData("col", "01"), new AllocationHistory(new Date(), "p1"), new Alerts("Hi"));
    Employee kamal = new Employee(1, "kamal", 23, new AllocationData("col", "01"), new AllocationHistory(new Date(), "p1"), new Alerts("Good"));
    Employee nimal = new Employee(1, "nimal", 24, new AllocationData("col", "01"), new AllocationHistory(new Date(), "p1"), new Alerts("Bad"));
    Employee sunil = new Employee(1, "sunil", 25, new AllocationData("col", "01"), new AllocationHistory(new Date(), "p1"), new Alerts("Go"));
    Employee ruwan = new Employee(1, "ruwan", 26, new AllocationData("col", "01"), new AllocationHistory(new Date(), "p1"), new Alerts("Come"));
    session.save(saman);
    session.save(kamal);
    session.save(nimal);
    session.save(sunil);
    session.save(ruwan);

    session.getTransaction().commit();
    session.close();

}
```

- On hibernate tab you can expand object and generate HQL



Add query to program

```
static void loadQueryBased() {
    System.out.println("I am in");
    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();
    Query query = session
            .createQuery("from com.virtusa.training.hibernate.objects.Employee");
    List<Employee> employees = query.list();
    for (Employee employee : employees) {
        System.out.println(employee.getName());

    }

    session.getTransaction().commit();
    session.close();
}
```

```
Hibernate: /* load com.
saman
kamal
nimal
sunil
ruwan
```

Fine tune query

```
static void loadQueryBased() {
    System.out.println("I am in");
    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();
    Query query = session
            .createQuery("from Employee");
    List<Employee> employees = query.list();
    for (Employee employee : employees) {
        System.out.println(employee.getName());

    }

    session.getTransaction().commit();
    session.close();
}
```

```
Hibernate: /* loa
saman
kamal
nimal
sunil
ruwan
```

Filter data

```
static void loadQueryBased() {
    System.out.println("I am in");
    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();
    Query query = session
            .createQuery("from Employee employee where employee.age=22");
    List<Employee> employees = query.list();
    for (Employee employee : employees) {
        System.out.println(employee.getName());

    }

    session.getTransaction().commit();
    session.close();
}
```

```
Hibernate: /* 1
Hibernate: /* 1
saman
nimal
```

Fetch Nested Table

```
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();
    Query query = session
            .createQuery("select employee.alerts from Employee employee ");
    List<Alerts> alerts = query.list();

    for (Alerts alerts2 : alerts) {
        System.out.println(alerts2.getMessage());
    }

    session.getTransaction().commit();
    session.close();
}
```

```
Hibernat
Hibernat
Hibernat
Hibernat
Hibernat
Hibernat
Hibernat
Hibernat
Hibernat
Hi
Good
Bad
Go
Come
```

Working with Parameter

```
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();
    Query query = session
            .createQuery("select employee.alerts from Employee employee where employee.age= :age").setInteger("age", 22);
    List<Alerts> alerts = query.list();

    for (Alerts alerts2 : alerts) {
        System.out.println(alerts2.getMessage());
    }

    session.getTransaction().commit();
    session.close();
}
```

```
Hibernate: /* select
Hi
Bad
```

Paging

- You can change setFirstResult to move to next page

```
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();
    Query query = session
            .createQuery("select employee.alerts from Employee employee").setFirstResult(0).setMaxResults(2);

    List<Alerts> alerts = query.list();

    for (Alerts alerts2 : alerts) {
        System.out.println(alerts2.getMessage());
    }

    session.getTransaction().commit();
    session.close();
}
```

Named Query

- Objective is to get query out from code
- Get query out and put in mapping file

```
        <property name="message" type="java.lang.String">
            <column name="MESSAGE" />
        </property>

    </class>
     <query name="getAllAllerts">
        <![CDATA[select employee.alerts from Employee employee]]>
    </query>
</hibernate-mapping>
```

```
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();
    Query query = session.getNamedQuery("getAllAllerts");

    List<Alerts> alerts = query.list();

    for (Alerts alerts2 : alerts) {
        System.out.println(alerts2.getMessage());
    }

    session.getTransaction().commit();
    session.close();
}
```

Joins

- When we use query hibernate it self create joints call implicit joints

```
select employee.alerts from Employee employee where employee.age= 22
```

Error Log | Hibernate Query Result | Hibernate Dynamic SQL Preview | Console

```
0: java.util.Set(com.virtusa.training.hibernate.objects.Employee.alerts)
-----------------
select
  alerts2_.ID as ID4_,
  alerts2_.MESSAGE as MESSAGE4_
 from
  hibernate_sample.EMPLOYEE employee0_
 inner join
  hibernate_sample.EMPLOYEE_ALERTS alerts1_
   on employee0_.EMPID=alerts1_.EMPID
 inner join
  hibernate_sample.ALERTS alerts2_
   on alerts1_.ALERT_ID=alerts2_.ID
 where
  employee0_.AGE=22
```

Explicit join

hibernate Max results:

```
select employee from Employee employee join
employee.allocationData allocation where
allocation.id=1
```

Error Log | Hibernate Query Result | Hibernate Dynamic SQL Preview

```
0: com.virtusa.training.hibernate.objects.Employee
----------------
select
 employee0_.EMPID as EMPID0_,
 employee0_.NAME as NAME0_,
 employee0_.AGE as AGE0_,
 employee0_.ALLOCATION_ID as ALLOCATION4_0_
from
 hibernate_sample.EMPLOYEE employee0_
inner join
 hibernate_sample.ALLOCATIONDATA allocation1_
  on employee0_.ALLOCATION_ID=allocation1_.ID
where
 allocation1_.ID=1
```

Explicit join for two tables

Employee.java | Application.... | AllocationD... | *hibernat

hibernate | Max results:

```
select e from Employee e ,AllocationData al
where e.empid=al.id
```

Error Log | Hibernate Query Result | Hibernate Dynamic SQL Preview

```
0: com.virtusa.training.hibernate.objects.Employee
-----------------
select
 employee0_.EMPID as EMPID0_,
 employee0_.NAME as NAME0_,
 employee0_.AGE as AGE0_,
 employee0_.ALLOCATION_ID as ALLOCATION4_0_
from
 hibernate_sample.EMPLOYEE employee0_ cross
join
 hibernate_sample.ALLOCATIONDATA allocation1_
where
 employee0_.EMPID=allocation1_.ID
```

virtusa

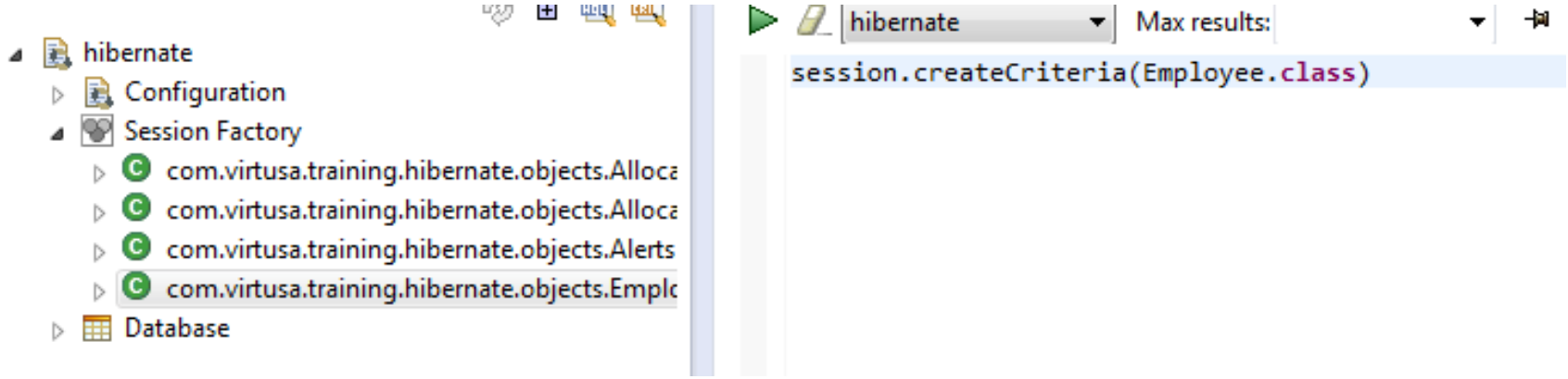
Dynamic Instantiations

```java
package com.virtusa.training.hibernate.objects;

public class ProjectData {
    private String name;
    private String city;

    public ProjectData(String name, String city) {

        this.name = name;
        this.city = city;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }


}
```

```java
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();

    Query query = session
            .createQuery("select new com.virtusa.training.hibernate.objects.ProjectData( e.name,e.allocationData.city) from Employee e");

    List<ProjectData> projectDatas = query.list();

    for (ProjectData projectData : projectDatas) {

        System.out.println(projectData.getName() + "#"
                + projectData.getCity());
    }

    session.getTransaction().commit();
    session.close();
}
```

Criteria

- It is other way to query
- More OO approach
- On hibernate window right click on object and select criteria editor
- NO HQL preview



Use criteria

```
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();

    //Query query = session
        //  .createQuery("select new com.virtusa.training.hibernate.objects.ProjectData

    Criteria criteria = session.createCriteria(Employee.class);

    List<Employee> employees = criteria.list();

    for (Employee employee : employees) {
        System.out.println(employee.getName());
    }

    session.getTransaction().commit();
    session.close();
}
```

Criteria with Restrictions

- Add equal and greater than

```
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();

    Criteria criteria = session.createCriteria(Employee.class);
    criteria.add(Restrictions.eq("name", "saman"));
    criteria.add(Restrictions.gt("age", 20));

    List<Employee> employees = criteria.list();

    for (Employee employee : employees) {
        System.out.println(employee.getName());
    }

    session.getTransaction().commit();
    session.close();
}
```

Criteria with OR

```java
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();

    Criteria criteria = session.createCriteria(Employee.class);
    criteria.add(Restrictions.or(
    Restrictions.eq("name", "saman"),
    Restrictions.gt("age", 24)));

    List<Employee> employees = criteria.list();

    for (Employee employee : employees) {
        System.out.println(employee.getName());
    }

    session.getTransaction().commit();
    session.close();
}
```

Error Log | Hibernate Query Result | Hibernate Dynamic SQL Preview | Console

```
<terminated> Application (4) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Nov 27, 2013, 10:38:53 A
Hibernate: /* load com.virtusa.training.hibernate.objects.AllocationData */ select all
Hibernate: /* load com.virtusa.training.hibernate.objects.Employee */ select employee0_
Hibernate: /* load com.virtusa.training.hibernate.objects.AllocationData */ select all
Hibernate: /* load com.virtusa.training.hibernate.objects.Employee */ select employee0_
saman
sunil
ruwan
```

Criteria with Projection

```
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();

    Criteria criteria = session.createCriteria(Employee.class);
    criteria.add(Restrictions.or(Restrictions.eq("name", "saman"),
            Restrictions.gt("age", 24))).setProjection(Projections.avg("age"));

    System.out.println(criteria.list().size());
    System.out.println(criteria.list().get(0));

    /*List<Employee> employees = criteria.list();

    for (Employee employee : employees) {
        System.out.println(employee.getName());
    }

    session.getTransaction().commit();
    session.close();
}
```

```
Hibernate: /* inse
Hibernate: /* inse
Hibernate: update
Hibernate: /* inse
Hibernate: update
Hibernate: /* inse
Hibernate: update
Hibernate: /* inse
Hibernate: update
Hibernate: /* inse
Hibernate: update
Hibernate: /* inse
Hibernate: /* crit
1
Hibernate: /* crit
27.6667
```

Criteria with Projection List

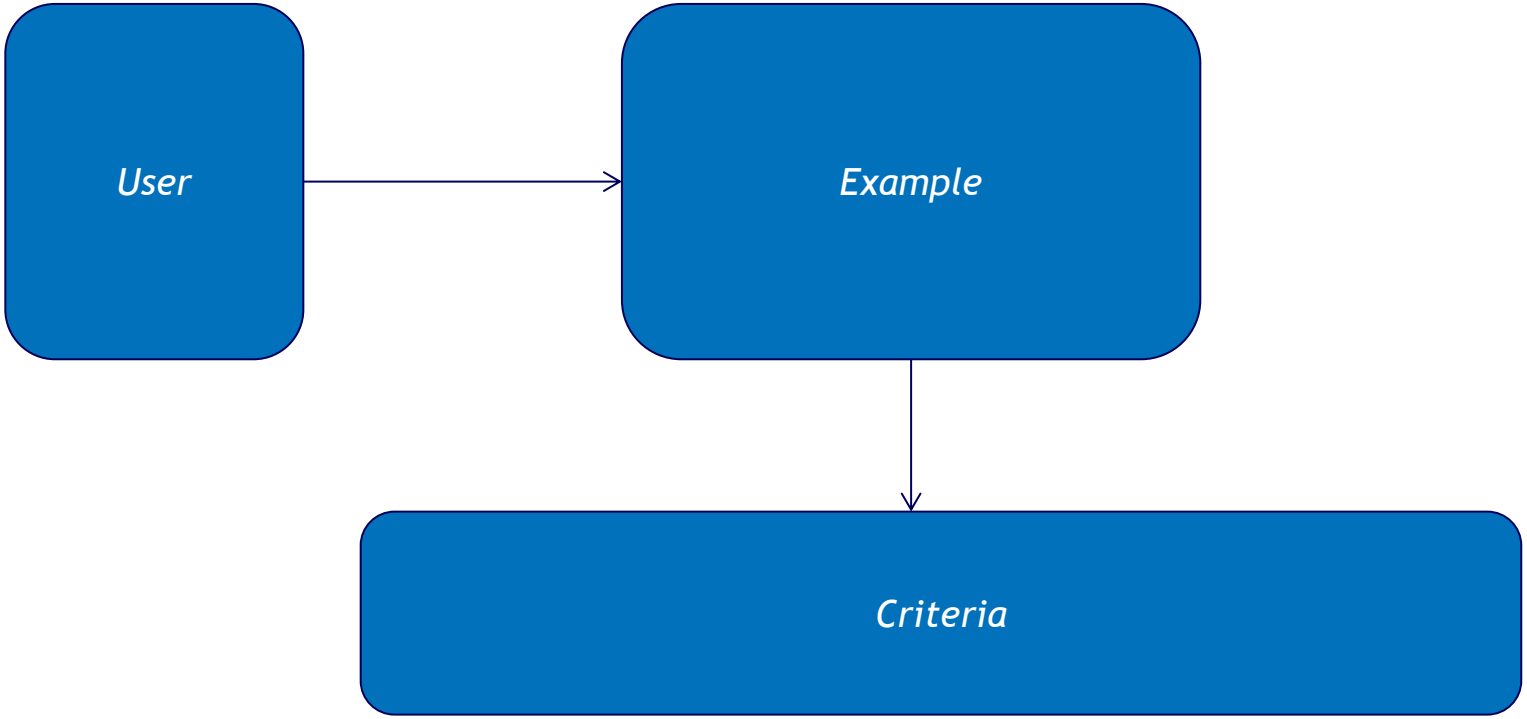
```
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();

    Criteria criteria = session.createCriteria(Employee.class);
    criteria.add(
            Restrictions.or(Restrictions.eq("name", "saman"),
                    Restrictions.gt("age", 24))).setProjection(
            Projections.projectionList().add(Projections.property("name"))
                    .add(Projections.property("empid")));

    List<Object[]> employee = criteria.list();

    for (Object[] objects : employee) {
        for (Object object : objects) {
            System.out.println(object.toString());
        }
    }

    session.getTransaction().commit();
    session.close();
}
```

Criteria with Join

```
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();

    Criteria criteria = session.createCriteria(Employee.class);
    criteria.createAlias("allocationData", "ad");
    criteria.add(
            Restrictions.or(Restrictions.eq("name", "saman"),
                    Restrictions.gt("age", 24))).setProjection(
            Projections.property("ad.city"));

    List<Object> employee = criteria.list();

    for (Object object : employee) {
        System.out.println(object.toString());
    }

    session.getTransaction().commit();
    session.close();
}
```

Query by Example

Query by Example



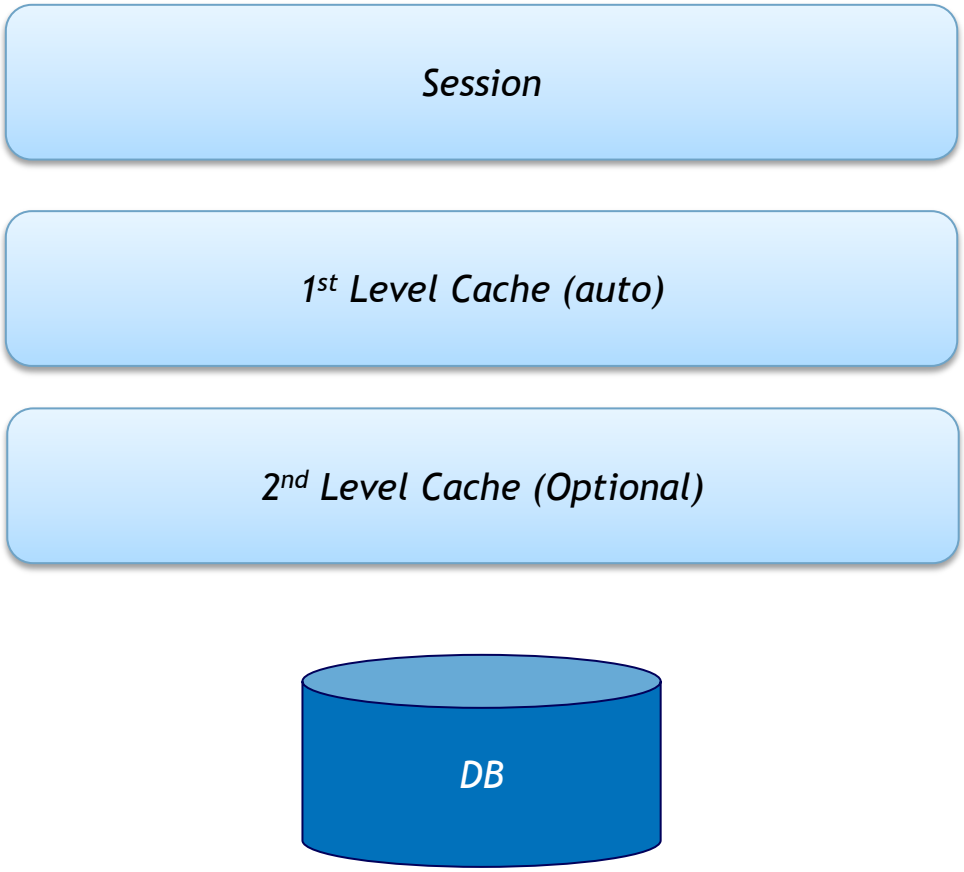
```java
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();

    Employee employee = new Employee();
    employee.setAge(32);
    Example e = Example.create(employee).ignoreCase();
    Criteria criteria = session.createCriteria(Employee.class).add(e);
    List<Employee> employees = criteria.list();
    for (Employee employee2 : employees) {
        System.out.println(employee2.getName());
    }
    session.getTransaction().commit();
    session.close();
}
```

virtusa

Advance Hibernate [not extreme advance]

virtusa

Cache



virtusa

EHCACHE

- For second level cache we need to have cache provider

```
<repository>
    <id>terracotta-releases</id>
    <url>http://www.terracotta.org/download/reflector/releases</url>
    <releases>
        <enabled>true</enabled>
    </releases>
    <snapshots>
        <enabled>false</enabled>
    </snapshots>
</repository>
```

```
<dependency>
    <groupId>net.sf.ehcache</groupId>
    <artifactId>ehcache-core</artifactId>
    <version>2.6.5</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-ehcache</artifactId>
    <version>4.2.7.SP1</version>
</dependency>
```

virtusa

Add properties for config file

- <property name="hibernate.cache.region.factory_class">org.hibernate.cache.ehcache.EhCacheRegionFactory</property>
- <property name="hibernate.cache.use_second_level_cache">true</property>

virtusa

```xml
<hibernate-mapping>
    <class name="com.virtusa.training.hibernate.objects.Employee"
        table="EMPLOYEE">
        <cache usage="read-write"/>
        <id name="empid" type="int">
            <column name="EMPID" />
            <generator class="identity" />
        </id>
        <property name="name" type="java.lang.String">
            <column name="NAME" />
        </property>
        <property name="age" type="int">
            <column name="AGE" />
        </property>
        <!-- <component name="AllocationData"> <property name="city" type="java.lang.String">
            <column name="CITY" /> </property> <property name="mobile" type="java.lang.String">
            <column name="MOBILE" /> </property> </component> -->

        <many-to-one name="allocationData"
            class="com.virtusa.training.hibernate.objects.AllocationData" column="ALLOCATION_ID"
            cascade="save-update" unique="true" />

        <list name="allocation" table="EMPLOYEE_HISTORY" inverse="true"
            cascade="save-update">
            <cache usage="read-write"/>
            <key column="EMPID" />
            <list-index column="ROWID" />
            <one-to-many
                class="com.virtusa.training.hibernate.objects.AllocationHistory" />
        </list>
        <set name="alerts" table="EMPLOYEE_ALERTS" cascade="save-update">
        <key column="EMPID"/>
        <many-to-many class="com.virtusa.training.hibernate.objects.Alerts" column="ALERT_ID"/>
        </set>
    </class>
</hibernate-mapping>
```

```xml
<hibernate-mapping>
    <class name="com.virtusa.training.hibernate.objects.AllocationHistory" table="ALLOCATIONHISTORY">
    <cache usage="read-only"/>
        <id name="employeeid" type="int">
            <column name="id" />
            <generator class="increment" />
        </id>
        <property name="allocationDate" type="java.util.Date">
            <column name="ALLOCATIONDATE" />
        </property>
        <property name="projectCode" type="java.lang.String">
            <column name="PROJECTCODE" />
        </property>
        <many-to-one name="employee" class="com.virtusa.training.hibernate.objects.Employee" not-null="true">
            <column name="EMPID" />
        </many-to-one>
    </class>

</hibernate-mapping>
```

Batch Processing

- When u use batch processing if you use session then your application could crash.
- To overcome this we have 2 options

1. Use HQL to handle data
2. Periodically flush L1 cache

virtusa

HQL update

```
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();

    Employee employee = new Employee();
    employee.setAge(32);
    Example e = Example.create(employee).ignoreCase();
    Criteria criteria = session.createCriteria(Employee.class).add(e);
    List<Employee> employees = criteria.list();
    for (Employee employee2 : employees) {
        System.out.println(employee2.getName());
    }

    Query query = session.createQuery("update AllocationData ad set ad.city='Colombo'");
    query.executeUpdate();

    session.getTransaction().commit();
    session.close();
}
```

virtusa

Manual Batch

- Edit property file as
- `<property name="hibernate.jdbc.batch_size">20</property>`
- This number can be anything based on performance
- And change application code as follows (next page)

virtusa

Scroll {Cursor}

- Normally hibernate wait till commit to clear session. But here we clear it manually

```
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();

    Criteria criteria = session.createCriteria(Employee.class);

    ScrollableResults employee = criteria.setCacheMode(CacheMode.IGNORE)
            .scroll();
    int run = 0;

    while (employee.next()) {
        Employee employee2 = (Employee) employee.get(0);
        employee2.setAllocationData(new AllocationData("Galle",
                "071456XXXX"));
        session.save(employee2);
        System.out.println(employee2.getName());
        if (++run % 2 == 0) {
            session.flush();
            session.clear();
        }

    }
```

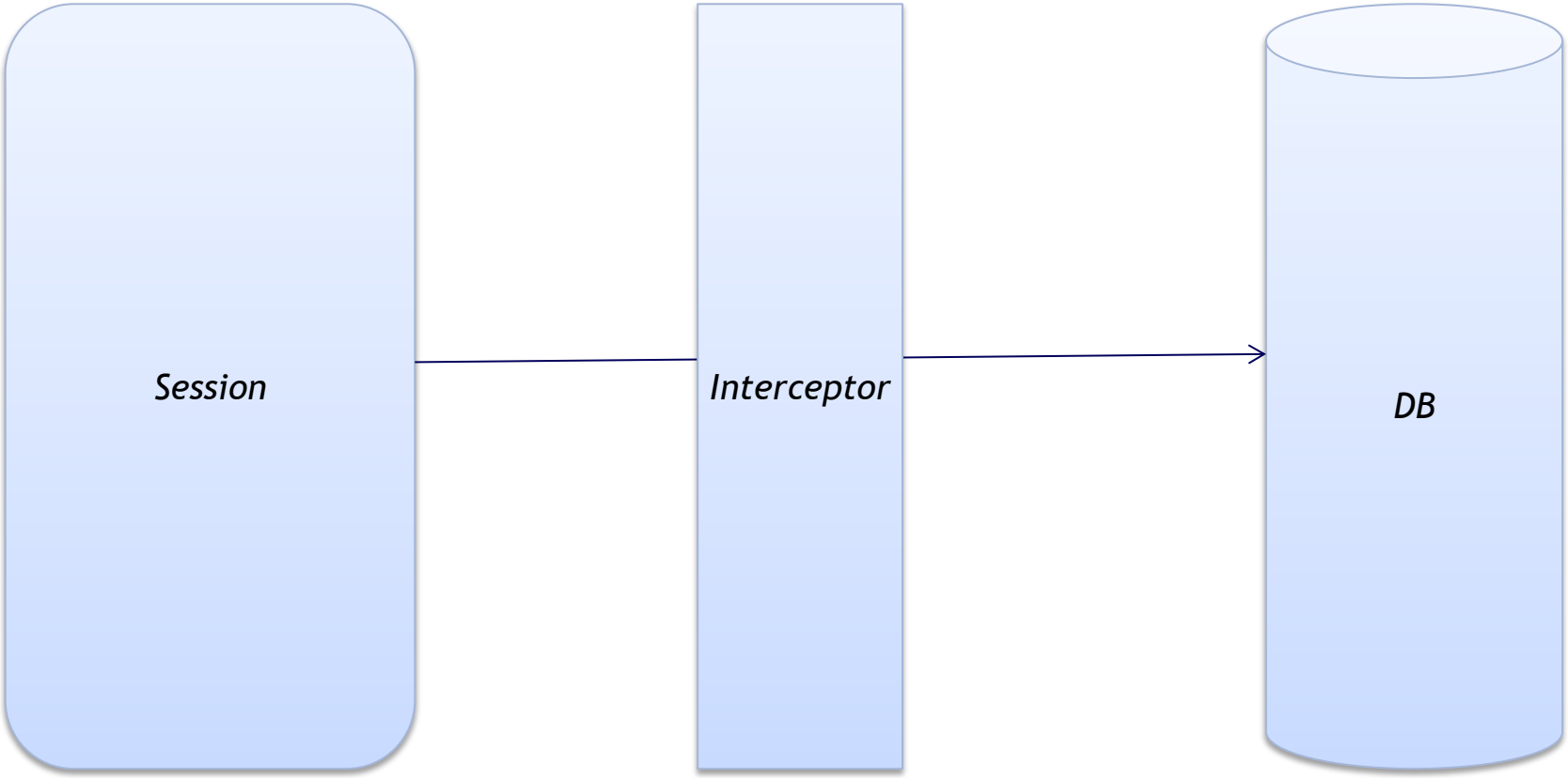
Native SQL

- Make sure you have database name

```
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.password">dbsys</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernate_sample</proper
<property name="hibernate.connection.username">system</property>
```

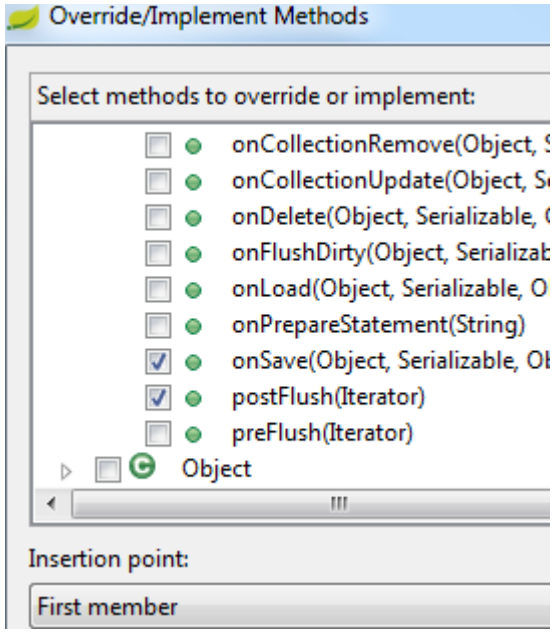
```
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.beginTransaction();

    Query query = session.createSQLQuery("Select * from EMPLOYEE")
            .addEntity(Employee.class);
    List<Employee> employees = query.list();

    for (Employee employee : employees) {
        System.out.println(employee.getName());
    }
    session.getTransaction().commit();
    session.close();
}
```

Interceptor



virtusa

- Create new class as
- `public class Interceptor extends EmptyInterceptor`
- Create following override



```java
public class AuditInterceptor extends EmptyInterceptor  {

    @Override
    public boolean onSave(Object entity, Serializable id, Object[] state,
            String[] propertyNames, Type[] types) {

        System.out.println("Wowwww.... Its SAVED... :)");
        return false;
    }

    @Override
    public void postFlush(Iterator entities) {

        System.out.println("its FlUsHeD.... Grrrrrr");
    }


}
```

Modify util class and execute

```
public class HibernateUtilities {

    private static SessionFactory sessionFactory;
    private static ServiceRegistry serviceRegistry; // new from version 4

    static {
        try {
            Configuration configuration = new Configuration().setInterceptor(new AuditInterceptor()).configure();
            serviceRegistry = new ServiceRegistryBuilder().applySettings(
                    configuration.getProperties()).buildServiceRegistry();
            sessionFactory = configuration.buildSessionFactory(serviceRegistry);
        } catch (HibernateException exception) {
            exception.printStackTrace();
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

}
```

Listener

From Hibernate 4 its changed!!!

- Need few basic things
  1. Listener
  2. Integrator
  3. META-INF

virtusa

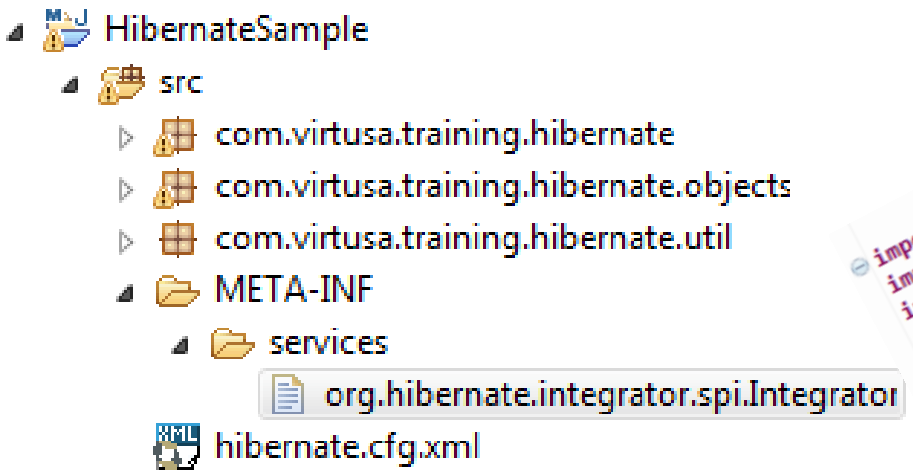
Listener

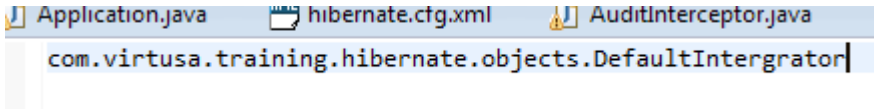
```
public class LoadEventListner extends DefaultLoadEventListener {
    @Override
    public void onLoad(LoadEvent event, LoadType arg1)
            throws HibernateException {
        System.out.println("Entity Loaded.. name is > "
                + event.getEntityClassName());
    }

}
```

Integration

```java
public class DefaultIntergrator implements Integrator {

    @Override
    public void integrate(Configuration configuration,
            SessionFactoryImplementor sessionFactory,
            SessionFactoryServiceRegistry serviceRegistry) {

        EventListenerRegistry registry=serviceRegistry.getService(EventListenerRegistry.class);
        registry.prependListeners(EventType.LOAD, new LoadEventListner());
    }

    @Override
    public void integrate(MetadataImplementor metadata,
            SessionFactoryImplementor sessionFactory,
            SessionFactoryServiceRegistry serviceRegistry) {
    }

    @Override
    public void disintegrate(SessionFactoryImplementor sessionFactory,
            SessionFactoryServiceRegistry serviceRegistry) {
    }

}
```

- Create file in exact name and exact case and exact path
- Add FQCN (fully qualified class name of here and execute





Data Filter

- If we need system wide filter for data this is ideal.
- Imagine you need to prevent to load word call “KILL” ☹
- But this will not work with native SQL ☹☹☹

virtusa

```java
static void loadQueryBased() {

    Session session = HibernateUtilities.getSessionFactory().openSession();
    session.enableFilter("fileterByName").setParameter("ename", "s%");

    session.beginTransaction();



    Query query = session.createQuery("from Employee");

    List<Employee> employees = query.list();

    for (Employee employee : employees) {
        System.out.println("Emp name is "+employee.getName());
    }
    session.getTransaction().commit();
    session.close();
}
```

virtusa

```xml
<hibernate-mapping>
    <filter-def name="fileterByName" condition="name  like :ename">
        <filter-param name="ename" type="string" />
    </filter-def>
    <class name="com.virtusa.training.hibernate.objects.Employee"
        table="EMPLOYEE">
        <cache usage="read-write" />
        <id name="empid" type="int">
            <column name="EMPID" />
            <generator class="identity" />
        </id>
        <property name="name" type="java.lang.String">
            <column name="NAME" />
        </property>
        <property name="age" type="int">
            <column name="AGE" />
        </property>
        <many-to-one name="allocationData"
            class="com.virtusa.training.hibernate.objects.AllocationData" column="ALLOCATION_ID"
            cascade="save-update" unique="true" />

        <list name="allocation" table="EMPLOYEE_HISTORY" inverse="true"
            cascade="save-update">
            <cache usage="read-write" />
            <key column="EMPID" />
            <list-index column="ROWID" />
            <one-to-many
                class="com.virtusa.training.hibernate.objects.AllocationHistory" />
        </list>
        <set name="alerts" table="EMPLOYEE_ALERTS" cascade="save-update">
            <key column="EMPID" />
            <many-to-many class="com.virtusa.training.hibernate.objects.Alerts"
                column="ALERT_ID" />
        </set>
        <filter name="fileterByName"></filter>
    </class>
</hibernate-mapping>
```

Happy Ma”qp”ing ☺

virtusa