第二次迭代说明：

项目是跑在glassfish里，glassfish整合了tomcat，不需要手动部署。并且glassfish也支持ejb和websocket，不需要额外的glassfish和tomcat之间的rmi

第三次迭代说明

环境：

Intellij 14.0.2

JDK1.8.0

Hibernate 4.2.2

Glassfish 4.1.0

MySQL 5.6

完成功能：

1. MemCached
   1. MemCached功能被组装到ejb.Utils.MemCachedUtil中，可以实现多个MemCached Server的分布式缓存。本地测试过程中使用多个cmd进程，每一个cmd窗口用memcached –p [port]把cache server绑定到不同的端口上。
   2. 阅读了MemCachedClient源码后发现，cache server之间的一致性hash由MemCachedClient定义，不需要手动定义。
   3. 使用中，对UserEntity进行了缓存，具体代码在ejb.DAO.UserMgrBean中。对每一个正在活动的用户的信息进行缓存，对数据库做修改或者删除操作时，也会对memcached进行同步。
2. 国际化

对首页进行了国际化，所有页面上的文字由Locale.LangBundle进行获取，初始化时，会把语言存放到session中，每次get会获取session中存放的语言偏好，并用这个偏好来更改jsp上的提示信息

1. AspectJ：

对用户下订单这一行为做了logging，实现在ejb.Utils.SaleLogging中。

日志会记录下该订单的用户名和交易金额。

1. WebService

将用户查询书籍详细信息行为添加到了webservice中。并且服务器打开后启动首页时把服务publish出去。（publish代码在Util.WebService.WSPublisher中）

调用了本地的现有的BookMgrBean。BookMgrBean自动注入失败，原因是因为不同的应用服务器对EJB的注入方式不同。所以为了避免代码和应用服务器关联过于紧密，通过context lookup手动注入。实现在Util.WebService.BookServiceImpl中。

WSDL文件在Util.WebServiceClient.BookServiceImpl.wsdl中。通过java的wsimport工具自动生成服务端代码对应的wsdl文件。

（似乎jdk1.8.0有bug，要在java目录里手动添加jaxp.properties并且赋予访问外部schema的权限才能正常使用wsdl）

可以使用Util.WebServiceClient.Test.ClientTest进行测试.

1. RestfulService同样实现了用户查询书籍详细信息的接口。对于用户查询，返回200 code，并返回书籍的Json信息。实现在Util.RestWebService中。

通过访问书籍详细信息的URI

<https://localhost:8181/BookStore_war_exploded/rest/Test/book/2>

可以获得书籍的Json格式信息

{"bid":2,"bookname":"fdafdas","catagory":"1212","price":222.0}