

Bài thực hành 02

Phát triển Big Web services

1. Mục tiêu

- Tạo ứng dụng Big web services có tương tác với database.
- Cung cấp các chức năng xử lý CRUD với CSDL

2. Bài thực hành Step by Step

Bài thực hành 1: Tạo Big web services tương tác với database để thực hiện xử lý theo các phương thức:

- Get all data
- Get data by id
- Insert
- Update
- Delete

Bước 1: Tạo database cho ứng dụng

```
create database DB_BigWebservice
go
use DB_BigWebservice
go
create table Book(
    Isbn varchar(15) not null primary key,
    BookName nvarchar(100),
    Author nvarchar(70),
    Publisher nvarchar(100),
    YearPubish int,
    Pages int,
    Price float)

insert into Book values ('B01','Hai Van Dam Duoi Bien','Nguyen Thuy Linh','NXB Kim Dong',2019,500,120000)
insert into Book values ('B02','Lap trinh HCJ','Nguyen Duc Nam','NXB Giao Duc',2020,200,50000)
```

Bước 2: Tạo ứng dụng Dynamic Web Project có tên WEBSERV_Lab02_Webservices

File → New → Other → Web → Dynamic Web Project . Đặt tên project

“WEBSERV_Lab02_Webservices”



New Dynamic Web Project

Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location
☐ Use default location
Location:

Target runtime

Dynamic web module version

Configuration

A good starting point for working with Apache Tomcat v9.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership
☐ Add project to an EAR
EAR project name:

Working sets
☐ Add project to working sets
Working sets:

Bước 3: Convert project to maven project

Thêm vào các thư viện

- Hibernate core
- Hibernate entity manager

- Sqljdbc4.jar

Bước 4: Tạo class ánh xạ với bảng trong database

```
package entity;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "Book")
public class Book implements Serializable{
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    @Id
    @Column(name = "Isbn")
    private String isbn;
    @Column(name = "BookName")
    private String bookName;
    @Column(name = "Author")
    private String author;
    @Column(name = "Publisher")
    private String publisher;
    @Column(name = "YearPublish")
    private Integer yearPublish;
    @Column(name = "Pages")
    private Integer pages;
    @Column(name = "Price")
    private Double price;

    public Book() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Book(String isbn, String bookName, String author, String publisher,
        Integer yearPublish, Integer pages,
        Double price) {
        super();
        this.isbn = isbn;
        this.bookName = bookName;
        this.author = author;
        this.publisher = publisher;
        this.yearPublish = yearPublish;
        this.pages = pages;
        this.price = price;
    }

    public String getIsbn() {
```



```
        return isbn;
    }

    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }

    public String getBookName() {
        return bookName;
    }

    public void setBookName(String bookName) {
        this.bookName = bookName;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getPublisher() {
        return publisher;
    }

    public void setPublisher(String publisher) {
        this.publisher = publisher;
    }

    public Integer getYearPublish() {
        return yearPublish;
    }

    public void setYearPublish(Integer yearPublish) {
        this.yearPublish = yearPublish;
    }

    public Integer getPages() {
        return pages;
    }

    public void setPages(Integer pages) {
        this.pages = pages;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }
}
```



Bước 5: Tạo file cấu hình của hibernate

Src / New / Other / Hibernate / Hibernate Configuration File (cfg.xml)

Hibernate Configuration File (cfg.xml)

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

Container: /WEBSERV_Lab02_Webservices/src

File name: hibernate.cfg.xml

Hibernate version: 3.5

Session factory name:

[Get values from Connection](#)

Database dialect: SQL Server

Driver class: com.microsoft.sqlserver.jdbc.SQLServerDriver

Connection URL: jdbc:sqlserver://localhost:1433;databaseName=DB_BigWebservice

Default Schema:

Default Catalog:

Username: sa

Password: 1234\$

☐ Create a console configuration

< Back Next > Finish Cancel

Thêm vào các thông số cấu hình của hibernate

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN"
"http://www.hibernate.org/dtd/hibernate-
configuration-3.0.dtd">
<hibernate-configuration>
```



```
<session-factory>
<property
name="hibernate.connection.driver_class">com.microsoft.sqlserver.jdbc.SQLServerDriver<
/property>
<property name="hibernate.connection.password">1234$</property>
<property
name="hibernate.connection.url">jdbc:sqlserver://localhost:1433;databaseName=DB_BigWeb
service</property>
<property name="hibernate.connection.username">sa</property>
<property
name="hibernate.dialect">org.hibernate.dialect.SQLServer2008Dialect</property>
<property name="hibernate.show_sql">true</property>
<property name="hibernate.current_session_context_class">thread</property>
<property name="hibernate.use_sql_comments">true</property>
<mapping class="entity.Book"/>
</session-factory>
</hibernate-configuration>
```

Bước 6: Tạo file HibernateUtil.java

```
package util;

import org.hibernate.SessionFactory;
import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.service.ServiceRegistry;

public class HibernateUtil {
    private static SessionFactory sessionFactory;
    static {
        if(sessionFactory==null) {
            ServiceRegistry reg = new
StandardServiceRegistryBuilder().configure().build();
            MetadataSources source = new MetadataSources(reg);
            Metadata metadata = source.getMetadataBuilder().build();
            sessionFactory = metadata.getSessionFactoryBuilder().build();
        }
    }
    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Bước 7: Tạo giao diện BookDAO.java

```
package dao;

import java.util.List;

import entity.Book;

public interface BookDAO {
    public List<Book> getListBooks();
    public Book getBookById(String isbn);
}
```



```
public boolean insertBook(Book book);  
public boolean updateBook(Book book);  
public boolean deleteBook(String isbn);  
}
```

Bước 8 : Tạo class BookDAOImpl.java

```
package dao.impl;  
  
import java.util.List;  
  
import org.hibernate.Session;  
  
import dao.BookDAO;  
import entity.Book;  
import util.HibernateUtil;  
  
public class BookDAOImpl implements BookDAO{  
  
    @Override  
    public List<Book> getListBooks() {  
        Session session = HibernateUtil.getSessionFactory().openSession();  
        try {  
            session.beginTransaction();  
            List list = session.createQuery("from Book").list();  
            session.getTransaction().commit();  
            return list;  
        } catch (Exception e) {  
            e.printStackTrace();  
            session.getTransaction().rollback();  
        } finally {  
            session.close();  
        }  
        return null;  
    }  
  
    @Override  
    public Book getBookById(String isbn) {  
        Session session = HibernateUtil.getSessionFactory().openSession();  
        try {  
            session.beginTransaction();  
            Book book = session.get(Book.class, isbn);  
            session.getTransaction().commit();  
            return book;  
        } catch (Exception e) {  
            e.printStackTrace();  
            session.getTransaction().rollback();  
        } finally {  
            session.close();  
        }  
        return null;  
    }  
  
    @Override  
    public boolean insertBook(Book book) {  
        Session session = HibernateUtil.getSessionFactory().openSession();  
        try {
```




```
        session.beginTransaction();
        session.save(book);
        session.getTransaction().commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        session.getTransaction().rollback();
    } finally {
        session.close();
    }
    return false;
}

@Override
public boolean updateBook(Book book) {
    Session session = HibernateUtil.getSessionFactory().openSession();
    try {
        session.beginTransaction();
        session.update(book);
        session.getTransaction().commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        session.getTransaction().rollback();
    } finally {
        session.close();
    }
    return false;
}

@Override
public boolean deleteBook(String isbn) {
    Session session = HibernateUtil.getSessionFactory().openSession();
    try {
        session.beginTransaction();
        session.delete(getBookById(isbn));
        session.getTransaction().commit();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        session.getTransaction().rollback();
    } finally {
        session.close();
    }
    return false;
}
}
```

Bước 9: Tạo file BookService.java

```
package service;

import java.util.List;
```





```
import dao.impl.BookDAOImpl;
import entity.Book;

public class BookService {
    public Book[] getListBooks(){
        List<Book> listBooks = new BookDAOImpl().getListBooks();
        Book[] arr = new Book[listBooks.size()];
        listBooks.toArray(arr);
        return arr;
    }
    public Book getBookById(String isbn) {
        return new BookDAOImpl().getBookById(isbn);
    }
    public boolean insertBook(Book book) {
        return new BookDAOImpl().insertBook(book);
    }
    public boolean updateBook(Book book) {
        return new BookDAOImpl().updateBook(book);
    }
    public boolean deleteBook(String isbn) {
        return new BookDAOImpl().deleteBook(isbn);
    }
}
```

Bước 10: Tạo Big Web services từ class BookService.java

Right click BookService.java / New / Other / Web services / Web service




 Web Service

Web Services
Select a service implementation or definition and move the sliders to set the level of service and client generation.

Web service type: Bottom up Java bean Web Service

Service implementation: service.BookService Browse...


Test service



Configuration:
[Server runtime: Tomcat v9.0 Server](#)
[Web service runtime: Apache Axis \(Deprecated\)](#)
[Service project: WEBSERV_Lab02 Webservices](#)

Client type: Java Proxy

No client




Configuration: No client generation.

☐ Publish the Web service

☐ Monitor the Web service

☒ Overwrite files without warning



< Back Next > Finish Cancel

Web Service

Web Service Java Bean Identity

Configure the Java bean as a Web service.

WSDL file:

Methods

- ☒ deleteBook(java.lang.String)
- ☒ getListBooks()
- ☒ getBookById(java.lang.String)
- ☒ updateBook(entity.Book)
- ☒ insertBook(entity.Book)

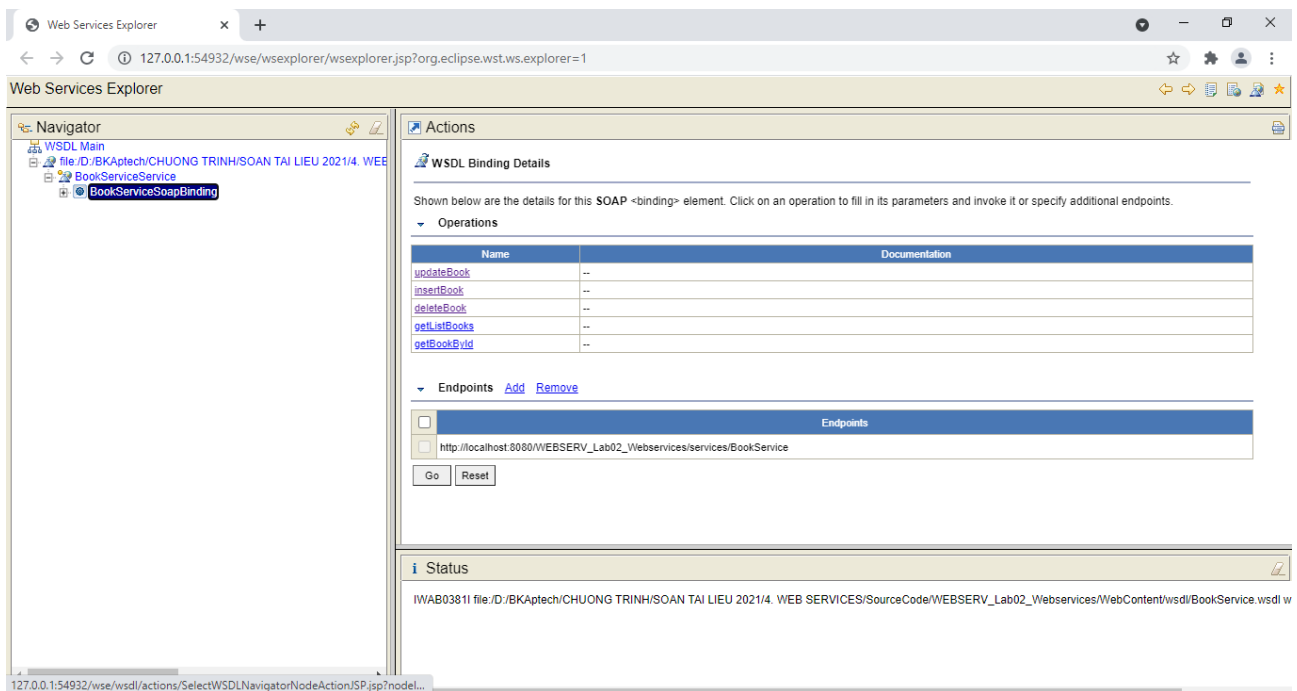
Style and use

- ☒ document/literal (wrapped)
- ☐ document/literal
- ☐ RPC/encoded

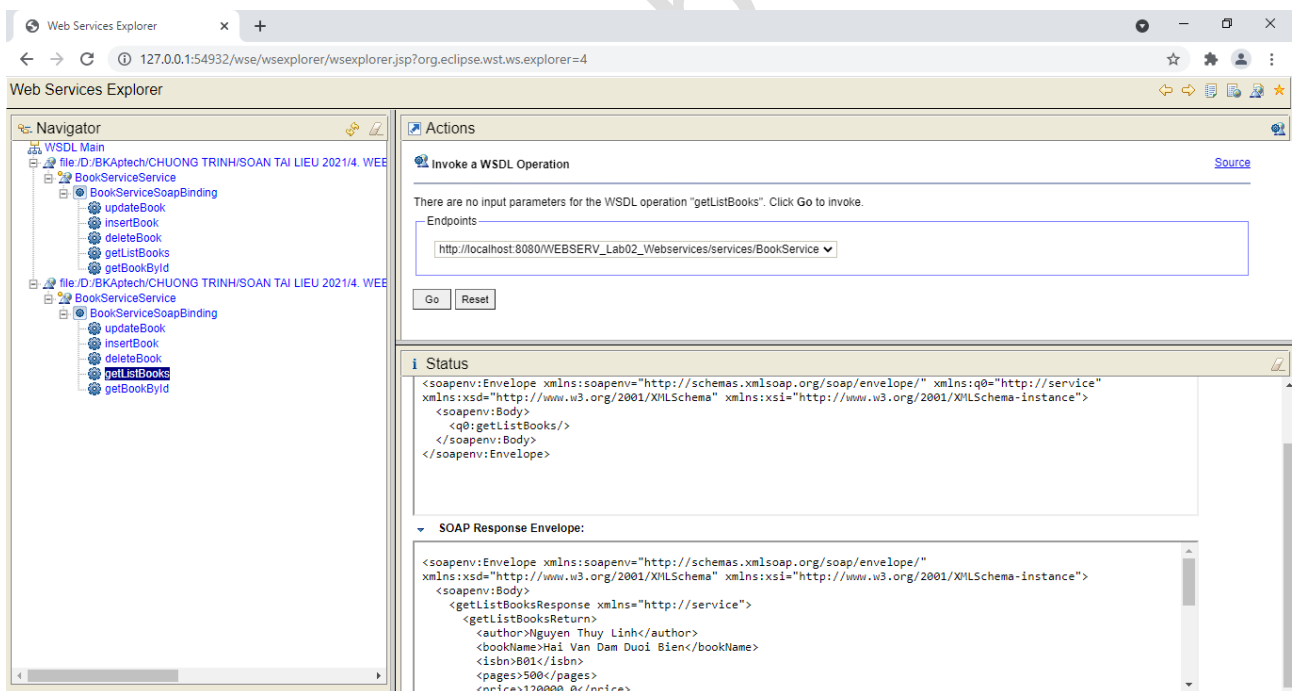
☐ Define custom mapping for package to namespace.

Big Web services được tạo thành công





Chạy thử chức năng getListBooks



3. Bài tập tự làm

Bài 1. Tạo database như sau



Tên cột	Kiểu dữ liệu	Mô tả
CusId	Int identity	Khoá chính
CusName	Nvarchar(70)	
Gender	Bit	
Birthday	Datetime	
Address	Nvarchar(200)	
Email	Varchar(100)	
Telephone	Varchar(15)	

Tạo một Big web services cung cấp các phương thức xử lý sau:

- Get list customers
- Get customer by id
- Insert customer
- Update customer
- Delete customer

