PHÁT TRIỂN HỆ PHẦN MỀM

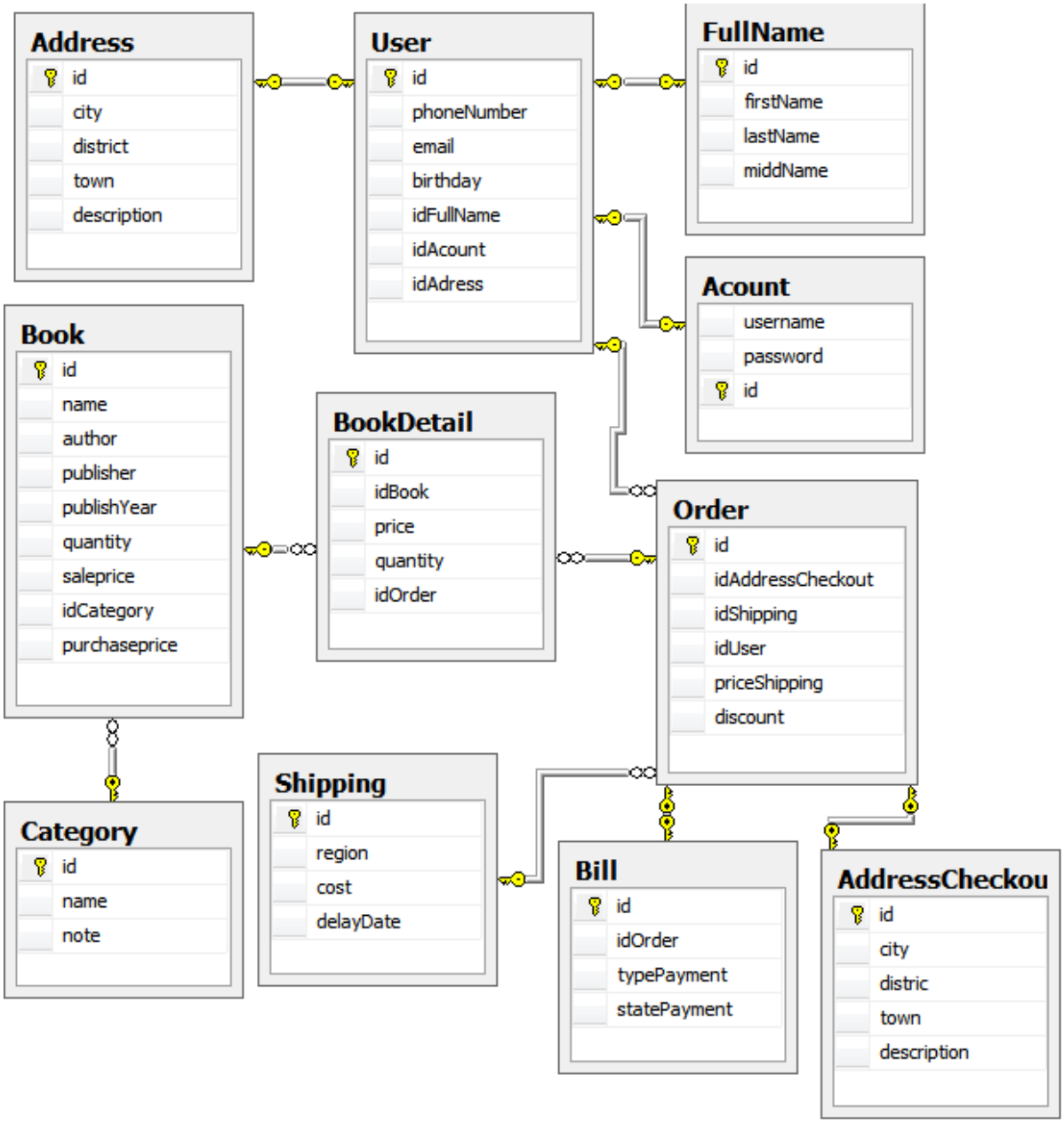
DỰA TRÊN EJB-J2EE

Tài liệu tham khảo này dành cho sinh viên chuyên ngành Công nghệ phần mềm, Học viện CNBCVT

Giả sử ta có lược đồ CSDL xây dựng từ Biểu đồ lớp. Bài toán đặt ra là xây dựng các bean cho J2EE như thế nào? Tài liệu này hướng dẫn xây dựng session bean và entity bean từ lược đồ trên. Chú ý lược đồ CSDL ở đây chỉ là để tham khảo nhằm thể hiện các bước xây dựng bean như thế nào nên sinh viên không nên xem là chuẩn mực để tuân theo! Hy vọng nó có ích cho các bạn khi hoàn thành Bài tập lớn. Các bạn có ý kiến trao đổi hãy mail: tdque@yahoo.com

1. Cơ sở dữ liệu

1.1 Lược đồ



1.2 Giải thích

1.2.1 Book

Quantity:số lượng sách còn trong kho

Purchaseprice:giá cửa hàng nhập sách vào.

Saleprice:giá cửa hàng bán ra.

1.2.2 BookDetail

Chi tiết từng quyển sách khách hàng mua.

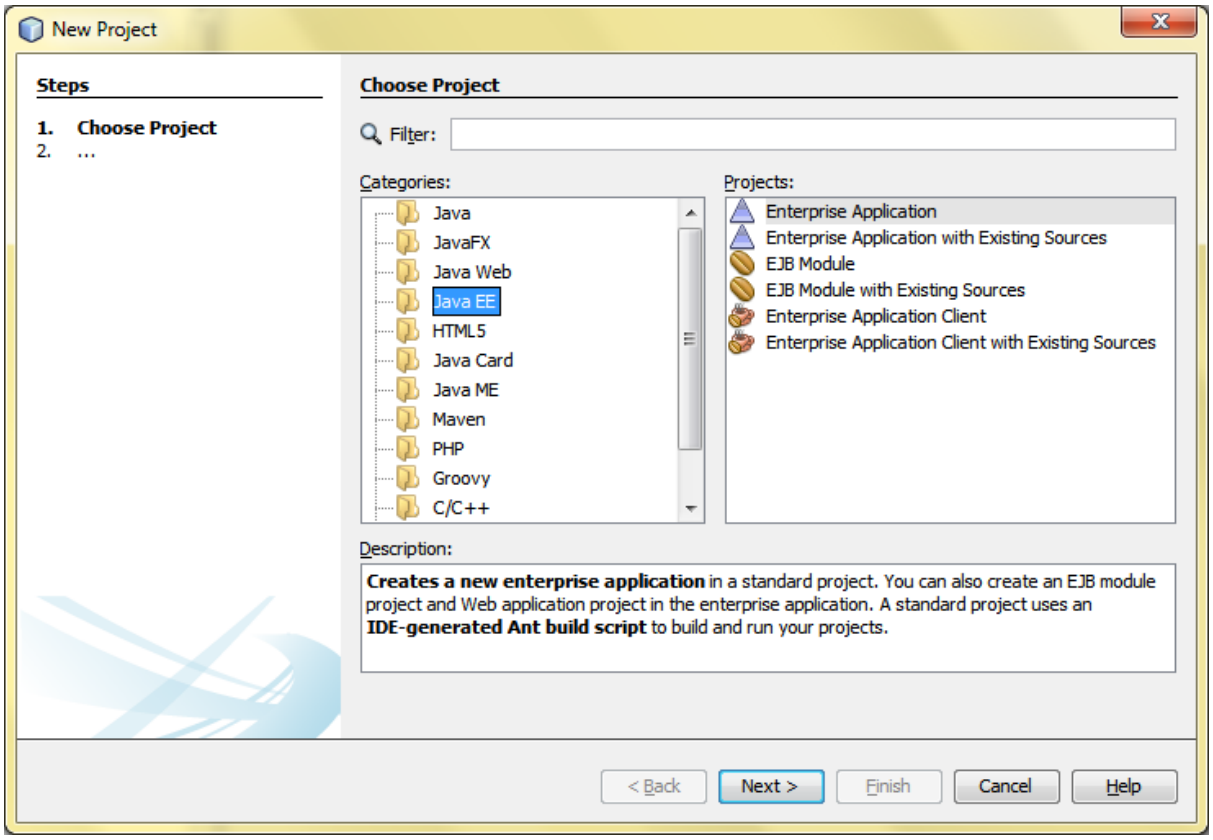
Price:giá khách hàng mua sách.

Quantity:số lượng quyển cho cuốn sách khách hàng mua.

2. Các bước tạo project,entity bean,session bean từ database

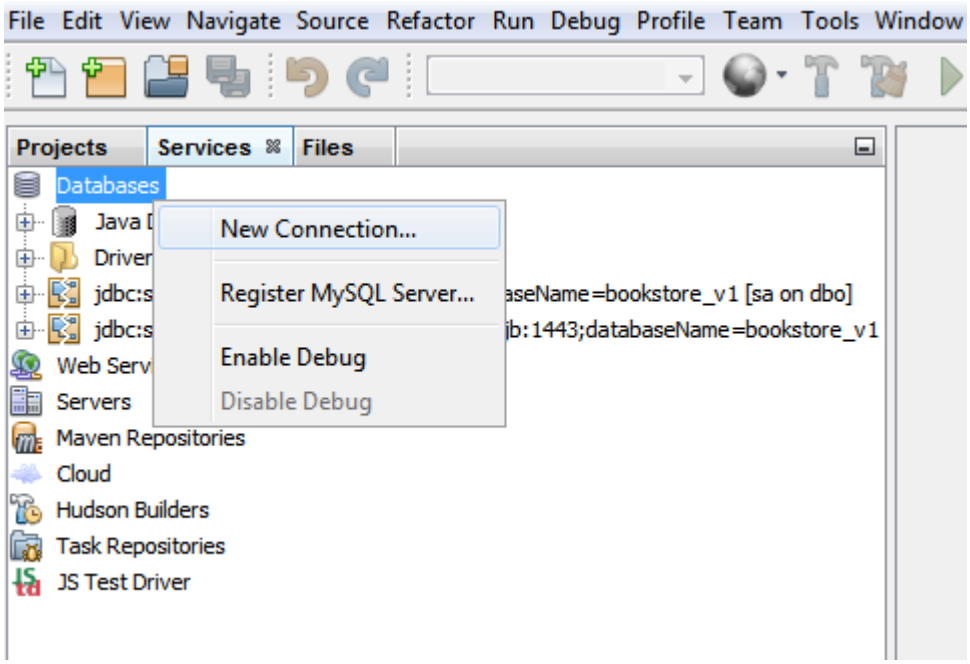
2.1 Tạo project gồm ejbmodul và webapplication

- Chọn new Project
- Chọn JavaEE
- Chọn enterprise Application.
- Next
- Điền tên project
- Next
- finish

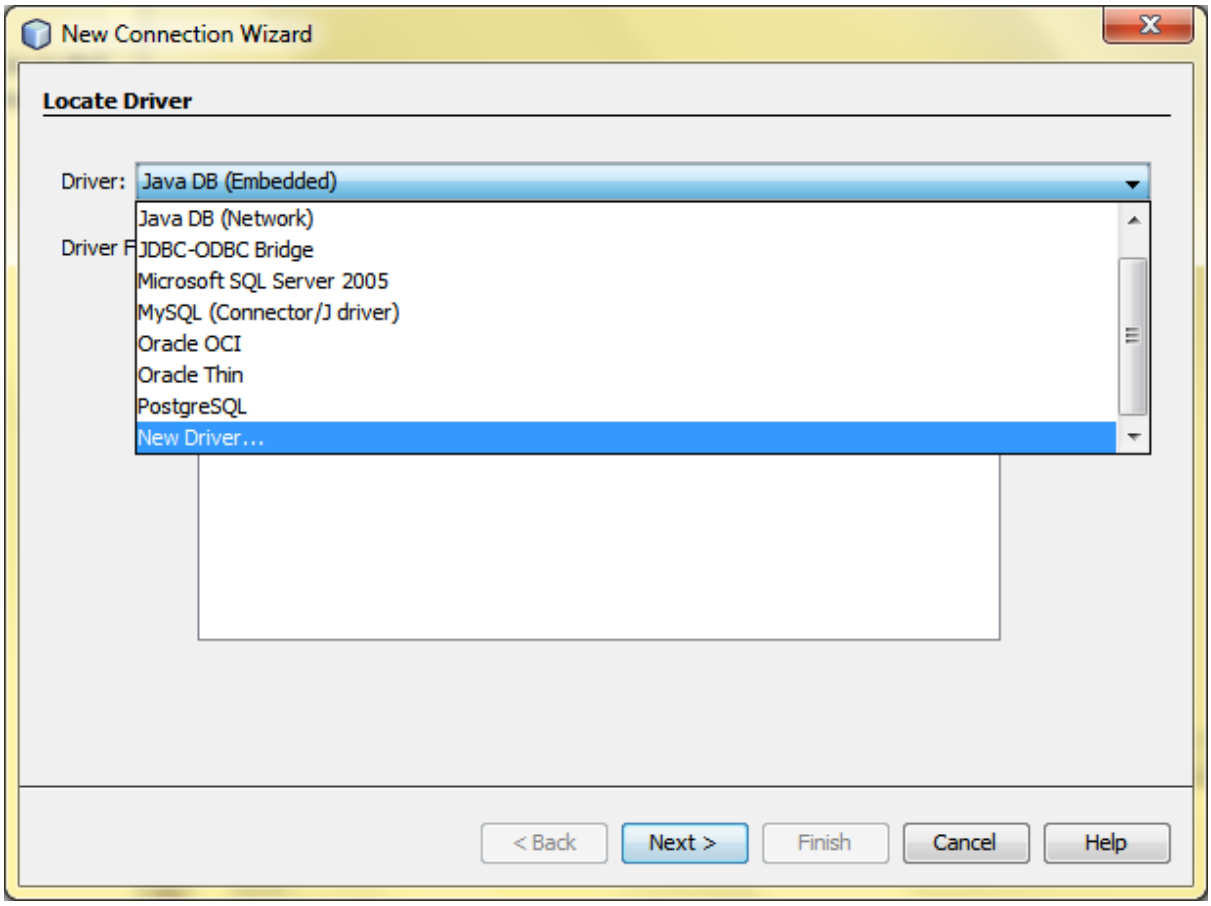


2.2 Tạo kết nối đến database

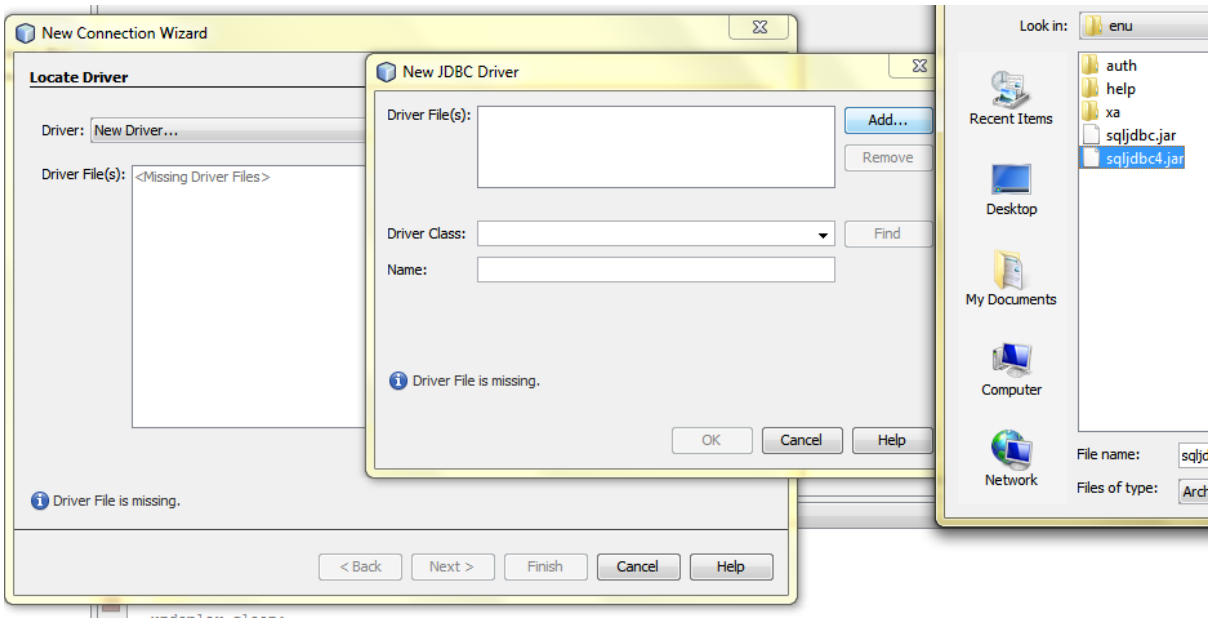
- Chọn thẻ Services
- Chuột phải vào Databses/New connection



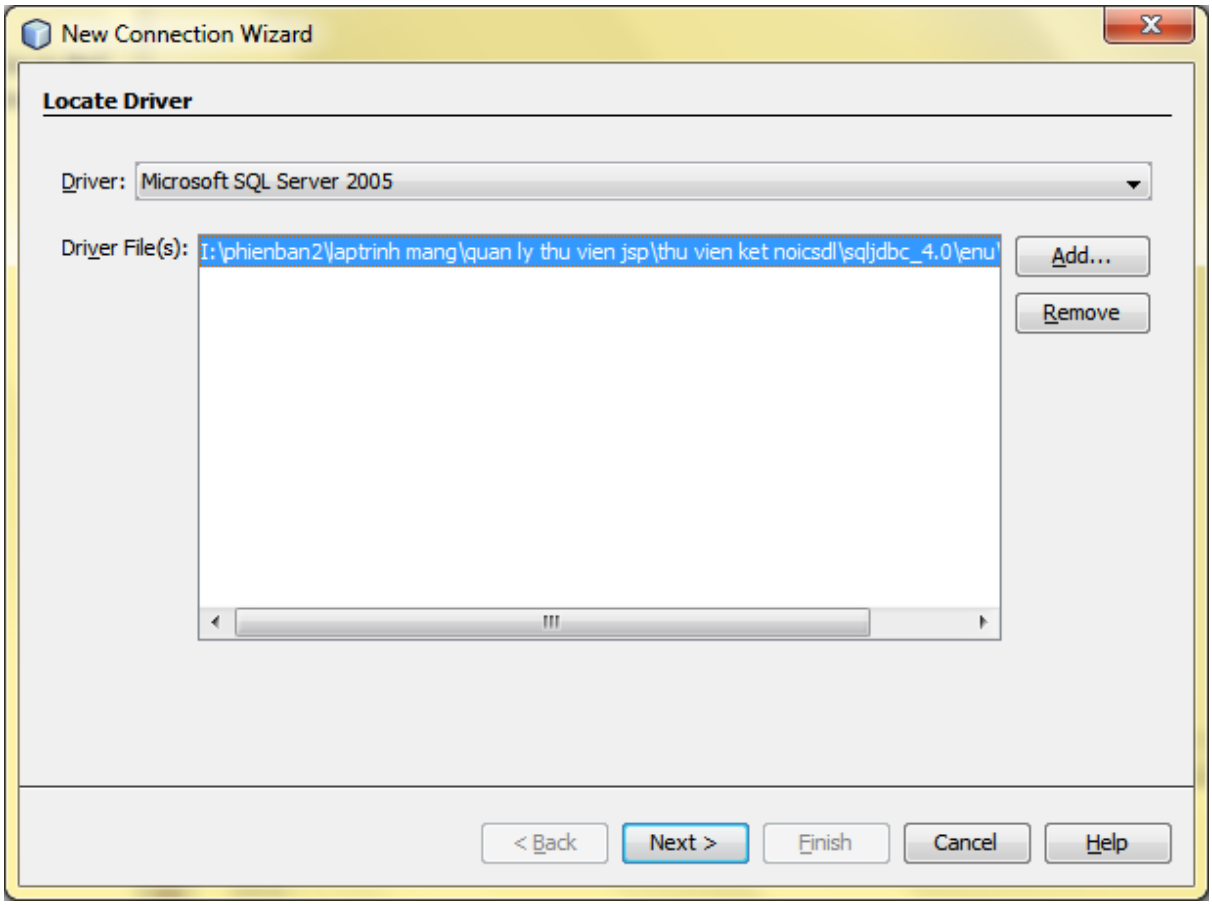
- Chọn new Driver/next



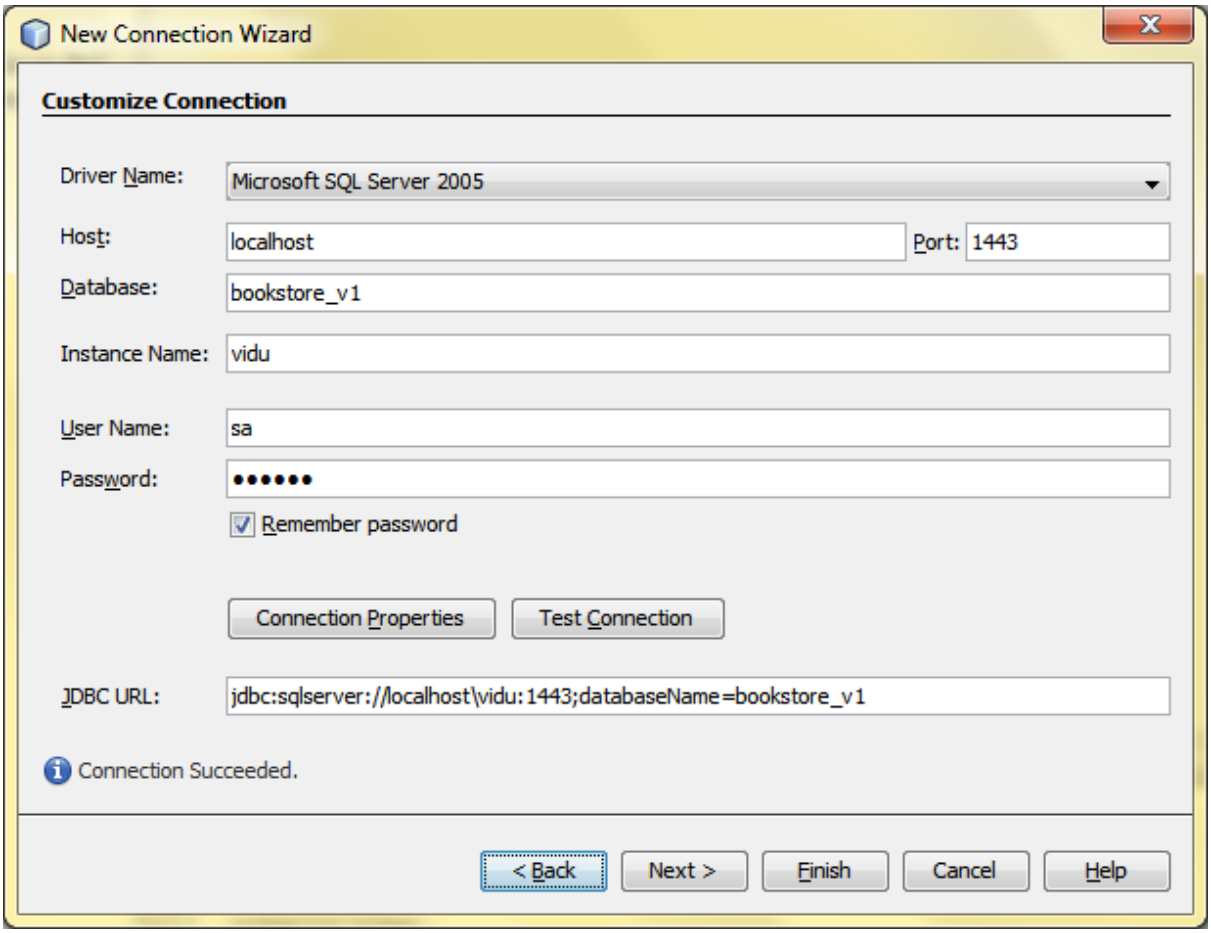
- Chọn add
- Add thư viện kết nối cơ sở dữ liệu /ok



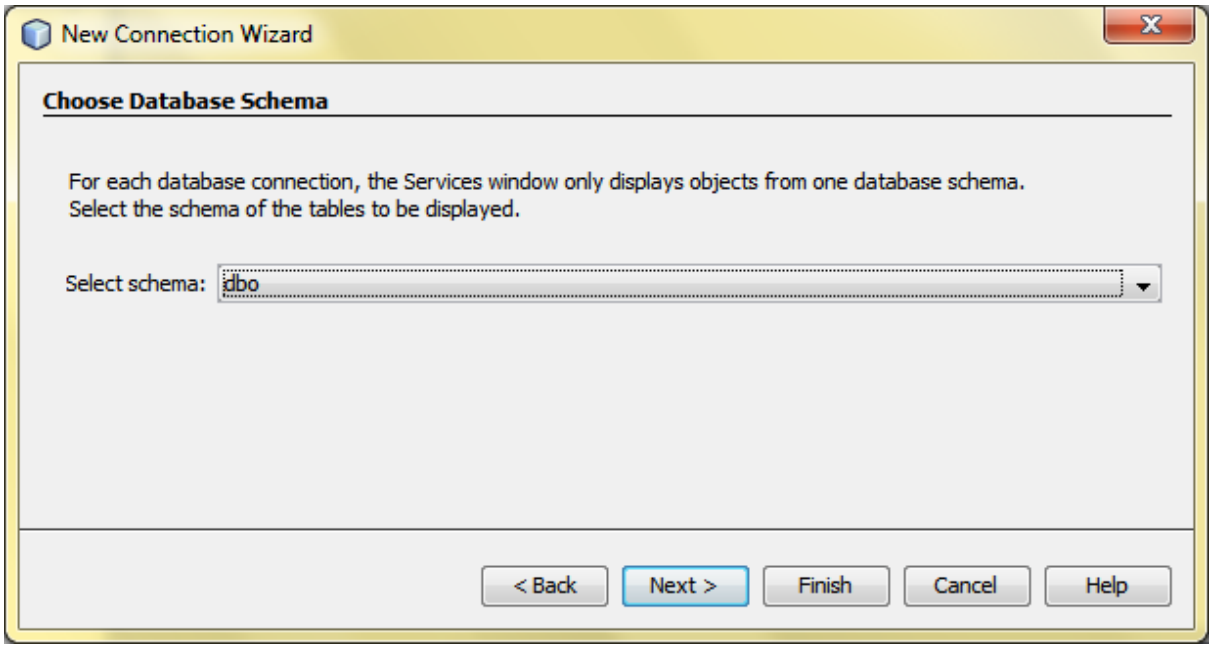
- Chọn sqlserver 2005 trong combobox /next



- Điền thông tin phù hợp như hình dưới
- Chọn testconnection
- Nếu thành công thì chọn next

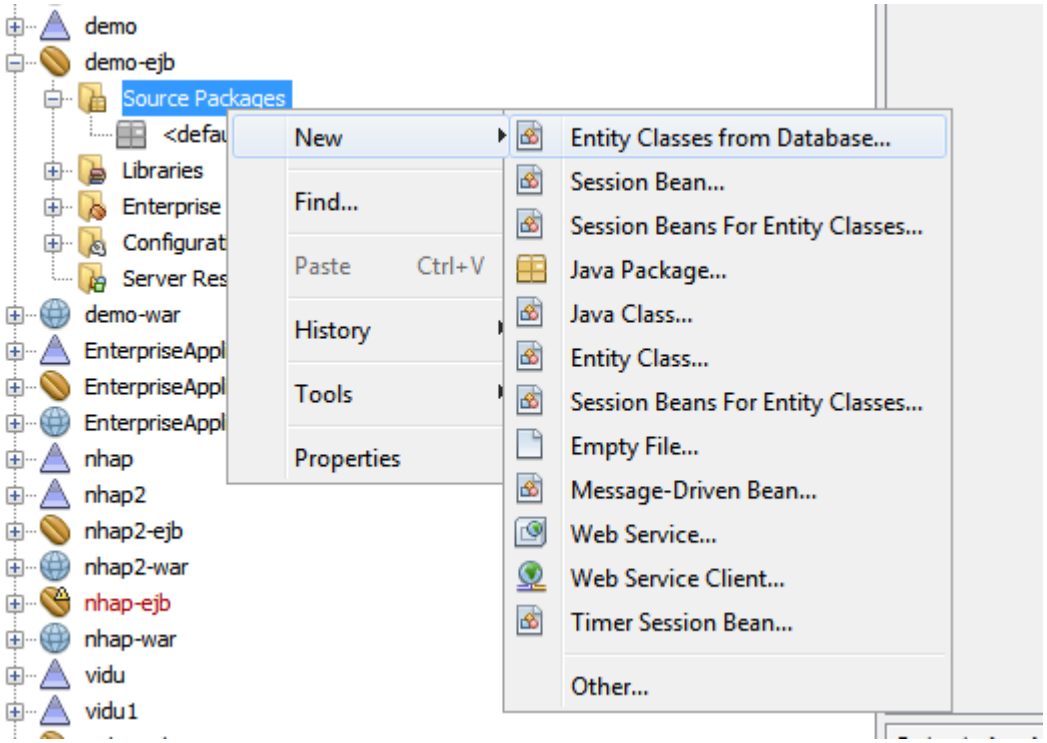


- Chọn dbo/next/finish

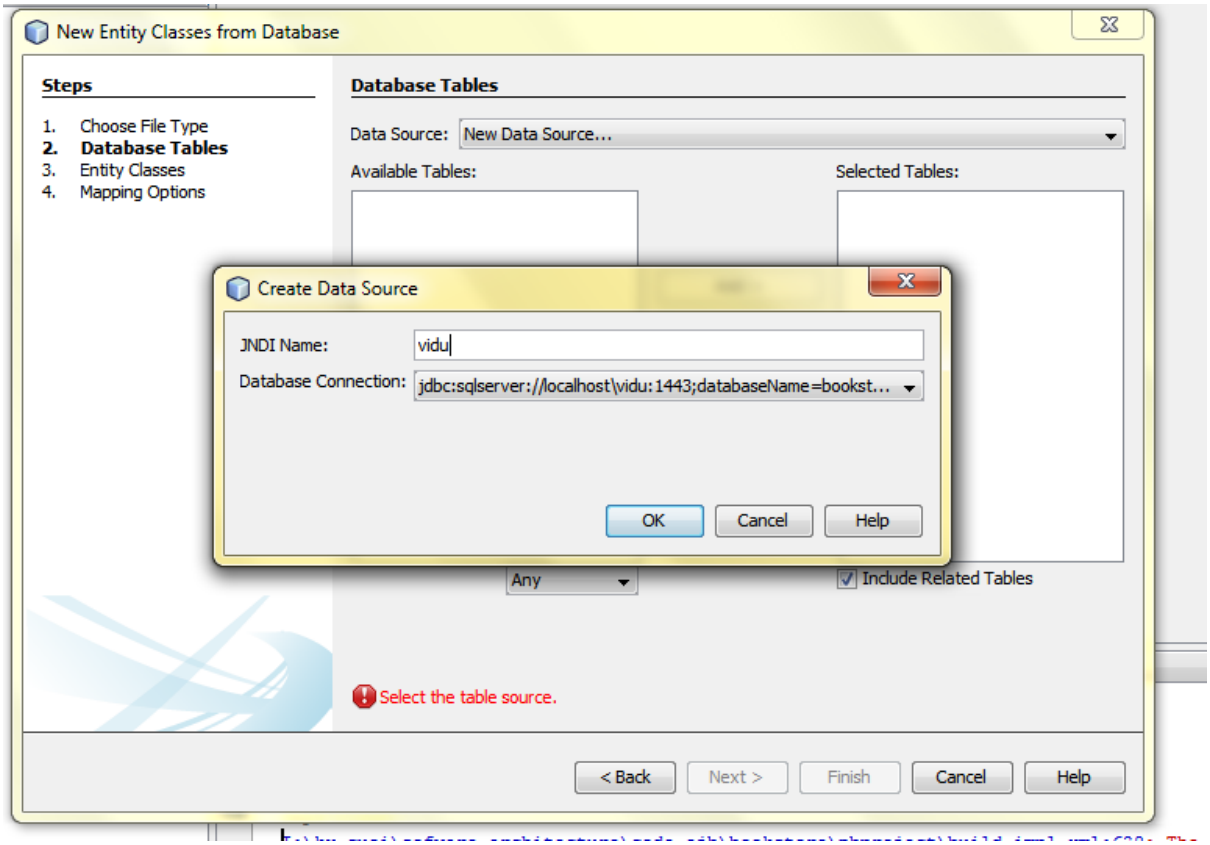


2.3 Tạo entitybean từ database

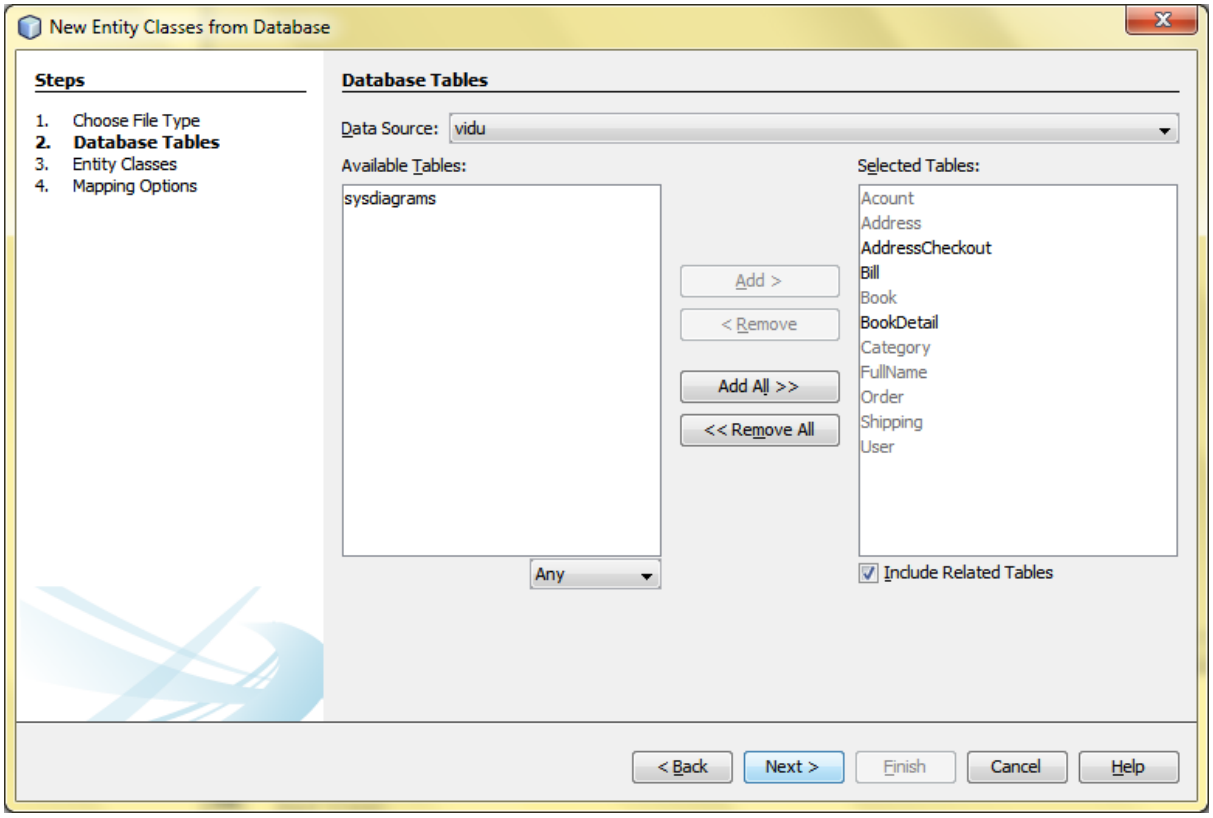
- Chuột phải vào source Packages



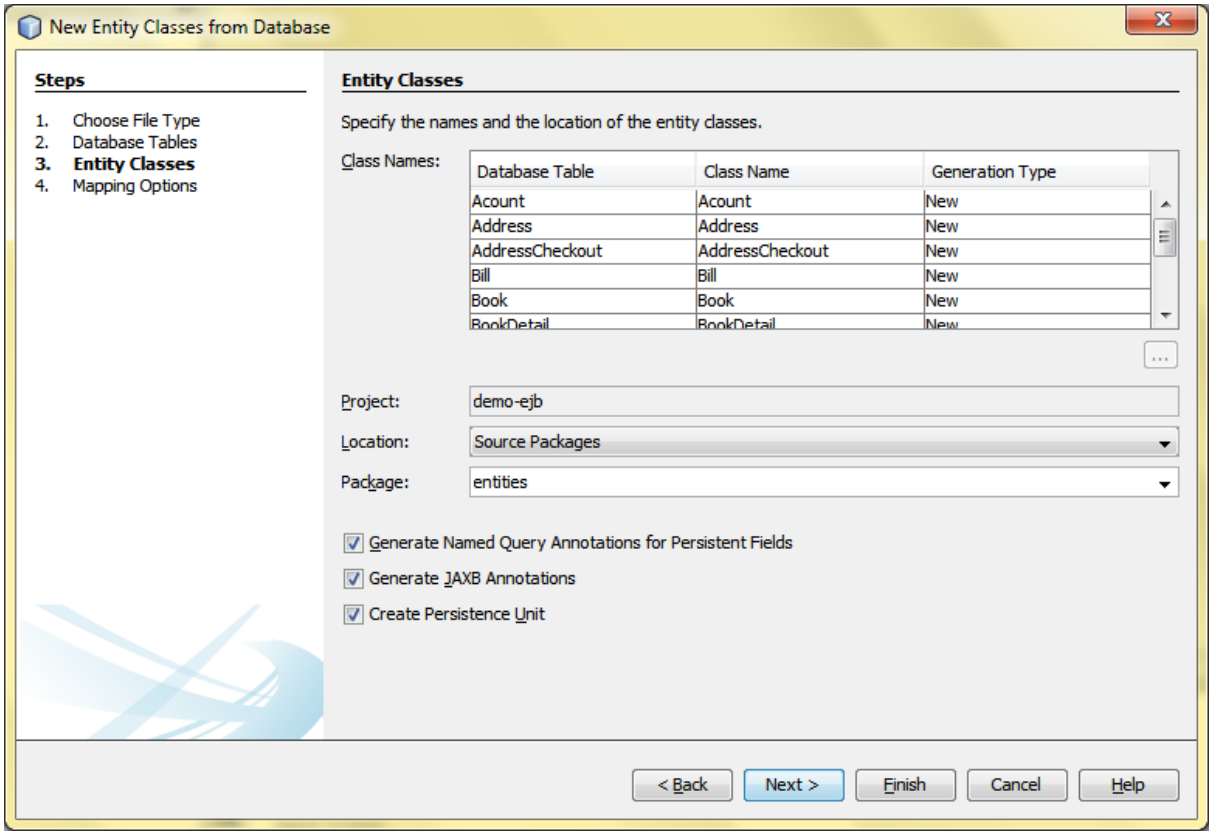
- Chọn new datasource
- Chọn đến kết nối vừa tạo được ở trên
- Đặt tên cho jndi Name



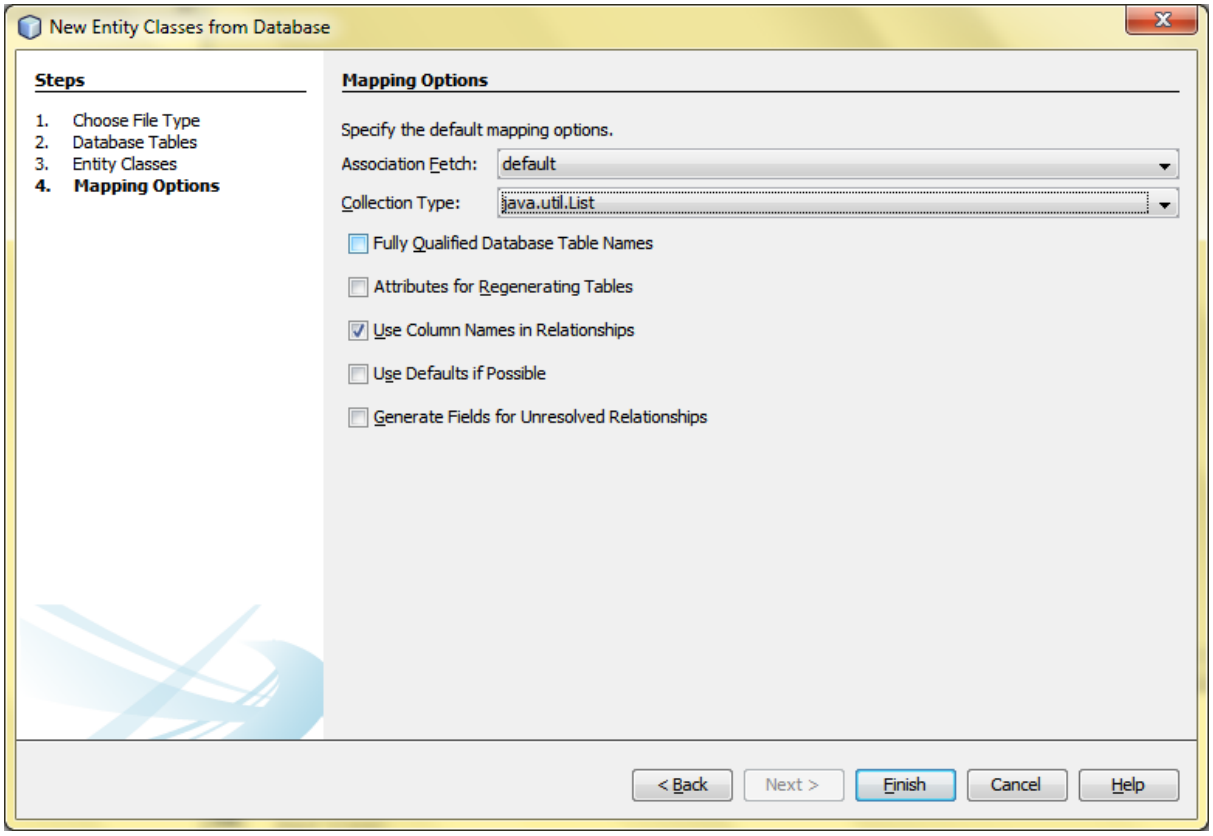
- Add các table mà muốn tạo thành entity bean tương ứng/next



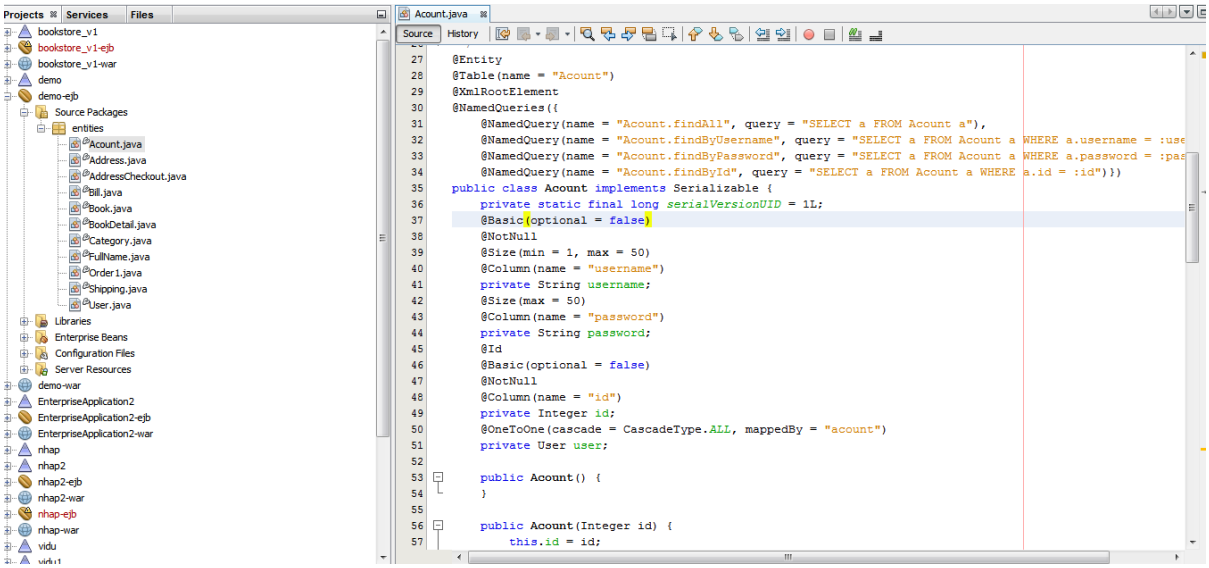
- Đặt tên cho package



- Chọn trong collection type là java.until.List/finish



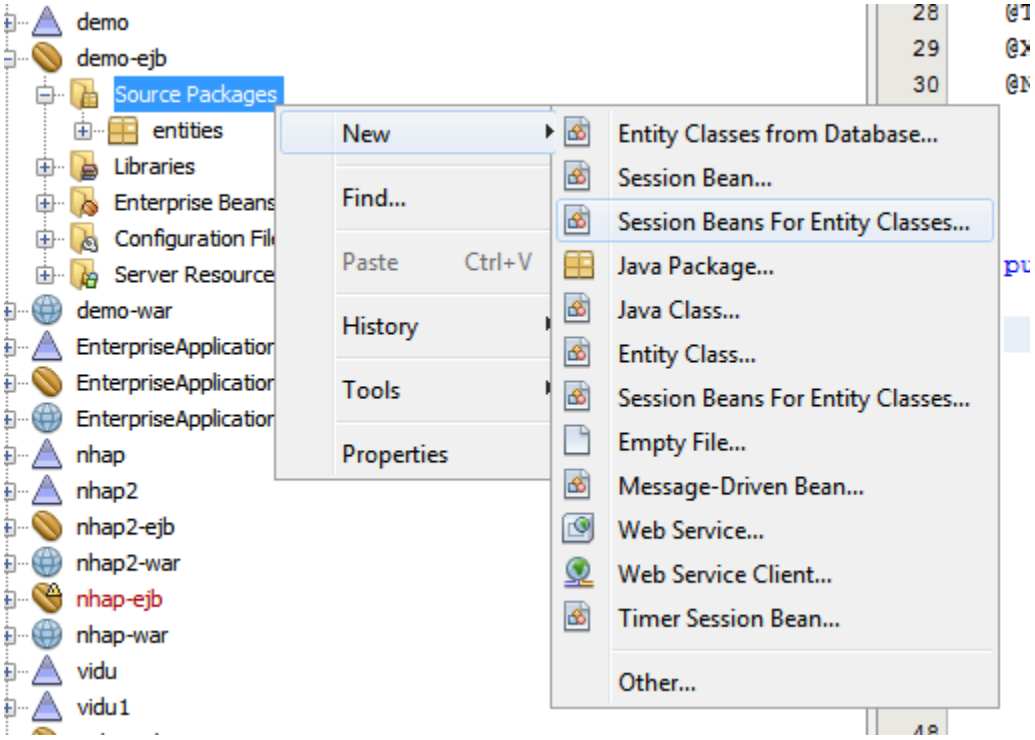
- Kết quả được các lớp thực thể tương ứng
- Nội dung bên trong mỗi lớp thực thể này gồm có tên các câu truy vấn vào cơ sở dữ liệu
- Các thuộc tính và phương thức get/set



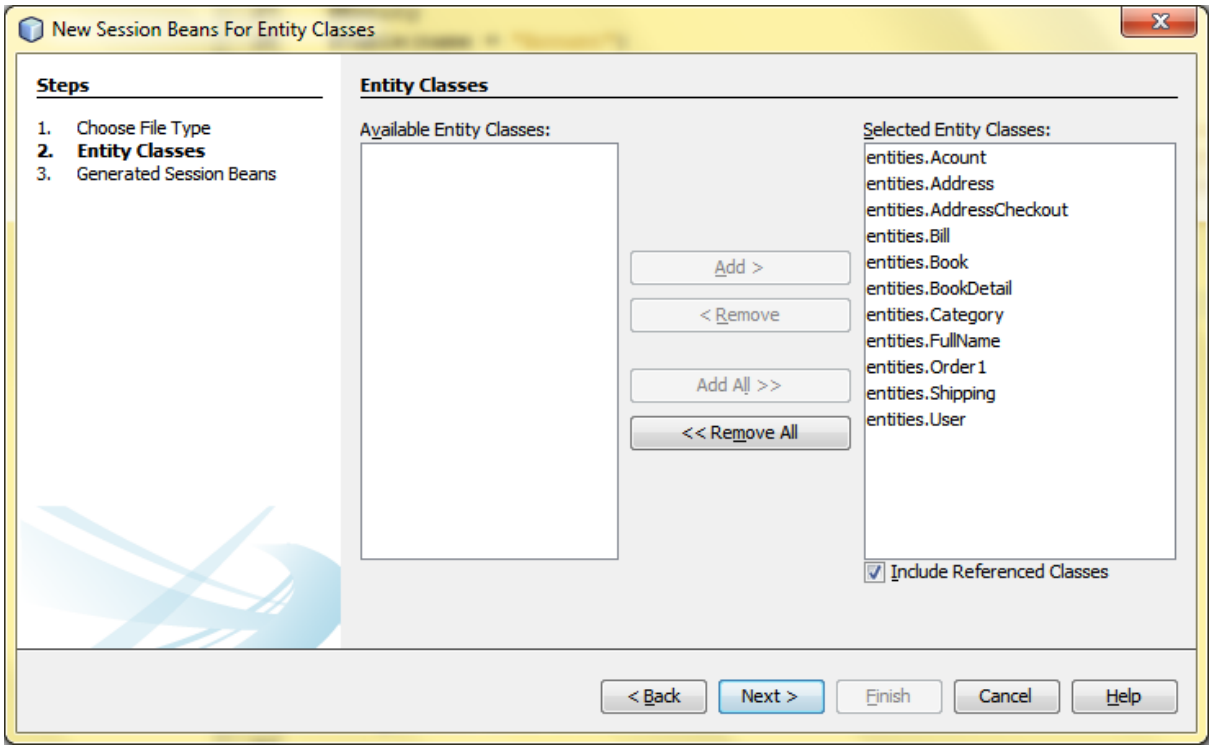
2.4 Tạo sessionbean từ entitybean

Sau khi có entity bean tạo từ cơ sở dữ liệu thì cần phải có các session bean để lưu trữ trạng thái của client bên phía server hoặc chứa các phương thức để thực hiện các câu truy vấn đến cơ sở dữ liệu

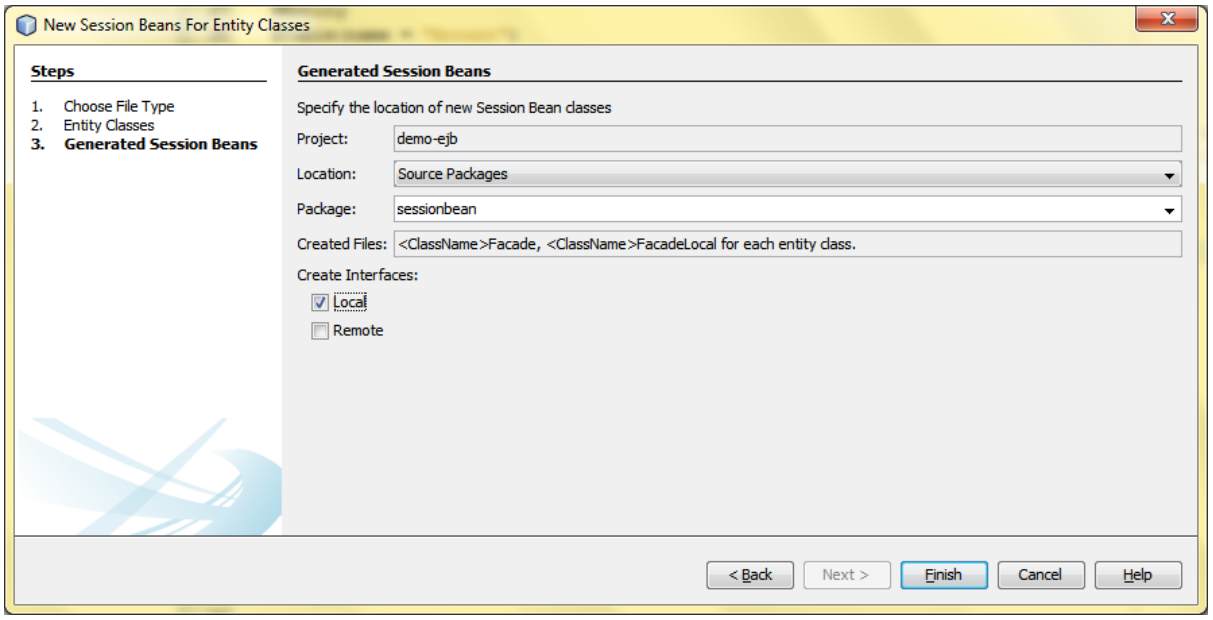
- Chuột phải vào source package
- Chọn new/session bean for entity classes



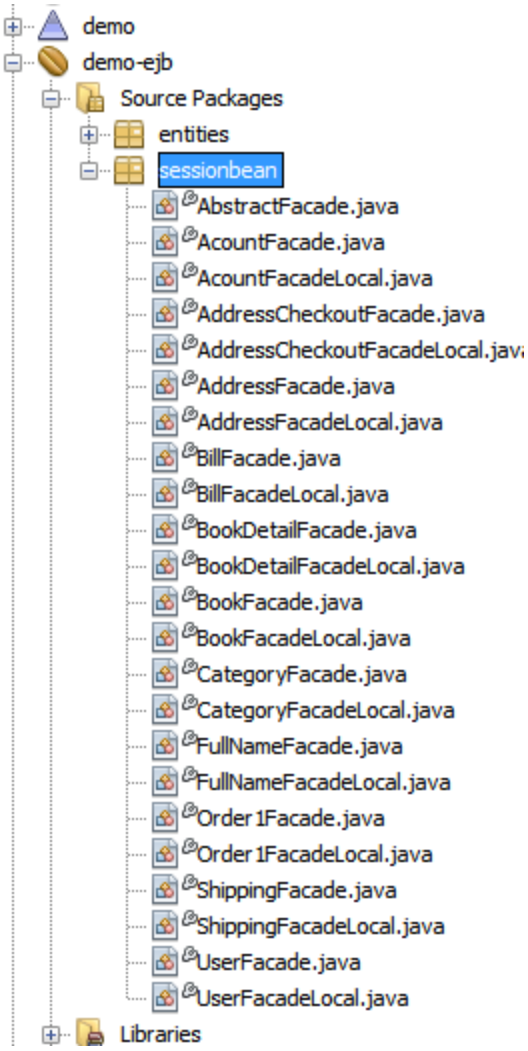
- Add các entity mà muốn tạo session bean sang
- Next



- Đặt tên cho package
- Chọn local hoặc remote tùy vào mục đích sử dụng



- Kết quả đươc như hình bên dưới
- Gồm các interface facadelocal tương ứng với mỗi lớp để chứa tên các phương thức.
- Các lớp façade định nghĩa chi tiết các phương thức trong interface facadelocal
- 1 lớp abstracfacde duuy nhất để định nghĩa các phương thức dấu nhau của các lớp.



2.5 Clean and build /deploy ejb modul

Chuột phải vào ejb modul chọn clean and build

Lean and build xong thì chon deploy

3. Ví dụ ứng dụng hiển thị danh sách thông tin các quyển sách

get full book

name	author	publisher	publishYear	category	quantity	price	select
toan cao cap	tran van ha	NXB giao duc	1998	khoa hoc tu nhien	27	50000	add cart
So so	Vu trong phung	NXB giao duc	1990	van hoc nghe thuat	9	40000	add cart

3.1 Tạo servlet có tên là doBook

- Khai báo biến bookfacelocal có annonation @EJB
- Gọi đến hàm findAll() đã được định nghĩa sẵn.
- Lưu danh sách lấy được vào session và chuyển đến trang jsp để hiển thi

```
package controller;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import entity.*;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.EJB;
import javax.servlet.http.HttpSession;
import sessionBean.*;
@WebServlet(name = "doBook", urlPatterns = {"/doBook"})
public class doBook extends HttpServlet {

    @EJB
    BookFacadeLocal bookDAO;

    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        getFullBook(request, response);

    }

    private void getFullBook(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        HttpSession session = request.getSession();
        session.setAttribute("lisBooks", bookDAO.findAll());
        response.sendRedirect("listBook.jsp");
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }


    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
```

```
            throws ServletException, IOException {
        processRequest(request, response);
    }


    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

3.2 Tạo file jsp listbook hiển thị danh sách sách lấy được

```
<%@page import="entity.*"%>
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>get full book</h1>
        <table border="1">
            <tr>
                <th>name</th>
                <th>author</th>
                <th>publisher</th>
                <th>publishYear</th>
                <th>category</th>
                <th>quantity</th>
                <th>price</th>
                <th>select</th>
            </tr>
            <%
                List<Book> lisBooks = (List<Book>) session.getAttribute("lisBooks");
                for (int i = 0; i < lisBooks.size(); i++) {
            %>
            <tr>
                <td><%=lisBooks.get(i).getName()%></td>
                <td><%=lisBooks.get(i).getAuthor()%></td>
                <td><%=lisBooks.get(i).getPublisher()%></td>
                <td><%=lisBooks.get(i).getPublishYear()%></td>
                <td><%=lisBooks.get(i).getIdCategory().getName()%></td>
                <td><%=lisBooks.get(i).getQuantity()%></td>
                <td><%=lisBooks.get(i).getPrice()%></td>
                <td><a href="doCart?index=<%=i%>">add cart</a></td>
            </tr>
            <%
                }
```

```
        %>
      </table>
   </body>
</html>
```

4. Ví dụ ứng dụng tạo cart

4.1 Tạo 1 interface CartBeanLocal trong package sessionbean

Sau khi có danh sách các quyển sách trong kho hiện lên trang listBook.jsp

Thì ta chọn addcart.

```
package sessionBean;

import entity.Book;
import entity.BookDetail;
import java.util.ArrayList;
import javax.ejb.Local;

@Local
public interface CartBeanLocal {

    void addBookToCart(Book book);

    ArrayList<BookDetail> getCart();

    void removeBookInCart(int index);

}
```

4.2 Tạo 1 CartBean implement từ CartBeanLocal để đĩnh nghĩa các phương thức

- Vì cart phải duy trì trạng thái của client bên phiá server lên phải để trạng thái của cartbean là @stateful
- Cartbean sẽ chứa 1 list các bookdetail của khách hàng tương ứng
- Phương thức addBookToCart() để thêm 1 quyển sách vào cart
- Nếu cart =null thì cartbean sẽ gọi đến hàm khởi tạo init(),sau đó thêm bookdetail vào
- Nếu cart!=null mà quyển sách này đã có trong cart thì ta thay cộng số lượng quyển sách này thêm 1
- Nếu quyển sách này chưa có thì thêm bookdetail với số lượng ban đầu là 1 quyển

```
package sessionBean;
```

```
import entity.Book;
import entity.BookDetail;
import java.util.ArrayList;
import javax.annotation.PostConstruct;
import javax.ejb.Stateful;

@Stateful
public class CartBean implements CartBeanLocal {

    ArrayList<BookDetail> lisBooks;

    @PostConstruct
    void init() {
        lisBooks = new ArrayList<BookDetail>();
    }

    @Override
    public void addBookToCart(Book book) {
        for (int i = 0; i < lisBooks.size(); i++) {
            if (lisBooks.get(i).getIdBook().getId() == book.getId()) {
                lisBooks.get(i).setQuantity(lisBooks.get(i).getQuantity() + 1);
                return;
            }
        }
        BookDetail bd = new BookDetail();
        bd.setIdBook(book);
        bd.setPrice(book.getPrice());
        bd.setQuantity(1);
        lisBooks.add(bd);
    }

    @Override
    public ArrayList<BookDetail> getCart() {
        return lisBooks;
    }

    @Override
    public void removeBookInCart(int index) {
        lisBooks.remove(index);
    }

}
```

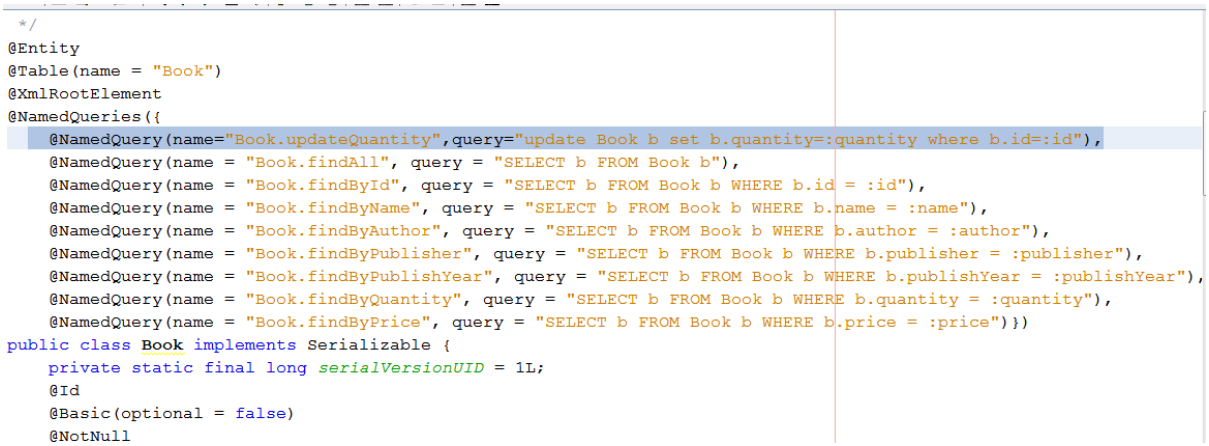
5. Ứng dụng cập nhật số lượng sách

Trong trường hợp này thì không có sẵn phương thức nào định nghĩa để làm chức năng này.

Cách làm

5.1 Viết câu truy vấn

Viết câu truy vấn vào lớp thực thể Book như sau



```
 */
@Entity
@Table(name = "Book")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name="Book.updateQuantity",query="update Book b set b.quantity=:quantity where b.id=:id"),
    @NamedQuery(name = "Book.findAll", query = "SELECT b FROM Book b"),
    @NamedQuery(name = "Book.findById", query = "SELECT b FROM Book b WHERE b.id = :id"),
    @NamedQuery(name = "Book.findByName", query = "SELECT b FROM Book b WHERE b.name = :name"),
    @NamedQuery(name = "Book.findByAuthor", query = "SELECT b FROM Book b WHERE b.author = :author"),
    @NamedQuery(name = "Book.findByPublisher", query = "SELECT b FROM Book b WHERE b.publisher = :publisher"),
    @NamedQuery(name = "Book.findByPublishYear", query = "SELECT b FROM Book b WHERE b.publishYear = :publishYear"),
    @NamedQuery(name = "Book.findByQuantity", query = "SELECT b FROM Book b WHERE b.quantity = :quantity"),
    @NamedQuery(name = "Book.findByPrice", query = "SELECT b FROM Book b WHERE b.price = :price")})
public class Book implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
```

Trong đó cấu trúc:

@NamedQuery(name=”Tên lớp.tên câu truy vấn”,query=”chi tiết câu truy vấn”);

5.2 Thêm phương thức updateQuantity vào interface BookFacadeLocal

```
package sessionBean;

import entity.Book;
import java.util.List;
import javax.ejb.Local;

@Local
public interface BookFacadeLocal {

    void create(Book book);

    void edit(Book book);

    void remove(Book book);

    Book find(Object id);

    List<Book> findAll();

    List<Book> findRange(int[] range);

    int count();
    void updateQuantity(Object id,Object quantity);

}
```

5.3 Định nghĩa phương thức updateQuantity

```
package sessionBean;

import entity.Book;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;


@Stateless
public class BookFacade extends AbstractFacade<Book> implements
BookFacadeLocal {

    @PersistenceContext(unitName = "bookstore_v1-ejbPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public BookFacade() {
        super(Book.class);
    }

    public void updateQuantity(Object id, Object quantity) {

        javax.persistence.Query qr =
getEntityManager().createNamedQuery(Book.class.getSimpleName() +
".updateQuantity");
// truyền các đối số vào
        qr.setParameter("id", id);
        qr.setParameter("quantity", quantity);
        qr.executeUpdate();
    }

}
```

- Cú pháp

```
javax.persistence.Query qr = getEntityManager().createNamedQuery(Tên
lớp.class.getSimpleName() + ".tên câu truy vấn");
```