MyEclipse 5.0GA에서 SSH(Struts Spring Hibernate)연동하기

Email : yustit골벵이gmail.com

MSN : lgmoim골벵이hotmail.com

개발환경 및 사용한 도구들 :

개발언어 : Java(J2SDK 1.5.08)

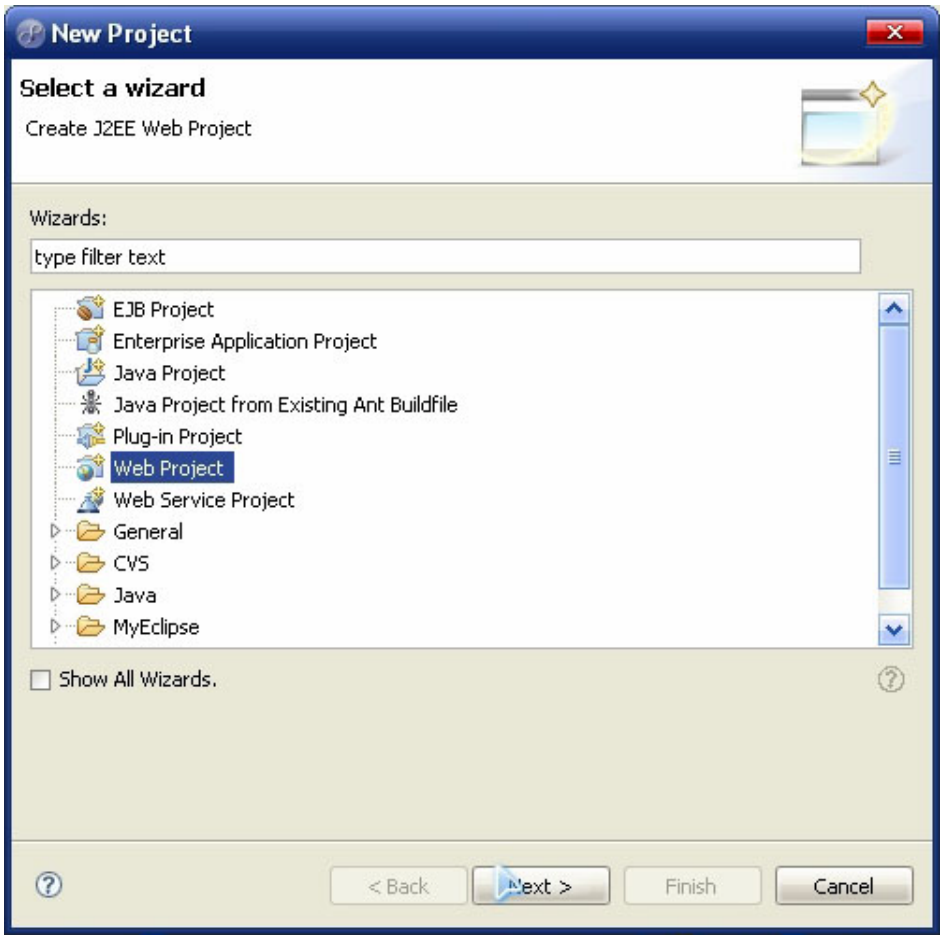
데이타베이스 : MySQL(5.0.24)

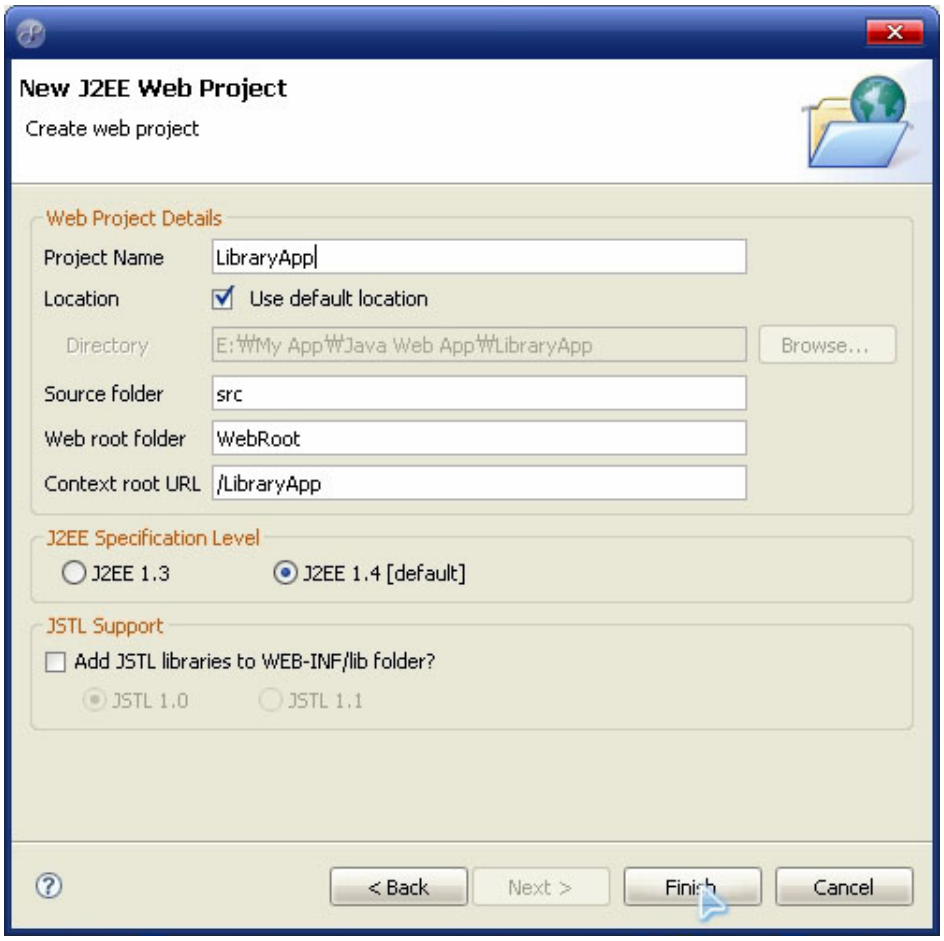
JDBC: mysql-connector-java-5.0.3-bin

WAS (Web Application Server) : Tomcat 5.0.28

IDE : Eclipse 3.2 With MyEclipse 5.0GA

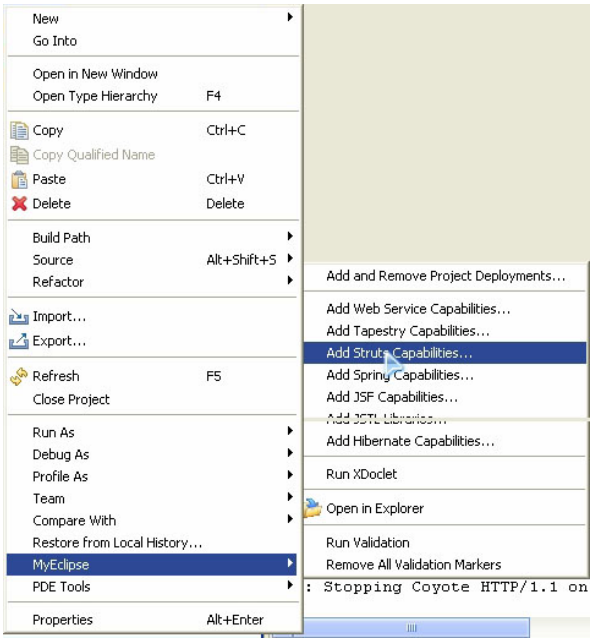
1. 새 프로젝트를 만듭니다. 프로젝트명은 LibraryApp로 합니다.



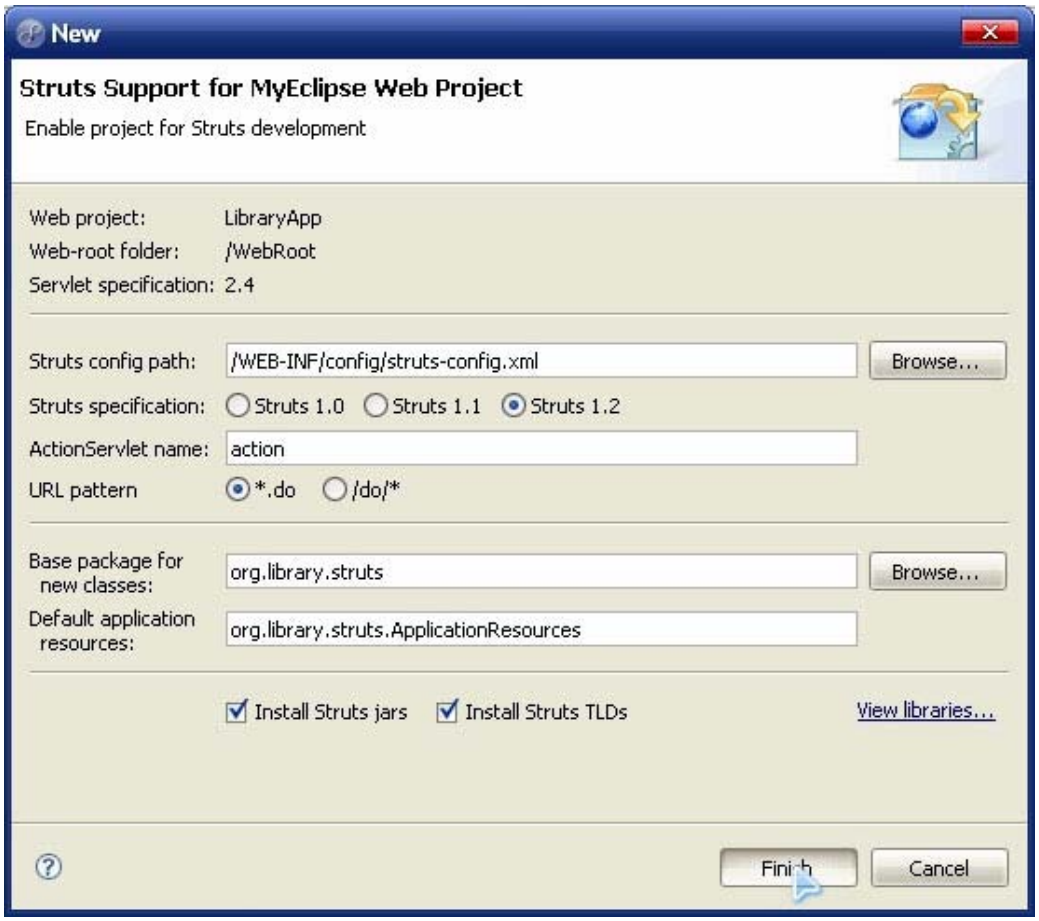


2. 먼저 Struts FrameWork을 추가해보도록 하겠습니다.

생성한 프로젝트에서 오른쪽 마우스를 클릭하여 MyEclipse => Struts Capabilities를 클릭합니다. 아래 그림에서와 같이…



그러면 Struts설치 관련 창이 뜹니다. 여기서 아래 그림에서와 같이 설정하고 Finish를 클릭하면 Struts FrameWork이 설치됩니다. 여기서는 사용한 Struts 버젼은 1.2.9인데 지금까지 나온 최신버젼 1.3.x도 사용가능합니다. 그리고 struts form과 action이 기본으로 생성될 패키지를 지정합니다. 간편하죠?



3. index.jsp 메인페이지를 만들어줍니다.

Struts Framework을 구성하였으니 필요한 페이지를 만들어보도록 하겠습니다. 편의상 저는 index.jsp, bookList.jsp 그리고 필요한 FormBean과 ActionForm을 추가하도록 하겠습니다.

WebRoot아래에 index.jsp 파일을 생성해주시고 내용은 다음과 같이 합니다.

```
<%@ page language="java"%>
<%@ taglib uri=http://struts.apache.org/tags-logic prefix="logic" %>
<logic:forward name="welcome"/>
```

이페이지에서는 요청을 받으면 welcome페이지로 Forwarding하는 작용을 합니다.

Welcome Forwarding은 차후에 struts-config.xml에서 추가합니다.

다음 welcome으로 포워딩했을때 이동할 페이지를 추가하는데 파일이 위치할 경로는 /WebRoot/JSP/이고요 파일명은 index.jsp입니다.

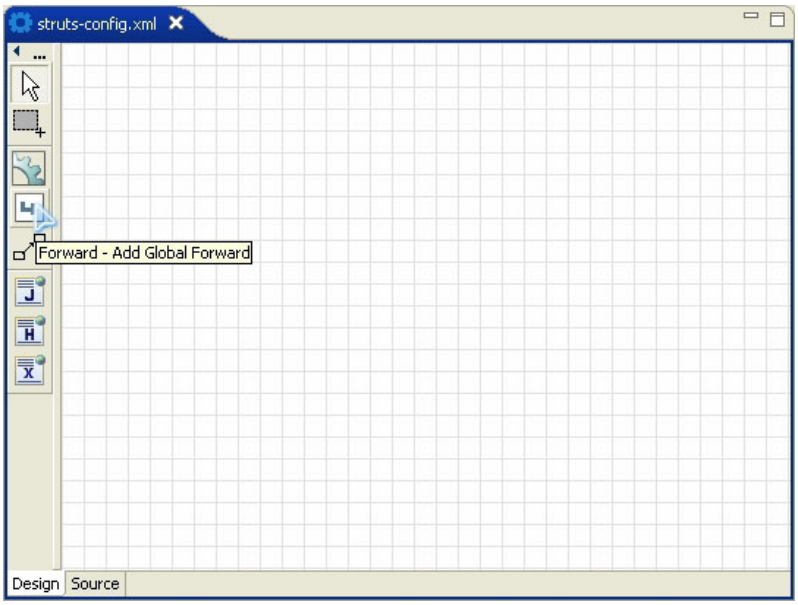
index.jsp 파일내용은 다음과 같습니다.

```
<%@ page language="java"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic" prefix="logic" %>
```

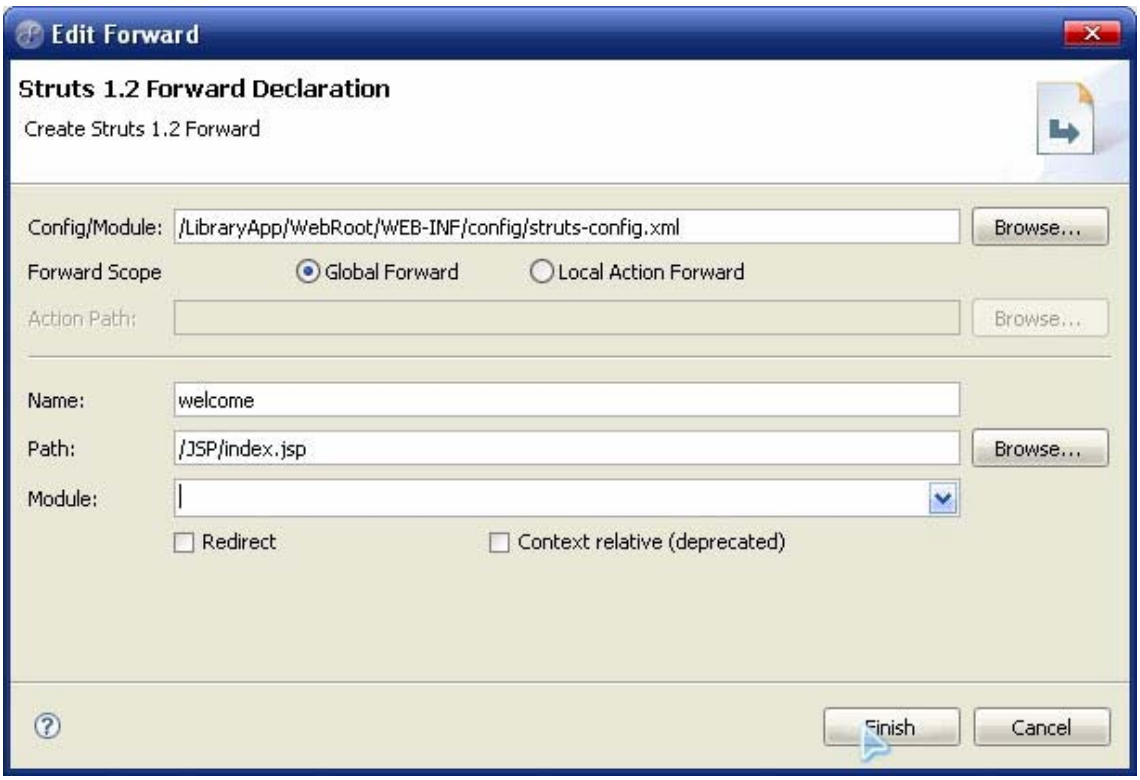
```
<html>
	<head>
		<title>Welcome to Library Web</title>
	</head>
	<body>
  <body>
    Welcome!
    <br>
    <html:link action="bookList">Show the book list</html:link>
    <br>
    <html:link action="customerList">Show the customer list</html:link>
  </body>
</html>
```

다음 생성한 index.jsp파일에 포워딩을 설정해줍니다.

Struts-config.xml을 열면 아래와 같이 공백 Grid창이 뜨는데 Forward버튼을 클릭하여 필요한 Global Forward를 추가합니다.



아래 그림에 보시는것 처럼 Name에는 welcome 그리고 Path에는 /JSP/index.jsp를 입력합니다.(옆의 Browse를 클릭하여서 찾아도 됩니다.)



다음 bookList.jsp 파일을 WebRoot/JSP/폴더에 생성합니다.

bookList.jsp파일 내용은 다음과 같습니다.

```
<%@ page language="java"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic" prefix="logic" %>

<html>
	<head>
		<title>Show book list</title>
	</head>
	<body>

	<table border="1">
	<tbody>
	<%-- set the header --%>
	<tr>
		<td>Author</td>
```

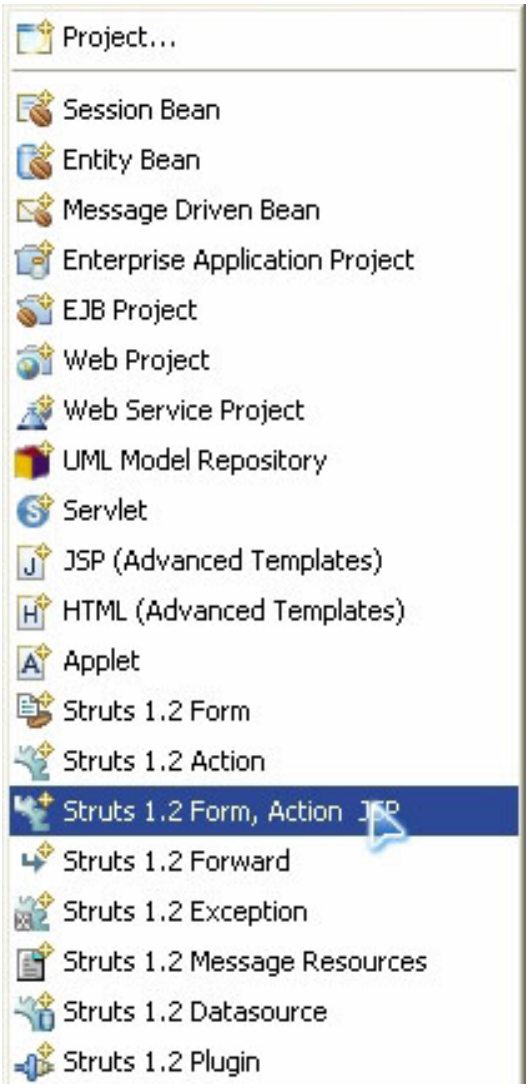
```
            <td>Book name</td>
            <td>Available</td>
            <td>Borrow by</td>
            <td> </td>
            <td> </td>
            <td> </td>
</tr>
<%-- start with an iterate over the collection books --%>
<logic:iterate name="books" type="org.library.bean.Book" id="book">
<tr>
            <%-- book informations --%>
            <td><bean:write name="book" property="author" /></td>
            <td><bean:write name="book" property="title" /></td>
            <td><html:checkbox disabled="true"
                                                name="book"
                                                property="borrowallowed"/>
            </td>
            <td>
                        <%-- check if a customer borrowed a book,
                                    when its true display his name
                                    otherwise display nothing --%>
                        <logic:notEmpty name="book" property="customer">
                                    <bean:write name="book" property="customer.name" />,
                                    <bean:write name="book" property="customer.lastname" />
                        </logic:notEmpty>
                        <logic:empty name="book" property="customer">
                                    -
                        </logic:empty>
            </td>
            <%-- borrow, edit and delete link for each book --%>
            <td>
                        <%-- check if a user borrowed a book,
                                    when its true display the return link
                                    otherwise display the borrow link --%>
                        <logic:notEmpty name="book" property="customer">
                                    <html:link action="bookEdit.do?do=returnBook"
```

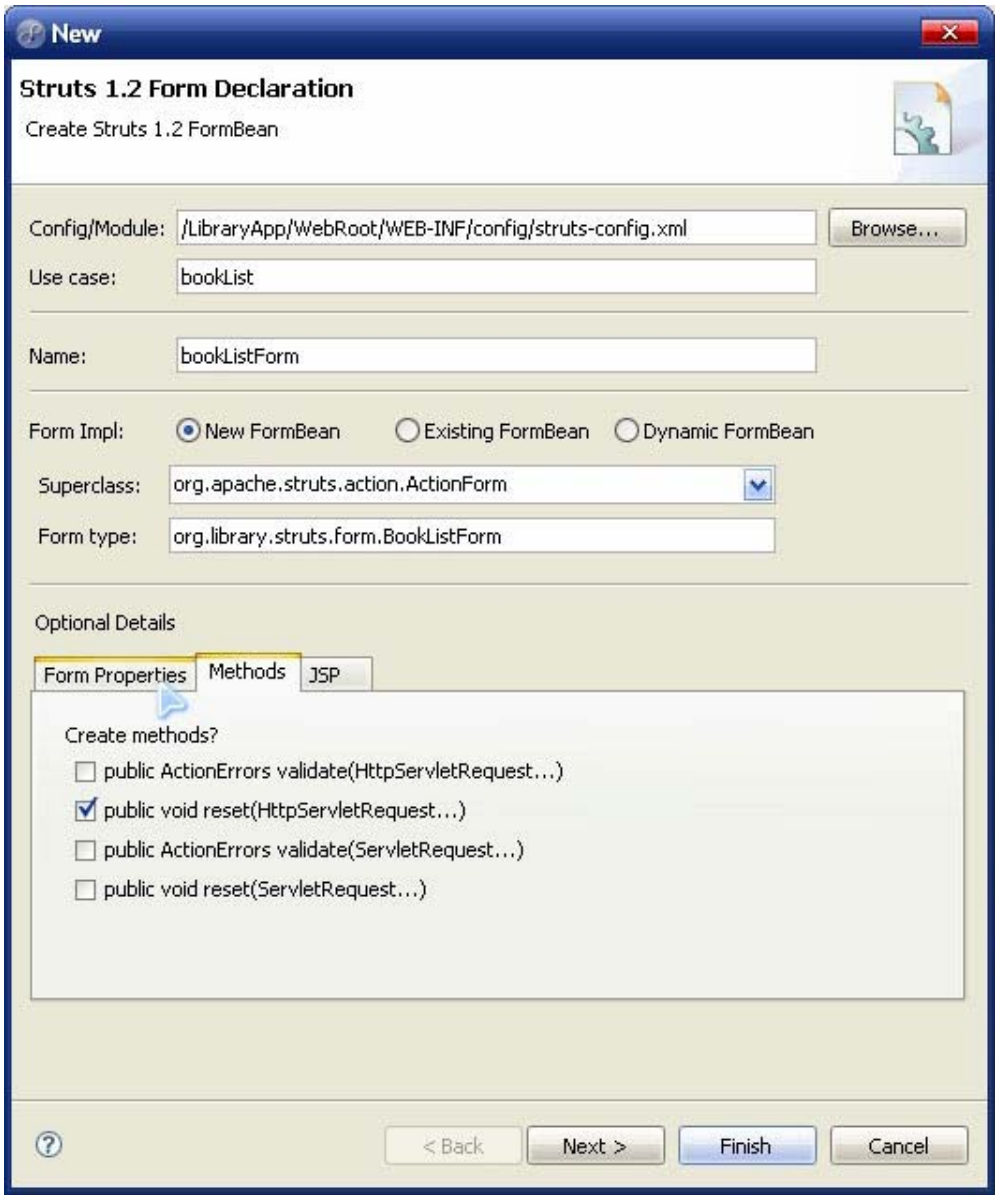
```
                                                    paramName="book"
                                                    paramProperty="id"
                                                    paramId="id">Return
book</html:link>
                        </logic:notEmpty>
                        <logic:empty name="book" property="customer">
                                <html:link action="bookEdit.do?do=borrowBook"
                                                    paramName="book"
                                                    paramProperty="id"
                                                    paramId="id">Borrow
book</html:link>
                        </logic:empty>
                </td>
                <td><html:link action="bookEdit.do?do=editBook"
                                                paramName="book"
                                                paramProperty="id"
                                                paramId="id">Edit</html:link>
                </td>
                <td><html:link action="bookEdit.do?do=deleteBook"
                                                paramName="book"
                                                paramProperty="id"
                                                paramId="id">Delete</html:link>
                </td>
        </tr>
        </logic:iterate>
        <%-- end interate --%>

        <%-- if books cannot be found display a text --%>
        <logic:notPresent name="book">
                <tr>
                        <td colspan="5">No books found.</td>
                </tr>
        </logic:notPresent>

        </tbody>
        </table>
```

```
        <br>
        <%-- add and back to menu button --%>
        <html:button property="add"
                                        onclick="location.href='bookEdit.do?do=addBook'">Add a new book
        </html:button>
         
        <html:button property="back"
                                        onclick="location.href='default.do'">Back to menu
        </html:button>
        </body>
</html>
```

그리고 bookList관련 Form과 Action을 만들어줍니다.

Struts 1.2 Form,Action JSP 를 클릭하여 관련 Form과 Action을 만듭니다.



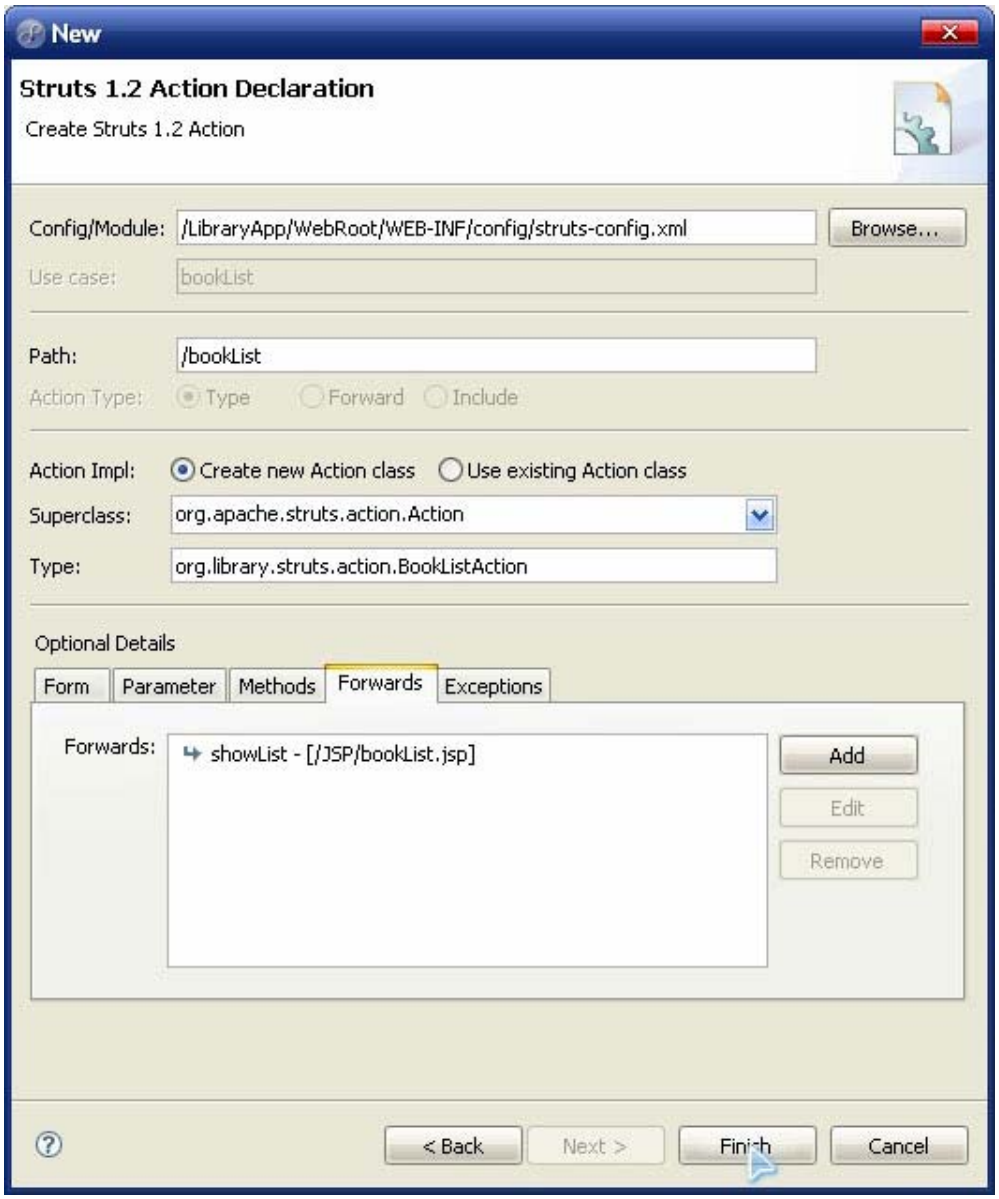


ActionErrors에 있는 체크는 해제합니다.

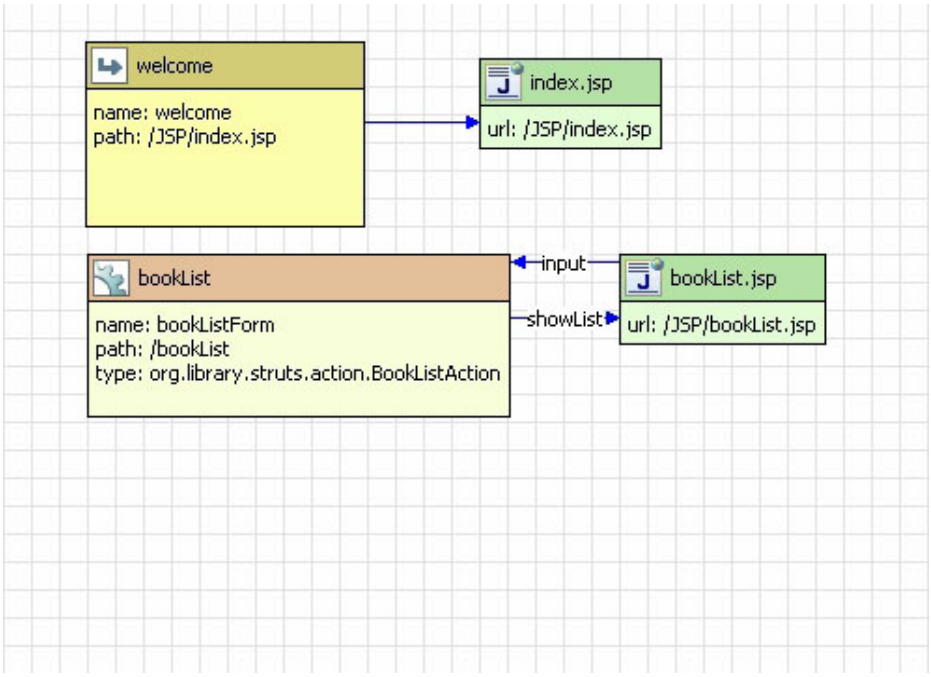
다음 Action 을 구성하기 위하여 Next를 클릭합니다.

단 유의해야할것은 Forwards에서 showList 포워즈를 추가해줍니다.

이것은 BookListAction에서 책목록을 DB에서 Fetch한후 넘겨줄 페이지를 미리 지정하기 위해서입니다.



Form과 Action 및 JSP페이지를 다 만들어 주셨으면 아마 아래와 같은 Struts-config.xml파일이 생길겁니다.



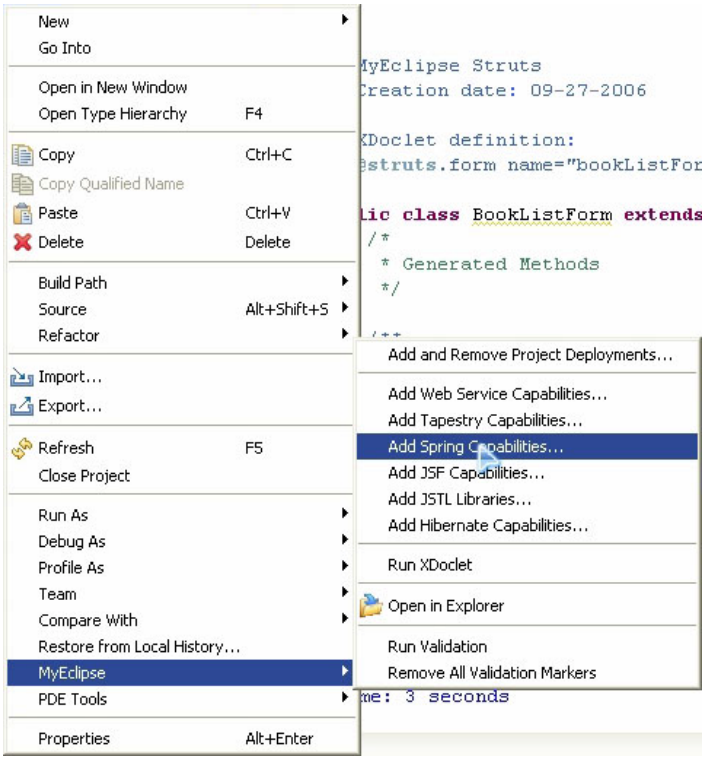
4. 초보적 테스트를 하여봅니다. 전체 프로젝트를 빌드하기 위하여 ant빌드 설정을 해줍니다. build.xml파일은 기존의 tomcat의 샘플을 그대로 복사하여서 쓸수있습니다. Tomcat 5.0=>webapps=>tomcat-docs=>appdev=>sample 안의 build.xml파일을 그대로 복사하여 project 루트 디렉토리에 넣는다. 다음 build.properties를 만드시고 관리자 아이디와 비밀번호를 적어넣습니다..

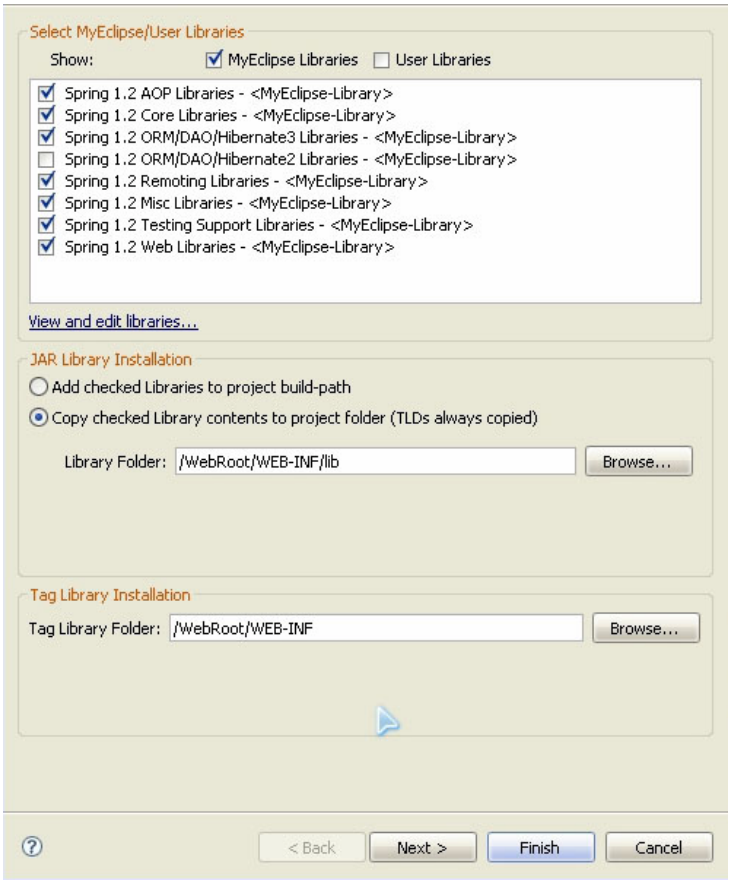
   manager.username = id

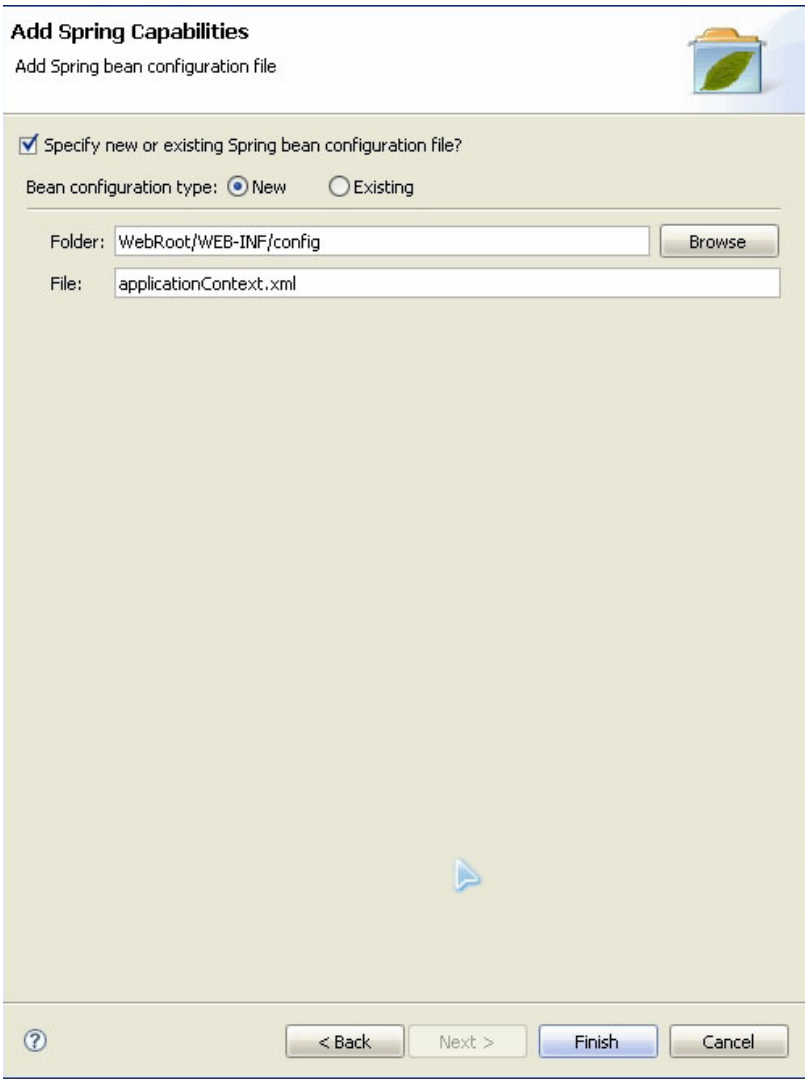
   manager.password = pw

   다음 build.xml파일을 여시고 app.name에는 project 이름 즉 LibraryApp를 적으시고 web.home에는 WebRoot를 적습니다. 다음 target을 install로 설정하시고 빌드합니다. 단 먼저 Tomcat이 이미 시작된 상태여야 되겠죠. 프로젝트가 빌드되면 웹브라우저로 접속하여 테스트해봅니다. 단 현재는 booklist.do 페이지는 접속불가입니다.

5. 여기까지 따라오셨으면 이번엔 Struts 와 Spring을 연동하는 단계입니다. 그림에서처럼 Spring Capabilities를 추가합니다.


New
Go Into
Open in New Window
Open Type Hierarchy F4
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor
Import...
Export...
Refresh F5
Close Project
Run As
Debug As
Profile As
Team
Compare With
Restore from Local History...
MyEclipse
PDE Tools
Properties Alt+Enter
Add and Remove Project Deployments...
Add Web Service Capabilities...
Add Tapestry Capabilities...
Add Spring Capabilities...
Add JSF Capabilities...
Add JSTL Libraries...
Add Hibernate Capabilities...
Run XDoclet
Open in Explorer
Run Validation
Remove All Validation Markers
MyEclipse Struts
Creation date: 09-27-2006
XDoclet definition:
@struts.form name="bookListFor
lic class BookListForm extends
/*
* Generated Methods
*/
me: 3 seconds


Select MyEclipse/User Libraries
Show: MyEclipse Libraries User Libraries
Spring 1.2 AOP Libraries - <MyEclipse-Library>
Spring 1.2 Core Libraries - <MyEclipse-Library>
Spring 1.2 ORM/DAO/Hibernate3 Libraries - <MyEclipse-Library>
Spring 1.2 ORM/DAO/Hibernate2 Libraries - <MyEclipse-Library>
Spring 1.2 Remoting Libraries - <MyEclipse-Library>
Spring 1.2 Misc Libraries - <MyEclipse-Library>
Spring 1.2 Testing Support Libraries - <MyEclipse-Library>
Spring 1.2 Web Libraries - <MyEclipse-Library>
View and edit libraries...
JAR Library Installation
Add checked Libraries to project build-path
Copy checked Library contents to project folder (TLDs always copied)
Library Folder: /WebRoot/WEB-INF/lib
Browse...
Tag Library Installation
Tag Library Folder: /WebRoot/WEB-INF
Browse...
< Back
Next >
Finish
Cancel



6. Spring이 설치되였으면 다음 보조는 Struts와 Spring을 연동시키는것이다. 먼저 Struts의 설정파일인

struts-config.xml을 열고서 기본의 action맵핑부분을 수정합니다

```
<action
 attribute="bookListForm"
 input="/JSP/bookList.jsp"
 name="bookListForm"
 path="/bookList"
 scope="request"
 type="org.library.struts.action.BookListAction"
 validate="false">
 <forward name="showList" path="/JSP/bookList.jsp" />
</action>
```

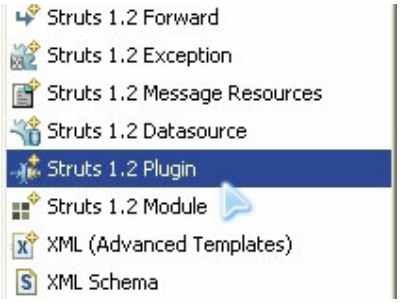
이렇게 되여있는것을 다음과 같이 고칩니다.

```
<action
 attribute="bookListForm"
 input="/JSP/bookList.jsp"
 name="bookListForm"
 path="/bookList"
 scope="request"
 type="org.springframework.web.struts.DelegatingActionProxy"
 validate="false">
 <forward name="showList" path="/JSP/bookList.jsp" />
</action>
```

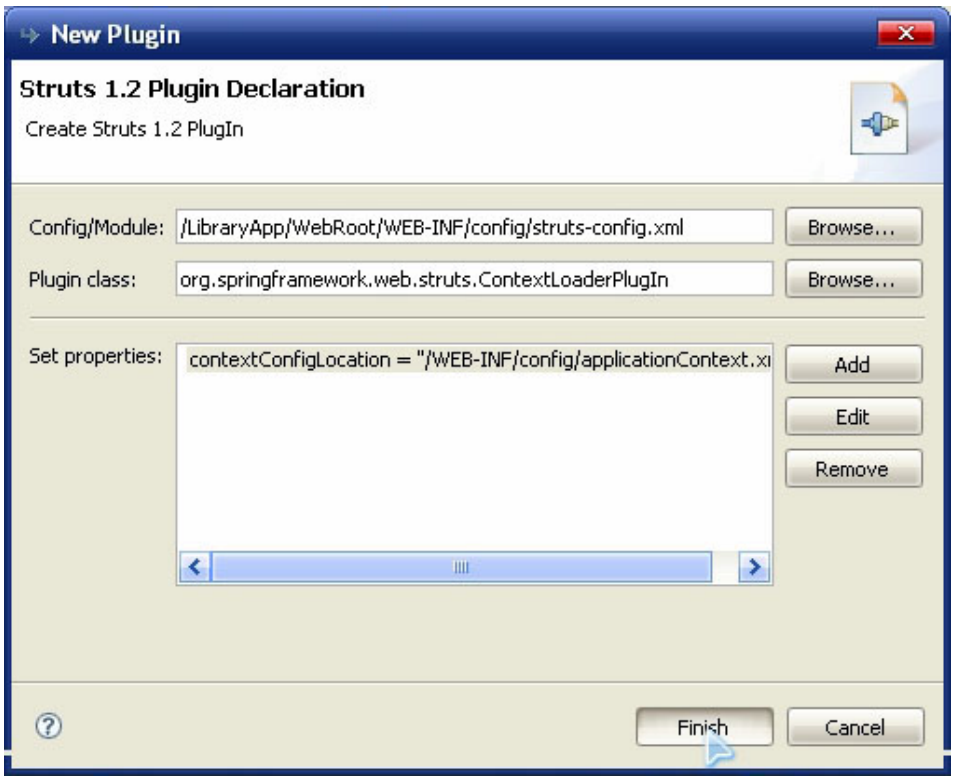
즉 고쳐진 부분은 type="org.library.struts.action.BookListAction" 이 부분입니다. 다음 spring의 설정파일인 applicationContext.xml을 여시고 아래와 같은 부분을 추가합니다.

```
<bean name="/bookList" class="org.library.struts.action.BookListAction" singleton="false">
</bean>
```

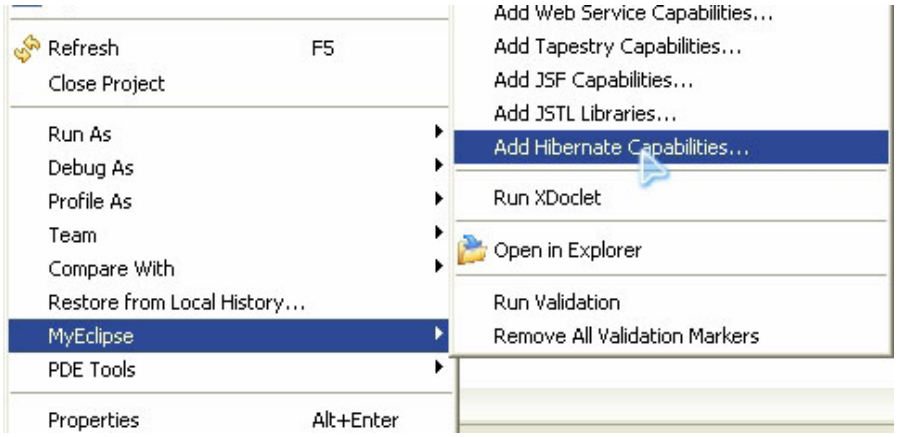
다음 struts-config.xml에 plugin을 등록합니다.



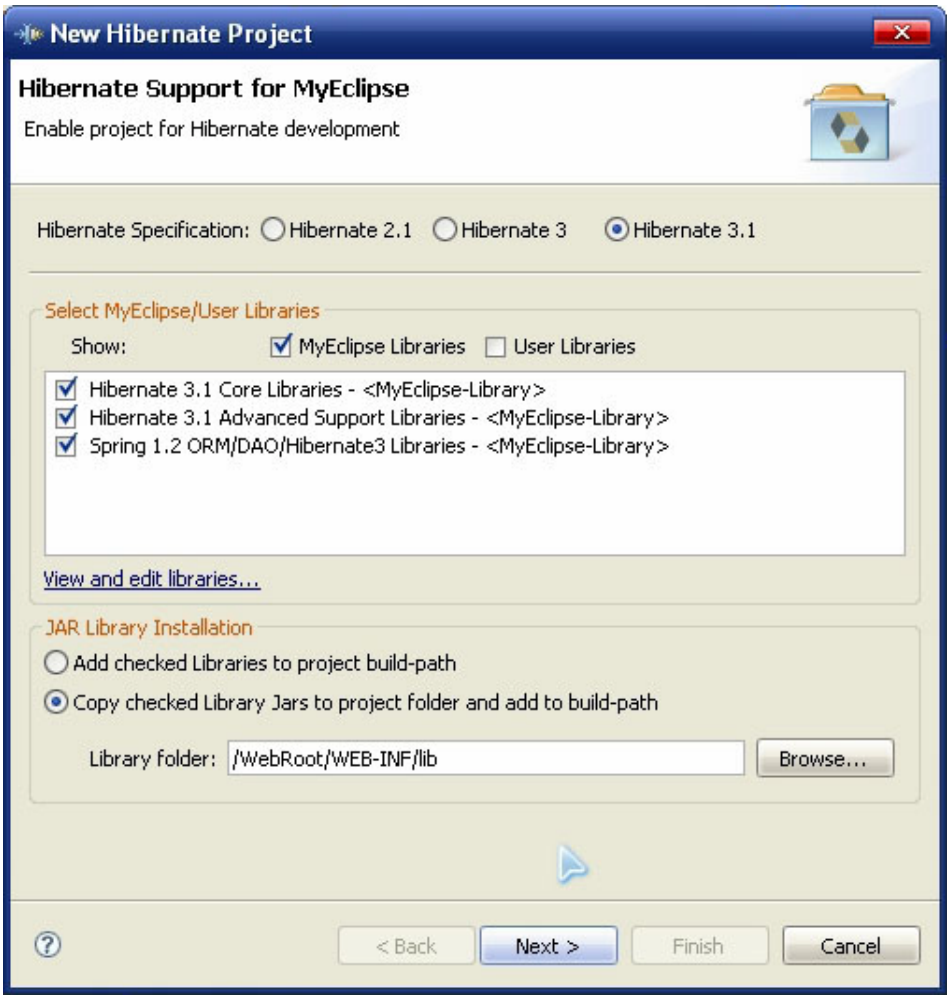
다음 아래 그림에서처럼 설정합니다.



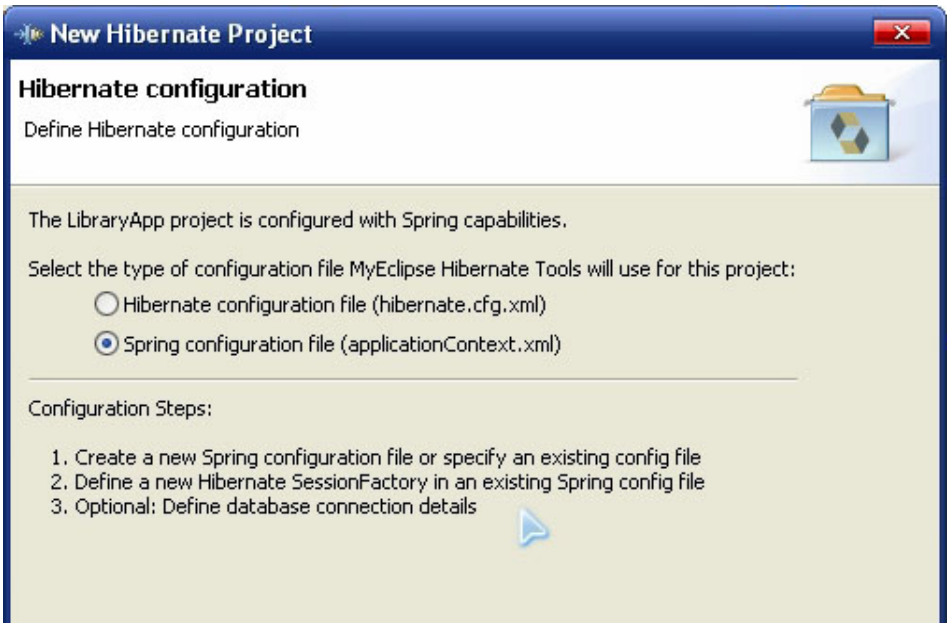
7. 이상 Struts와 Spring연동은 끝났습니다. 다음 Hibernate을 설치하고 Spring하고 연동하겠습니다. 먼저 Hibernate를 설치하겠습니다. 그림에서처럼 Hibernate Capabilities를 클릭합니다.



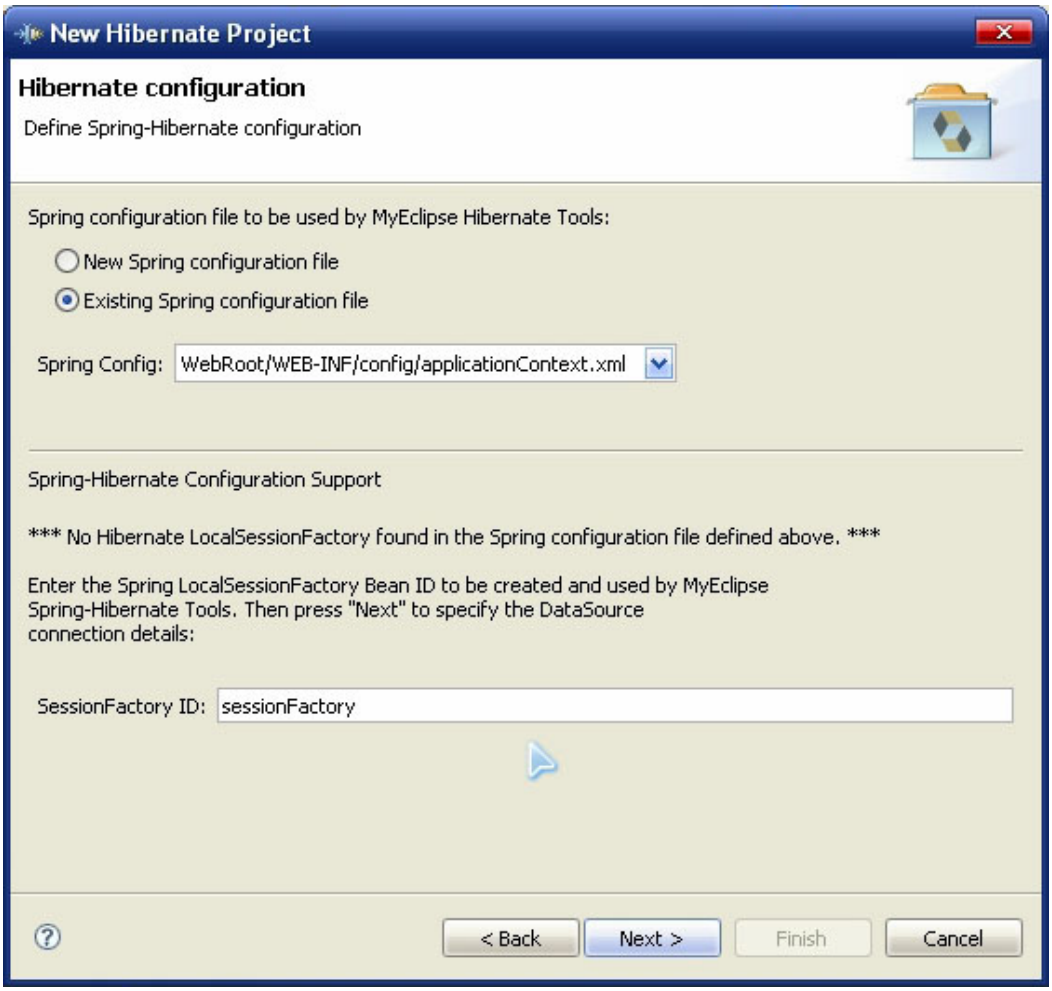
다음 hibernate설치를 위한 설정을 해줍니다. 설치한 hibernate버젼을 선택하고 라이브러리 저장경로를 설정해줍니다.



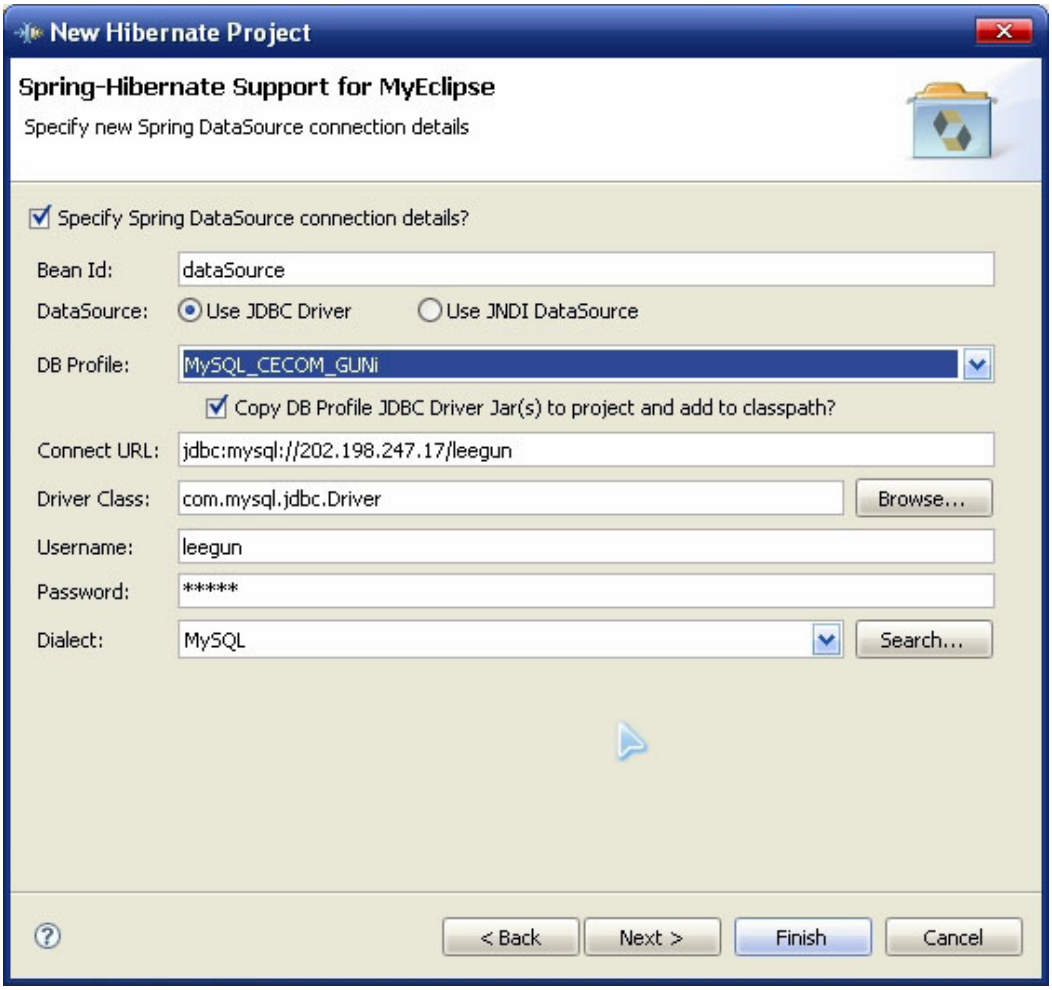
다음 hibernate 설정파일을 hibernate.cfg.xml로 할건지 아니면 applicationContext.xml로 할건지 선택합니다. 편의상 여기서 applicationContext.xml로 선택하고 다음을 클릭합니다.



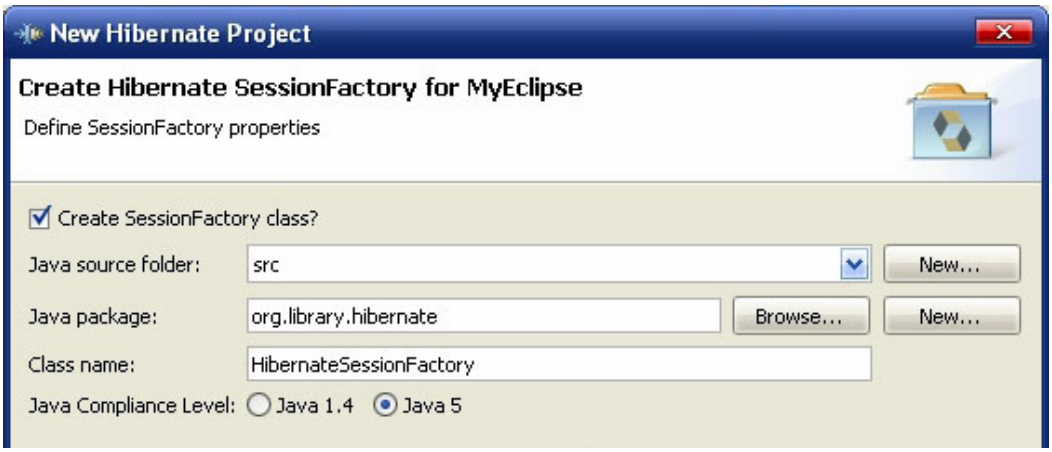
기존에 spring Framework을 이미 설치하였으므로 새로운 spring파일을 생성할 필요없이 기존으 spring 설정파일을 씁니다.다음 SessionFactory ID를 설정해줍니다. 여기서는 간단히 sessionFactory로 해주겠습니다.



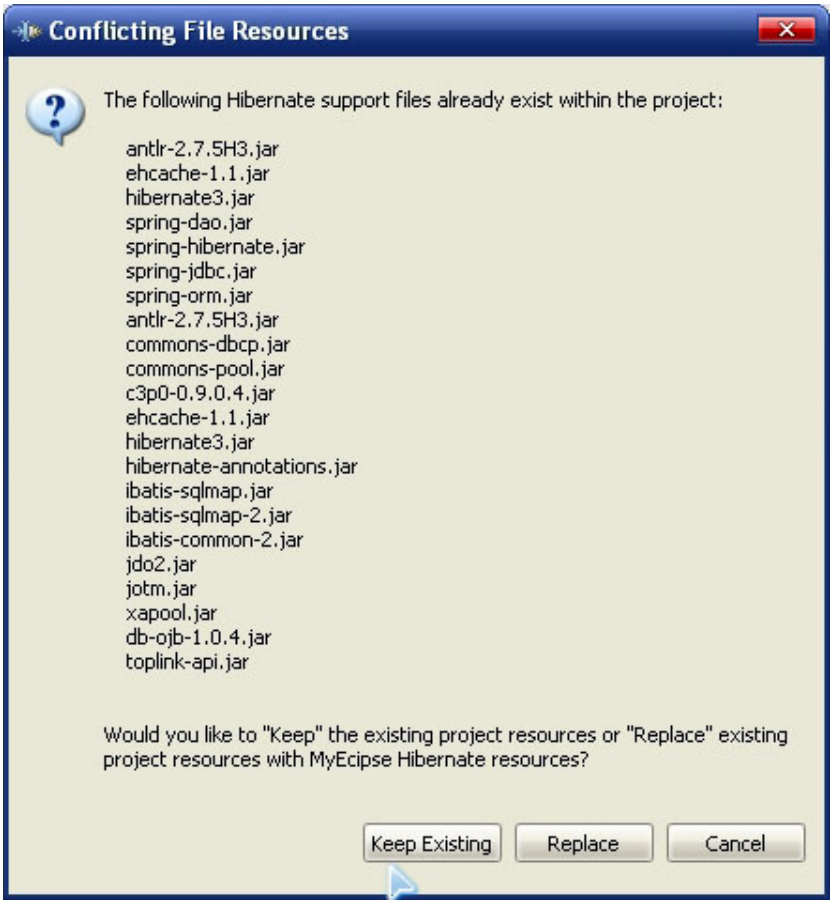
다음 DB접속을 위한 설정을 합니다. 저는 이미 DB Connect Profile을 만든 상태라서 그대로 가져다 씁니다. 만약 아직 이 설정을 하지 않으셨으면 MyEclipse=>DB=>Driver에 가셔서 필요한 드라이버를 등록하고 DB접속 프로필을 만들어 주십시오.



다음은 SessionFactory를 만들어주는데 여기서는 org.library.hibernate패키지속에 생성시키겠습니다.



설치를 시작하면 일부 라이브러리들이 기존에 존재하므로 덮어 쓰겠는가 물어봅니다. 그대로 놔두고 설치를 계속합니다.



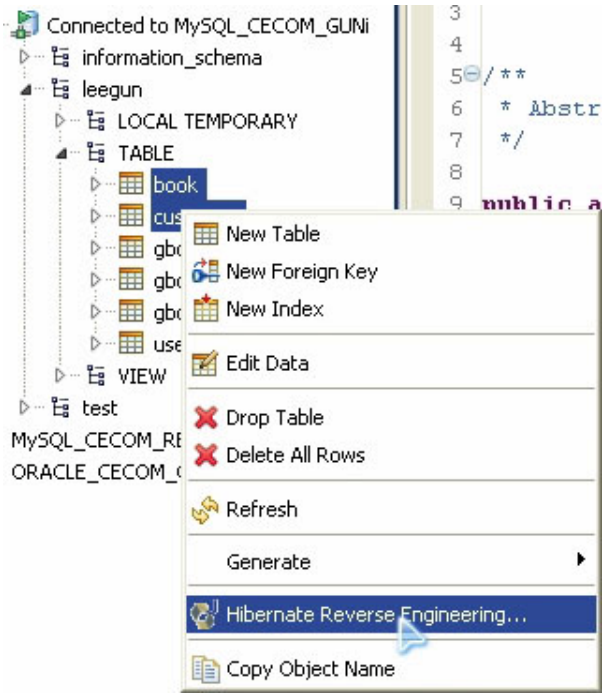
8. 이상 hibernate설치 및 spring하고 기본적인 연동이 되여졌습니다. 테스트에 필요한 테이블을 만들어 봅시다.

MySQL Query 문은 다음과 같습니다.

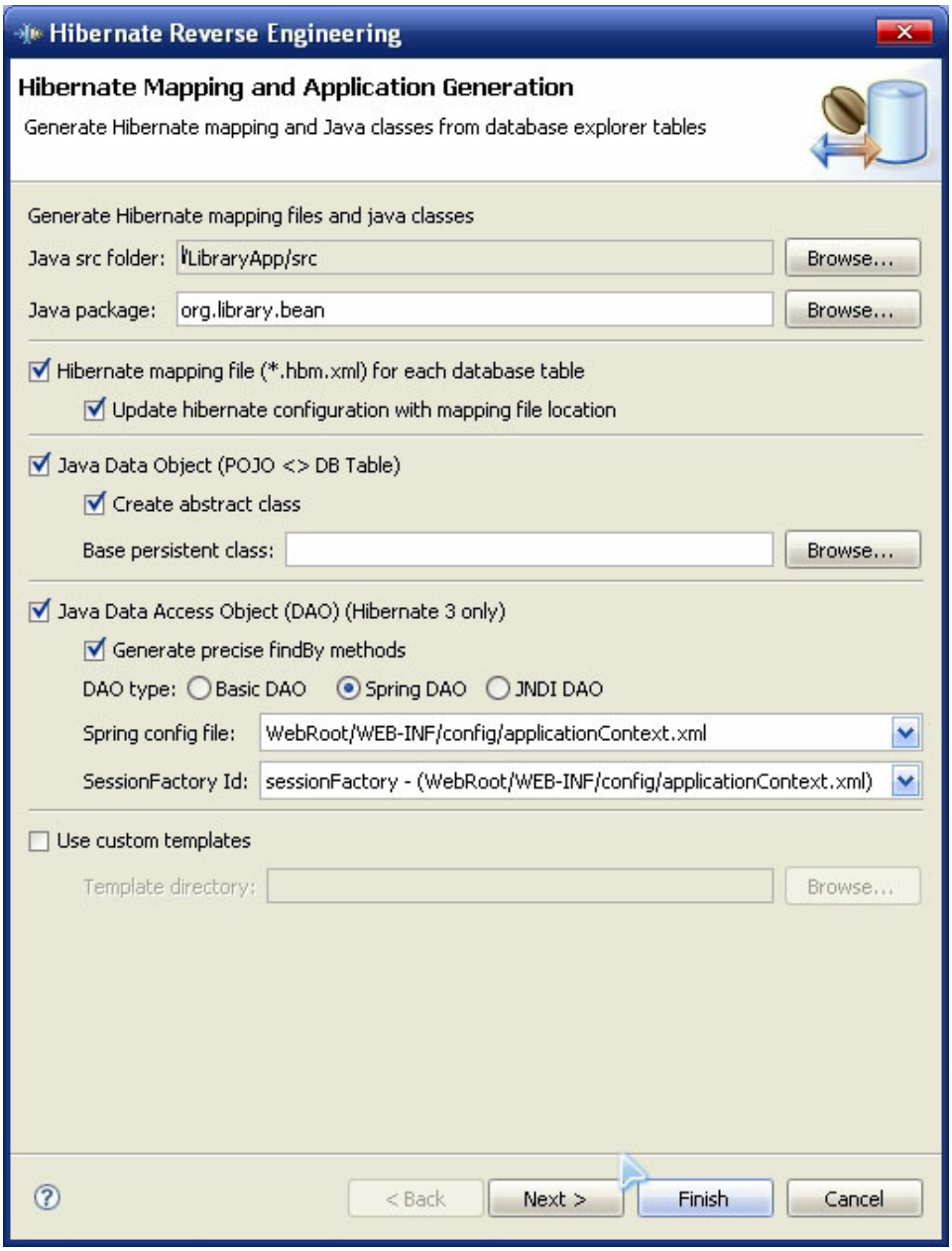
```
create table `leegun`.`book`(
          `id` int not null auto_increment,
         `title` varchar(255),
         `author` varchar(255),
         `customer_fk` int,
         `borrowallowed` tinyint default '' not null,
          primary key (`id`)
    );


 alter table `leegun`.`book`
          add index `book_customer`(`customer_fk`),
          add constraint `book_customer`
          foreign key (`customer_fk`)
          references `leegun`.`customer`(`id`);
```

```
create unique index `PRIMARY` on `leegun`.`book`(`id`);
                create index `customer_fk` on `leegun`.`book`(`customer_fk`);
create table `leegun`.`customer`(
    `id` int not null auto_increment,
    `firstname` varchar(255),
    `lastname` varchar(255),
    `age` int,
    primary key (`id`)
);

create unique index `PRIMARY` on `leegun`.`customer`(`id`);
```

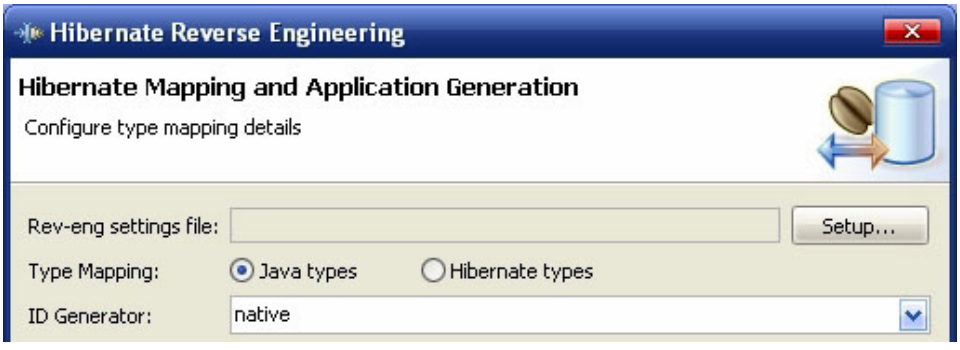
테이블이 정상적으로 생성이 되여졌으면 Hibernate Reverse Engineering을 해줍니다. 아래 그림에서와 같이…



Mapping파일이 저장될 패키지 및 Spring관련 DAO생성을 설정하는 페이지입니다.아래 그림에서처럼 설정합니다.



ID Generator를 native로 설정해줍니다.그리고 Finish클릭하면 필요한 Mapping파일 및 추상클래스 와 DAO가 만들어 집니다.



9. Hibernate Reverse Engineering과정을 거치면 도합 모두 8개파일이 생성됩니다. 각각 AbstractBook.java AbstractCustomer.java Book.java Customer.java BookDAO.java CustomerDAO.java book.hbm.xml customer.hbm.xml 입니다.

   여기서 AbstarctBook.java는 말 그래도 추상클래스입니다. 이 클래스에서는 book.hbm.xml 즉 맵핑파일에서 맵핑한 각 properties에 대하여 setter/getter 매소드를 선언해주고있습니다. 그리고 Book.java는 AbstractBook클래스를 상속한 자식클래스입니다. 여기서는 Book에 대한 생성자를 오버로딩해 주고 있습니다. Customer도 마찬가지므로 설명은 략하겠습니다. 그리고 한가지 기존의 코드를 수정할것이라면 기존의 Borrowable 이것이 Tinyint타입이 설정되였는데 Mapping시 보면 MyEclipse가 자동으로 Short타입으로 맵핑해주었습니다. 이것을 모두 boolean으로 고쳐줘야합니다. 다음 현재 파일 8개가 모두 bean 패키지속에 있는데 DAO파일 2개는 별도로 dao패키지속으로 이동하면 파일 관리에 좋습니다. 또한 프로젝트가 커지면 DAO외에 business로직을 구현하는 service클래스도 별도로 만들어줄 필요성이 있습니다. 여기서는 간단한 예제를 보여주니까 service클래스까지는 만들지 않겠습니다. 하지만 기존에 생성된 DAO파일을 두개의 파일 즉 예를 들자면 BookDAO.java 이파일을 BookDAO.java 및 BookDAOImpl.java 두 파일로 나눕니다. BookDAO.java속에는 간단한 interface만 만들어줍니다. 그리고 BookDAOImpl.java에는 BookDAO.java 에서 만들어진 interface를 구현해줍니다. 보통 IBookDAO.java 이렇게 명명할때도 있습니다.

   BookDAO.java의 내용은 다음과 같습니다.

```
package org.library.dao;

import java.util.List;
import org.library.bean.Book;

/**
 * Data access object (DAO) for domain model class Book.
 * @see org.library.bean.Book
 * @author MyEclipse - Hibernate Tools
 */
public interface BookDAO {

    public abstract void save(Book transientInstance);
public abstract void delete(Book persistentInstance);
public abstract List findAllBook();
    public abstract Book findById( java.lang.Integer id);
    public abstract List findByExample(Book instance);
    public abstract List findByProperty(String propertyName, Object value);
public abstract List findByTitle(Object title);
```

```
public abstract List findByAuthor(Object author);
public abstract List findByBorrowallowed(Object borrowallowed);
    public abstract Book merge(Book detachedInstance);
    public abstract void attachDirty(Book instance);
    public abstract void attachClean(Book instance);
}
```

그리고 BookDAOImpl.java 파일속이 내용은 다음과 같습니다.

```
package org.library.dao;

import java.util.List;
import java.util.ArrayList;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.hibernate.LockMode;
import org.library.bean.Book;
import org.springframework.context.ApplicationContext;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;

/**
 * Data access object (DAO) for domain model class Book.
 * @see org.library.bean.Book
 * @author MyEclipse - Hibernate Tools
 */
public class BookDAOImpl extends HibernateDaoSupport implements BookDAO {

    private static final Log log = LogFactory.getLog(BookDAO.class);

//property constants
    public static final String ALL = "from Book b";
public static final String TITLE = "title";
public static final String AUTHOR = "author";
public static final String BORROWALLOWED = "borrowallowed";

protected void initDao() {
        //do nothing
}
```

```
    public void save(Book transientInstance) {
        log.debug("saving Book instance");
        try {
            getHibernateTemplate().save(transientInstance);
            log.debug("save successful");
        } catch (RuntimeException re) {
            log.error("save failed", re);
            throw re;
        }
    }

public void delete(Book persistentInstance) {
        log.debug("deleting Book instance");
        try {
            getHibernateTemplate().delete(persistentInstance);
            log.debug("delete successful");
        } catch (RuntimeException re) {
            log.error("delete failed", re);
            throw re;
        }
    }

public List findAllBook() {
           try {
                    return this.getHibernateTemplate().find(ALL);
           } catch (Exception e) {
                    e.printStackTrace();
                    return new ArrayList();
           }
}

    public Book findById( java.lang.Integer id) {
        log.debug("getting Book instance with id: " + id);
        try {
            Book instance = (Book) getHibernateTemplate()
```

```
                    .get("org.library.bean.Book", id);
            return instance;
        } catch (RuntimeException re) {
            log.error("get failed", re);
            throw re;
        }
    }


    public List findByExample(Book instance) {
        log.debug("finding Book instance by example");
        try {
            List results = getHibernateTemplate().findByExample(instance);
            log.debug("find by example successful, result size: " + results.size());
            return results;
        } catch (RuntimeException re) {
            log.error("find by example failed", re);
            throw re;
        }
    }

    public List findByProperty(String propertyName, Object value) {
      log.debug("finding Book instance with property: " + propertyName
            + ", value: " + value);
      try {
          String queryString = "from Book as model where model."
                                                              + propertyName + "= ?";
            return getHibernateTemplate().find(queryString, value);
      } catch (RuntimeException re) {
          log.error("find by property name failed", re);
          throw re;
      }
}

public List findByTitle(Object title) {
            return findByProperty(TITLE, title);
```

```
}

public List findByAuthor(Object author) {
            return findByProperty(AUTHOR, author);
}

public List findByBorrowallowed(Object borrowallowed) {
            return findByProperty(BORROWALLOWED, borrowallowed);
}

    public Book merge(Book detachedInstance) {
        log.debug("merging Book instance");
        try {
            Book result = (Book) getHibernateTemplate()
                    .merge(detachedInstance);
            log.debug("merge successful");
            return result;
        } catch (RuntimeException re) {
            log.error("merge failed", re);
            throw re;
        }
    }

    public void attachDirty(Book instance) {
        log.debug("attaching dirty Book instance");
        try {
            getHibernateTemplate().saveOrUpdate(instance);
            log.debug("attach successful");
        } catch (RuntimeException re) {
            log.error("attach failed", re);
            throw re;
        }
    }

    public void attachClean(Book instance) {
        log.debug("attaching clean Book instance");
```

```
            try {
                getHibernateTemplate().lock(instance, LockMode.NONE);
                log.debug("attach successful");
            } catch (RuntimeException re) {
                log.error("attach failed", re);
                throw re;
            }
        }

 public static BookDAO getFromApplicationContext(ApplicationContext ctx) {
            return (BookDAO) ctx.getBean("BookDAO");
 }
}
```

CustomerDAO.java파일 내용

```
package org.library.dao;

import java.util.List;
import org.library.bean.Customer;

/**
 * Data access object (DAO) for domain model class Customer.
 * @see org.library.bean.Customer
 * @author MyEclipse - Hibernate Tools
 */
public interface CustomerDAO {

    public abstract void save(Customer transientInstance);
 public abstract void delete(Customer persistentInstance);
    public abstract Customer findById( java.lang.Integer id);
    public abstract List findByExample(Customer instance);
    public abstract List findByProperty(String propertyName, Object value);
 public abstract List findByFirstname(Object firstname);
 public abstract List findByLastname(Object lastname);
 public abstract List findByAge(Object age);
    public abstract Customer merge(Customer detachedInstance);
```

```
    public abstract void attachDirty(Customer instance);
    public abstract void attachClean(Customer instance);
}
```

CustomerDAOImpl.java파일의 내용

```
package org.library.dao;

import java.util.List;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.hibernate.LockMode;
import org.library.bean.Customer;
import org.springframework.context.ApplicationContext;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;

/**
 * Data access object (DAO) for domain model class Customer.
 * @see org.library.bean.Customer
 * @author MyEclipse - Hibernate Tools
 */
public class CustomerDAOImpl extends HibernateDaoSupport implements CustomerDAO {

    private static final Log log = LogFactory.getLog(CustomerDAO.class);

 //property constants
 public static final String FIRSTNAME = "firstname";
 public static final String LASTNAME = "lastname";
 public static final String AGE = "age";

 protected void initDao() {
            //do nothing
 }

    public void save(Customer transientInstance) {
        log.debug("saving Customer instance");
```

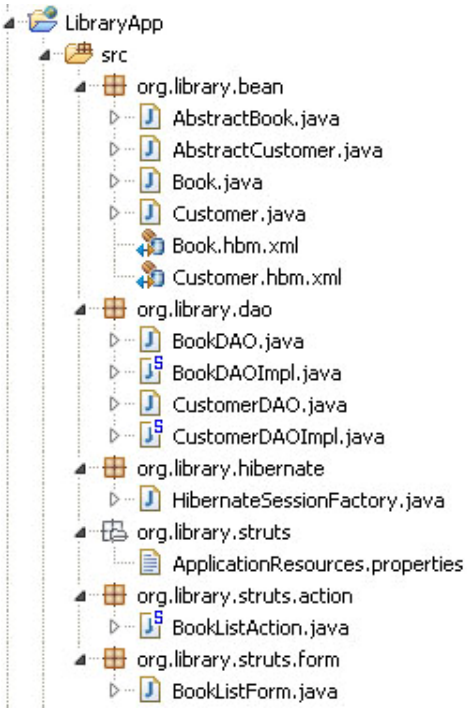
```
            try {
                getHibernateTemplate().save(transientInstance);
                log.debug("save successful");
            } catch (RuntimeException re) {
                log.error("save failed", re);
                throw re;
            }
        }

    public void delete(Customer persistentInstance) {
            log.debug("deleting Customer instance");
            try {
                getHibernateTemplate().delete(persistentInstance);
                log.debug("delete successful");
            } catch (RuntimeException re) {
                log.error("delete failed", re);
                throw re;
            }
        }

        public Customer findById( java.lang.Integer id) {
            log.debug("getting Customer instance with id: " + id);
            try {
                Customer instance = (Customer) getHibernateTemplate()
                        .get("org.library.bean.Customer", id);
                return instance;
            } catch (RuntimeException re) {
                log.error("get failed", re);
                throw re;
            }
        }


        public List findByExample(Customer instance) {
            log.debug("finding Customer instance by example");
            try {
```

```
            List results = getHibernateTemplate().findByExample(instance);
            log.debug("find by example successful, result size: " + results.size());
            return results;
        } catch (RuntimeException re) {
            log.error("find by example failed", re);
            throw re;
        }
    }

    public List findByProperty(String propertyName, Object value) {
      log.debug("finding Customer instance with property: " + propertyName
            + ", value: " + value);
      try {
          String queryString = "from Customer as model where model."
                                                                + propertyName + "= ?";
            return getHibernateTemplate().find(queryString, value);
      } catch (RuntimeException re) {
          log.error("find by property name failed", re);
          throw re;
      }
}

public List findByFirstname(Object firstname) {
            return findByProperty(FIRSTNAME, firstname);
}

public List findByLastname(Object lastname) {
            return findByProperty(LASTNAME, lastname);
}

public List findByAge(Object age) {
            return findByProperty(AGE, age);
}

    public Customer merge(Customer detachedInstance) {
        log.debug("merging Customer instance");
```

```
        try {
            Customer result = (Customer) getHibernateTemplate()
                    .merge(detachedInstance);
            log.debug("merge successful");
            return result;
        } catch (RuntimeException re) {
            log.error("merge failed", re);
            throw re;
        }
    }

    public void attachDirty(Customer instance) {
        log.debug("attaching dirty Customer instance");
        try {
            getHibernateTemplate().saveOrUpdate(instance);
            log.debug("attach successful");
        } catch (RuntimeException re) {
            log.error("attach failed", re);
            throw re;
        }
    }

    public void attachClean(Customer instance) {
        log.debug("attaching clean Customer instance");
        try {
            getHibernateTemplate().lock(instance, LockMode.NONE);
            log.debug("attach successful");
        } catch (RuntimeException re) {
            log.error("attach failed", re);
            throw re;
        }
    }

public static CustomerDAO getFromApplicationContext(ApplicationContext ctx) {
        return (CustomerDAO) ctx.getBean("CustomerDAO");
}
```

}

수정후 패키지구조를 보면 다음과 같습니다.



10. 다음 Hibernate과 Spring이 통합되여 작동하게 하기 위하여 applicationContext.xml에 다음과 같이 추가합니다.

```
<bean id="transactionManager"
class="org.springframework.orm.hibernate3.HibernateTransactionManager">
            <property name="sessionFactory">
                        <ref local="sessionFactory" />
   </property>
 </bean>
```

그리고

```
<bean id="BookDAO" class="org.library.bean.BookDAO">
            <property name="sessionFactory">
                        <ref bean="sessionFactory" />
            </property>
 </bean>
```

이부분을

```
<bean id="BookDAO" class="org.library.dao.BookDAOImpl">
            <property name="sessionFactory">
                        <ref bean="sessionFactory" />
```

```
        </property>
    </bean>
```

이렇게 수정합니다.

다음 아래부분도 추가합니다.

```
<bean id="bookDAOProxy"
class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
        <property name="transactionManager">
            <ref bean="transactionManager" />
        </property>
        <property name="target">
            <ref bean="BookDAO"/>
        </property>
        <property name="transactionAttributes">
            <props>
                <prop key="insert*">PROPAGATION_REQUIRED</prop>
                <prop key="get*">PROPAGATION_REQUIRED,readOnly</prop>
                <prop key="is*">PROPAGATION_REQUIRED,readOnly</prop>
            </props>
        </property>
    </bean>
```

다음 예전에 만들었던 login액션부분을 아래것처럼 수정합니다.

```
<bean name="/bookList" class="org.library.struts.action.BookListAction" singleton="false">
        <property name="bookDAO">
            <ref bean="bookDAOProxy" />
        </property>
    </bean>
```

11. 다음 FormBean부분을 다음과 같이 수정합니다.

```
/*
 * Generated by MyEclipse Struts
 * Template path: templates/java/JavaClass.vtl
 */
package org.library.struts.form;


import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
```

```
import org.library.bean.Book;
import org.library.bean.Customer;

/**
 * MyEclipse Struts
 * Creation date: 09-27-2006
 *
 * XDoclet definition:
 * @struts.form name="bookListForm"
 */
public class BookListForm extends ActionForm {
 /*
  * Generated Methods
  */

 private static final long serialVersionUID = 7929309963980301882L;
 private Book[] books = new Book[0];
 private Book book = new Book();


 public Book[] getBooks() {
        return books;
 }

 public void setBooks(Book[] books) {
        this.books = books;
 }

 /**
  * Method reset
  * @param mapping
  * @param request
  */
 public void reset(ActionMapping mapping, HttpServletRequest request) {
        // TODO Auto-generated method stub
        books = new Book[0];
```

```
        book = new Book();
}

public boolean equals(Object obj) {
        return book.equals(obj);
}

public String getAuthor() {
        return book.getAuthor();
}

public Boolean getBorrowallowed() {
        return book.getBorrowallowed();
}

public Customer getCustomer() {
        return book.getCustomer();
}

public Integer getId() {
        return book.getId();
}

public String getTitle() {
        return book.getTitle();
}

public int hashCode() {
        return book.hashCode();
}

public void setAuthor(String author) {
        book.setAuthor(author);
}

public void setBorrowallowed(Boolean borrowallowed) {
```

```
		book.setBorrowallowed(borrowallowed);
	}

	public void setCustomer(Customer customer) {
		book.setCustomer(customer);
	}

	public void setId(Integer id) {
		book.setId(id);
	}

	public void setTitle(String title) {
		book.setTitle(title);
	}
}
```

12. 그리고 Action부분은 다음과 같이 수정합니다.

```
/*
 * Generated by MyEclipse Struts
 * Template path: templates/java/JavaClass.vtl
 */
package org.library.struts.action;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.library.dao.BookDAO;

/**
 * MyEclipse Struts
 * Creation date: 09-27-2006
 *
 * XDoclet definition:
 * @struts.action path="/bookList" name="bookListForm" input="/JSP/bookList.jsp" scope="request"
```

```
 * @struts.action-forward name="showList" path="/JSP/bookList.jsp"
 */
public class BookListAction extends Action {
/*
 * Generated Methods
 */
private BookDAO bookDAO;

public BookDAO getBookDAO() {
        return bookDAO;
}

public void setBookDAO(BookDAO bookDAO) {
        this.bookDAO = bookDAO;
}

/**
 * Method execute
 * @param mapping
 * @param form
 * @param request
 * @param response
 * @return ActionForward
 */
public ActionForward execute(ActionMapping mapping, ActionForm form,
                HttpServletRequest request, HttpServletResponse response) {
        //BookListForm bookListForm = (BookListForm) form;// TODO Auto-generated method stub
        request.getSession().setAttribute("books", bookDAO.findAllBook());
        //bookListForm.setBooks(bookDAO.findAllBook());
        return mapping.findForward("showList");
}
}
```

13. 마지막으로 전체 컴파일 시키고 Tomcat재시동시키고서 테스트하면 정상적으로 작동할겁니다. 동영상 강좌로 만든것이 있지만 파일이 너무 커서 올리지 못하겠네요.이상 허접한 SSH연동 강좌였습니다.