Aprameyah
Technologies Pvt. Ltd.

More Hibernate

Aprameyah
Technologies Pvt. Ltd.

HQL

```
String hql = "FROM Users";
Query query = session.createQuery(hql);
List results = query.list();
```

```
hql = "SELECT U.name FROM Users U”;
```

```
String hql = "FROM Users U WHERE U.id = :uid”;
Query query = session.createQuery(hql);
query.setParameter(“uid",10);
```

```
String hql = "UPDATE Users set age = :age "  +
                "WHERE id = :user_id";
Query query = session.createQuery(hql);
query.setParameter("age", 12);
query.setParameter("user_id", 10);
int result = query.executeUpdate();
```

Aprameyah Technologies Pvt. Ltd.

HQL

```
String hql = "DELETE FROM Employee "  +
                "WHERE id = :employee_id”;
```

```
String hql = "INSERT INTO Employee(firstName, lastName, salary)"  +
                "SELECT firstName, lastName, salary FROM old_employee";
Query query = session.createQuery(hql);
int result = query.executeUpdate();
```

```
String hql = "SELECT count(distinct E.firstName) FROM Employee E”;
```

```
String hql = "FROM Employee";
Query query = session.createQuery(hql);
query.setFirstResult(1);
query.setMaxResults(10);
```

Apprameyah
Technologies Pvt. Ltd.

Criteria Queries

```
Criteria cr = session.createCriteria(Employee.class);
      List results = cr.list();
```

```
Criteria cr = session.createCriteria(Employee.class);
      cr.add(Restrictions.eq("salary", 2000));
      List results = cr.list();
```

```
Criteria cr = session.createCriteria(Employee.class);
Criterion salary = Restrictions.gt("salary", 2000);
Criterion name = Restrictions.ilike("firstNname","zara%");
// To get records matching with OR condistions
LogicalExpression orExp = Restrictions.or(salary, name);
cr.add( orExp );
// To get records matching with AND condistions
LogicalExpression andExp = Restrictions.and(salary, name);
cr.add( andExp );
List results = cr.list();
```

Aprameyah Technologies Pvt. Ltd.

Criteria Queries

```
Criteria cr = session.createCriteria(Employee.class);
// To get records having salary more than 2000
cr.add(Restrictions.gt("salary", 2000));
// To sort records in descening order
crit.addOrder(Order.desc("salary"));
// To sort records in ascending order
crit.addOrder(Order.asc("salary"));
List results = cr.list();
```

```
Criteria cr = session.createCriteria(Employee.class);
// To get total row count.
cr.setProjection(Projections.rowCount());
Criteria cr = session.createCriteria(Employee.class);
// To get maximum of a property.
cr.setProjection(Projections.max("salary"));
```

Apprameyah Technologies Pvt. Ltd.

Batch Strategies

- Try loading all users and accessing their orders and watch how hibernate generates sql statements

```
List<Users> users = session.createQuery("select u from Users u").list();
System.out.println("Loaded users...");
for (Users user : users) {
   System.out.println(user.getOrders());
}
```

Batching

- Comment out previous code, apply a batch size to the orders collection and retry the previous sample

```
<set name="orders" batch-size="2">
   <key column="user_id" />
   <one-to-many class="com.mydomain.model.Orders" />
</set>
```

- Batch size specifies how many uninitialised collections are loaded into memory

Fetching in Many to One

- Try to load all orders and access its corresponding users this way:

```
List<Orders> orders = session.createQuery("select o from Orders
o").list();
System.out.println("Loaded orders...");
for (Orders order : orders) {
   System.out.println(order.getUser());
}
```

Batching in Single Associations

- Apply a batch size to the users class to see how the previous sample behaves

```
<class name="com.mydomain.model.Users" table="USERS" schema="APP" batch-size="3">
```

- Uncomment the previous code above to load all users and watch queries
    - Remember session is a first level cache

Aprameyah
Technologies Pvt. Ltd.

Fetch Strategy

- Default strategy is to lazy load associations using additional select statements
- Can be optimised to load in a join
- Try the below code with and without the fetchmode setting

```
User user = (User) session.createCriteria(User.class)
                .setFetchMode("orders", FetchMode.JOIN)
                .add( Restrictions.idEq(309))
                .uniqueResult();
```

Aprameyah Technologies Pvt. Ltd.

Multi Tenancy

- Multi Tenancy is a reality in many SaaS implementations
- There are stringent US govt restrictions of how and where you can maintain user data in SaaS models
- It gets difficult to code if a Multi tenant DB model is not followed

Aprameyah Technologies Pvt. Ltd.

Multi Tenancy Strategy

- Database
    - A different physical database is used for each tenant
- Schema
    - A different schema within the same physical database is used for each tenant
- Differentiator
    - A differentiator field is used in all tables to identify as to whom the data belongs to. (Not supported in hibernate 4)

Setup Multi-Tenancy

- Identify Strategy to be used

```
<property name="hibernate.multiTenancy">DATABASE</property>
```

- Provide a Connection provider that implements MultiTenantConnectionProvider interface AND use

```
<property
name=“hibernate.multi_tenant_connection_provider">com.mydomain.biz.Tenan
tBasedConnectionProvider</property>

Session ses =
sessionFactory.withOptions().tenantIdentifier("mydb").openSession();
```

Aprameyah
Technologies Pvt. Ltd.

Auditing

- Auditing refers to keeping a track of changes that happen to entities where keeping a history is required

- With the Envers Auditing feature in Hibernate 4.x, we can maintain an audit trail for any entity

Apprameyah Technologies Pvt. Ltd.

Listener Configuration

- Envers auditing works based on listeners in hibernate session factory

```
<property name="hibernate.ejb.event.post-insert">
org.hibernate.ejb.event.EJB3PostInsertEventListener,org.hibernate.envers.event.AuditEventListener</property>
<property name="hibernate.ejb.event.post-update">
org.hibernate.ejb.event.EJB3PostUpdateEventListener,org.hibernate.envers.event.AuditEventListener</property>
<property name="hibernate.ejb.event.post-delete">
org.hibernate.ejb.event.EJB3PostDeleteEventListener,org.hibernate.envers.event.AuditEventListener</property>
<property name="hibernate.ejb.event.pre-collection-update">
org.hibernate.envers.event.AuditEventListener</property>
<property name="hibernate.ejb.event.pre-collection-remove">
org.hibernate.envers.event.AuditEventListener</property>
<property name="hibernate.ejb.event.post-collection-recreate">
org.hibernate.envers.event.AuditEventListener</property>
```

Aprameyah
Technologies Pvt. Ltd.

Auditing Entities

- Decide on entities that need an audit trial and annotate it with @Audit

- Any related entity that does not need auditing should be annotated with @Audited(targetAuditMode = RelationTargetAuditMode.NOT_AUDITED)

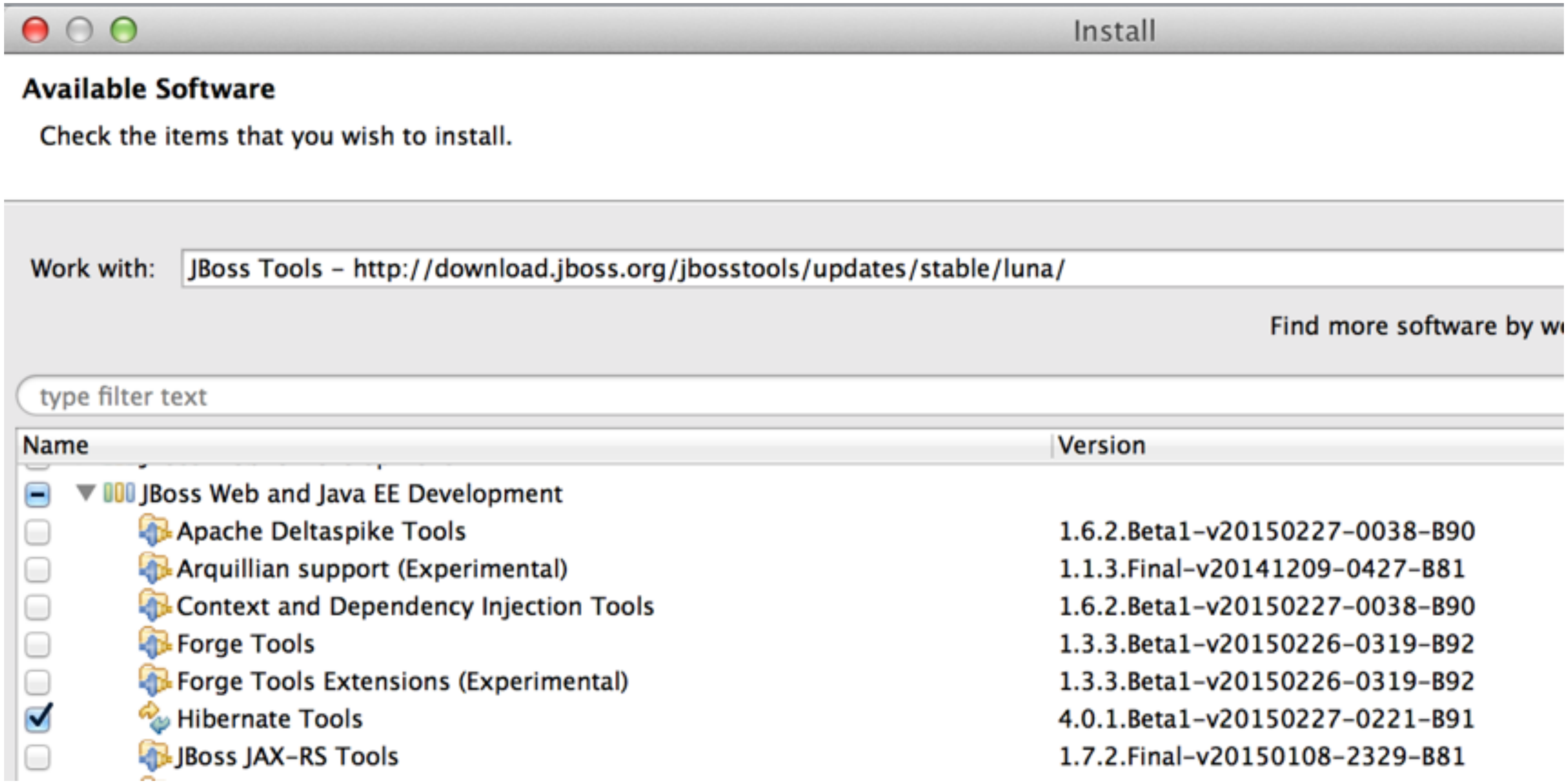
- Finally regenerate database schema to include the auditing tables using this setting:

```
<property name="hibernate.hbm2ddl.auto">create-drop</property>
```

Aprameyah
Technologies Pvt. Ltd.

Reverse Engineering

- Hibernate tools allow a lot of reverse engineering and schema generation abilities



Install

Available Software

Check the items that you wish to install.

Work with: JBoss Tools – http://download.jboss.org/jbosstools/updates/stable/luna/

Find more software by w

type filter text

	Name	Version
▬	▼ JBoss Web and Java EE Development	
☐	Apache Deltaspike Tools	1.6.2.Beta1-v20150227-0038-B90
☐	Arquillian support (Experimental)	1.1.3.Final-v20141209-0427-B81
☐	Context and Dependency Injection Tools	1.6.2.Beta1-v20150227-0038-B90
☐	Forge Tools	1.3.3.Beta1-v20150226-0319-B92
☐	Forge Tools Extensions (Experimental)	1.3.3.Beta1-v20150226-0319-B92
✓	Hibernate Tools	4.0.1.Beta1-v20150227-0221-B91
☐	JBoss JAX-RS Tools	1.7.2.Final-v20150108-2329-B81

Aprameyah
Technologies Pvt. Ltd.

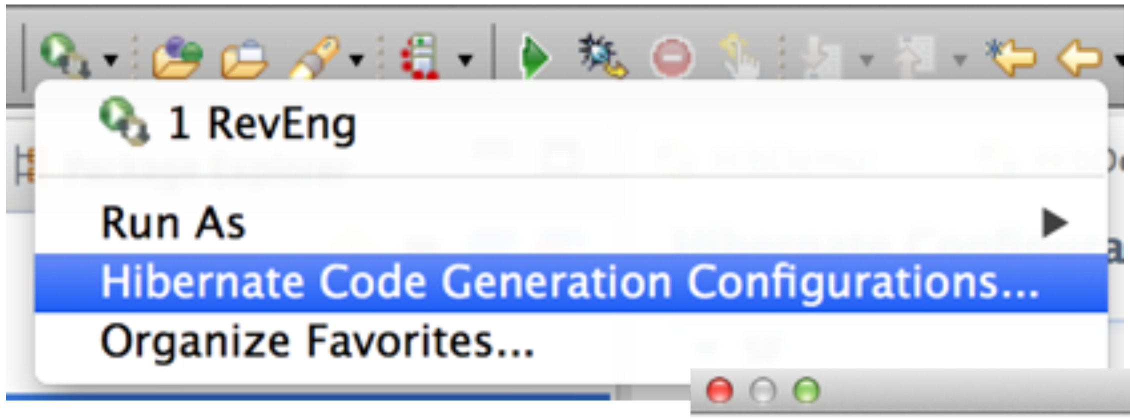
Reverse Engineering

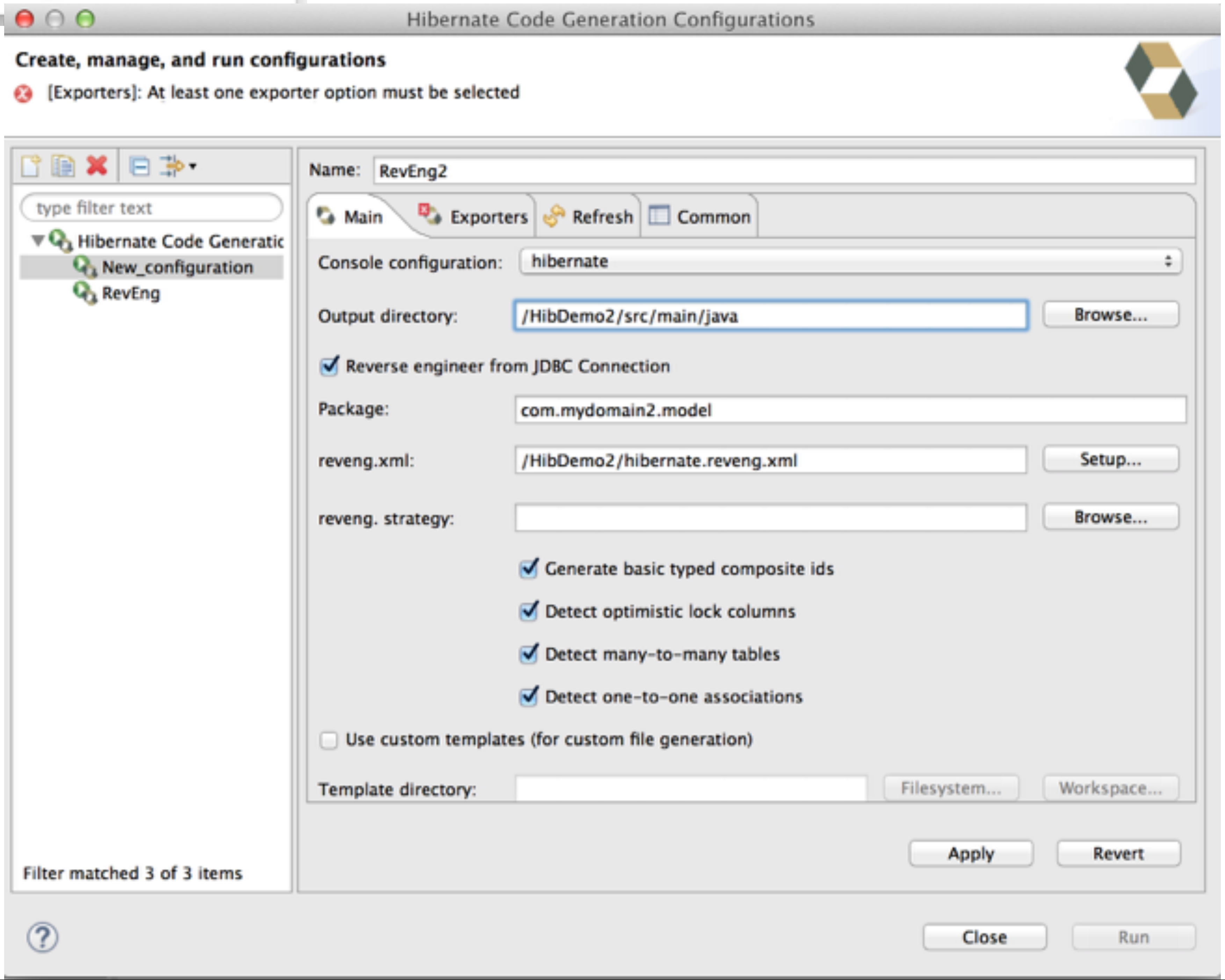
- Switch to Hibernate perspective and right-click to add configuration

- Select a project and setup to create a new configuration.

- Provide connection information and create a session factory configuration

Reverse Engineering





Hibernate Code Generation Configurations

Create, manage, and run configurations

[Exporters]: At least one exporter option must be selected

type filter text

- Hibernate Code Generatic
  - New_configuration
  - RevEng

Filter matched 3 of 3 items

Name: RevEng2

Main | Exporters | Refresh | Common

Console configuration: hibernate

Output directory: /HibDemo2/src/main/java

✓ Reverse engineer from JDBC Connection

Package: com.mydomain2.model

reveng.xml: /HibDemo2/hibernate.reveng.xml

reveng. strategy:

✓ Generate basic typed composite ids

✓ Detect optimistic lock columns

✓ Detect many-to-many tables

✓ Detect one-to-one associations

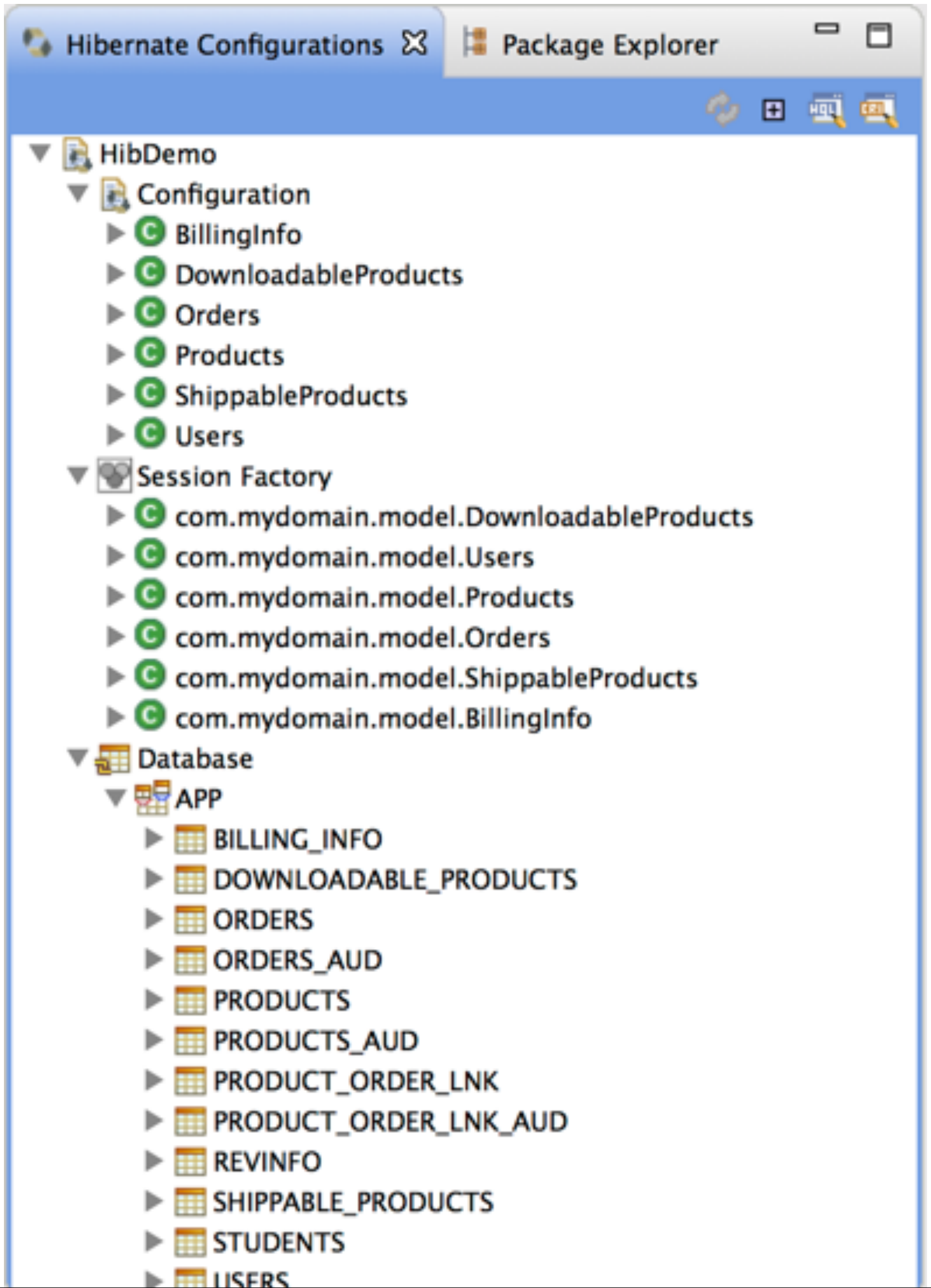
☐ Use custom templates (for custom file generation)

Template directory:

Aprameyah Technologies Pvt. Ltd.

Hibernate Config View

- View Entities, Configuration and Database tables

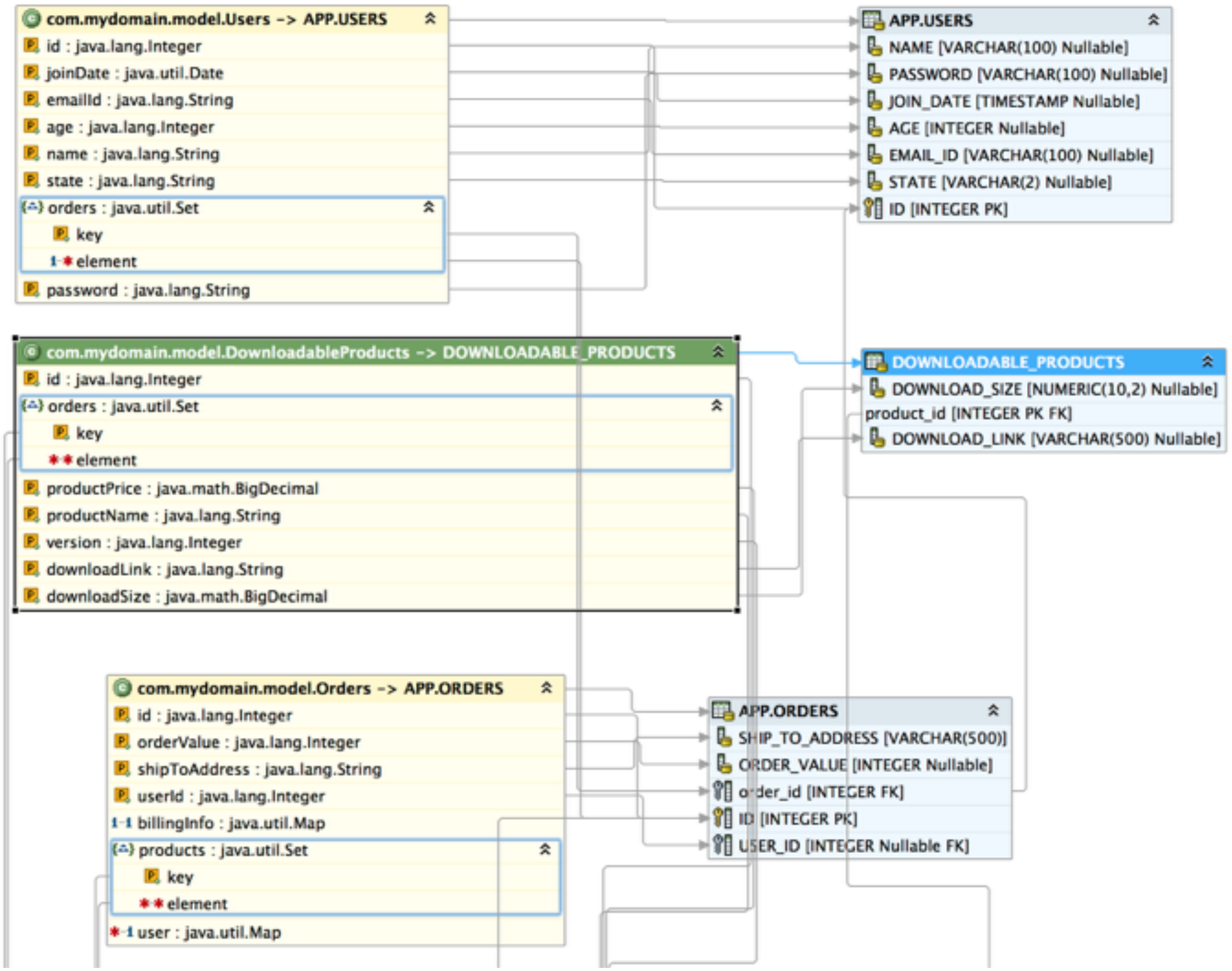


```
▼ HibDemo
  ▼ Configuration
    ▶ BillingInfo
    ▶ DownloadableProducts
    ▶ Orders
    ▶ Products
    ▶ ShippableProducts
    ▶ Users
  ▼ Session Factory
    ▶ com.mydomain.model.DownloadableProducts
    ▶ com.mydomain.model.Users
    ▶ com.mydomain.model.Products
    ▶ com.mydomain.model.Orders
    ▶ com.mydomain.model.ShippableProducts
    ▶ com.mydomain.model.BillingInfo
  ▼ Database
    ▼ APP
      ▶ BILLING_INFO
      ▶ DOWNLOADABLE_PRODUCTS
      ▶ ORDERS
      ▶ ORDERS_AUD
      ▶ PRODUCTS
      ▶ PRODUCTS_AUD
      ▶ PRODUCT_ORDER_LNK
      ▶ PRODUCT_ORDER_LNK_AUD
      ▶ REVINFO
      ▶ SHIPPABLE_PRODUCTS
      ▶ STUDENTS
      ▶ USERS
```

Aprameyah
Technologies Pvt. Ltd.

Mapping Diagram

- Right click configuration and run “mapping diagram”


com.mydomain.model.Users -> APP.USERS
id : java.lang.Integer
joinDate : java.util.Date
emailId : java.lang.String
age : java.lang.Integer
name : java.lang.String
state : java.lang.String
orders : java.util.Set
key
element
password : java.lang.String
APP.USERS
NAME [VARCHAR(100) Nullable]
PASSWORD [VARCHAR(100) Nullable]
JOIN_DATE [TIMESTAMP Nullable]
AGE [INTEGER Nullable]
EMAIL_ID [VARCHAR(100) Nullable]
STATE [VARCHAR(2) Nullable]
ID [INTEGER PK]
com.mydomain.model.DownloadableProducts -> DOWNLOADABLE_PRODUCTS
id : java.lang.Integer
orders : java.util.Set
key
element
productPrice : java.math.BigDecimal
productName : java.lang.String
version : java.lang.Integer
downloadLink : java.lang.String
downloadSize : java.math.BigDecimal
DOWNLOADABLE_PRODUCTS
DOWNLOAD_SIZE [NUMERIC(10,2) Nullable]
product_id [INTEGER PK FK]
DOWNLOAD_LINK [VARCHAR(500) Nullable]
com.mydomain.model.Orders -> APP.ORDERS
id : java.lang.Integer
orderValue : java.lang.Integer
shipToAddress : java.lang.String
userId : java.lang.Integer
billingInfo : java.util.Map
products : java.util.Set
key
element
user : java.util.Map
APP.ORDERS
SHIP_TO_ADDRESS [VARCHAR(500)]
ORDER_VALUE [INTEGER Nullable]
order_id [INTEGER FK]
ID [INTEGER PK]
USER_ID [INTEGER Nullable FK]