**JPA Tutorial**

JPA tutorial မွာ Java Persistence API ရဲ႕ basic နဲ႔ advance concept မ်ားကို သိရမွာ ၿဖစ္ပါတယ္။ JPA က အဓိက ဘာကိုလုပ္ေဆာင္ေပးသလဲဆိုရင္ Java applications မ်ားထဲမွာ ရွိတဲ့ relational data မ်ားကို manage လုပ္ရန္ object-relational mapping ကို လုပ္ေဆာင္ေပးၿခင္း ၿဖစ္ပါတယ္။ SQL statements မ်ားကို အသံုးမွဳမ်ားရဲ႕ objects instead မ်ားကို တိုက္ရိုက္ လုပ္ေဆာင္ႏိုင္ရန္ platform တစ္ခု ေပးစြမ္းပါတယ္။

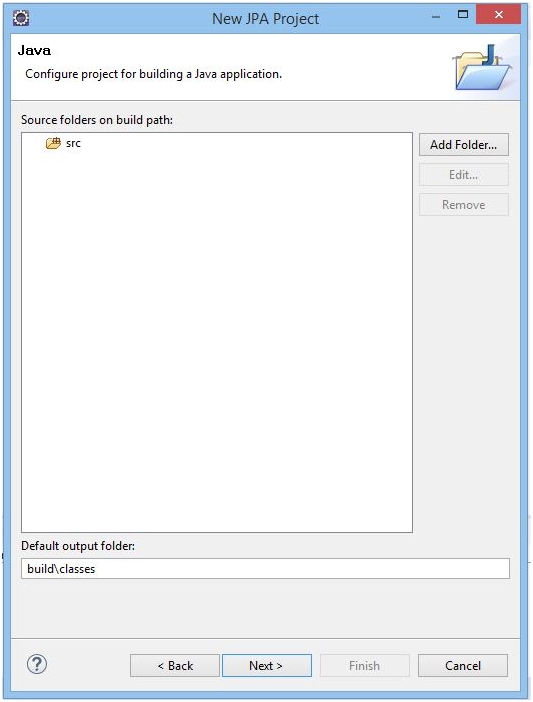
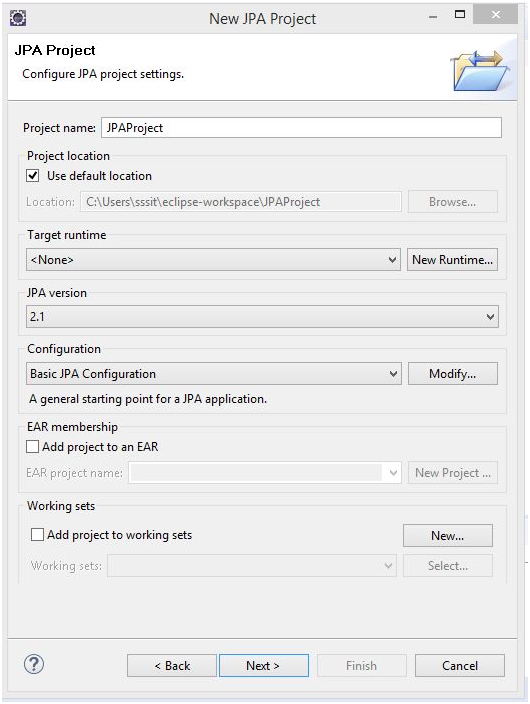
1. JPA Introduction

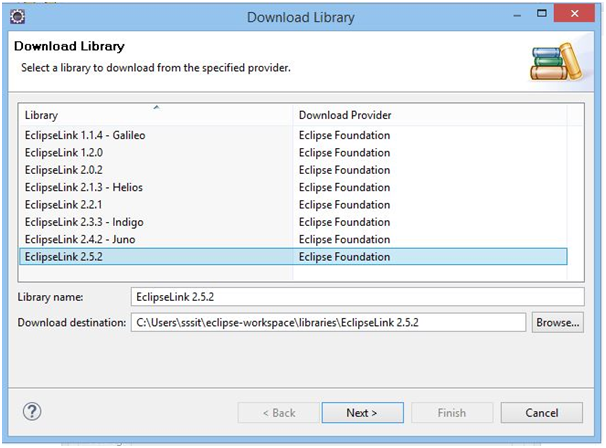
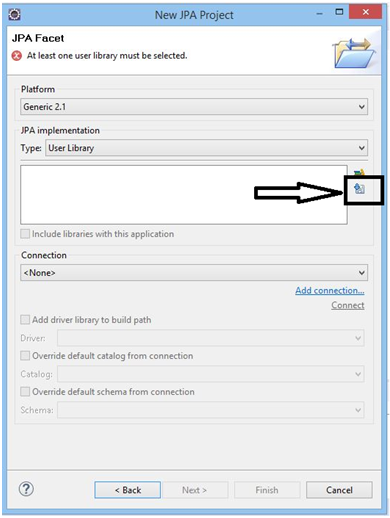
JPA ကို Java object နဲ႔ relational database ၾကားထဲမွာ data မ်ား persist လုပ္ေဆာင္ႏိုင္ရန္ အတြက္အသံုးၿပဳပါတယ္။ ဆိုလိုတာက JPA ဆိုတာ object-oriented domain models မ်ားနဲ႔ relational database systems မ်ားကို ဆက္သြယ္ေပးထားတဲ့ တံတားတစ္ခု ၿဖစ္ပါတယ္။

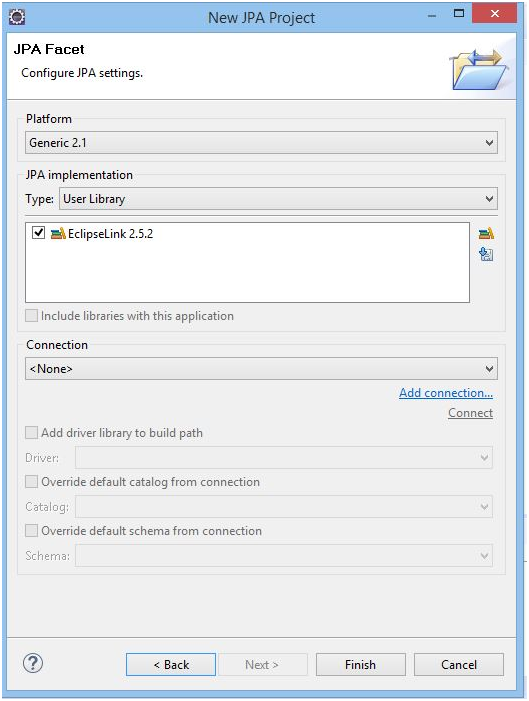
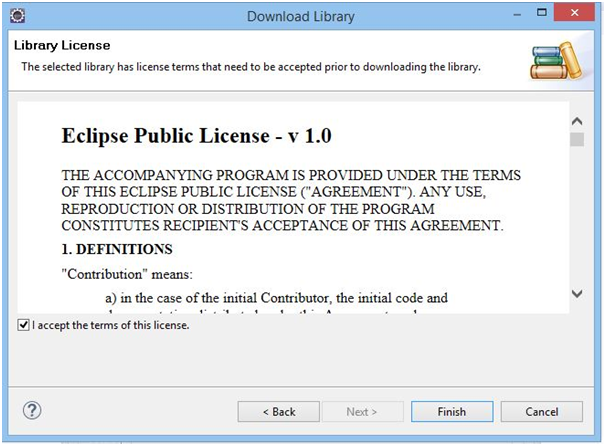
JPA ဆိုတာက specification တစ္ခုပဲ ၿဖစ္တယ္။ သူ႕ဟာနဲ႔သူ operation မလုပ္ေဆာင္ႏိုင္ဘူး။ ဆိုလိုတာက သူအေနနဲ႔ implementation တစ္ခု လိုအပ္တယ္။ ဒါေၾကာင့္ ORM tools မ်ားၿဖစ္ၾကတဲ့ Hibernate, TopLink နဲ႔ iBatis အစရွိသည္တို႔က data persistence အတြက္ JPA specifications မ်ား implements လုပ္ၾကၿခင္း ၿဖစ္ပါတယ္။

2. JPA Installation

File>New>JPA Project







3. JPA ORM

JPA Object Relational Mapping

Object Relational Mapping (ORM) ဆိုတာ functionality တစ္ခု ၿဖစ္ပါတယ္။ ORM ကို object တစ္ခု နဲ႔ relational database ၾကားထဲမွာ object state နဲ႔ database column လုပ္ၿခင္းအားၿဖင့္ အသံုးၿပဳလို႔ရပါတယ္။ အမ်ိဳးမ်ိဳးေသာ database ေတြနဲ႔လည္း အဆင္ေၿပေၿပ

ခ်ိတ္ဆက္ႏိုင္ပါတယ္။



ORM Frameworks

ေအာက္မွာၿပထားတဲ့ အမ်ိဳးမ်ိဳးေသာ frameworks မ်ားက ORM mechanism ေပၚမွာ လုပ္ေဆာင္ၾကပါတယ္။

1. Hibernate
2. TopLink
3. ORMLite
4. iBATIS
5. JPOX

Mapping Directions

Mapping Directions ကို အပိုင္း ၂ ပိုင္းခြဲၿခားထားပါတယ္။

1. **Unidirectional relationship -** In this relationship, only one entity can refer the properties to another. It contains only one owing side that specifies how an update can be made in the database.
2. **Bidirectional relationship -** This relationship contains an owning side as well as an inverse side. So here every entity has a relationship field or refer the property to other entity.

Types of Mapping

1. **One-to-one -** @OneToOne annotation အေနနဲ႔ ကိုယ္စားၿပဳပါတယ္။ entity တစ္ခုခ်င္းစီတုိင္းရဲ႕ instance ဟာ အၿခား entity တစ္ခုရဲ႕ single instance နဲ႔ ဆက္စပ္ေနမွာ ၿဖစ္ပါတယ္။
2. **One-to-many -** @OneToMany annotation အေနနဲ႔ ကိုယ္စားၿပဳပါတယ္။ entity တစ္ခုရဲ႕ instance တစ္ခုဟာ အၿခား entity တစ္ခုရဲ႕ တစ္ခုထက္ပိုေသာ instance မ်ားနဲ႔ relate ၿဖစ္ခဲ့မယ္ဆိုရင္ ဒီ relationship ကို အသံုးၿပဳမွာ ၿဖစ္ပါတယ္။
3. **Many-to-one -** @ManyToOne annotation အေနနဲ႔ ကိုယ္စားၿပဳပါတယ္။ entity တစ္ခုရဲ႕ multiple instances မ်ားက အၿခား entity တစ္ခုရဲ႕ single instance နဲ႔ relate ၿဖစ္ေနမယ္ဆိုရင္ ဒီ relationship ကို အသံုးၿပဳပါတယ္။
4. **Many-to-many -** @ManyToMany annotation အေနနဲ႔ ကိုယ္စားၿပဳပါတယ္။ entity တစ္ခုရဲ႕ multiple instances မ်ားက အၿခား entity တစ္ခုရဲ႕ multiple instances မ်ားနဲ႔ ဆက္စပ္ေနမယ္ဆိုရင္ ဒီ relationship ကို အသံုးၿပဳႏိုင္ပါတယ္။ ဒီ mapping မွာဆိုရင္ မည္သည့္ side မဆို owing side ၿဖစ္ႏိုင္ပါတယ္။

**JPA Entity**

1. JPA Entity Introduction

ေယဘုယေၿပာရမယ္ဆိုရင္ entity ဆိုတာက single unit တစ္ခုထဲမွာ states ေတြကို အတူတကြစုေပါင္းထားတဲ့ group တစ္ခုၿဖစ္ပါတယ္။ entity တစ္ခုဆိုတာက object တစ္ခုအေနနဲ႔ ၿပဳမူၿပီးေတာ့ object-oriented paradigm ရဲ႕ အဓိက အစိတ္အပိုင္းအေနနဲ႔ ၿဖစ္လာတယ္။ အဲ့ေတာ့ entity တစ္ခုဆိုတာက Java Persistence Library ထဲမွာ application-defined object တစ္ခုၿဖစ္ပါတယ္။

Entity Properties

Persistability – object တစ္ခုၿဖစ္ၿပီးေတာ့ သူ႕ကို database ထဲမွာ သိမ္းခဲ့မယ္ဆိုရင္ေတာ့ persistent လို႔ေခၚပါတယ္။ ၿပီးေတာ့ သူ႕ကို အခ်ိန္မေရြး access လုပ္ႏိုင္ပါတယ္။

Persistent Identity -