**Eclipse Study Notes**

@author Count Monte Cristo

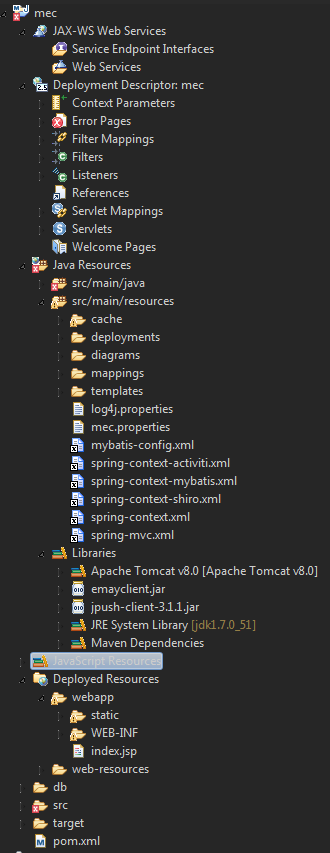
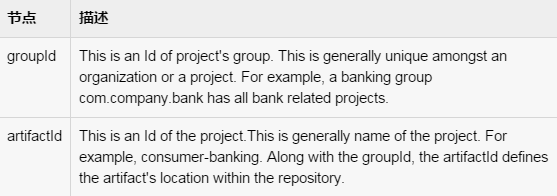
@create D20150616

@version ~~D20150715~~ ~~D20151024~~ D20151106

@description 此处介绍关于Eclipse的重要知识

@motto VICTORY LOVES PREPARATION

@criteria 凯旋基诺

1. **Eclipse教程**
   1. @Reference <http://www.w3cschool.cc/eclipse/eclipse-tutorial.html>
2. **含义**
   1. Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台，是 Java 的集成开发环境（IDE），同时Eclipse 也可以作为其他开发语言的集成开发环境，如C，C++，PHP，和 Ruby 等。
3. **Eclipse添加书签**
   1. <http://www.w3cschool.cc/eclipse/eclipse-add-bookmarks.html> @Reference；
   2. Eclipse 中可以在编辑器的任意一行添加书签，可以使用书签作为提示信息，或者使用书签快速定位到文件中的指定的行；
   3. 可以在Bookmarks视图中管理书签。
4. **Eclipse任务管理**
   1. <http://www.w3cschool.cc/eclipse/eclipse-task-management.html> @Reference
   2. 在Eclipse中用TODO标签管理任务，利用这个功能可以方便地将项目中一些需要处理的任务记录下来；
   3. 可以在Tasks视图中管理任务。
5. **Eclipse快捷键** 
   1. Eclipse 的很多操作都提供了快捷键功能，我们可以通过键盘就能很好的控制 Eclipse 各个功能；
   2. <http://www.w3cschool.cc/eclipse/eclipse-shortcuts.html> @Reference。
6. **Eclipse Maven Webapp项目目录结构剖析**
   1. **实例视图**
   2. **具体剖析**
      1. mec
         1. 项目名称
      2. JAX-WS Web Services
         1. @百度百科(JAX-WS)
      3. Deployment Descriptor(配置描述)
         1. 根据web.xml的内容进行生成，主要生成web.xml里面的filter,listener,servlet等信息，方便查找工程的配置。
      4. Java Resources
         1. src/main/java
            1. 存放Java代码
         2. src/main/resources
            1. 存放配置信息等
         3. libraries
            1. 存放项目依赖的jar包
      5. JavaScript Resources
         1. 存放JS资源
      6. Deployed Resources
         1. 已经部署到Web服务器上的资源
         2. 其中jar包已经通过Eclipse中的Deployment Assembly装配到Tomcat
      7. target
         1. target是有存放项目构建（通过Maven）后的文件和目录，jar包、war包、编译的class文件等。
      8. pom.xml
         1. POM 代表项目对象模型。它是工作在Maven的基本单位。这是一个XML文件。它始终驻留在该项目 pom.xml 基本目录。
         2. POM 包含的项目使用Maven来构建该项目和各种配置的详细信息。
         3. POM 也包含了目标和插件。在执行任务或目标时，Maven 会在当前目录中的POM。它读取POM，得到所需要的配置信息，然后执行的目标。
         4. goropId、artifactId介绍
7. **Eclipse视图** @version D20151024
   1. **Package Explorer：包视图**
      1. Eclipse 删除的内容是不进回收站的，可以通过Eclipse 找回。
         1. Restore from local history。
      2. 在包视图下可以排除源代码。
         1. build path exclude == 除源代码;
         2. build path include == 恢复被排除的源代码。
   2. **Navigator：导航视图**
      1. 导航视图，显示文件的样式，跟windows资源管理器的文件结构是一样的。
   3. **Tasks：任务视图**
      1. 在任务视图下，可以对代码做标记,//TODO标记。
      2. 自定义标记，编辑标记，移除标记。
         1. Window 🡪 Preferences 🡪 java 🡪 compiler 🡪 Task tag。
      3. 在自定义时可以添加优先级 Priority。
      4. 标记的内容配置。
         1. 鼠标点击右边的三角形 🡪 configure contents;
         2. 1 点击Properties 🡪 Java Compiler 🡪 ask Togs 🡪 选中单选框然后在new一个标签输入名字之后在选下面的级别然后点击OK;
         3. 然后在打开Task视图然后点击右上角的一个往下的三角形点击之后在选择Configure Contents点击打开，打开之后在点击new然后在选中Rename更改名字完成之后在到类里面注释。
         4. Tasks的好处比如可以注释作者名字标记时间，关闭窗口后可以打开Tasks视图点击你设置的Tasks标签名字点击就可以进到你注释的那行代码去了。
   4. **Bookmark:书签视图**
      1. 在书签视图下可以给代码做上书签便于查找，比如我们看一本书看到100页的时候我们用书签标记下，同样的道理。
      2. 添加书签,移除书签。鼠标点击编辑区的在侧边 Add Bookmark Remove Bookmark。
      3. 为什么要使用bookmark
         1. 写代码一般不是从上往下写，经常在几个模块之间变换的写，你可能使用搜索功能ctrl+f，ctrl+c，这样查找位置将会变的非常痛苦。

因为重要的位置一般就那么几个，如定义部分，核心函数部分，那么不妨使用bookmark插入bookmark

* + - 1. 把光标放在重要代码位置的最左边一栏，右击 add Bookmark，设置bookmark名称。
      2. 显示bookmark视图
      3. 点击eclipse菜单的window-》show view-》Bookmarks 将会显示bookmark视图。
      4. 定位bookmark
      5. 双击bookmark里面的 名称，就能定位到那里去了。
  1. **Outline:提纲摘要视图**

1. 可以查看类的结构。

 按字母排序。

 隐藏字段。

 隐藏用静态修饰的。

 显示用public 修饰的。

**用途:** 每个编辑器和 Outline 视图间有种特殊的关系。当编辑器打开时，Outline 视图会被关联到编辑器，如果编辑器为 Outline 视图提供了一个模型，该模型就会在 Outline 视图中显示，无论编辑器是否处于活动状态，Outline 视图可以用来浏览内容，或者是在更高抽象层上与编辑器的内容进行交互。

例如，当我们打开在编辑器中打开一个 java 类文件，Outline 视图中就会显示类中的成员变量，函数等。当在 Outline 视图中选中某个元素时，该元素在 Java 编辑中所对应的模型元素也将被高亮选中

同时，随着当前编辑器的变化，Outline 视图中的工具栏与上下文菜单将随之发生变化，以便于对编辑器内容进行操作。当编辑器中的数据太多而无法在有限的屏幕范围内显示时，或者当编辑器的内容为非 结构化时 Outline 视图能够有效地提供一个结构化的模型，并以概念角度来帮助用户操作定位、编辑器中内容元素。

其中：红色方块 代表 private

黄色菱形 代表 protected

绿色圆形 代表 public

实心 代表 函数

空心 代表 变量

此外：

向上的小三角就表示这是个函数，并且会抛出异常。

类似表型的圆圈就表示该变量或函数是synchronized（线程同步）。

一个大C，表示一个类

带一个放音键似的三角，那就代表这个类是存在main函数的类

* 1. **Problems :问题视图**

1. 显示项目的一些警告和一些错误。

用途:在Problems窗口中可以看到错误的信息

* 1. **Snippet:片段视图**

1. 将常用的代码收集起来做这程序片段。

例: 右键单出视图区域 Customize new category

选中代码，右键单击 Add to Snippet。

**用途:** Snippets主要是体现了代码重用，可以搜集进来一些比较常用的代码存放在Snippets视图里 随时想用 随时可以从Snippets中点击加到页面中去.如果想创建一个代码段，也就是 snippets可以在Eclipse的editor中选中一块代码段，然后右键点击Add to Snippets....就打开了创建代码段的对话框，比如说数据库的连接什么的，弹出的对话框让你选择代码段的种类，你可以从原有的种类中选也可以输入 新的种类，然后点ok，出现地对话框Customize palette 左边是代码段的树形分类，右边的name是起个好记得名字，Template Pattern 是你的选中代码，可以随意修改，还可以在Variables:里面添加变量，基本上就最简单的变量名，缺省值格式，如果想在代码段中引用，可以用${变量 名}的形式

* 1. **Templates :模版视图**

1. 列出了Eclipse 的所有模版。我们可以将常用的一些写成模版,这样提高了编程的速度。
2. 自定义模版，编辑模版。

 点击可以自定义模版。

 点击修改模版。

**用途:**Eclipse Template对我们是一个相当有用的工具，能节省我们很多写重复代码的时间；也能减少我们对copy&paste的使用。关于Eclipse Templage的基础知识，详见我的Blog：Eclipse Template用法探讨。而模式在我们的面向对象的编程的一个重要手段，特别是Java编程，更加离不开模式。然而，在模式的使用过程中，我们也会遇到 很多重复代码的问题。这篇文章就是试图将Eclipse Template和模式结合起来，来解决我们在使用模式过程中遇到的重复代码的问题。

* 1. **Error Log :Eclipse本身的错误**

1. 当我们的类，包，代码没有错的时候，项目出错。或Eclipse上的按扭点击没有效果，打开就自动关掉。 我们可以通过Error Log视图来查看Eclipse的错误。

**用途:**

* 1. **Markers : 市场**

1. 是一个所有视图的一个集合,列出了所有视图的一些信息。
   1. **Project explorer :项目视图**

在Eclipse IDE的Package Explorer视图中单击右键，在弹出的快捷菜单中选择命令

选择“Existing Projects into Workspace”

选择“Select archive file”，单击“Browse”按钮，定位到之前下载的SWT工具包，此时该对话框的项目列表中会显示一个org.eclipse.swt项目，选定该项目并单击“Finish”按钮。

打开项目的构建路径配置窗口，在该对话框中选择“Projects”选项卡

在该页面中，单击“Add”按钮，在弹出的项目列表中选择“org.eclipse.swt”，单击“OK”按钮完成构建路径配置。注意org.eclipse.swt中包含SWT的源代码压缩文件，通过将Eclipse SWT项目加入到项目的“依赖项目”列表中，可以方便地查看这些源代码以及相应的Java Doc文件。

1. **Eclipse快捷键** @version D20151106
   1. Ctrl + Shift + F 格式化，可以设置具体的格式化模板
   2. Ctrl + Shift + O 自动导入需要的包、移除多余的包
   3. Ctrl + 1 快速解决问题，只要把光标移动到错误上然后点之就会出现解决方案
   4. Ctrl + O 大纲视图
   5. Ctrl + Shift + H 打开对象在语法上的层级关系Hierarchy， （Ctrl + T）/F4 打开当前对象在语法上的层级关系
   6. Ctrl + L 定位到具体的行
   7. Ctrl + F6 切换Editor
   8. Ctrl + F8 切换视图
   9. Ctrl + D 删除一行
   10. Alt + 上下箭头 移动当前行
   11. Ctrl + 3 快速查询
   12. Ctrl + W 关闭当前Editor
   13. Ctrl + Shift + T 查询类型，可以直接在当前对象上进行查询
   14. Ctrl + Shift + R 查询资源，可以直接在当前对象上进行查询
   15. Ctrl + F 当前文本查找
   16. Ctrl + E 显示已经打开Editor列表
       1. Ctrl + Page Up / Page Down 浏览前后选项卡
   17. Ctrl + Z/Y 返回上一次编辑状态/前进到下一个编辑状态
   18. Ctrl + G 搜索当前对象在项目中的位置
   19. Ctrl + H 复杂搜索
   20. Ctrl + Alt + H 打开引用关系层级，就是能够看到那些地方使用了该方法对象之类
   21. Shift + Alt + S 打开Source，也就是打开和代码相关的一些辅助
   22. Alt + / Word Completion， 自动补充，加上Ctrl之后是强制补充
   23. F3 打开声明定义的地方
   24. Ctrl + M 放大/缩小当前（perspective）视图
   25. Ctrl + Q 定位到最后编辑的地方
2. **Eclipse Project Clean和Tomcat Clean**
   1. 都是重新编译（虽然已经设置了自动编译，但是有时候eclipse没有检测到自动编译或者不符合eclipse自动编译条件导致没有重新编译）
   2. Eclipse Project Clean重新编译的范围是eclipse中打开的项目，而Tomcat clean是部署到Tomcat服务器上的项目
   3. 一般出现找不到Class文件而该文件存在而且没有错误，可能是新的文件没有重新编译，此时可以eclipse project clean和tomcat clean二者选一，然后可以选择性地执行tomcat publish（如果设置了自动部署就不用了）