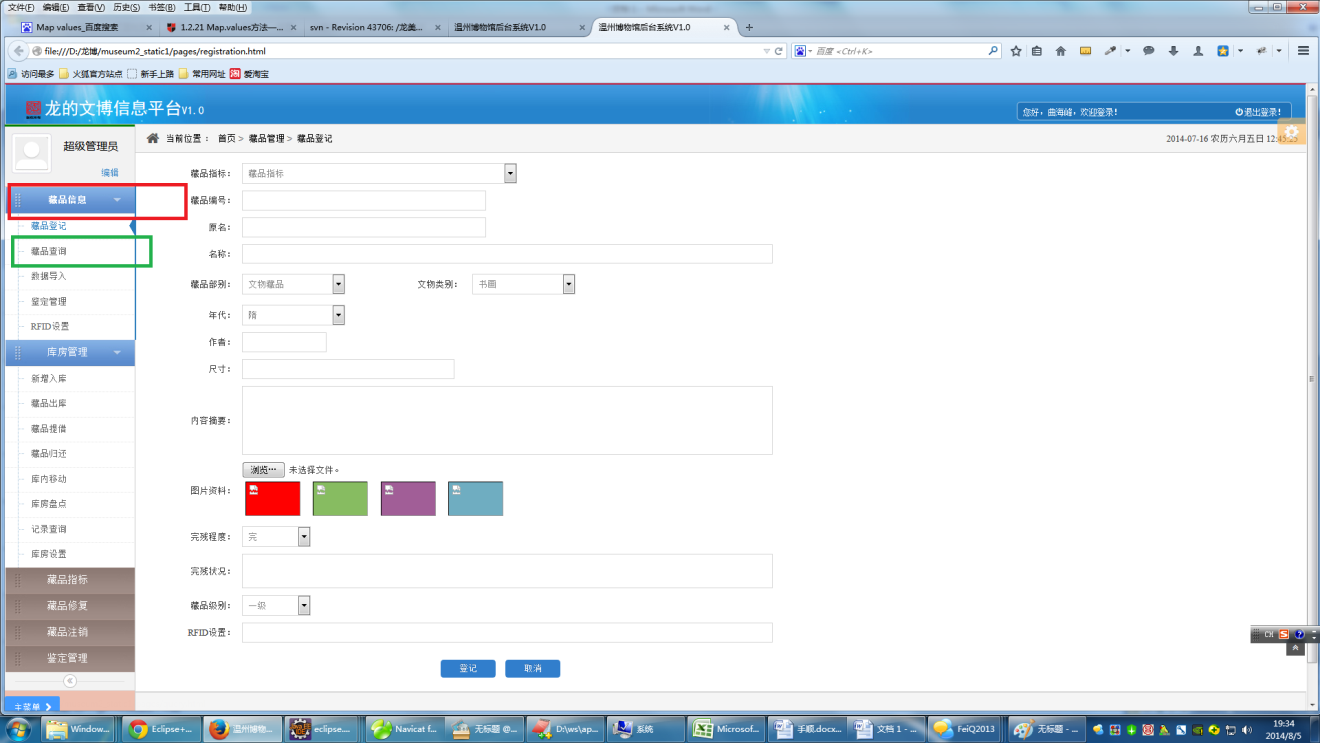
# 包的定义、划分：

1. com longdi.XXXX，XXXX是子系统名（例如藏品信息，库房管理）
2. 实体层entity，数据库访问层dao跟表一一对应（单独一个人做）
3. 业务层service和控制层action一一对应，包括类和方法（一个人对应一或多个action）



1. jsp页面 webapp/WEB-INF/page中建一个与系统名（XXXX）同名的文件夹，用于存放jsp页面

注意：项目中有两个包是共通包：

1. common包，存放可以独立于项目存在的共通代码，例如对Hibernate的初步封装。
2. basic包，存放对基本数据表（例如Code表，user表，权限表）的增删改查和初步封装操作（相当于基盘）。

# Struts2要点：

1. 配置文件：

包名，名称空间，继承包

Action名，class名，对应的方法名

result 名，result类型（dispatcher，redirect，json）

|  |
| --- |
| <package name=*"user"* namespace=*"/user"* extends=*"longdi-default"*>    <action name=*"list"* class=*"userAction"* method=*"list"*>  <result name=*"success"*>/WEB-INF/page/user/user\_list.jsp</result>  </action>    <action name=*"detail"* class=*"userAction"* method=*"detail"*>  <result name=*"success"*>/WEB-INF/page/user/user\_detail.jsp</result>  </action>  <action name=*"create"* class=*"userAction"* method=*"create"*>  <result name=*"success"*>/WEB-INF/page/user/user\_edit.jsp</result>  </action>  <action name=*"edit"* class=*"userAction"* method=*"edit"*>  <result name=*"success"*>/WEB-INF/page/user/user\_edit.jsp</result>  </action>  <action name=*"save"* class=*"userAction"* method=*"save"*>  <result name=*"success"* type=*"redirect"*>list</result>  </action>  <action name=*"delete"* class=*"userAction"* method=*"delete"*>  <result name=*"success"* type=*"redirect"*>list</result>  </action>    <action name=*"delete\_all\_tasks"* class=*"userAction"* method=*"deleteTasksByUserId"*>  <result name=*"success"* type=*"redirect"*>detail?id=${id}</result>  </action>    <action name=*"user\_tasks"* class=*"userAction"* method=*"getUserTaskPage"*>  <result name=*"success"*>/WEB-INF/page/user/task\_search\_results.jsp</result>  </action>    <action name=*"search"* class=*"userAction"* method=*"search"*>  <result name=*"success"*>/WEB-INF/page/user/user\_list.jsp</result>  </action>  </package> |

1. 基类

BasicActionSupport

|  |
| --- |
| /\*\* 400エラーのビジュアルウィンドウ. \*/  **protected** **final** String ERROR\_400 = "400";  /\*\* 403エラーのビジュアルウィンドウ. \*/  **protected** **final** String ERROR\_403 = "403";  /\*\* 404エラーのビジュアルウィンドウ. \*/  **protected** **final** String ERROR\_404 = "404";  **protected** **final** String EXPORT\_EXCEL="excel";  /\*\* HttpServletRequest対象. \*/  **protected** HttpServletRequest request;  /\*\* HttpServletResponse対象. \*/  **protected** HttpServletResponse response;  /\*\* session map, sessionのデータ. \*/  **protected** Map<String, Object> session;  /\*\* 結果フラグ.[0の場合: success, 0以外の場合: カスタム] \*/  **protected** Integer resultFlag = 0; |

1. Action的定义

|  |
| --- |
| @Controller  @Scope("prototype")  **public** **class** TaskAction **extends** BasicActionSupport { |

1. Action中的属性

简单类型，对象类型，List类型，有get，set方法

|  |
| --- |
| **private** Task task;    **private** List<Task> taskList;    **private** Long id; |

页面上如何使用

参照：4常用标签

调用Service，没有get，set方法

|  |
| --- |
| @Autowired  **private** TaskService taskService; |

1. 常用struts2标签

<s:property value=*"task.title"* />

<s:label name=*"task.id"*></s:label>

<s:textfield name="task.title"></s:textfield>

<s:iterator value=*"taskList"* status=*"tmst"*>

|  |
| --- |
| <s:if test="templateType == 'case'">  <td>ライフサイクルテンプレート</td>  </s:if>  <s:elseif test="templateType == 'event'">  <td>    </td>  </s:elseif>  <s:else>  <td>タスクテンプレート</td>  </s:else> |

|  |
| --- |
| <s:select name="templateDistinguish" list="templateDistinguishList" listKey="pulldownItemId" listValue="pulldownItemCategoryName" headerKey="0" headerValue="" cssClass="group\_linkage"></s:select> |

<s:hidden name=*"taskQuerySettingDTO.taskQuerySetting.dispTaskIdSort"* cssClass=*"sorter"*/>

s:checkbox name=*"taskQuerySettingDTO.taskQuerySetting.dispTaskStatus"*></s:checkbox>

|  |
| --- |
| <s:checkboxlist name=*"taskQuerySettingDTO.roleArray"*  list=*"taskQuerySettingDTO.roleList"*  listKey=*"pulldownItemId"* listValue=*"pulldownItemName"*  value=*"taskQuerySettingDTO.roleArray"*>  </s:checkboxlist> |

<s:textarea name=*"taskDetailDTO.taskInfo.taskSummary"* readonly=*"true"* cssClass=*"readonly"*/>

<s:radio list=*"caseEditDTO.caseTypeList"*

listKey=*"pulldownItemId"*

listValue=*"pulldownItemName"*

name=*"caseEditDTO.caseInfo.caseType"*>

</s:radio>

<s:date name=*"caseInfoDTO.caseInfo.updateDateTime"* format=*"YYYY/MM/dd"* />

# Service层

如何调用dao层代码

|  |
| --- |
| @Component  @Transactional  **public** **class** TaskService {  @Autowired  **private** TaskDao taskDao; |

# Hibernate Dao层

1. 定义

**public** **interface** TaskDao **extends** PagingAndSortingRepository<Task, Long>, JpaSpecificationExecutor<Task> {

1. 可以直接使用的方法

接口PagingAndSortingRepository中的方法：（常规操作）

|  |
| --- |
| /\*\*  \* Returns all entities sorted by the given options.  \*  \* **@param** sort  \* **@return** all entities sorted by the given options  \*/  Iterable<T> findAll(Sort sort);  /\*\*  \* Returns a {@link Page} of entities meeting the paging restriction provided in the {@code Pageable} object.  \*  \* **@param** pageable  \* **@return** a page of entities  \*/  Page<T> findAll(Pageable pageable); |

|  |
| --- |
| /\*\*  \* Saves a given entity. Use the returned instance for further operations as the save operation might have changed the  \* entity instance completely.  \*  \* **@param** entity  \* **@return** the saved entity  \*/  <S **extends** T> S save(S entity);  /\*\*  \* Saves all given entities.  \*  \* **@param** entities  \* **@return** the saved entities  \* **@throws** IllegalArgumentException in case the given entity is (@literal null}.  \*/  <S **extends** T> Iterable<S> save(Iterable<S> entities);  /\*\*  \* Retrieves an entity by its id.  \*  \* **@param** id must not be {@literal null}.  \* **@return** the entity with the given id or {@literal null} if none found  \* **@throws** IllegalArgumentException if {@code id} is {@literal null}  \*/  T findOne(ID id);  /\*\*  \* Returns whether an entity with the given id exists.  \*  \* **@param** id must not be {@literal null}.  \* **@return** true if an entity with the given id exists, {@literal false} otherwise  \* **@throws** IllegalArgumentException if {@code id} is {@literal null}  \*/  **boolean** exists(ID id);  /\*\*  \* Returns all instances of the type.  \*  \* **@return** all entities  \*/  Iterable<T> findAll();  /\*\*  \* Returns all instances of the type with the given IDs.  \*  \* **@param** ids  \* **@return**  \*/  Iterable<T> findAll(Iterable<ID> ids);  /\*\*  \* Returns the number of entities available.  \*  \* **@return** the number of entities  \*/  **long** count();  /\*\*  \* Deletes the entity with the given id.  \*  \* **@param** id must not be {@literal null}.  \* **@throws** IllegalArgumentException in case the given {@code id} is {@literal null}  \*/  **void** delete(ID id);  /\*\*  \* Deletes a given entity.  \*  \* **@param** entity  \* **@throws** IllegalArgumentException in case the given entity is (@literal null}.  \*/  **void** delete(T entity);  /\*\*  \* Deletes the given entities.  \*  \* **@param** entities  \* **@throws** IllegalArgumentException in case the given {@link Iterable} is (@literal null}.  \*/  **void** delete(Iterable<? **extends** T> entities);  /\*\*  \* Deletes all entities managed by the repository.  \*/  **void** deleteAll(); |

接口JpaSpecificationExecutor中的方法：（复杂查询，及分页）

|  |
| --- |
| /\*\*  \* Returns a single entity matching the given {@link Specification}.  \*  \* **@param** spec  \* **@return**  \*/  T findOne(Specification<T> spec);  /\*\*  \* Returns all entities matching the given {@link Specification}.  \*  \* **@param** spec  \* **@return**  \*/  List<T> findAll(Specification<T> spec);  /\*\*  \* Returns a {@link Page} of entities matching the given {@link Specification}.  \*  \* **@param** spec  \* **@param** pageable  \* **@return**  \*/  Page<T> findAll(Specification<T> spec, Pageable pageable);  /\*\*  \* Returns all entities matching the given {@link Specification} and {@link Sort}.  \*  \* **@param** spec  \* **@param** sort  \* **@return**  \*/  List<T> findAll(Specification<T> spec, Sort sort);  /\*\*  \* Returns the number of instances that the given {@link Specification} will return.  \*  \* **@param** spec the {@link Specification} to count instances for  \* **@return** the number of instances  \*/  **long** count(Specification<T> spec); |

1. 如果以上方法不满足，可以定义新的接口，添加新的方法

例如：（再讨论）

|  |
| --- |
| **public** **interface** TaskDao **extends** PagingAndSortingRepository<Task, Long>, JpaSpecificationExecutor<Task> {  Page<Task> findByUserId(Long id, Pageable pageRequest);  @Modifying  @Query("delete from Task task where task.user.id=?1")  **void** deleteByUserId(Long id);  } |

1. SearchDTO的使用

规则

|  |
| --- |
| **public** **enum** Operator {  ***EQ***, ***LIKE***, ***GT***, ***LT***, ***GTE***, ***LTE***  } |

DTO举例

|  |
| --- |
| **public** **class** TaskSearchDTO {  **private** String search\_LIKE\_title;  **private** String search\_EQ\_id;  **private** String search\_LIKE\_user\_name;  **private** Long search\_EQ\_user\_id; |

使用举例：

Demo1，不带分页

com.longdi.inventory.service.UserService

|  |
| --- |
| **public** List<User> search(UserSearchDTO searchDTO) {  Map<String, Object> searchParams = SearchUtils.*buildSearchParams*(searchDTO);  Specification<User> spec = SearchUtils.*buildSpecification*(searchParams, User.**class**);  **return** userDao.findAll(spec);  } |

Demo2，带分页

com.longdi.inventory.service.TaskService

|  |
| --- |
| **public** Page<Task> getUserTaskPage(TaskSearchDTO searchDTO, PageBean pageBean) {  PageRequest pageRequest = SearchUtils.*buildPageRequest*(pageBean);  Specification<Task> spec = SearchUtils.*buildSpecification*(searchDTO, Task.**class**);  **return** taskDao.findAll(spec, pageRequest);  } |

URL

|  |
| --- |
| <http://192.168.1.102:7080/LDDemo/user/user_tasks?id=2&pb.pageSize=6&pb.pageNumber=1&taskSearchDTO.search_LIKE_title=est&taskSearchDTO.search_EQ_id=3&taskSearchDTO.search_LIKE_user_name=B&taskSearchDTO.search_EQ_user_id=2>  <http://192.168.1.102:7080/LDDemo/user/search?userSearchDTO.search_LIKE_name=A&userSearchDTO.search_EQ_id=1&userSearchDTO.search_LIKE_loginName=A> |

SearchDTO不够用的情况