Nirava概要和开发文档

1. 插件系统

基于eclipse的extension 和 extension-point的系统。

1. 实现了jsp 绘制的ui
2. 基于struts的和改写了struts的json的插件

项目介绍

Client 部分

1. 是js 实现了页面里面view 之间的交互主要实现了selectionService。
2. Rpc的实现依赖 jquery.jsonrpcclient.js插件
3. 全局的monitor的实现

Server 部分

1、nirvana-core 实现了插件平台 platform 和 workbench 定义了系统中的组件模型

模型如下

product 1:\* theme 一个产品可以多个主题

theme 1:1 desktop 一个主题 包含第一个 桌面

product 1:\* page 一个产品定义为多个page组成

page 1:\* view 一个page 包含多个view

2、nirvana-ui 用jsp实现了绘制组。

3、nirvana-web 主要就是把workbench启动起来，绑定到session中，还有一些以来具体部署服务器的部署服务。页面访问的入口

4、nirvana-workbench 启动的项目，最终生成war包

5、nirvana-plugins 这里面有三个插件，主要就是为了演示项目建立了三个插件

nirvana-inspinia 具体项目，功能很简单

**5.1监控图数据获取**

com.wxxr.nirvana.inspinia.ChartDemoService 这个类提供前台数据，依赖struts-json插件实现的，作为一个action的扩展点具体配置片段如下

<extension point="com.wxxr.nirvana.platform.actions" id="pages.actions">

<action id="chartData"

class="com.wxxr.nirvana.inspinia.ChartDemoService">

</action>

</extension>

这样这个action就注册进来了

注：每一个extention都有唯一id来标识它，id的规则是插件id 加上 extension id

例如nirvana-inspinia插件的插件id是com.wxxr.nirvana.test 如下

片段

<plugin id="com.wxxr.nirvana.test" name="nirvana"

version="1.0.0" provider-name="wxxr.com.cn"

>

那么唯一的id就是 com.wxxr.nirvana.test. chartData.

客户端调用：

$.invorkRpc = function( method, params, rpcId, successCb, errorCb, isMonitor)；

method --- 调用的方法名字 就是java类ChartDemoService 的getDataList方法名字

params --- 方法的参数，参照例子

rpcId --- 就是刚刚提到的actionid：com.wxxr.nirvana.test. chartData

successCb --- 成功回调函数

errorCb ---- 失败回调函数

isMonitor --- 是否需要monitor

**5.2** com.wxxr.nirvana.inspinia.PageLayoutRender 这个类实现了布局，简单实现bootstrap的网格布局

<extension

id="nirvana"

point="com.wxxr.nirvana.platform.pagelayouts">

<pageLayout

id="nirvanapagelayout2"

name="layout"

description="layoutdesc"

render="com.wxxr.nirvana.test.pageLayoutRender">

</pageLayout>

</extension>

以上是layout的扩展的片段 render 是有哪个render来绘制这个layout 这个com.wxxr.nirvana.test.pageLayoutRender id就是唯一的id (这个就是用代码来绘制layout，如果layout 的 有uri属性指向jsp可以直接用jsp绘制)

<extension id="renders" point="com.wxxr.nirvana.platform.uirenders">

<uirender id="pageLayoutRender"

description="nirvanapagelayout"

uri="menu.jsp"

class="com.wxxr.nirvana.inspinia.PageLayoutRender">

</uirender>

</extension>

**5.3 page定义**

<extension point="com.wxxr.nirvana.platform.pages" id="pages.niravanaPage">

<page

id="niravanaPage"

label="tiger"

name="tiger"

layout="com.wxxr.nirvana.style.nirvanapagelayout"

>

<view ref="com.wxxr.nirvana.test.chart1" attachment="headerone"></view>

<view ref="com.wxxr.nirvana.test.chart2" attachment="headertwo"></view>

<view ref="com.wxxr.nirvana.test.chart3" attachment="headerthree"></view>

</page>

</extension>

指定 layout 的id是com.wxxr.nirvana.style.nirvanapagelayout 这个是在nirvana-inspinia-style这个插件中定义的layout所以要依赖nirvana-inspinia-style这个项目依赖定义

<requires>com.wxxr.nirvana.style</requires>

View的指定： 包含了三个view ，view中有个attachment 属性这个属性指定了view要附属在哪里，看看具体的layout使用在nirvana-inspinia-style项目中的pageContainer.jsp页面

Html片段如下

<div class="col-sm-3"><!-- view1wrap -->

<nirvana:ui name="view" anchor="headerone"/>

</div><!-- end view1wrap -->

<div class="col-sm-6"><!--view2wrap-->

<nirvana:ui name="view" anchor="headertwo"/>

</div><!-- end view2wrap -->

<div class="col-sm-3"><!--view3wrap-->

<nirvana:ui name="view" anchor="headerthree"/>

</div>

如果要动态布局最好用render来绘制比较灵活。

**5.4 产品定义**

<extension id="nirvana" point="com.wxxr.nirvana.platform.products">

<product

name="nirvana"

description="nirvana"

themeref="com.wxxr.nirvana.style.nirvanaStyle\_theme"

id="nirvana"

navRender="com.wxxr.nirvana.style.nirvanapagelayout"

>

<page ref="com.wxxr.nirvana.test.niravanaPage" index="1" defaulPage="true" />

<page ref="com.wxxr.nirvana.test.lionPage" index="2" />

</product>

Themeref 指定用哪个theme

navRender 导航的render

page标签 引用了哪个page。

**5.5 PageAction**

对于需要对页面的重绘，比如动态增加个view

com.wxxr.nirvana.inspinia.SomethingAction 这个类就是动态添加了view的例子

nirvana-inspinia-style – 样式项目

定义了引用哪些资源，定义了theme用了那个桌面

nirvana-resources -- 定义了一些资源大部分是js css资源，样式引用的资源差不多都在这个插件中