EC2P代码开发手册

1. UI前端代码
   1. URL及VM规范
      1. 通用规则说明

**通用规则说明：**

     1：三种页面类型对应URL标记：

          A>**Page**   独立完整的HTML页面，包含完整HTML标签，如Head，Body等。

          B>**AjaxPage**  部分HTML标签代码的页面，如一个Div。

          C>**JsonPage** 要求返回Json格式的操作结果。

     2：VM的保存路径，总体上要求与URL一一对应。另外，建议在开发功能时，尽量为VM提供个性化支持的余地，以便为同功能里的不同数据提供差异化的VM，以供显示。

* + 1. 前台URL规范

**前台URL规范**

     1：域名/模块名/\*\*功能名+页面类型(不含最后单词Page).html

     2：页面类型为Page时，可省略这个后缀。

     3：示例如下：

          3.1：普通登录页面：/common/login.html（正确） ， /common/loginPage.html（错误）。

          3.2：被其它页面内嵌的异步登录代码片断：/common/loginAjax.html（正确），/common/loginAjaxPage.html（错误）。

          3.3：被其它页面通过Json请求的登录配置：/common/loginJson.html（正确），/common/loginJsonPage.html（错误）。

* + 1. 后台URL规范

**后台URL规范**

     1：域名/**admin/**模块名/\*\*功能名+页面类型(不含最后单词Page).html

     2：页面类型为Page时，可省略这个后缀。

     3：示例参考前台URL规范中的示例：

     4：为了支撑操作成功后关闭当前页面和刷新指定页面，建议在URL上添加以下几个参数，Query里已集成了这些属性。

          【dwzId】（DWZ中打开面板的ID，主要用于UI端标记操作来源Tab或Dialog，以便后续操作完成时，通知或刷新源页面。如果是lookup，表示打开的是弹出选择页面。 ）.addQueryData("dwzId","newdwzId")

          【navTabId】（操作成功后,要重新加载的Tab或者dialog的 ID号。 [把指定navTab页面标记为需要“重新载入”。注意navTabId不能是当前navTab页面的. 否则Tab重新载入当前navTab页面）.addQueryData("navTabId","newnavTabId")

          【callbackType】（EnumDwzJsonCallbackType .“forward”在当前面板中跳转到【forwardUrl】，“closeCurrent”关闭当前面板Tab或者dialog（默认）   ）.addQueryData("callbackType","newcallbackType")

          【forwardUrl】（ callbackType=“forward”时在当前面板中跳转到forwardUrl ）。.addQueryData("forwardUrl","newforwardUrl")

如： $appServer.get('/admin/dynamicModel/bizData/addAjax.html').addQueryData("dwzId","dwz\_tab\_dynamicModel\_bizData").addQueryData("navTabId","newnavTabId").addQueryData("callbackType","newcallbackType").addQueryData("forwardUrl","newforwardUrl")

          具体实现原理：

               以上几个参数通过Query对象传入Action，Action处理完成，返回DwzJson对象时，通用dwzJson.setQuery(query)方法，将以上几个参数设置给DwzJson对象，Action返回Json数据到前端，前端Dwz解析并按照这几个参数的设置，进行UI上的处理。

**常用功能名：**

     1：列表搜索页面：【**list**】，对应三种页面类型：/admin/bizCode/**listAjax**【**list**|**listJson**】.html

     2：进入添加：【**add**】

     3：保存添加：【**save**】

     4：进入修改：【**edit**】

     5：保存修改：【**update**】

     6：删除：【**del**】

     7：快速选择：【**lookup**】（一般，可共用list搜索列表页面，通过特殊DwzId（lookup）来标记。）

     8：查看详情：【**view**】

**关于Query中的DwzId说明：**

     特殊ID：lookup 表示打开的VM是为弹出选择框使用的，与普通搜索列表页面有一些差异。

* + 1. Layout规范

**Layout规范：**

     views.layout.模块名

          blank.vm  空白模板，相当于不使用Layout，后台大部分页面可使用此模板。

          bodyLite.vm 空白Body模板，拥有完整HTML标签，只是不包含Body里面的代码。

          default.vm  前台默认，包含页头页脚等公共页面模块

          dwzAdminDefault.vm  后台DWZ前端UI框架下的完整模板。使用此模板的页面，只需要提供右侧工作区的代码就可以。

          dwzAdminLite.vm   后台DWZ前端UI框架下精简的模板，此模板不包含页头，页脚，侧栏，与BodyLite.vm相同，只是引入了Dwz相关的JS和CSS。

* + 1. contain  复用组件规范

**复用组件规范：**

     前台公用组件： views.screen.模块名.contain  （后台可使用前台组件，但是，前台不可使用后台组件，所以，前后台都要用的组件，且这个组件不依赖后台业务或VM时，请放在前台组件下。）

     后台公用组件： views.screen.**admin.**模块名.contain

* + 1. 模块后台侧栏菜单

**如何在首页侧栏和头部增加菜单。**

*目前侧栏和头部都是在Common模块里页面，暂时无法支持动态增加，需要一个良好的设计，能动态增加菜单。*

          目录先支持在后台首页URL上带参数（module）来打开各业务模块的左栏，如：/admin/{module}index.html。

          各业务模块在各自的VM或URL上提供类似这样的URL(/admin/**模块名**/contain/leftSidebar.xhtml) 或VM（views/screen/admin/**模块名/**contain/leftSidebar.vm），这个页面就是左侧栏目AjaxPage。

     另外，请同时提供对应的首页VM到：/admin/模块名/index.vm

* + 1. 模块引入JS的Idemand动态增加

正在设计中。

* + 1. JavaScript开发规范

各功能模块，开发的JS必须归类，按照指定规则进行整理，杜绝直接定义全局的JS变量。

因为Dwz框架下，所有页面都是在一个页面中打开，任何一个JS代码，都是全局的，如随意编写，很可能干扰其它模块功能。另外，在页面中，杜绝使用指定的ID，原因同上，如不控制，很可能出现一个页面上，不同Tab子页中，出现相同ID的控件，导致页面控制出现异常。

如果一定要使用ID，建议使用指定ID+随机后缀。对应的JS方法增加参数，将ID后缀传入，以便精准操作。（可参考Common工程里的：statics/common/javaScript/ace.js）

JS规范如下：

/\*\*

 \* 请注意定义规范，ec2p为全局JS对象，本产品所有JS必须在这个对象内，不得擅自创建顶级对象。其它模块里不用写下面这行代码。

 \*/

**var** ec2p = {};

/\*\*

 \* 各产品模块自己再定义成ec2p对象的子对象，如果某一产品模块下功能很多，建议再往下分子对象。

 \*/

ec2p.common = {};

/\*\*

 \* 然后建议在模块下，分两块，分别为函数方法（以fn为方法名结尾），和变量（注意，方法级的临时变量不要放到这里，这里只放模块间，或方法间要互用的变量。）。

 \*/

ec2p.common.fns = {

};

/\*\*

 \* 可交互的变量。**TODO** 需要提供一个机制，能适时的清理缓存中的变量，否则，可能会不停的增加变量缓存，最终导致内存消耗过量。

 \*/

ec2p.common.vars = {

};

* + 1. 通用的VM代码

将枚举组装成下拉选择框#enumSelectEl

##将枚举组装成下拉选择框, 使用方式如：#enumSelectEl( $enumSex,"sexInput", "sexInputId","M","请选择性别","","sexSelectClass" )

## $enumObj枚举对象,

## $elName为下拉框的Name,默认为:enumSelect.

## $elId 为下拉框的Id ,为空时，不使用Id

## $curValue为当前值,则默认选中这项,且,如果页面中有类目组件,自动关联类目可选数据.

## $waitText ，初始时类似“请选择”方案, 如果需要这个选项,请传入要显示的文字,为Null或空串，则不显示这个值选项.

## $readOnly表示是否是只读,如果是只读的话,下拉中就只显示当前值.非空白表示只读,null或空白表示非只读

## $class Select控件的附加Class样式名

DWZ后台分页条件

##生成DWZ页面上的分页条组件，

## $pageObj 分页对象,就是DwzPage对象

## $pageType 为窗口dialog，导航Tab：navTab（默认）.

#macro (dwzPageBar $pageObj $pageType)

过滤HTML标签，输出纯文字内容

##注意 ,此宏中的参数$htmlStr在传入时,不要用引号或双引号包裹

##过滤HTML标签，输出纯文字内容，如果需要截取字数，请传递第二个参数$count。

#macro ( htmlText $htmlStr $count)

输出HMTL代码串

#macro ( direct $direct-output )$!direct-output#end

##注意 ,此宏中的参数$htmlStr在传入时,不要用引号或双引号包裹

##转回HMTL输出，通常用在HTML内容显示。

#macro ( html $direct-output )$!direct-output#end

* 1. 前台UI端
  2. 后台UI端-DWZ
     1. VM开发

后台UI已由DWZ框架搭建，加上Melody的VM开发方案，则开发代码方式如下：

1：后台页面必须使用后台DWZ的layout，后台页面，一般会用到三种layout,如下：

      一： views/layout/common/dwzAdminDefault.vm：后台新窗口中打开的完整页面，有页头页脚侧栏。其页面的HTML代码结构是完整的，有Head，有Body。对应的页面类型为Page

      二： views/layout/common/dwzAdminLite.vm：这个主要用于后台页面中使用Iframe或单独的页面使用，其页面是一个独立的完整的HTML页面，有Head，Body等标签，但是其页面上不需要显示页头页脚侧栏等。这个一般很少用到。对应的页面类型为Page

      三： 不使用Layout：这个是后台功能页面中，最常用的方式，页面只是片断的代码，如只有一个Div。它将通过DWZ的前端JS，将HTML片断代码嵌入到后台框架上显示。对应页面类型为AjaxPage，或JsonPage

2： 如何在右侧工作区打开页面：

     一：对应页面类型AjaxPage的：在a标签上，增加这两个属性：target="navTab" rel="dwz\_tab\_role"

            其中Target表示URL在哪里打开，navTab表示在右侧工作区打开。

            rel是这个页面对应工作区Tab的ID，以便后面操作这个页面，如刷新，关闭这个Tab。 这些标签的详细介绍，请看DWZ的操作手册。

     二：对应页面类型Page的：如果URL不是本站，则可直接和第一条一样处理。DWZ将会自动以Iframe的方式嵌入这个页面到工作区。  如果URL是本站的，请务必在第一条之外，再加一个标签： external="true" 。这样的话，DWZ会强制使用Iframe的方式将网页嵌入在工作区。

     三： 对应页面类型为JsonPage的，无法在工作区打开，这种类型，一般只用在Ajax操作。

3： 如何通过Ajax提交表单，且操作成功后，关闭当前页面，并刷新列表页面。

示例：

     VM：

     Java：

lookUp选择框

Dwz的原来这功能不太方便，框架已经对这块代码进行了修改，在Common工程下的：ec2p.dwz.database.js文件里，扩展了Input一个属性lookupName.此属性用来标记选择返回时，这个Input框加载的对应值的Key。这样就能支持一个选择框共用给不同的页面使用，这些页面的Input的Name可以不相同。

示例如下：

  1：表单页面：

            #springBind("dynamicModelBizData.categoryCode")

            <dl>

                <dt>类型：</dt>

                <dd>

                      <input type="hidden"  lookupName="${groupId}.code" name="${status.expression}" value="$!{status.value}"/>

                    <input  type="text"

                          lookupPk="${groupId}.code" lookupName="${groupId}.shortName" suggestfields="${status.expression}\_txt,${status.expression}" value="$!{status.value}"

                          name="${status.expression}\_txt" title="如不需要修改，请保持空白"  class="textInput" maxlength="64" size="30"  readonly  />

                    <a class="**btnLook**" href="$!{appServer.get('/admin/dynamicModel/category/listAjax.html').addQueryData("dwzId","lookup")}" warn="请选择类型" lookupgroup="${groupId}">选择</a>

                </dd>

            </dl>

       说明：lookupName 属性是标记这个Input的值对应选择传回的Key，注意，这个属性可以加groupId做前缀，以区分同名的组。

  2：对应的LookUp页面，可以是一个完整的搜索列表页面，在每行数据后，加一个选择按钮，按钮链接格式如下就可以了：

<a href="javascript:jQuery.bringBack({code:'$!rs.code', shortName:'$!rs.shortName'})" warn="请选择一个数据" title="查找带回" class="btnSelect">选择</a>

注意传回的数据为Json格式，Key就对应着前文中的LookUpName属性。

如不清楚的，可参数views.screen.admin.dynamicModel.category下的VM页面：domainForm.vm和lookupAjax.vm

1. 分层代码
   1. DoMain层规范

1：所有的DoMain数据库对象都要继承BaseObject类。因为BaseObject提供好了通用的ToString()、Equals()、HashCode()三个方法。子类不再需要重写，除非有特殊要求。

2：对象命名：不需要添加表名（ec2p\_模块名\_表名）前缀ec2p，即只要“模块名表名”

3：必须为每一个字段都写好字段说明，如果某字段有对应枚举，请一定在注释中写明枚举类。

12：注意，对象中以下三种类型的字段，必须提供多种Get和Set扩展方法，以便能直观的使用属性，而不产生歧义：

     12.1：金钱：Long get\*\*\*Fen(), set\*\*\*Fen(Long fen), Double get\*\*\*Yuan(), set\*\*\*Yuan(Double yuan), 原属性默认就是Fen处理。可通过CommonUtil提供的静态方法进行转换。

     12.2：比例：比例有三个概念，一个是小数值real，一个是百分值percent，一个是数据库中值DbRate，对应比例关系为：1：100：10000。原属性默认就是real值来处理，对象的这个字段就要提供相应Get和Set的6个方法。可通过CommonUtil提供的静态方法进行转换。

     12.3：时间：

* 1. Query

1:后端DWZ的查询Query统一继承DwzBaseQuery类，这个类针对DWZ的分页和排序，以及操作完成后刷新或关闭或跳转等特性提供了支持。

2：Query字段，请按照以下要求进行扩展。

     Query对象上，不使用数据字段名做属性，而是统一按照查询约束进行扩展查询字段，如有一个Name字段，对应查询条件可能就有：【等于】，【不等于】，【左模糊】，【右模糊】，【全模糊】，那对应就在Query上扩展成6个查询字段。分别为：name\_eq, name\_ne, name\_ew, name\_bw, name\_cn

     主要条件如下：

     \* <pre>

     \* "eq"=等于    , "ne"!=不等

     \* "lt"<小于    , "le"<=小于等于

     \* "gt">大于    , "ge">=大于等于

     \* "bw" 开始于 （右模糊） , "bn" 不开始于

     \* "in" 属于    , "ni" 不属于

     \* "ew" 结束于（左模糊）  , "en" 不结束于

     \* "cn" 包含（全模糊）    , "nc" 不包含

     \* </pre>

* 1. Action层规范

1：Action类要点说明：

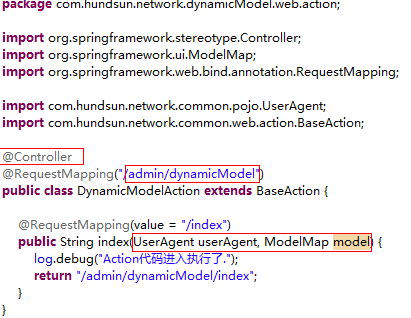
     》1：注解为Action容器，@Controller

     》2：类头注解URL路径，如果是后台类，可直接在类头上注解后台路径，否则注解为根目录。（可选）

     》3：方法头注解URL，相对类头URL的地址，不用带网页后缀.html。

     》4：方法体参数注解，请先参考SpringMVC的注解说明。本框架规范要求每个方法体必须要有以下两个注入参数：UserAgent userAgent, ModelMap model ，其中UserAgent是用户登录信息，将完整的传递到Service上进行业务逻辑处理。

     》5：注意，后台各模块首页URL已经被框架集成了，所以，建议不要在Action上注解类似这样的URL：/admin/业务模块名/index.html。如果你对业务模块后台首页有特殊要求，请直接提供相应的VM到对应这个URL的目录下就可以。



* + 1. Page类型注解
    2. 活动 2
  1. Service层规范
     1. 任务 1
     2. 任务 2
  2. Dao层规范
     1. 任务 1
     2. 任务 2
  3. Mapper层规范

正在考虑是否需要这一层。

* 1. SqlMap层规范
  2. Pojo
  3. Utils
  4. Enums
     1. 是否（yes/no）EnumYesNo.java

*YES*(**new** String("yes"), "是"), *NO*(**new** String("no"), "否");

* + 1. 是否（1/0）EnumYesNoShort.java

*YES*(1, "是"), *NO*(0, "否");

* + 1. EnumSex.java

*man*(**new** String("m"), "男"), *woman*(**new** String("w"), "女"), *unknown*("u", "未知");

* + 1. EnumSexCalled.java

*SIR*(**new** String("m"),"先生"), *LADY*(**new** String("w"),"女士"), *Unknown*("u","先生或女士") ;

* + 1. EnumEnableStatus.java

*ENABLED*(**new** Integer("1"), "有效")// 有效

    , *WAIT*(**new** Integer("0"), "待查")// 待查

    , *DISABLED*(**new** Integer("-1"), "禁用")// 禁用

    ;

* + 1. EnumServiceError.java

ServiceResult 返回时,错误号枚举, Service方法中,对于操作结果中的错误信息通用枚举

* + 1. EnumErrorType.java

*LOGIN*("login", "登录页面"), // 登录页面(login),

*PERMISSION*("permission", "无权限页面"), // 无权限页面(permission),

*ERROR*("error", "错误异常页面")// 错误异常页面(error)

* + 1. EnumPageType.java

*PAGE*("Page", "普通页面"), // 普通页面 (Page)(默认)

*AJAXPAGE*("AjaxPage", "AJAX页面"), // AJAX页面(AjaxPage)

*JSONPAGE*("JsonPage", "Json页面"),// Json页面(JsonPage)

* + 1. EnumDwzJsonCallbackType.java

    /\*\* 不指定,默认. \*/

*DEFAULT*("", "默认"),

    /\*\* forward在当前面板中跳转到forwardUrl \*/

*FORWARD*("forward", "在当前面板中跳转到forwardUrl"),

    /\*\* 关闭当前面板或者dialog \*/

*CLOSECURRENT*("closeCurrent", "关闭当前面板或者dialog");

* + 1. EnumDwzJsonStatusType.java

    /\*\* 表示操作成功, 做页面跳转等操作. \*/

*OK*("200", "操作成功"),

    /\*\* 表示操作失败, 提示错误原因 \*/

*ERROR*("300", "操作失败"),

    /\*\* 表示session超时，下次点击时跳转到DWZ.loginUrl \*/

*TIMEOUT*("301", "登录已过期");

1. 表结构规范
   1. 表名命名

表名结构说明：

     ec2p\_模块名\_表名

          其中，【模块名】建议精简字母，要求在6个字母内，最好是简写到3个字母内。

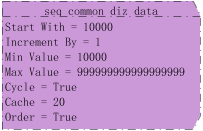
* 1. 序列号字义规范

     SEQ\_表名

          其中表名规范请参阅【表名命名】规范。

  序列号统一从10000开始，循环，最大值为20个9。

例如：



* 1. 字段定义

ID：Number(20), 且从10000开始

金钱：以分为基础单位保存。最大19位数字，Number(19,0)

比例：以原始数据扩大10000倍的整数保存，Number(10,0)

1. 其它基础代码规范
   1. Error
   2. Page

1：后台DWZ的UI，请使用DwzPage来返回查询结果，这个类对DWZ结构进行了支持。

* 1. UserAgent
  2. 常量类
  3. 缓存类
  4. Json
  5. Ajax

1. 产品模块基础配置

主要配置在src/main/resources目录下。

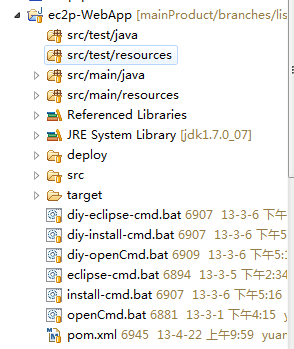
1：

* 1. Eclispe设置优化

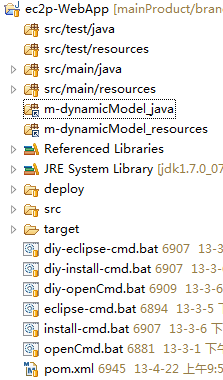
1：建议安排Maven插件。

2：在WebApp工程上，源码引用的方式，引入模块工程的Java和Resources目录做源码。这样的话，既可以像普通工程一样开发，又可将代码按照业务模块进行划分。（注意，如果重新生成Eclipse工程时，引入的源码配置会丢失，请重新配置）

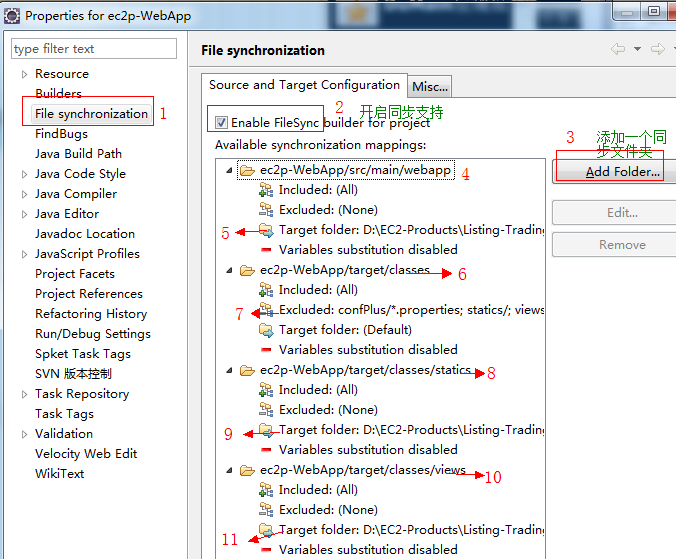
  如原工程结构如下，

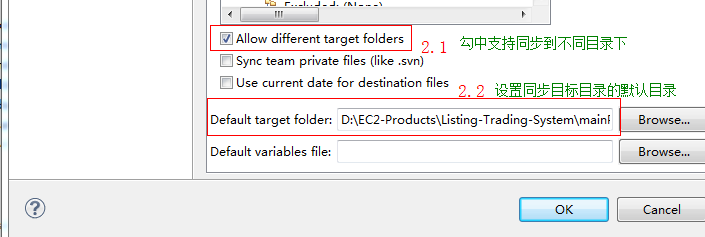


  引入业务模块工程源码后如下：



3：引入其它工程源码后，为了便于调试，请添加同步插件（FileSync，在SVN上已经提供了，https://cm07.hundsun.com/integ/Listing-Trading-System/mainProduct/branches/listing-2.0-dev/Bin/Eclipse常用插件/FileSync），并且按照以下方式进行配置：





     安装好插件后，在工程上点击右键--》【属性】，打开工程属性设置页面。

     1：在左侧点击菜单：【File synchronization】，打开天窗说亮话同步插件设置页面。如图中1

     2：在右侧启用插件。如图中2

       2.1：在右侧下方勾中：【Allow different target folders】，表示支持同步到不同的目标目录下。如图中2.1

       2.2：在右下方选择默认的同步目标目录。就是WebApp运行的WebRoot\WEB-INF\classes目录(\Maven-Sources\ec2p-WebApp\target\ec2p-WebApp-2.0.0.0\WEB-INF\classes)。如图中2.2

     3：点击右侧按钮【Add Folder...】，添加四个要同步的原始目录，分别如图中的：4，6，8，10，并分别对这四个目录进行个性配置，分别对应5，7，9，11. 详细说明如下：

       3.1：源目录：  \Maven-Sources\ec2p-WebApp\src\main\webapp 。如图4

             目标目录（Target Folder）：\Maven-Sources\ec2p-WebApp\target\ec2p-WebApp-2.0.0.0\    如图5

     3.2： 源目录：\Maven-Sources\ec2p-WebApp\target\classes\    如图6

             目标目录（Target Folder）:默认，不用设置

             不包含（Exclude）： confPlus/webApp/m-log4j.properties, confPlus/webApp/m-mail.properties, confPlus/webApp/m-server.properties,statics/, views/     如图7

     3.3： 源目录：\Maven-Sources\ec2p-WebApp\target\classes\statics\    如图8

             目标目录（Target Folder）:   \Maven-Sources\ec2p-WebApp\target\ec2p-WebApp-2.0.0.0\statics     如图9

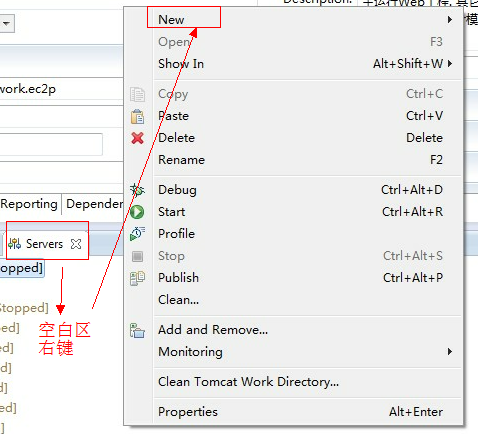
     3.4： 源目录：\Maven-Sources\ec2p-WebApp\target\classes\views\    如图10

             目标目录（Target Folder）:   \Maven-Sources\ec2p-WebApp\target\ec2p-WebApp-2.0.0.0\WEB-INF\views   如图11

     4：好了，经过以上配置，就可以同步了，一般在保存时，会自动进行同步，也可手动同步，在工程上点右键，再点击一下graphic就可以了

     5：在Eclipse中调试代码：

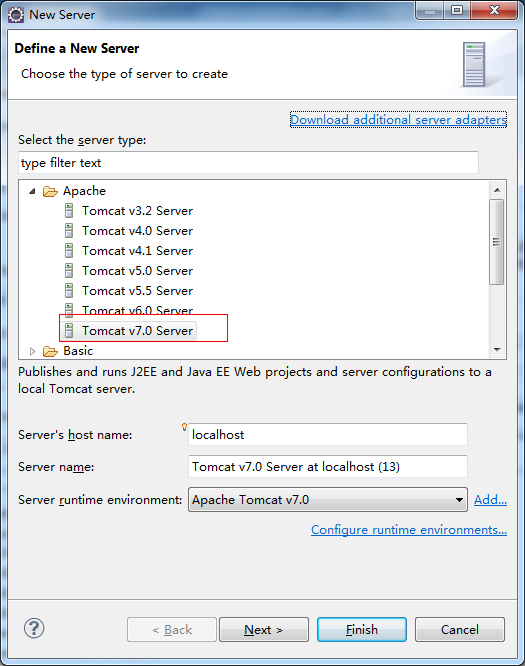
          5.1：在Eclipse右下区打开Servers工作台，在空白区右键，再选中【New】



          5.2：再选择Server：

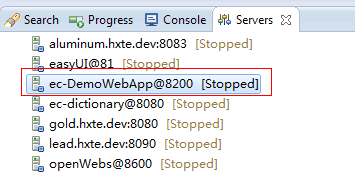
graphic

          5.3：打开服务器创建窗口，如下：

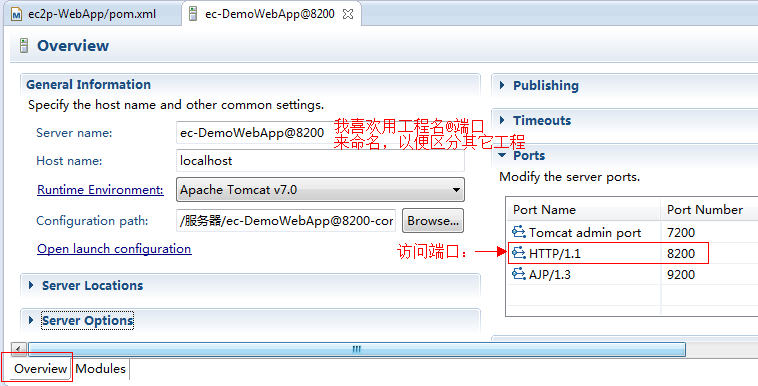


          点击【Finish】保存此服务器。

          5.4：到右下角工作台中找到刚才添加的服务器，以我本机为例（我修改了服务器名称），双击刚刚添加的服务器，打开服务器设置页面。

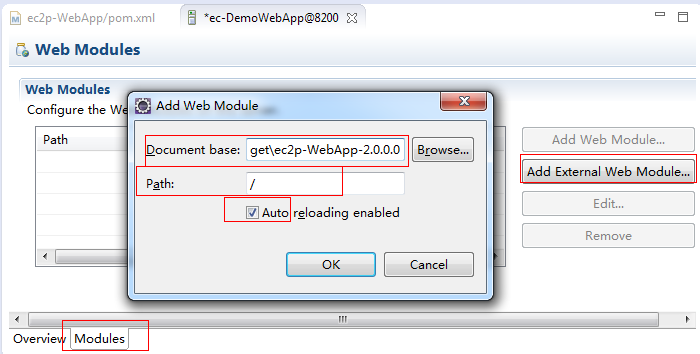


          上图为服务器列表，下图为双击打开服务器设置页面：



          在设置页面，可设置Server Name，我一般喜欢使用工程名@端口号来命名，以便区分多个Web项目。也可设置Web应用的端口号。

          5.5：在设置页面下方，点击【Modules】Tab页打开Web应用设置页面：



          Document Base设置为发布目录：\Maven-Sources\ec2p-WebApp\target\ec2p-WebApp-2.0.0.0

          Path：设置根目录\

          设置好后，点击OK，再保存服务器设置就可以了。

注意：

     如果同步后，启动Tomcat出错，请手动将WebRoot的Libs中，引入源码的依赖工程的Jar包删除。