**Eds**

**Enumeration of all possible remote method types.**

**Store的方法**

**Gxt编译速度慢，使用java写界面太别扭，uibind也相当不美观。**

**Xamp,apache来查看extjs的文档，速度非常快。**

**Deft JS**

视图控制器扩展自抽象基类Deft.mvc.ViewController，它提供了一个“control”配置项用于简化视图组件的访问器的创建及为它们绑定事件。

示例中，Gird和按钮可通过itemId（也支持自定义选择器）引用，它们的单击事件句柄则定义为视图控制器的方法。两个访问器函数会自动创 建：getContactsGrid() 和getEditButton()。当视图被销毁时，视图控制器的destroy方法会被调用，通过它可以阻止视图销毁。如果该方法返回true，视图将 被销毁，所有在视图控制器通过“control”定义创建的引用和监听事件会自动被删除。

MVC

使用mvc与创建普通的Ext界面组件一样，同样是create创建view对象,view中有controller和store两个对象的定义。并会同时创建该两个对象。Ccontroller创建时，也传入了view的对象引用，因此，在controller对像中能使用getview（）方法获得对应的view对象。

Edit.js 对象都没有对应的controller,这样合理吗？

Store与view的整合，及调用方法与store的关系。

Store里的方法是会view使用的，我们两只时应该仅仅调用view的方法？

Store里的数据变了，view会自动变化，

增加完成后，再

**idea 中的make实际是调用maven的build,其中**

**<resource>**

**<directory>src/main/java</directory>**

**<includes>**

**<include>\*\*/\*.xml</include>**

**<include>\*\*/\*.ftl</include><!-- jeff,将freemaker文件编译到目的目录-->**

**</includes>**

**</resource>**

**面向对象的设计，而不是面向数据表的设计，**

**优势：**

**灵活，可以通过entity生成表，没有的数据库注释之类的可以通过额外的注解进行添加，如增加@displayName,@comment等属性，displayName,用于生成前台的界面，comment用于生成数据库字段注释。**

**依赖的工具少：powerdesinger也不用了，可以用直接对象的UML图来替代。**

**便于跨数据库。**

**劣势：**

**对开发人员要求比较高。**

**开发数据库改成oracle，对代码同步没什么影响，**

**使用h2数据库，数据库同步通过文件方式来解决。**

**查询也有代码提示，只不过是没有数据编辑等功能。**

**extjs字典数据如何转换**

列如，性别在后台是F / M 在前台如何显示女 男

用renderer函数重新渲染啊

<http://shuimomo.blog.51cto.com/1141396/799738>

自动根据模型生成增删改查界面，就不太需要sencha archteect来做可

store不一定需要配置model,

<http://localhost:8080/api-0.0.1.js?apiNs>=

eds提供的js文件，根据@ExtDirectMethod动态生成direct的说明JSON

ch.ralscha.extdirectspring.controller.ApiController

webresources.txt

WebResourceProcessor processor = new WebResourceProcessor(!"development".equals(profile));

processor.process(servletContext);

用来整合压缩js和css文件。

如何把WebResourceProcessor和spring都改成debug模式？

System.setProperty("spring.profiles.active", "development");

平时开发时，使用maven生成文件，使用java类启动

/\* <debug> \*/

Ext.Loader.setConfig({

enabled: true,

paths: {

'xcore': 'app',

'Ext.ux': 'resources/extjs-gpl/4.2.2/ux'

}

});

/\* </debug> \*/

调试时才需要，生产环境下，直接会将所有代码合并到一个文件里，所以就不需要该配置信息了。

将一个panel放到top位置，里面嵌查询条件和toolbar

大量直接使用js,会造成许多不方便不安全的问题，如按钮的安全控制，国际化文件的转换，都需要将所有的控制文件下载到客户端，会不会有些安全的问题呢？当然也能在后端加以控制。

ExtDirectSpring contains an [APT Processor](http://app.rasc.ch/jenkins/job/ExtDirectSpring/site/apidocs/ch/ralscha/extdirectspring/generator/ModelAnnotationProcessor.html) that is build upon the [ModelGenerator](https://github.com/ralscha/extdirectspring/wiki/ModelGenerator). This processor creates Javascript model files during compile time.

In a Maven project add the following configuration into the pom.xml. The [apt-maven-plugin](https://github.com/mysema/maven-apt-plugin) runs by default during the generate-resources phase. Calling maven with **mvn generate-resources** on the command line will trigger this phase and creates the source code files if not already created.

idea 中，要make，才会重新编译类

**mvn generate-resources基于编译后的类文件来进行生成，在执行前一定要make**

Filter filter = request.getFilters().iterator().next();

String fieldName = filter.getField();

String oper = "";

String[] strs = fieldName.split("\\\_\\\_");

if(strs.length>1){

fieldName = strs[0];

oper = strs[1];

}

if(filter instanceof StringFilter){

if(oper ==null)oper = "like";

StringFilter stringFilter = (StringFilter)filter;

QUser.user.

if ()

}

}

hibernate查询出的对象序列化时，会不会触发懒加载？

ext.direct是不是通过model来控制，库列化的对象属性，根据model中的属性进行json化?

spring security3

servlet3.0增加了@WebServlet, @WebFilter, @WebListener等注解，servlet容器会在classpath扫描并注册所有的标注好的servlet, filter和listener。

在spring3.1以servlet3中annotation已经是一等公民了，可以实现任何原先只能在xml文件中配置的功能，并具有简洁，静态检查 及重构友好等优点。总体上来讲spring提供的“魔法”还是太多了，尤其是跟hibernate，事务，open session in view等机制结合在一起的时候，简洁代码的背后隐藏着太多的依赖关系，如果程序出了问题，排除这些魔法，一层一层地还原程序的本来面目，将是一件很需要 耐心的事情

一个@Configuration 类，相当于一个applicationContext.xml,

其中bean声明方式，包括注解和方法两种形式

|  |  |  |
| --- | --- | --- |
| @ComponentScan | <context:component-scan>， |  |
| @PropertySource | <context:property-placeholder> |  |