# 目标定义

结合现在的新的版本要求，需要现在如下几个方面进行预先的技术验证，形成示范的工程，然后整体项目按照这个规范进行

* 系统前端使用bootstrap
  + 查找比较简洁好用的模板
  + 对话框、表格Grid组件
  + 流式布局
* 母板布局和bootstrap集合
  + 从多种布局引擎中选择1个
* Spring MVC
  + 基于SpringMVC4整合1个完整的工程，包括：
    - 如上布局的form提交和验证
    - 后台Controller的@Valid，@Service的配合使用
    - 服务部分的组织规范

# 工具

一些windows下测试验证比较好用的工具【可从老吴处拷贝，如下的例子工程也已经下载】

ideaIC-2016.1.3.exe

spring-tool-suite-3.7.3.RELEASE-e4.6-win32-x86\_64.zip

jdk-8u60-windows-x64.exe

mysql-5.5.47-win32.zip

mysql-query-browser-1.1.20-win-noinstall.zip

apache-maven-3.3.9-bin.tar.gz

# 布局引擎

## 母板-inlcude

Github 上的spring-init

### web.xml

*<!-- JSP 母版页设置 -->*<**jsp-config**>  
 <**jsp-property-group**>  
 <**url-pattern**>\*.jsp</**url-pattern**>  
 <**el-ignored**>false</**el-ignored**>  
 <**page-encoding**>UTF-8</**page-encoding**>  
 <**scripting-invalid**>false</**scripting-invalid**>  
 <**include-prelude**>/WEB-INF/template/include.jsp</**include-prelude**>  
 </**jsp-property-group**>  
</**jsp-config**>

### include.jsp

<%@ page language=**"java"** contentType=**"text/html; charset=UTF-8"**%>  
<%@ page pageEncoding=**"UTF-8"**%>  
<%@ taglib uri=**"http://java.sun.com/jsp/jstl/core"** prefix=**"c"**%>  
<%@ taglib uri=**"http://java.sun.com/jsp/jstl/fmt"** prefix=**"fmt"**%>  
<%@ taglib uri=**"http://amateras.sf.jp/functions"** prefix=**"f"**%>  
<%@ taglib uri=**"http://www.springframework.org/tags"** prefix=**"spring"**%>  
<%@ taglib uri=**"http://www.springframework.org/tags/form"** prefix=**"form"**%>  
<fmt:requestEncoding value=**"UTF-8"** />

### Layout.jsp

<!DOCTYPE html>  
<html>  
<head>  
<meta http-equiv=**"Content-Type"** content=**"text/html; charset=UTF-8"**>  
<link href=**"<c:url value="**/resources/bootstrap/bootstrap.min.css**" />"** rel=**"stylesheet"** />  
<script type=**"text/javascript"** src=**"<c:url value="**/resources/js/jquery-1.10.2.min.js**" />"**></script>  
<script type=**"text/javascript"** src=**"<c:url value="**/resources/bootstrap/bootstrap.min.js**" />"**></script>  
<title>${param.title}</title>  
</head>  
<body>  
 <nav class=**"navbar navbar-default"** role=**"navigation"**>  
 <!-- Brand and toggle get grouped for better mobile display -->  
 <div class=**"navbar-header"**>  
 <button type=**"button"** class=**"navbar-toggle"** data-toggle=**"collapse"** data-target=**"#bs-example-navbar-collapse-1"**>  
 <span class=**"sr-only"**>Toggle navigation</span> <span class=**"icon-bar"**></span> <span class=**"icon-bar"**></span> <span  
 class=**"icon-bar"**></span>  
 </button>  
 <a class=**"navbar-brand"** href=**"${pageContext.request.contextPath}/home"**>Spring-Init</a>  
 </div>  
  
 <!-- Collect the nav links, forms, and other content for toggling -->  
 <!-- menu -->  
 <div class=**"collapse navbar-collapse"** id=**"bs-example-navbar-collapse-1"**>  
 <ul class=**"nav navbar-nav"**>  
 <li><a class=**"navbar-brand"** href=**"${pageContext.request.contextPath}/person/list"**>Person</a></li>  
 <li><a class=**"navbar-brand"** href=**"${pageContext.request.contextPath}/security/user/list"**>用户</a></li>  
 <li><a class=**"navbar-brand"** href=**"${pageContext.request.contextPath}/security/role/list"**>权限</a></li>  
 </ul>  
 <shiro:authenticated>  
 <form class=**"navbar-form navbar-right"** role=**"search"** style=**"margin-top: 15px;"**>  
 </form>  
 </shiro:authenticated>  
 </div>  
 <!-- /.navbar-collapse -->  
 </nav>  
  
 <div>${param.body}</div>  
  
 <nav class=**"navbar-default navbar-fixed-bottom"** role=**"navigation"**>  
 <div style=**"height: 17px; font-size: 9px;"** align=**"center"**>  
 <strong>Copyright &nbsp;&copy;&nbsp; 2014 &nbsp;&bull;&nbsp; TaiYuan Credo Software Co., Ltd. &nbsp;All  
 rights reserved.</strong>  
 </div>  
 </nav>  
</body>  
</html>

### 实际页面

<c:import url=**"/WEB-INF/template/layout.jsp"** charEncoding=**"UTF-8"**>  
 <c:param name=**"title"** value=**"PersonList"** />  
 <c:param name=**"body"**>

....

</c:param>  
</c:import>

*<!-- 根目录跳转到home页面 -->*<**mvc:view-controller path="/" view-name="redirect:/home"** />

*<!-- Handles HTTP GET requests for /resources/\*\* by efficiently serving   
 up static resources in the ${webappRoot}/resources directory -->*<**mvc:resources mapping="/resources/\*\*" location="/resources/"** />  
  
*<!-- 默认解析器,解析JSP+JSTL -->*<**bean id="defaultViewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  
 <**property name="viewClass" value="org.springframework.web.servlet.view.JstlView"**/>  
 <**property name="contentType" value="text/html"**/>  
 <**property name="prefix" value="/WEB-INF/views/"**/>  
 <**property name="suffix" value=".jsp"**/>  
</**bean**>

E:\java\spring-init-master\spring-init\src\main\resources\META-INF\spring\spring-shiro-web.xml

## 母板-SiteMesh

Github上的 springmvc-bootstrap

### Web.xml

<**filter**>  
 <**filter-name**>sitemesh</**filter-name**>  
<**filter-class**>com.opensymphony.sitemesh.webapp.SiteMeshFilter</**filter-class**>  
</**filter**>

<**filter-mapping**>  
 <**filter-name**>sitemesh</**filter-name**>  
 <**url-pattern**>/\*</**url-pattern**>  
</**filter-mapping**>

### decorators.xml

src/main/webapp/WEB-INF/decorators.xml

<**decorators defaultdir="/WEB-INF/decorators"**>  
 *<!-- Any urls that are excluded will never be decorated by Sitemesh -->* <**excludes**>  
 <**pattern**>/about.jsp</**pattern**>  
 </**excludes**>  
   
 <**decorator name="main-layout" page="main-layout.jsp"**>  
 <**pattern**>/\*</**pattern**>   
 </**decorator**>   
</**decorators**>

src/main/webapp/WEB-INF/decorators/main-layout.jsp

<%@ taglib uri=**"http://java.sun.com/jsp/jstl/core"** prefix=**"c"** %>  
<%@ taglib uri=**"http://www.opensymphony.com/sitemesh/decorator"** prefix=**"decorator"** %>

<decorator:body />

@RequestMapping(value=**"pathVariables/{foo}/{fruit}"**, method=RequestMethod.***GET***)  
**public** String pathVars(@PathVariable String foo, @PathVariable String fruit) {  
 *// No need to add @PathVariables "foo" and "fruit" to the model  
 // They will be merged in the model before rendering* **return "views/html"**;  
}  
  
@RequestMapping(value=**"dataBinding/{foo}/{fruit}"**, method=RequestMethod.***GET***)  
**public** String dataBinding(@Valid JavaBean javaBean, Model model) {  
 *// JavaBean "foo" and "fruit" properties populated from URI variables* **return "views/dataBinding"**;  
}

<%@ taglib uri=**"http://java.sun.com/jsp/jstl/core"** prefix=**"c"** %>  
<%@ page session=**"false"** %>  
<html>  
<head>  
 <title>Data Binding with URI Template Variables</title>  
 <link href=**"<c:url value="**/resources/form.css**" />"** rel=**"stylesheet"** type=**"text/css"** />   
</head>  
<body>  
<div class=**"success"**>  
 <h3>javaBean.foo: ${javaBean.foo}</h3>

## Thymeleaf

Github上的spring-boot-thymeleaf

<html xmlns:th=*"http://www.thymeleaf.org"*>

<head>

<link href=*"http://cdn.jsdelivr.net/webjars/bootstrap/3.3.6/css/bootstrap.min.css"*

th:href=*"@{/webjars/bootstrap/3.3.6/css/bootstrap.min.css}"*

rel=*"stylesheet"* media=*"screen"* />

</head>

<body>

<div th:fragment=*"footer"*>

**包含页**

<footer>

<div th:replace=*"fragments/footer :: footer"*>&copy; 2016 blog.codeleak.pl</div>

</footer>

<http://www.thymeleaf.org/documentation.html>

# 前端

## Bootstrap

<http://v3.bootcss.com/>



<!doctype html>

<html>

<head>

<meta charset="utf-8"/>

<meta http-equiv="X-UA-Compatible" content="IE=edge"/>

<meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1, maximum-scale=1"/>

<title> </title>

<meta name="keywords" content="跨平台开发,应用号,混合应用,混合APP,Hybrid APP,HTML5,移动应用开发平台,企业移动信息化"/>

<meta name="description" content="中国领先的移动应用云服务平台" />

<link rel="apple-touch-icon-precomposed" sizes="144x144" href=""/>

<!-- Bootstrap -->

<link href="../static/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!--[if lt IE 9]>

<script src="../assets/js/ie8-responsive-file-warning.js"></script><![endif]-->

<script src="../assets/js/ie-emulation-modes-warning.js"></script>

<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->

<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->

<!--[if lt IE 9]>

<script src="../static/bootstrap/js/html5shiv.min.js"></script>

<script src="../static/bootstrap/js/respond.min.js"></script>

<![endif]-->

</head>

<body class="home-layout ">

<header class="">

<div class="nav-left">

<a href="/" class="logo" title='APICLOUD-重新定义APP移动应用开发工具'>

</a>

</div>

<div class="nav-right mw-8">

<div class="nav-right">

<div class="unauthenticated">

<a href="/signin">

登录

</a>

<span>

</span>

<a href="/signup">

注册

</a>

</div>

</div>

</header>

<section class="number">

<div class="container">

</section>

<footer>

</footer>

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->

<script src="../static/bootstrap/js/jquery.min.js"></script>

<!-- Include all compiled plugins (below), or include individual files as needed -->

<script src="../static/bootstrap/js/bootstrap.min.js"></script>

<script src="../static/js/handlebars.js"></script>

</body>

</html>

<http://www.cnblogs.com/sunhaoyu/p/4275190.html>

<!-- bootstarp 的导航栏默认高度为 50px

class="nav navbar-inverse" 设置导航的条样式（当前北京为黑色） 还可以写成class="nav navbar-default"

role="navigation" 表示当前div 已设置为导航栏

class="container-fluid" 也就是告诉导航栏采用的是流布局方式

navbar-fixed-top 设置导航条始终显示在页面顶部 ，对应的navbar-fixed-bottom始终显示在下面

-->

<div class="nav navbar-inverse navbar-fixed-top" role="navigation">

<div class="container-fluid">

<div class="navbar-header">

<!-- 隐藏按钮当页面小于默认为768像素时，隐藏响应式布局里的内容，显示此按钮

data-target=".nav-collapse" 找的是下面设置响应布局的 div

-->

<button class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="icon-bar"></span> <!--每一个”span 标签就是一个小横线 这里的按钮就是由3小横线组成“-->

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<!-- 添加一个logo 因为导航栏的高度默认为50pk 为了以防外一将logo的高度设为50px-->

<a href="#" class="navbar-brand">

<img src="img/{1Y(FB52``7)ZKN(0EYQUZY.jpg" style="height: 50px; margin-top:-15px;"/>

</a>

</div>

<!--

class="collapse nav-collapse" 将当前div里的模块变成响应式布局

-->

<div class="collapse navbar-collapse">

<!-- 导航列表-->

<ul class="nav navbar-nav">

<!-- class="active" 为默认被选中的菜单 -->

<li class="active"><a href="#">首页</a></li>

<li><a href="#">导航一</a></li>

<li><a href="#">导航二</a></li>

</ul>

<!--导航的搜索框

class="navbar-form navbar-right" 第一个类标签标记为搜索栏

第二个"navbar-right"设置为靠右显示

-->

<div class="navbar-form navbar-right">

<input type="text" class="form-control" placeholder="请输入关键字"/>

<button class="btn btn-primary">搜索</button>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<a href="#" class="navbar-link" style="font-size: 18px; color: #FFFFFF">登 陆</a>

&nbsp;&nbsp;&nbsp;&nbsp;

<a href="#" class="navbar-link" style="font-size: 18px; color: #FFFFFF">注 册</a>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

</div>

最基本的一个页面效果

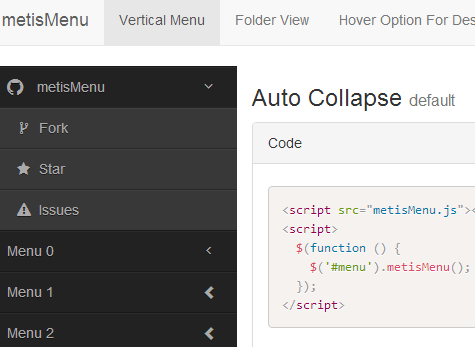
<http://blackrockdigital.github.io/startbootstrap-bare/>

## 组件

### 导航Menu

https://github.com/onokumus/metisMenu

<http://mm.onokumus.com/event.html#> 左导航和提示窗口



### 表单校验

jqBootstrapValidation-master.zip ReactiveRaven-jqBootstrapValidation-1.3.6-0-gd66d033.zip bootstrap2的版本，较老

bootstrap-validator-master.zip 采用，支持H5的形式，如果兼容一以前的有兼容方案

### 表单设计

Bootstrap-Form-Builder-gh-pages.zip FormbuildV1.0\_20140714.zip[中文版]

编写表单时，可以使用这个直接拖放形成表单的html内容

### Grid

bootstrap-table-develop.zip bootstrap-table-examples-master.zip

### 对话框

bootstrap3-dialog-master.zip

### 提示框

noty-master.zip 这个更简洁一些

<https://github.com/sciactive/pnotify>

### 整体布局

<http://startbootstrap.com>

<https://github.com/BlackrockDigital/startbootstrap> 总计有35个网站模板 其中sb2的质量和流行度好

startbootstrap-sb-admin-2-1.0.8.zip

### Dropdown类的悬停自动打开

<https://github.com/CWSpear/bootstrap-hover-dropdown>

bootstrap-hover-dropdown-master.zip

### TreeView

<https://github.com/jonmiles/bootstrap-treeview>

### 图标 fontawesome

用法：首先引入CSS文件：<link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-awesome.min.css" rel="stylesheet">  
然后用 <i class="fa fa-XXXX"></i> 输出图标  
比如用 <i class="fa fa-home"></i> 可输出   
更多内容请访问 **[http://fontawesome.io/](http://fontawesome.io/" \o "" \t "http://9iphp.com/_blank)**

<iclass="fa fa-camera-retro"></i> **fa-camera-retro**

To increase icon sizes relative to their container, use the **fa-lg** (33% increase), **fa-2x**, **fa-3x**, **fa-4x**, or **fa-5x** classes.

Use fa-fw to set icons at a fixed width.

<http://fontawesome.io/icons/>

<http://fontawesome.io/examples/>

# Thymeleaf和后台

直接的一个URL请求 **signup**

@RequestMapping(value = **"signup"**)  
**public** String signup(Model model) {  
 model.addAttribute(**new** SignupForm());  
 **return *SIGNUP\_VIEW\_NAME***;  
}

<!DOCTYPE **html**>  
<**html xmlns:th="http://www.thymeleaf.org"**>  
<**head**>  
 <**title**>Signup</**title**>  
 <**meta http-equiv="Content-Type" content="text/html; charset=UTF-8"**/>  
 <**link href="../../../resources/css/bootstrap.min.css" rel="stylesheet" media="screen" th:href="@{/resources/css/bootstrap.min.css}"**/>  
 <**link href="../../../resources/css/core.css" rel="stylesheet" media="screen" th:href="@{/resources/css/core.css}"** />  
 <**script src="http://code.jquery.com/jquery-latest.js"**></**script**>  
 <**script src="../../../resources/js/bootstrap.min.js" th:src="@{/resources/js/bootstrap.min.js}"**></**script**>  
</**head**>  
<**body**>  
<**form class="form-narrow form-horizontal" method="post" th:action="@{/signup}" th:object="${signupForm}"**>  
 *<!--/\* Show general error message when form contains errors \*/-->* <**th:block th:if="${#fields.hasErrors('${signupForm.\*}')}"**>  
 <**div th:replace="fragments/alert :: alert (type='danger', message='Form contains errors. Please try again.')"**>Alert</**div**>  
 </**th:block**>  
 <**fieldset**>  
 <**legend**>Please Sign Up</**legend**>  
 <**div class="form-group" th:classappend="${#fields.hasErrors('email')}? 'has-error'"**>  
 <**label for="email" class="col-lg-2 control-label"**>Email</**label**>  
 <**div class="col-lg-10"**>  
 <**input type="text" class="form-control" id="email" placeholder="Email address" th:field="\*{email}"** />  
 <**span class="help-block" th:if="${#fields.hasErrors('email')}" th:errors="\*{email}"**>Incorrect email</**span**>  
 </**div**>  
 </**div**>  
 <**div class="form-group" th:classappend="${#fields.hasErrors('password')}? 'has-error'"**>  
 <**label for="password" class="col-lg-2 control-label"**>Password</**label**>  
 <**div class="col-lg-10"**>  
 <**input type="password" class="form-control" id="password" placeholder="Password" th:field="\*{password}"**/>  
 <**span class="help-block" th:if="${#fields.hasErrors('password')}" th:errors="\*{password}"**>Incorrect password</**span**>  
 </**div**>  
 </**div**>  
 <**div class="form-group"**>  
 <**div class="col-lg-offset-2 col-lg-10"**>  
 <**button type="submit" class="btn btn-default"**>Sign up</**button**>  
 </**div**>  
 </**div**>  
 <**div class="form-group"**>  
 <**div class="col-lg-offset-2 col-lg-10"**>  
 <**p**>Already have an account? <**a href="signin" th:href="@{/signin}"**>Sign In</**a**></**p**>  
 </**div**>  
 </**div**>  
 </**fieldset**>  
</**form**>  
</**body**>  
</**html**>

@RequestMapping(value = **"signup"**, method = RequestMethod.***POST***)  
**public** String signup(@Valid @ModelAttribute SignupForm signupForm, Errors errors, RedirectAttributes ra) {  
 **if** (errors.hasErrors()) {  
 **return *SIGNUP\_VIEW\_NAME***;  
 }  
 Account account = **accountService**.save(signupForm.createAccount());  
 **accountService**.signin(account);  
 *// see /WEB-INF/i18n/messages.properties and /WEB-INF/views/homeSignedIn.html* MessageHelper.*addSuccessAttribute*(ra, **"signup.success"**);  
 **return "redirect:/"**;  
}

# Hibernate

@Temporal标签的作用很简单：

       (1) 如果在某类中有Date类型的属性，数据库中存储可能是'yyyy-MM-dd hh:MM:ss'要在查询时获得年月日，在该属性上标注@Temporal(TemporalType.DATE) 会得到形如'yyyy-MM-dd' 格式的日期。

      (2)如果在某类中有Date类型的属性，数据库中存储可能是'yyyy-MM-dd hh:MM:ss'要获得时分秒，在该属性上标注 @Temporal(TemporalType.TIME) 会得到形如'HH:MM:SS' 格式的日期。

      (3)如果在某类中有Date类型的属性，数据库中存储可能是'yyyy-MM-dd hh:MM:ss'要获得'是'yyyy-MM-dd hh:MM:ss'，在该属性上标注 @Temporal(TemporalType.TIMESTAMP) 会得到形如'HH:MM:SS' 格式的日期

@Column(name = "UPDATED\_AT", nullable = false)

@Temporal(TemporalType.TIMESTAMP)

private Date updatedAt;

@Transient表示该属性并非一个到数据库表的字段的映射,ORM框架将忽略该属性.  
    如果一个属性并非数据库表的字段映射,就务必将其标示为@Transient,否则,ORM框架默认其注解为@Basic  
    示例:  
    //根据birth计算出age属性  
    @Transient  
    public int getAge() {  
       return getYear(new Date()) - getYear(birth);  
    }

<http://blog.sina.com.cn/s/blog_935ebb670101dnre.html>

<http://docs.jboss.org/hibernate/orm/5.2/userguide/html_single/Hibernate_User_Guide.html>

## 1.类级别注解

@Entity     映射实体类

@Table    映射数句库表

 @Entity(name="tableName") - 必须，注解将一个类声明为一个实体bean。

属性：

name - 可选，对应数据库中的一个表。若表名与实体类名相同，则可以省略。

 @Table(name="",catalog="",schema="")  - 可选，通常和@Entity 配合使用，只能标注在实  体的 class 定义处，表示实体对应的数据库表的信息。

属性：

name - 可选，表示表的名称，默认地，表名和实体名称一致，只有在不一致的情况下才需            要指定表名

catalog - 可选，表示Catalog名称，默认为 Catalog("").

schema - 可选 , 表示 Schema 名称 , 默认为Schema("").

## 2.属性级别注解

@Id     映射生成主键

@Version    定义乐观锁

@Column     映射表的列

@Transient     定义暂态属性

### 2.1 与主键相关注解

@Id - 必须，定义了映射到数据库表的主键的属性，一个实体只能有一个属性被映射为主  键，置于 getXxxx()前。

@GeneratedValue(strategy=GenerationType,generator="") - 可选，用于定义主键生成策略。

属性：

Strategy - 表示主键生成策略，取值有：

GenerationType.*AUTO* - 根据底层数据库自动选择（默认），若数据库支持自动              增                                       长类型，则为自动增长。

GenerationType.*INDENTITY* - 根据数据库的Identity字段生成，支持DB2、MySQL、MS、SQL Server、SyBase与HyperanoicSQL数据库的Identity                                          类型主键。

GenerationType.*SEQUENCE -* 使用Sequence来决定主键的取值，适合Oracle、DB2等                                      支持Sequence的数据库，一般结合@SequenceGenerator使用。

(Oracle没有自动增长类型，只能用Sequence)

GenerationType.*TABLE*  - 使用指定表来决定主键取值，结合@TableGenerator使用。

如：

**@Id**

**@TableGenerator(name="tab\_cat\_gen",allocationSize=1)**

**@GeneratedValue(Strategy=GenerationType.*Table*)**

Generator - 表示主键生成器的名称，这个属性通常和ORM框架相关 , 例如：

Hibernate 可以指定 uuid 等主键生成方式

@SequenceGenerator — 注解声明了一个数据库序列。

属性：  
name - 表示该表主键生成策略名称，它被引用在@GeneratedValue中设置的“gernerator”值中。  
sequenceName - 表示生成策略用到的数据库序列名称。  
initialValue - 表示主键初始值，默认为0.  
allocationSize - 每次主键值增加的大小，例如设置成1，则表示每次创建新记录后自动加1，默认为50.

示例 :

**@Id**

**@GeneratedValues(strategy=StrategyType.*SEQUENCE*)**

    public int getPk() {

       return pk;

    }

Hibernate的访问类型为field时，在字段上进行注解声；

访问类型为property时，在getter方法上进行注释声明。

### 2.2 与非主键相关注解

@Version - 可以在实体bean中使用@Version注解,通过这种方式可添加对乐观锁定的支持

@Basic - 用于声明属性的存取策略：

@Basic(fetch=FetchType.EAGER)   即时获取（默认的存取策略）

@Basic(fetch=FetchType.LAZY)    延迟获取

@Temporal - 用于定义映射到数据库的时间精度：

@Temporal(TemporalType=DATE)       日期

@Temporal(TemporalType=TIME)       时间

@Temporal(TemporalType=TIMESTAMP) 两者兼具

@Column - 可将属性映射到列，使用该注解来覆盖默认值，@Column描述了数据库表中           该字段的详细定义，这对于根据 JPA 注解生成数据库表结构的工具非常有作用。

属性：

name - 可选，表示数据库表中该字段的名称，默认情形属性名称一致

nullable - 可选，表示该字段是否允许为 null，默认为 true

unique - 可选，表示该字段是否是唯一标识，默认为 false

length - 可选，表示该字段的大小，仅对 String 类型的字段有效，默认值255.

insertable - 可选，表示在ORM框架执行插入操作时，该字段是否应出现INSETRT               语句中，默认为 true

updateable - 可选，表示在ORM 框架执行更新操作时，该字段是否应该出现在               UPDATE 语句中，默认为 true. 对于一经创建就不可以更改的字段，该           属性非常有用，如对于 birthday字段。

columnDefinition - 可选，表示该字段在数据库中的实际类型。通常ORM 框架可以根     据属性类型自动判断数据库中字段的类型，但是对于Date 类型仍无法确定数据       库中字段类型究竟是 DATE,TIME 还是TIMESTAMP. 此外 ,String 的默认映射类型为 VARCHAR, 如果要将 String 类型映射到特定数据库的 BLOB或 TEXT 字段类型，该属性非常有用。

示例 :

**@Column(name="BIRTH",nullable="false",columnDefinition="DATE")**

    public String getBithday() {

       return birthday;

    }

@Transient - 可选，表示该属性并非一个到数据库表的字段的映射，ORM框架将忽略该属性，如果一个属性并非数据库表的字段映射，就务必将其标示为@Transient，否则ORM 框架默认其注解为 @Basic

示例 :

    // 根据 birth 计算出 age 属性

**@Transient**

    public int getAge() {

       return getYear(new Date()) - getYear(birth);

    }

### 2.3无注解属性的默认值

如果属性为单一类型,则映射为@Basic，

否则,如果属性对应的类型定义了@Embeddable注解,则映射为@Embedded，

否则,如果属性对应的类型实现了Serializable, 则属性被映射为@Basic并在一个列中保存该对象的serialized版本，

否则,如果该属性的类型为java.sql.Clob 或 java.sql.Blob,则作为@Lob并映射到适当的LobType.。

## 3.映射继承关系

@Inheritance注解来定义所选择的策略. 这个注解需要在每个类层次结构(class hierarchy) 最顶端的实体类上使用

## 4.映射实体bean的关联关系

### 4.1关联映射的一些定义

单向一对多：一方有集合属性，包含多个多方，而多方没有一方的引用。用户--->电子邮件

单向多对一：多方有一方的引用，一方没有多方的引用。论文类别---> 类别

双向一对多：两边都有多方的引用，方便查询。班级---> 学生

双向多对一：两边都有多方的引用，方便查询。

单向多对多：需要一个中间表来维护两个实体表。论坛--->文章

单向一对一：数据唯一，数据库数据也是一对一。舰船---> 水手

主键相同的一对一：使用同一个主键，省掉外键关联。客户---> 地址

单向：关系写哪边，就由谁管理。

双向：一般由多方管理。

@OneToMany(mappedBy="对方") //反向配置，对方管理。

### 4.2 关联映射的一些共有属性

@OneToOne、@OneToMany、@ManyToOne、ManyToMany的共有属性：

fetch - 配置加载方式。取值有

Fetch.*EAGER* -  及时加载，多对一默认是Fetch.*EAGER*

Fetch.*LAZY* - 延迟加载，一对多默认是Fetch.*LAZY*

cascade - 设置级联方式，取值有：

CascadeType.*PERSIST* - 保存

CascadeType.*REMOVE* - 删除

CascadeType.*MERGE* - 修改

CascadeType.*REFRESH* - 刷新

CascadeType.*ALL* - 全部

targetEntity - 配置集合属性类型，如：@OneToMany(targetEntity=Book.class)

@JoinColumn - 可选，用于描述一个关联的字段。

@JoinColumn和@Column类似，介量描述的不是一个简单字段，而是一个关联字段，例如描述一个 @ManyToOne 的字段。

属性：

name - 该字段的名称，由于@JoinColumn描述的是一个关联字段，如ManyToOne, 则默认的名称由其关联的实体决定。

例如，实体 Order 有一个user 属性来关联实体 User, 则 Order 的 user 属性为一个外键 ,

其默认的名称为实体User的名称 + 下划线 + 实体User的主键名称

### 4.3 一对一关联

@OneToOne – 表示一个一对一的映射

1.主表类A与从表类B的主键值相对应。  
 主表：@OneToOne(cascade = CascadeType.ALL)  
     @PrimaryKeyJoinColumn     public B getB(){

  Return b;

    }  
从表：无

2.主表A中有一个从表属性是B类型的b

主表：@OneToOne(cascade = CascadeType.ALL)  
   @JoinColumn(name="主表外键")   //这里指定的是数据库中的外键字段。  
   public B getB(){

return b;

       }

从表：无

3.主表A中有一个从表属性是B类型的b，同时，从表B中有一个主表属性是A类型的a  
主表：@OneToOne(cascade = CascadeType.ALL)  
   @JoinColumn(name="主表外键")   //这里指定的是数据库中的外键字段。  
   public B getB(){

return b;

       }  
从表：@OneToOne(mappedBy = "主表类中的从表属性")  
   public 主表类 get主表类(){

return 主表对象

      }  
  注意：@JoinColumn是可选的。默认值是从表变量名+"\_"+从表的主键（注意，这里加的是主键。而不是主键对应的变量）。

### 4.4 多对一关联

@ManyToOne - 表示一个多对一的映射，该注解标注的属性通常是数据库表的外键。

1.单向多对一：多方有一方的引用，一方没有多方的引用。

           在多方

@ManyToOne(targetEntity=XXXX.class)   //指定关联对象

@JoinColumn(name="")                            //指定产生的外键字段名

2.双向多对一：配置方式同双向一对多。

示例 :

    // 订单 Order 和用户 User 是一个 ManyToOne 的关系

    // 在 Order 类中定义

    @ManyToOne()

    @JoinColumn(name="USER")

    public User getUser() {

       return user;

}

### 4.5 一对多关联

@OneToMany - 描述一个一对多的关联，该属性应该为集合类型，在数据库中并没有实际字段。

1.单向一对多：一方有集合属性，包含多个多方，而多方没有一方的引用。

@OneToMany  默认会使用连接表做一对多关联

添加@JoinColumn(name="xxx\_id") 后，就会使用外键关联，而不使用连接表了。

2.双向一对多

1）在多方

@ManyToOne

@JoinColumn(name="自己的数据库外键列名")

2）在一方

@OneToMany(mappedBy="多端的关联属性名")

@JoinColumn(name="对方的数据库外键列名")

### 4.6 多对多关联

@ManyToMany - 可选，描述一个多对多的关联。

属性：

targetEntity - 表示多对多关联的另一个实体类的全名，例如：package.Book.class

mappedBy - 用在双向关联中，把关系的维护权翻转。

1.单向多对多关联：

    在主控方加入@ManyToMany注解即可。

2.双向多对多关联：

    两个实体间互相关联的属性必须标记为@ManyToMany，并相互指定targetEntity属性。有且只有一个实体的@ManyToMany注解需要指定mappedBy属性，指向targetEntity的集合属性名称。

# 其他

<http://demo.jeesite.com/jeesite>