|  |
| --- |
| 恒生电子股份有限公司 |
| UI组件框架规范 |
| JRESPLUS-UI |
|  |
| **研发中心** |
| **2016/2/16** |

|  |
| --- |
| [在此处键入文档摘要。摘要通常为文档内容的简短概括。在此处键入文档摘要。摘要通常为文档内容的简短概括。] |

目录

[1. 概述 2](#_Toc444690287)

[1.1. 目的 2](#_Toc444690288)

[1.2. 范围 2](#_Toc444690289)

[1.3. 定义 2](#_Toc444690290)

[1.4. 读者 2](#_Toc444690291)

[1.5. 参考资料 2](#_Toc444690292)

[2. 基本原则 3](#_Toc444690293)

[2.1. 组件化封装 3](#_Toc444690294)

[2.2. 自描述定义 3](#_Toc444690295)

[2.3. 格式化注释 3](#_Toc444690296)

[3. 组件框架通用规范 4](#_Toc444690297)

[3.1. 组件框架概述 4](#_Toc444690298)

[3.2. 组件框架结构 5](#_Toc444690299)

[3.3. 组件生命周期 5](#_Toc444690300)

[3.3.1. 概述 5](#_Toc444690301)

[3.3.2. 开发期 6](#_Toc444690302)

[3.3.3. 引用期 8](#_Toc444690303)

[3.3.4. 运行期 9](#_Toc444690304)

[3.4. 组件运行容器 9](#_Toc444690305)

[3.4.1. 容器依赖技术 9](#_Toc444690306)

[3.4.2. 容器运行配置 10](#_Toc444690307)

[3.4.3. 容器运行时规范 10](#_Toc444690308)

[3.5. 组件样式分离规范 12](#_Toc444690309)

[4. 组件规范 13](#_Toc444690310)

[4.1. 组件定义 13](#_Toc444690311)

[4.1.1. 组件描述 13](#_Toc444690312)

[4.1.2. 组件包命名 16](#_Toc444690313)

[4.1.3. 组件包结构 16](#_Toc444690314)

[4.2. 组件开发 17](#_Toc444690315)

[4.3. 组件使用 17](#_Toc444690316)

[4.4. 系统组件封装 17](#_Toc444690317)

[5. 附录 18](#_Toc444690318)

[5.1. components.xsd对象描述 18](#_Toc444690319)

[5.2. components.xml组件定义样例 27](#_Toc444690320)

[5.3. 系统组件列表 28](#_Toc444690321)

# 概述

《UI组件框架规范》下文简称“规范”。

规范在JRESPLUS-UI的技术框架下，为开发人员提供从事UI组件开发、使用活动的标准依据。

其中，

第二章基本原则主要用来说明制定本规范的指导思想和设计依据；

第三章组件框架通用规范描述了组件框架全貌，框架全局性的约束和标准都在此定义；

第四章组件规范用于定义组件级的标准和约束，同时也列举了组件框架自身的系统组件和第三方组件内容。

## 目的

* **在组件的开发活动中：**组件开发者根据规范进行组件开发活动。
* **在组件的使用活动中：**组件使用者根据规范进行应用开发活动。

## 范围

规范适用公司内部所有使用JRESPLUS-UI框架进行页面开发的产品。

## 定义

**组件：**UI组件是为开发和构建用户界面部分所提供的，具备功能独立、接口规范的软件实体。

**组件包：**用来打包UI组件的文件结构，本规范中，组件均以jar包方式进行统一打包发布和引用。

**组件框架：**UI组件框架包含运行、加载和解析组件的容器，它定义组件的整体结构，以及组件间的协作控制流程。

## 读者

**组件开发者：**从事生产UI组件的软件开发人员。

**组件使用者：**使用UI组件从事业务应用开发的软件开发人员。

**组件框架维护者：**组件框架的开发和维护人员。

## 参考资料

《jresplus-ui组件开发指南.docx》

《jresplus-ui组件使用指南.docx》

《jresplus-ui组件样式分离规范.docx》

# 基本原则

## 组件化封装

通过组件化改造，JRESPLUS-UI将原来整合在一个工程中的所有资源按组件功能的划分，分解成可以单独发布和维护、自由组合和使用、统一标准纳入运行容器的独立组件发布包。组件化封装满足组件的开发、引用、运行有清晰的边界，与组件之外的代码实现隔离。

组件开发时，组件开发者可以建立独立的组件开发工程，开发结束后，可以通过组件开发工程提供的脚本打包成独立的组件发布包提交公司Maven仓库统一管理。

组件使用时，应用开发者通过maven坐标依赖所需组件包。应用开发工程提供针对组件包的资源合并压缩预编译脚本。

组件运行时，组件容器优先加载应用预编译的合并资源，当合并资源不存在时，组件包内容被解压成源代码，容器按改造前的逻辑加载资源。

组件化封装后，组件框架自身包含的系统组件在日后维护过程中可以进行增量发布，即修改了哪个组件，就单独发布此组件的更新版本，其它组件不受干扰。使用组件的具体应用工程自身需要区分应用的源码资源与运行时资源，针对源码资源的增量发放不受组件化的影响；运行时资源如果被包含在应用发布包内，后期单个组件的升级会影响到原发布包内的运行时资源（应用工程引入新组件版本后，需要通过预编译脚本重新集成应用工程才能将组件变更应用至工程页面代码中）。要解决应用工程的增量发布问题，可以将预编译活动从产品发布阶段移至部署阶段。

## 自描述定义

为了使运行预编译脚本、运行容器、开发工具能对组件包进行正确的识别，需要为组件定义一套可以自描述的语言，通过自描述定义可以将组件以下信息准确无误的表述给计算机程序：

* 组件标识：唯一标识组件
* 组件依赖：组件与其它组件的关系
* 组件资源：组件提供功能的具体资源分类和分类资源位置
* IDE工具信息：为用于组件开发的IDE工具提供必要的组件表现信息

## 格式化注释

为了简化组件对其内部的公开API的暴露，定义一套规范的格式化注释。组件开发者通过注释完成对组件API的自描述定义。组件开发工程和应用开发工程通过独立的DOC生成脚本为开发者提供从组件包提取API Document的工具。

* **组件类标注示例：**

|  |
| --- |
| /\*\*  \* @name Horn.Panel  \* @class  \* 面板组件</br>  \* 属于容器组件，一般做为单行组件的容器使用  \*/  /\*\*@lends Horn.Panel# \*/ |

* **组件属性标注示例：**

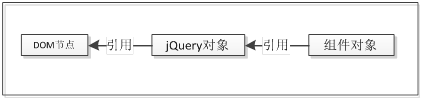
|  |
| --- |
| /\*\*  \* 是否展开(只有在expandable为true生效 ),默认不展开  \* @name Horn.Panel#<b>isExpand</b>  \* @type Boolean  \* @default false  \* @example  \* #@panel({"expandable":true,"isExpand":true})  \* #textfield({"label":"流程名称：", "name":"hsBpmNameLike"})  \* #end  \*/ |

* **组件方法标注示例：**

|  |
| --- |
| /\*\*  \* 隐藏<br/>  \* 隐藏面板容器及容器内的内容  \* @name Horn.Panel#hide  \* @function  \* @return {void}  \*/ |

# 组件框架通用规范

## 组件框架概述



从整体上说，UI组件由三部分组成：DOM节点、jQuery对象、组件对象。

* **DOM节点：**是指浏览器中HTML 节点的对象。它有基本的属性（例如：nodeType）和方法（例如：getElementById）。
* **jQuery对象：**是指用jQuery类库的选择器查询出来的对象,该对象中存放DOM节点的引用。它提供了很多操作DOM节点的方法。通过jQuery对象可以修改DOM节点的属性，也可以绑定DOM节点的事件。
* **组件对象：**是指通过执行组件的velocity宏进行封装，由jQuery创建出来的对象。组件对象中有很多方法，其中有一些方法是对组件功能的实现，供用户调用，例如：setValue、getValue方法。有一些方法是用于组件的初始化操作(初始化组件的属性、创建DOM节点、在特定的DOM节点上绑定事件。

## 组件框架结构

组件框架由三部分组成：组件规范、组件容器、标准组件。

* 组件规范：这部分是组件框架的核心，它包含组件相关的规范文档定义。其中，组件框架规范是框架的顶层规范文档；组件样式分离规范侧重描述UI组件与样式的代码分离，为实现组件的一键换肤提供规范保障；组件开发指南用于指导组件开发者如何实现组件的开发、调试和发布；组件使用指南用于指导组件使用者如何实现组件的引入、调用和应用调试、应用集成。
* 组件容器：组件容器定义了组件不同生命周期下赖以存在的环境，它包括：组件开发工程模板、应用开发工程模板、组件发布包、组件运行时容器，分别对应组件的开发期、引用期和运行期。
* 标准组件：组件框架项目在既定的规范基础上，会对原JRESPLUS-UI提供的系统组件进行规范化封装，形成松散耦合的标准组件。各标准组件实现独立发布，独立维护。按组件的实现方式不同，我们将标准组件划分为JS组件、CSS组件和VM组件三类，其中，JS组件可以直接对第三方中间件(如jQuery、ztree)进行封装，这类封装组件我们称之为第三方组件。

## 组件生命周期

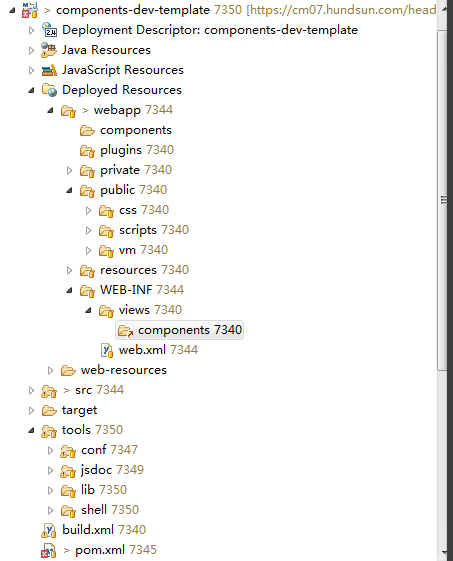
### 概述

组件从被生产到被引用，最后到运行，其生命周期可以划分为三个阶段：

1. 开发期：组件开发人员在组件开发工程模板指导下进行组件生产的过程。
2. 引用期：应用开发人员在应用开发工程模板指导下进行应用开发的过程。
3. 运行期：引用了组件的应用被部署到运行环境后，组件在运行容器中被解析和渲染的过程。

### 开发期

组件开发期内，组件开发者完成组件的开发、调试、发布事项。组件开发工程结构如下：



组件开发工程是一个maven管理的war工程，可以独立运行。组件开发者可以在此工程中编写组件相关代码，并进行调试工作。

**其中，开发的工作目录有：**

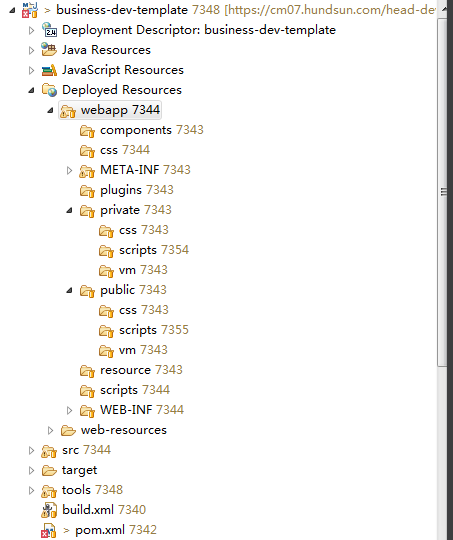
* **src/webapp/META-INF：**配置目录，开发者需要完成此目录下components.xml的配置。
* **src/webapp/private：**私有目录，开发者可以将组件内部逻辑代码放在此目录下，组件发布后，组件的使用者不能直接调用此目录下的资源。
* **src/webapp/public：**公共目录，开发者可以将组件对外暴露代码放在此目录下，组件发布后，组件的使用者可以直接调用此目录下的资源。
* **src/webapp/resource：**资源目录，可以将组件开发过程中引用的image、flash等资源放在此目录中。
* **src/webapp/plugins：**如果组件开发时使用了第三方代码，如jQuery，ztree等，需要将第三方代码放在plugins中。

**此外，组件开发工程在进行调试时还用到以下目录：**

* **src/webapp/components：**组件JS、CSS部分的运行时目录，运行调试时，组件代码中的js和css部分会被预编译脚本复制到此目录，同时public中的js和css还会被合并压缩成components.js、components-debug.js和components.css。同时，通过预编译脚本现在此目录上创建组件包命名空间对应的子目录名，然后将组件包内private目录下JS和CSS资源复制到对应命名空间的子目录下。
* **src/webapp/WEB-INF/views/components：**组件vm部分的运行时目录，运行调试时，组件代码中的vm部分会被预编译脚本复制到此目录，同时public中的vm还会被合并压缩成components.vm。同时，通过预编译脚本现在此目录上创建组件包命名空间对应的子目录名，然后将组件包内private目录下VM资源复制到对应命名空间的子目录下。

### 引用期

组件引用期内，应用开发者完成组件的依赖、调用、调试及应用发布事项。应用开发工程结构如下：



应用开发工程可以是一个maven管理的war工程，可以独立运行。应用开发者可以在此工程中编写组件相关代码，并进行调试工作。

**其中，开发的工作目录有：**

* **src/webapp/scripts：**用于放置应用开发中的js文件
* **src/webapp/css：**用于放置应用开发中的css文件
* **src/webapp/WEB-INF/views/layout：**用于放置应用开发中的视图层布局类文件。
* **src/webapp/WEB-INF/views/screen：**用于放置应用开发中的视图层内容类文件。
* **src/webapp/WEB-INF/views/components：**用于放置应用开发中的的公共宏定义文件。

**此外，应用开发工程在进行调试时还用到以下目录：**

* **src/webapp/components：**预编译工具会将组件包内的JS和CSS资源打包压缩到此目录的components.js、components-debug.js和components.css。
* **src/webapp/WEB-INF/views/components：**预编译工具会将组件包内的VM资源打包压缩到此目录的components.vm。
* **src/webapp/plugins：**预编译工具会将组件包内的第三方代码解压到此目录。
* **src/webapp/WEB-INF/lib：**用于放置依赖库下载的组件包

### 运行期

组件运行期内，应用工程结构与引用期一致。具体目录存放资源可参见引用期。

## 组件运行容器

组件运行容器是应用产品运行期的UI组件运行环境，组件运行容器的职责有：

* 应用启动时，对应用使用的组件进行统一装载
* 请求访问时，对页面组件的统一解析
* 为提高页面渲染性能，组件容器还需要对组件解析结果进行缓存

### 容器依赖技术

JRESPLUS-UI组件容器依赖了以下jar包：

|  |  |  |
| --- | --- | --- |
| groupId | artifactId | Version |
| com.hundsun.jresplus | jresplus-common | 1.0.9+ |
| com.hundsun.jresplus | jresplus-mvc | 2.0.1+ |
| com.hundsun.jresplus | jresplus-ui-runtime | 2.0.1+ |
| com.alibaba | fastjson | 1.1.23+ |
| com.caucho | hessian | 4.0.7+ |
| com.thoughtworks.xstream | xstream | 1.4.4+ |
| commons-codec | commons-codec | 1.4+ |
| commons-digester | commons-digester | 2.0+ |
| commons-fileupload | commons-fileupload | 1.2.1+ |
| javax.validation | validation-api | 1.0.0.GA+ |
| javax.xml.bind | jaxb-api | 2.2+ |
| org.apache.velocity | velocity | 1.7+ |
| org.apache.velocity | velocity-tools | 1.3+ |
| org.codehaus.jackson | jackson-mapper-asl | 1.5.3+ |
| org.hibernate | hibernate-validator | 4.2.0.Final+ |
| org.slf4j | slf4j-api | 1.5.6+ |
| org.springframework | spring-context | 3.1.4.RELEASE+ |
| org.springframework | spring-oxm | 3.1.4.RELEASE+ |
| org.springframework | spring-webmvc | 3.1.4.RELEASE+ |

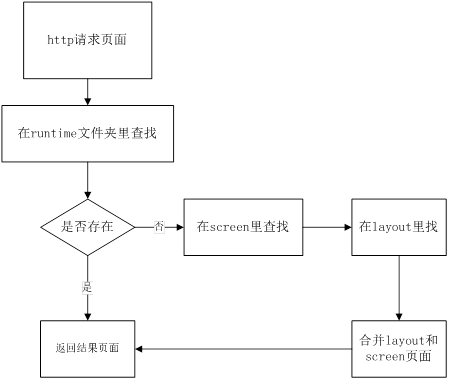
### 容器运行配置

要使用JRESPLUS-UI框架的运行时容器需要在WEB-INF/web.xml中进行以下过滤器配置：

|  |
| --- |
| <filter>  <filter-name>hornFilter</filter-name>  <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>  <init-param>  <param-name>targetBeanName</param-name>  <param-value>hornFilter</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>hornFilter</filter-name>  <url-pattern>\*.htm</url-pattern>  </filter-mapping>  <filter-mapping>  <filter-name>hornFilter</filter-name>  <url-pattern>\*.json</url-pattern>  </filter-mapping> |

### 容器运行时规范

组件化改造完成后，工程的JS、CSS、VM资源在预编译期已通过脚本进行合并和压缩。但是运行容器为了能够兼容历史工程，因此，在运行期解析逻辑需要做兼容处理：优先加载请求对应的压缩资源，如果找不到压缩资源，则继续按原加载逻辑加载screen和layout资源。



* JS资源加载顺序
* CSS资源加载顺序
* VM宏加载顺序

## 组件样式分离规范

参见《jresplus-ui组件样式分离规范》。

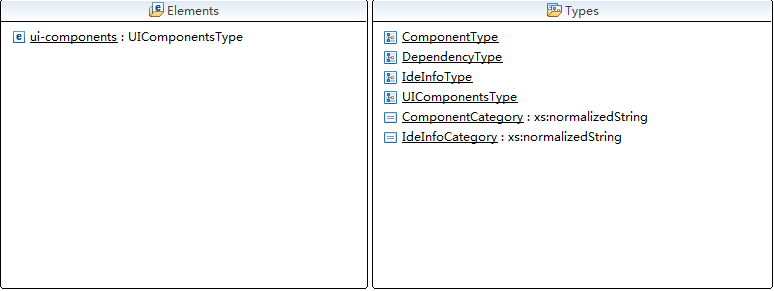
# 组件规范

## 组件定义

### 组件描述

组件开发和使用过程中，采用XML文件进行组件的描述工作。每个组件包的根目录下都使用一个components.xml组件定义文件来描述组件包本身以及包内组件的具体信息。

框架使用XSD文件(components.xsd)对组件定义文件格式进行描述。



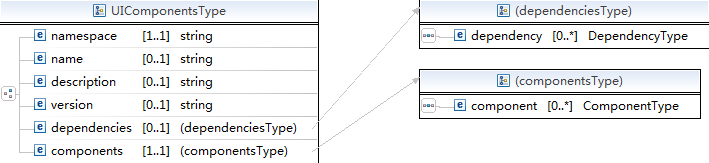
|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 标识 | 名称 | 描述 |
| element | ui-components | UI组件包 | XML文档的根节点，表示一个UI组件包 |
| complexType | UIComponentsType | 组件包类型 | 用来描述组件包具体信息 |
| complexType | ComponentType | 组件类型 | 用来描述组件具体信息 |
| complexType | DependencyType | 组件依赖项类型 | 用来描述组件包的依赖项信息 |
| complexType | IdeInfoType | 组件IDE工具配套信息类型 | 用来描述组件用于IDE工具中使用的基本信息 |
| simpleType | ComponentCategory | 组件分类 | 枚举值：  JS：js组件  VM：vm组件  CSS：css组件 |
| simpleType | IdeInfoCategory | IDE信息分类 | 枚举值： |

XSD具体内容参见附录：[5.1.components.xsd对象描述](#_components.xsd对象描述)。

1. **组件包描述：**

面对组件开发活动，组件包是组件的最小发布单位；面对组件使用活动，它也是组件的最小引用单位。框架的组件开发工程模板为组件开发者提供组件包的集成脚本，脚本可以将工程打包成一个jar格式用于发布的组件包，发布的组件包纳入maven仓库统一管理；框架的应用开发工程模板通过maven的POM文件声明组件包的引入，每一个maven坐标(groupId.artifactId)用来定位唯一的组件包。

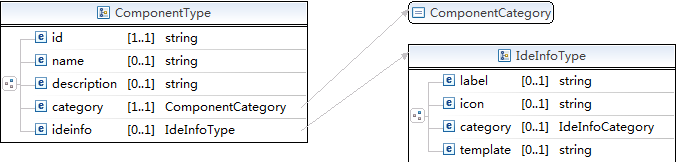
组件包描述包含以下内容：



|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 标识 | 名称 | 描述 |
| element | namespace | 组件包命名空间 | 用来唯一标识一个组件包，此元素是必选项，包内的所有组件会共用此命名空间 |
| element | name | 组件包名称 | 用来定义组件包的中文名称，此元素是可选项 |
| element | description | 组件包摘要 | 用来描述组件包的具体摘要信息，此元素是可选项 |
| element | version | 组件包版本 | 用来定义组件包的版本信息，此元素是可选项。  版本号用3位整型数字通过”.”号连接标识，规范格式：*[主版本号].[子版本号].[修正版本号]*。  如果组件开发着忽略version元素定义，则默认为[1.0.0]版本。 |
| element | dependencies | 组件包依赖集合 | 用来定义此组件包引入的其它包信息，此元素是可选项。每个集合项是DependencyType类型 |
| element | components | 组件包发布组件集合 | 用来定义此组件包对外发布的组件信息，此元素是必选项。每个集合项是ComponentType类型 |

1. **组件描述：**

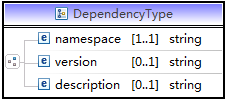
组件描述包含以下内容：



|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 标识 | 名称 | 描述 |
| element | id | 组件标识 | 用来在包内唯一标识一个组件，此元素是必选项 |
| element | name | 组件名称 | 用来定义组件的中文名称，此元素是可选项 |
| element | description | 组件摘要 | 用来描述组件的具体摘要信息，此元素是可选项 |
| element | category | 组件分类 | 用来描述组件分类，此元素是必选项  组件分类取值是一个枚举项，它包含*[JS,CSS,VM]*三个值，分别代表JS组件、CSS组件、VM组件 |
| element | ideinfo | 组件IDE工具配套信息 | 用来描述此组件在IDE工具中表现时需要额外定义的外观、代码模板等描述信息 |

1. **组件包依赖项描述：**

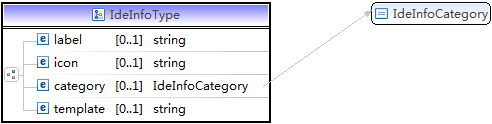
组件描述包含以下内容：



|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 标识 | 名称 | 描述 |
| element | namespace | 依赖项命名空间 | 所依赖的组件包对应的命名空间 |
| element | version | 依赖项版本 | 所依赖的组件包对应的版本 |
| element | description | 依赖项摘要 | 描述依赖项的摘要信息 |

1. **组件IDE基本信息描述：**

组件IDE基本信息描述包含以下内容：



|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 标识 | 名称 | 描述 |
| element | label | 组件显示标签 | 组件在IDE中的显示标签 |
| element | icon | 组件显示图标 | 组件在IDE中的显示图标 |
| element | category | 组件显示分类 | 组件在IDE中的显示区块分类，IDE按不同分类组织组件，category用来分类组件在IDE工具中的显示区块 |
| element | template | 组件模板 | 用来描述组件的模板代码，此元素是可选项，且只有在VM组件分类时才生效。模板代码用于为工具提供组件代码生成功能 |

### 组件包命名

首先，组件包以jar包格式发布，命名规则可参考java的包命名规则，由小写英文单词组成，单词之间使用”.”号分割。

其次，组件包采用maven仓库进行统一管理。因此，组件包命名也遵循maven命名规范：**groupId+artifactId+version**确定一个组件的maven仓库坐标。由于组件包的集成使用了maven的assembly插件，因此在组件包命名末尾还增加了**classifier的值[components]。**

**组件包名格式如下：**

**[组件包命名空间]-[组件包版本号]-components.jar**

**例如：**

com.hundsun.jresplus.ui.textfield-2.0.1-components.jar

### 组件包结构

组件包的结构如下图所示：

组件包内包含五个顶层目录：

**META-INF：**放置组件描述文件。

**public：**放置组件对外开放的js、vm和css资源。

**private：**放置组件内部私有的js、vm和css资源。

**plugins：**当组件包的thirdparty取值为true时，表示此组件包是第三方组件封装包，此时，需要将第三方源码放在plugins目录中。其它场景下，此目录为空。

**resources：**放置组件开发中涉及组件的其它资源文件，如images、flash等。resource下第一层目录按组件ID进行区分，每个组件的资源文件放在对应ID命名的目录下；第二层目录按资源分类进行区分，如image、flash等。

## 组件开发

参见《jresplus-ui组件开发指南》。

## 组件使用

参见《jresplus-ui组件使用指南》。

## 系统组件封装

组件框架提供对原JRESPUS-UI组件的封装升级，被封装后的系统组件将独立发布。

系统组件列表参见附录[5.3.系统组件列表](#_系统组件列表)。

# 附录

## components.xsd对象描述

|  |
| --- |
| <?xml version=*"1.0"*?>  <xs:schema xmlns:xs=*"http://www.w3.org/2001/XMLSchema"*  elementFormDefault=*"qualified"* attributeFormDefault=*"qualified"*>  <!-- 描述UI组件标签 -->  <xs:element name=*"ui-components"* type=*"UIComponentsType"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件包元素是XML描述符的根元素.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <!-- 描述UI组件包类型 -->  <xs:complexType name=*"UIComponentsType"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件包元素是XML描述符的根元素.  ]]>  </xs:documentation>  </xs:annotation>  <xs:all>  <xs:element minOccurs=*"1"* name=*"namespace"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件包的命名空间.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element minOccurs=*"0"* name=*"name"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件包的名称.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element minOccurs=*"0"* name=*"description"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件包的详细描述.该元素可以被指定为CDATA。  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element minOccurs=*"0"* name=*"version"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件包的版本.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element minOccurs=*"0"* name=*"dependencies"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件包的所有依赖项.  ]]>  </xs:documentation>  </xs:annotation>  <xs:complexType>  <xs:sequence>  <xs:element name=*"dependency"* minOccurs=*"0"* maxOccurs=*"unbounded"*  type=*"DependencyType"* />  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element minOccurs=*"1"* name=*"components"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件包内的所有组件.  ]]>  </xs:documentation>  </xs:annotation>  <xs:complexType>  <xs:sequence>  <xs:element name=*"component"* minOccurs=*"0"*  maxOccurs=*"unbounded"* type=*"ComponentType"* />  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:all>  </xs:complexType>  <!-- 描述组件依赖类型 -->  <xs:complexType name=*"DependencyType"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件包的依赖项类型.  ]]>  </xs:documentation>  </xs:annotation>  <xs:all>  <xs:element minOccurs=*"1"* name=*"namespace"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  所依赖的UI组件包的命名空间.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element minOccurs=*"0"* name=*"version"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  所依赖的UI组件包的版本.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element minOccurs=*"0"* name=*"description"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  依赖描述.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  </xs:all>  </xs:complexType>  <!-- 描述组件类型 -->  <xs:complexType name=*"ComponentType"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件包内的组件类型.  ]]>  </xs:documentation>  </xs:annotation>  <xs:all>  <xs:element minOccurs=*"1"* name=*"id"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件标识符.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element minOccurs=*"0"* name=*"name"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件名称.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element minOccurs=*"0"* name=*"description"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  UI组件描述.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element minOccurs=*"1"* name=*"category"* type=*"ComponentCategory"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  组件分类.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element name=*"ideinfo"* minOccurs=*"0"* maxOccurs=*"1"*  type=*"IdeInfoType"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  组件IDE工具配套信息.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  </xs:all>  </xs:complexType>    <!-- 描述组件IDE基本信息 -->  <xs:complexType name=*"IdeInfoType"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  组件IDE基本信息.  ]]>  </xs:documentation>  </xs:annotation>  <xs:all>  <xs:element minOccurs=*"0"* name=*"label"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  组件标签.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element minOccurs=*"0"* name=*"icon"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  组件图标.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element minOccurs=*"0"* name=*"category"* type=*"IdeInfoCategory"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  组件IDE分类.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  <xs:element minOccurs=*"0"* name=*"template"* type=*"xs:string"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  组件模板.  ]]>  </xs:documentation>  </xs:annotation>  </xs:element>  </xs:all>  </xs:complexType>  <!-- 描述组件分类类型 -->  <xs:simpleType name=*"ComponentCategory"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  组件分类类型.  ]]>  </xs:documentation>  </xs:annotation>  <xs:restriction base=*"xs:normalizedString"*>  <xs:enumeration value=*"JS"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  JS组件.  ]]>  </xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value=*"CSS"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  CSS组件.  ]]>  </xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value=*"VM"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  VM组件.  ]]>  </xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>    <!-- 描述组件IDE分类类型 -->  <xs:simpleType name=*"IdeInfoCategory"*>  <xs:annotation>  <xs:documentation source=*"version"*>1.0.0+</xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  组件IDE分类类型.  ]]>  </xs:documentation>  </xs:annotation>  <xs:restriction base=*"xs:normalizedString"*>  <xs:enumeration value=*"container"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  容器组件.  ]]>  </xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value=*"hidden"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  隐藏组件.  ]]>  </xs:documentation>  </xs:annotation>  </xs:enumeration>  <xs:enumeration value=*"normal"*>  <xs:annotation>  <xs:documentation source=*"version"*>  1.0.0+  </xs:documentation>  <xs:documentation source=*"description"*>  <![CDATA[  普通组件.  ]]>  </xs:documentation>  </xs:annotation>  </xs:enumeration>  </xs:restriction>  </xs:simpleType>  </xs:schema> |

## components.xml组件定义样例

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <ui-components xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:noNamespaceSchemaLocation=*"components.xsd"*>  <namespace>test-components</namespace>  <name>测试组件包</name>  <description><![CDATA[组件包的具体描述]]></description>  <dependencies>  <dependency>  <namespace>base-components</namespace>  <description><![CDATA[依赖基础组件包]]></description>  </dependency>  <dependency>  <namespace>field-components</namespace>  <version>1.0.1</version>  <description><![CDATA[依赖字段组件包]]></description>  </dependency>  </dependencies>    <components>  <component>  <category>JS</category>  <id>TestComponent</id>  <name>测试JS组件</name>  <description><![CDATA[测试组件的描述信息]]></description>  </component>  <component>  <category>CSS</category>  <id>TestComponent</id>  <name>测试CSS组件</name>  <description><![CDATA[测试组件的描述信息]]></description>  </component>  <component>  <category>VM</category>  <id>TestComponent</id>  <name>测试VM组件</name>  <description><![CDATA[测试组件的描述信息]]></description>  <ideinfo>  <label>测试组件</label>  <icon>images/component/test.png</icon>  <template>  <![CDATA[  #@screen({"title":"","columns":""})    #end  ]]>  </template>  </ideinfo>  </component>  </components>  </ui-components> |

## 系统组件列表

|  |  |
| --- | --- |
| 组件包命名空间 | 组件包名称 |
| com.hundsun.jresplus.ui.base | 框架基础组件 |
| com.hundsun.jresplus.ui.textfield | 普通文本输入组件 |
| com.hundsun.jresplus.ui.textarea | 普通文本输入组件 |
| com.hundsun.jresplus.ui.menu | 可绑定按钮的多级菜单组件 |
| com.hundsun.jresplus.ui.combox | 下拉选项组件 |
| com.hundsun.jresplus.ui.checkbox | 复选框组件 |
| com.hundsun.jresplus.ui.password | 密码录入组件 |
| com.hundsun.jresplus.ui.panel | 面板组件 |
| com.hundsun.jresplus.ui.calendar | 日期组件 |
| com.hundsun.jresplus.ui.window | 弹出窗口组件 |
| com.hundsun.jresplus.ui.pagebar | 分页栏组件 |
| com.hundsun.jresplus.ui.label | 用于显示标签的标签组件 |
| com.hundsun.jresplus.ui.typefield | 支持金额、卡号输入类型的input组件 |
| com.hundsun.jresplus.ui.canvas | canvas容器组件 |
| com.hundsun.jresplus.ui.form | FORM表单组件 |
| com.hundsun.jresplus.ui.ztree | ztree封装组件 |
| com.hundsun.jresplus.ui.datagrid | 数据列表展现组件 |
| com.hundsun.jresplus.ui.grid | 数据列表组件 |
| com.hundsun.jresplus.ui.querytable | 带查询表单的表格组件 |
| com.hundsun.jresplus.ui.buttonpanel | 按钮容器组件 |
| com.hundsun.jresplus.ui.hidden | 隐藏域组件 |