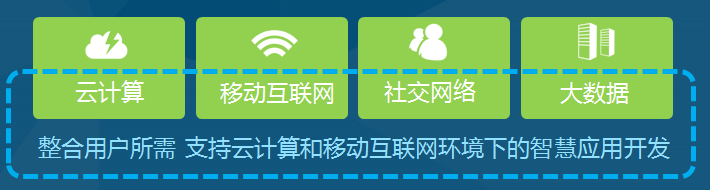
# Smart Cloud 开发指南

## 一、概述

#### 1.1什么是智慧云平台



（图 1-1 智慧云平台）

如图1-1所示：智慧云平台支持基于云计算、移动互联网、社交网络和大数据环境下的智慧应用开发，提供web或移动app应用所需的应用框架、常见界面组件、应用组件和数据分析等功能。

#### 1.2智慧云平台的技术架构



（图 1-2 平台架构）

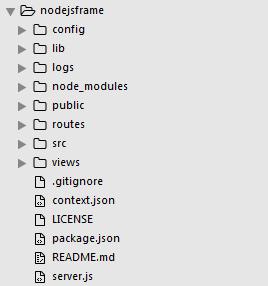
如图1-2所示：智慧云平台架构一共分为五层，分别是界面组建、分析功能、应用组件、应用框架以及数据库，其中界面组件是应用UI端，其它四层都是应用服务端。

本项目以nodejs为技术平台，提供应用服务，可利用浏览器或移动客户端等多种方式构建应用界面，方便用户使用，并提供对bootstrap、jquery等常用组件的支持，其访问过程如图1-3所示：



（图 1-3 访问过程）

## 二、工程目录结构



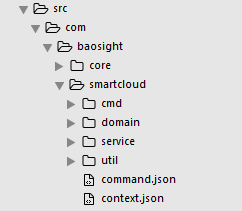
（图 2-1 总体目录结构）

如上图2-1所示：Smart Cloud系统主要具有如下目录文件：

* config目录：主要用来存放系统相关配置文件，包括系统初始化配置文件（config.json）、应用服务配置文件（server.json）以及日志配置文件（log4js.json）等。
* node\_modules目录：存放所有的项目依赖库；
* public目录：用来存放网站所有的静态资源和公共资源，包含images、javascripts、css、前端配置文件等；
* routes目录：用来存放地址访问控制文件；
* src目录：用来存放应用服务，所有的服务必须在src目录下；
* views目录：视图文件目录，用来存放模板引擎文件；
* context.json:基于bearcat框架的配置文件；
* server.js文件：项目入口文件，程序就是从这里开始运行；
* package.json文件：用来声明项目中使用的模块及相关信息，主要方便在新环境部署时，直接使用npm install进行项目的发布安装。

## 三、应用服务端

#### 3.1 目录结构



（图 3-1 应用服务端目录结构）

如上图3-1所示，smartcloud是一个应用服务：

* 为区分不同的应用服务，在src目录下应建立唯一地址（规范：公司域名+应用服务名，形如：com/baosight/employeemanage，其中com/baosight为公司的域名，smartcloud为应用服务名）；
* 在应用服务下，必须具有command.json以及context.json配置文件，分别是后台地址访问配置文件以及bearcat配置文件；
* 在应用服务下，应该具有cmd、domain以及service文件夹。cmd文件夹存放接口文件，即后台服务入口文件，domain文件夹用来存放数据模型，service文件夹用来存放业务处理文件；

#### 3.2 调用过程

应用服务端的调用过程如下图3-2所示：



（图3-2 应用服务端调用过程）

其详细过程如下所示：

1. http Server通过route请求数据，在route中匹配进行路由控制；
2. 在route中根据配置文件对url地址进行解析，解析过程如下所示：



（图3-3 路由解析）

1. route在url解析完成后，调用相应的command，处理页面请求中的逻辑；
2. command调用相应的业务服务处理业务逻辑；

4.1、command也可直接通过数据模型获取数据；

1. 业务处理通过数据模型获取数据；
2. command将服务处理返回的数据，据填充到ejs模板文件；
3. command将数据返回到route；
4. route将数据返回到http Server。

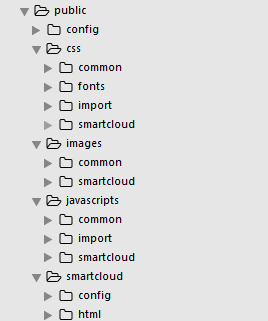
#### 3.3 对象容器

应用服务端的对象管理通过bearcat框架中的轻量级容器实现。bearcat是一个基于POJOs进行开发的应用层框架,提供了一个轻量级的容器来编写简单，可维护的[nodejs](http://nodejs.org/). Bearcat 提供了一个基础的底层来管理应用逻辑对象，使得开发者可以将精力放在应用层的逻辑编写上。核心容器包含核心，Beans，以及Context模块. 核心和Beans模块提供容器的基础部分，包含IoC容器和依赖注入. BeanFactory 是一个复杂的factory工厂模式实现. 它去除了手动去编写单例，允许你的实际程序逻辑从配置和依赖的管理中解耦。

bearcat框架地址：<https://github.com/bearcatnode/bearcat>

## 四、应用界面端

#### 4.1 目录结构



（图 4-1 应用界面端目录结构）

public文件夹下存在config、css、images、javascripts、smartcloud文件夹，其中config存放前端配置文件（包括：菜单、组件等配置文件）；css存放css样式文件（包括：common通用的自定义样式；import导入外部的样式，以及smartcloud（项目名）的自身样式）；同理images以及javascripts的结构与css结构类似；smartcloud表示一个项目名，此文件下存放两个文件夹，分别是config以及html，config 下存放了前端组件的相关配置信息，html存放了静态的html文件。

|  |
| --- |
| 项目下config中的前端配置信息如下所示：  {  "name":"智慧云平台",//项目名  "path":"smartcloud",//项目对应public下的路径  "code":"component\_smartcloudServer",//项目code 必须“component\_”+server "pCode":"component\_root",//父级目录，由于是前端组件 所以以component\_打头  "index":"user.list",//默认访问的页面  "modules":[  {"module":"user","caption":"用户管理","desc":"用户管理","js":"/javascripts/smartcloud/userController.js","pages":[  {"id":"list","caption":"用户列表","desc":"用户列表","url":"/html/user/userList.html","method":"init","components":[  {"component":"addUser","caption":"新增用户按钮","desc":"新增用户按钮","limit":false},//limit表示是否加入权限控制  {"component":"queryUser","caption":"查询用户按钮","desc":"查询用户按钮","limit":true}  ]},    ]}  ]  } |

|  |
| --- |
| 静态html页面注意点如下所示：   1. html的最顶层必须加入ng-page属性； 2. 所有的点击事件必须以如下命名：ng-click=”函数名” id=”函数名” |

#### 4.2 页面调用过程

页面调用过程如下图4-2所示：



（图4-2 页面调用过程）

其详细过程如下所示：

1. 浏览器输入url进行页面访问，如http://127.0.0.1:18080/#user.list；
2. App监听器中监听url地址变化，并对url地址进行解析，获取#后面的字符串（user.list）；
3. App实例读取AppConfig中的相关配置信息；
4. App实例利用第2步获取到的字符串与AppConfig中的配置信息pageId进行匹配，加载相匹配的templateUrl属性对应的html文件；
5. App实例显示加载的html文件；
6. App实例调用html对应的method属性的方法；
7. 方法至服务端获取数据；
8. 方法将获取到的数据填充至加载的html文件。

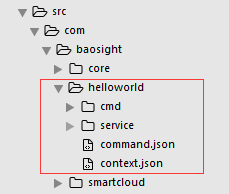
## 五、Hello world

#### 5.1 服务端

首先在config/server.json配置文件中配置新创建的服务信息。如下所示：

|  |
| --- |
| {  "name": "helloworld demo",//服务端名称  "code": "helloServer",//服务编码（注：不能与其他服务编码相同）  "path":"com/baosight/helloworld"//服务所在目录，必须在src目录下  } |

服务目录下必须有command.json和context.json文件，以及处理前端请求的cmd文件夹和进行业务处理的service文件夹，如下图所示：



其次，分别在cmd中创建helloCmd.js文件以及在service中创建helloService.js文件，其内容如下所示：

|  |
| --- |
| **HelloCmd.js**  var HelloCmd = function(){  this.helloService = null;//通过bearcat 手动注入  };  HelloCmd.prototype.hello = function(req,res){  var obj = {  "name":req.param("name")//获取传递的参数  }  //调用service 进行业务处理  this.helloService.hello(obj,function(error,result){  return res.json(error,result);//回调函数返回结果  })  }  module.exports = HelloCmd;//导出 |

|  |
| --- |
| **HelloService.js**  var HelloService = function(){  };  HelloService.prototype.hello = function(obj,callback){  callback(null,"hello "+obj.name);//回调返回结果  }  module.exports = HelloService;//导出 |

context.json和command.json配置文件内容所示：

|  |
| --- |
| **context.json**  {  "name": "helloworld",  "namespace":"helloworld",//命名空间  "beans":[  {"id": "helloCmd","func": "cmd.helloCmd","props": [{"name": "helloService","ref": "helloworld:helloService"}]},  {"id": "helloService","func": "service.helloService"}    ]  } |

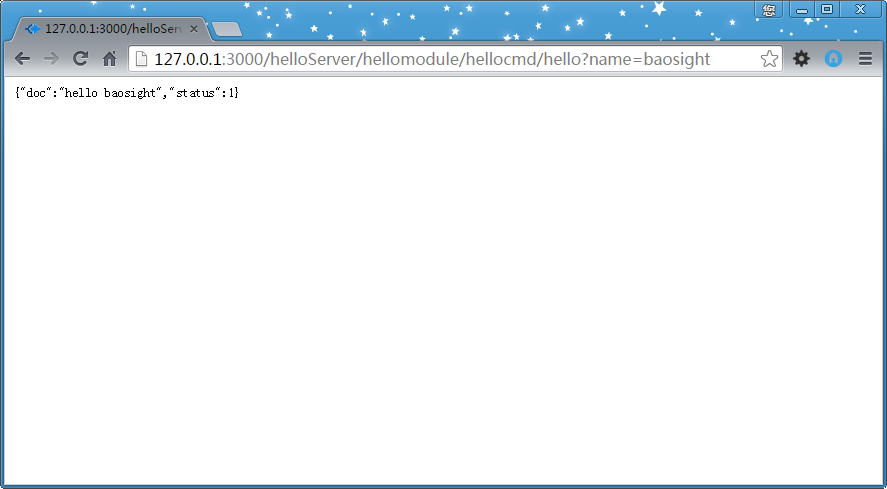
|  |
| --- |
| **command.json**  {  "modules": [  {"module":"hellomodule","caption":"hello模块","desc":"hello模块","commands":[  //cmd中的id对应context.json中配置的某个bean  {"cmd": "hellocmd","id":"helloworld:helloCmd","caption":"hello cmd","desc":"hello bean","methods":[  //对应bean中的方法  {"method":"hello","caption":"hello方法","desc":"hello方法"} ]}  ]}  ]  } |

注意：在项目根目录中的context.json文件中引用helloworld服务中的context.json 地址，如下所示：

|  |
| --- |
| "imports": [  "src/com/baosight/smartcloud/context.json",  "src/com/baosight/core/context.json",  "src/com/baosight/helloworld/context.json"  ] |

最后重启服务，访问地址：

<http://127.0.0.1:3000/helloServer/hellomodule/hellocmd/hello?name=baosight> 结果如下所示：



其中，status=1表示成功访问，doc里面的结果就是正确的返回结果。

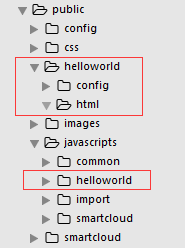
#### 5.2 界面端

首先，要定义helloworld服务对应的模板。定义方式如下所示：

在server.json中的helloworld服务的配置信息中加入indexPath配置信息，其值为渲染自定义模板的访问路径。

|  |
| --- |
| server.json中helloworld服务配置信息更改如下：  {  "name": "helloworld demo",  "code": "helloServer",  "path":"com/baosight/helloworld",  "indexPath":"hellomodule/hellocmd/index"//加载模板地址  }  其中，helloServer/hellomodule/hellocmd/index对应的方法如下所示：  HelloCmd.prototype.index = function(req,res){  return res.render("helloIndex");//helloIndex即是helloworld的ejs模板  };  模板内容如下所示：  <!DOCTYPE html>  <html>  <head>  <title></title>  </head>  <body>  <h1>hello world 自定义模板</h1>  <!-- 动态加载静态html页面的地方 -->  <div id="content"></div>  </body>  </html> |

其次，在public目录下创建对应helloworld服务的界面端，命名为helloworld，并在此目录下创建config目录以及html目录，分别存放配置文件以及静态页面文件，并在javascripts文件夹下创建对应于helloworld界面端的js文件目录，同样命名为helloworld。如下图所示：



1. 在html目录下创建页面，内容如下显示：

|  |
| --- |
| **Html/HelloList.html**  <div>  <h3>动态加载的html</h3>  <p id="msg"></p><!—内容从服务端获取 -->  </div> |

1. 在 javascripts/helloworld创建js入口文件main.js，以及helloController.js文件，内容如下：

|  |
| --- |
| **Main.js**  //由于公用的JS 引用了如下外部JS 因此必须配置如下外部js  requirejs.config({  baseUrl: '/javascripts',  paths: {  "jquery":'import/jquery-1.8.3',  "jquery.validform":'import/jquery.validform',  "hashchange":'import/jquery.ba-hashchange',  "pageslide":'import/jquery.pageslide.min',  "util":'common/util',  "datatables":'import/jquery.dataTables.min',  "datatables.bootstrap":'import/jquery.datatables.bootstrap'  },  shim: {  'hashchange': {  deps: ['jquery'],  exports: 'hashchange'  },  "jquery.validform":{  deps:['jquery']  },  "pageslide":{  deps:['jquery']  },  "datatables":{  deps:['jquery']  },  "datatables.bootstrap":{  deps:["datatables"]  }  }  });  // Start the main app logic.  requirejs(['jquery','util','hashchange'],function ($,util) {  $(window).hashchange( function(){  var hashStr = "";//页面定位  hashStr = location.hash.replace("#","");  util.parsePage(hashStr,"/helloworld/config/app.json","#content");//参数2：配置文件地址；参数3：ejs 中加载html的区域    });  $(window).hashchange();//触发事件  }); |

|  |
| --- |
| **helloController.js**  define(["common/component"],function (component) {    var init = function(){  component.get("/helloServer/hellomodule/hellocmd/hello",{"name":"baosight"},function(data){  $("#msg").text(data);//渲染html内容  })  }  return {  init:init  }  }) |

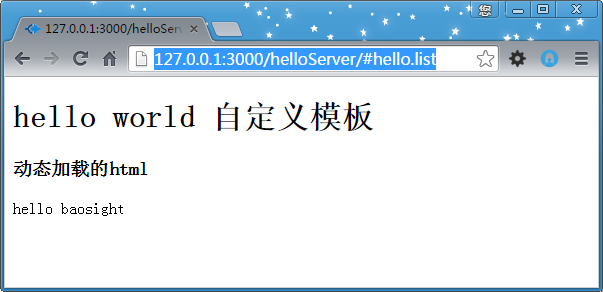
1. 在helloworld/config中创建app.json 内容如下：

|  |
| --- |
| **app.json**  {  "name":"helloworld平台",  "path":"/helloworld",  "code":"component\_helloServer",  "pCode":"component\_root",  "index":"hello.list",  "modules":[  {"module":"hello","caption":"hello module","desc":"hello mosule","js":"/javascripts/helloworld/helloController.js","pages":[  {"id":"list","caption":"hello","desc":"hello","url":"/html/helloList.html","method":"init","components":[  ]}  ]}  ]  } |

1. 最终别忘了在自定义模板中引用入口main.js文件，如下所示：

|  |
| --- |
| **helloIndex.ejs**  <!DOCTYPE html>  <html>  <head>  <title></title>  </head>  <body>  <h1>hello world 自定义模板</h1>  <div id="content"></div>  <!-- 加载入口文件 -->  <script data-main="/javascripts/helloworld/main.js" src="/javascripts/import/require.min.js"></script>  </body>  </html> |

最后通过访问查看效果，地址如下：[http://127.0.0.1:3000/helloServer/#hello.list](http://127.0.0.1:3000/helloServer/" \l "hello.list)



其中：“hello world 自定义模板”是模板中内容，“动态加载的html 和 hello baosight”是加载的html中的内容，并且“hello baosight”是从服务端获取的内容。

## 附：第三方组件

|  |  |
| --- | --- |
| 版本 | 名称 |
| 2.1.1 | require.js |
| 1.8.3 | jquery.js |
| 3.5 | jquery.ztree.all.js |
| 5.3.2 | jquery.validform.js |
| 2.0 | jqeury.pageslide.js |
| 1.10.0 | jquery.dataTables.js |
|  | jquery.dataTables.bootstrap.js |
| 1.3 | jquery.ba-hashchange.js |
| 3.0.3 | bootstrap.js |
| 4.2.0 | bootbox.js |
|  | ace.js |