# Eclipse常用功能及快速鍵介紹

日常java開發中實際寫代碼的時間並不會很多，花在整理思路，代碼重構，調試bug會佔用很大部分時間。  
熟練掌握調試工具，瞭解調試相關技巧，是java開發中減少調試時間，提高效率不可或缺的一部分。  
使用System.out.println作為調試主要工具的方法，是java初學者對著書本例子敲代碼的做法。

## 一、相關的常用窗口

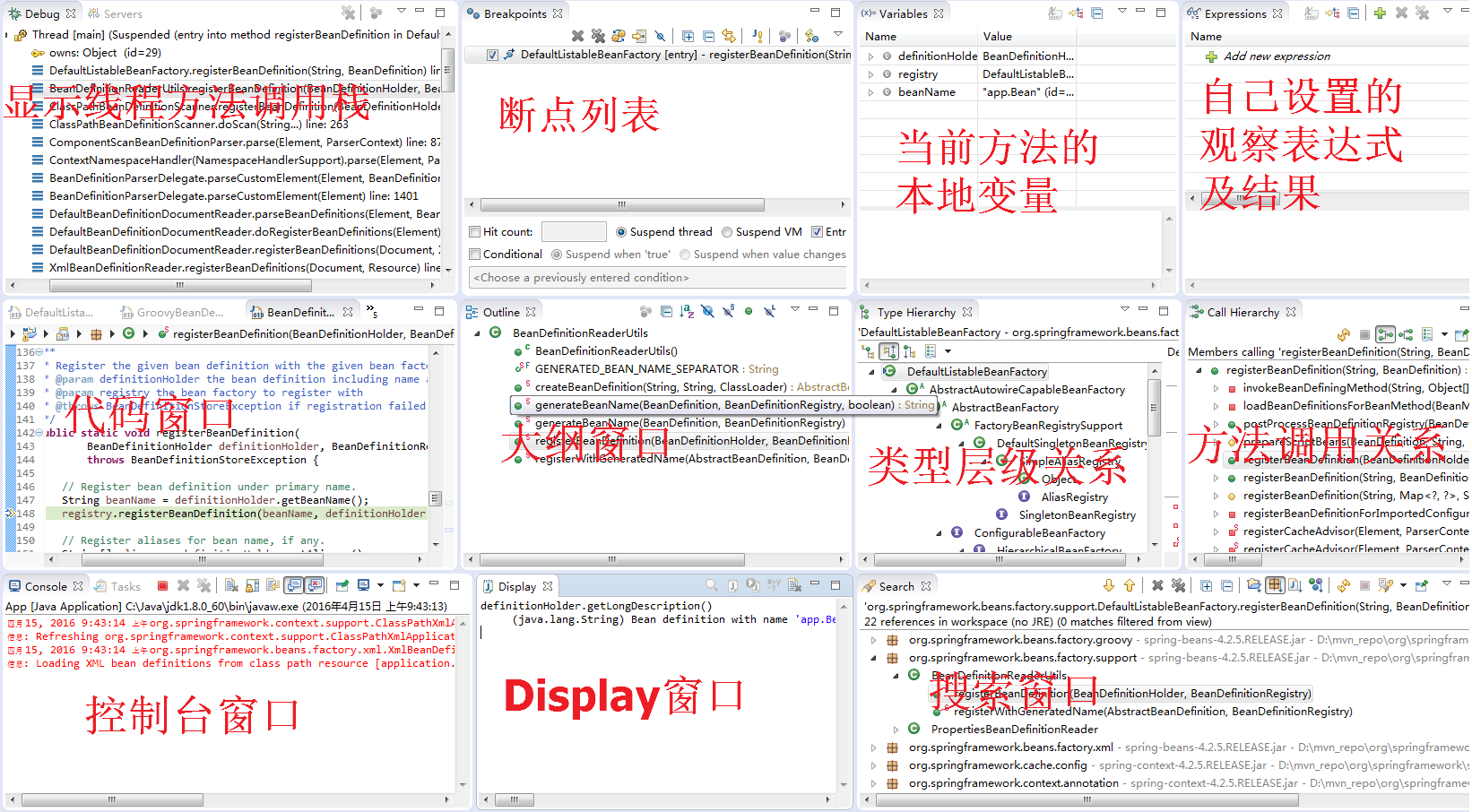
### 最常用的窗口:

| **窗口** | **說明** |
| --- | --- |
| Debug窗口 | 主要顯示當前執行緒方法調用棧， 以及代碼行數(有調試資訊的代碼) |
| 中斷點Breakpoints窗口 | => 中斷點清單視窗，可以方便增加中斷點，設置中斷點條件，刪除中斷點等 |
| 變數Variables視窗 | => 顯示當前方法的本地變數，非static方法，包含this應用，可以修改變數值 |
| 代碼編輯視窗 | => 這個不用多說了 |
| 輸出Console窗口 | => 日誌等輸出內容，調試時，可以將關注的[組件](http://www.2cto.com/kf/all/zujian/" \t "_blank)級別設置低一點，以便獲得跟多輸出資訊 |

### 輔助的窗口有:

| **窗口** | **說明** |
| --- | --- |
| 運算式expression窗口 | => 寫上自己需要觀察的資料的運算式，或者修改變數值 |
| Display窗口 | => 可以在display中執行代碼塊，輸出內容等 |
| 大綱Outline窗口 | => 查看當前類的方法，變數等 |
| 類型層級Type hierarchy窗口 | => 查看當前所在類的繼承層次，包括實現介面，類繼承層次 |
| 方法調用關係Call hierarchy視窗 | => 查看當前方法被哪些方法調用，調用方法在哪些類中、第幾行，可以直接打開對應的方法 |
| 搜索結果Search窗口 | => 結合快速鍵可以查看變數、方法等在工作空間、專案、工作集中被引用或定義的代碼位置 |

窗口全覽圖：



## 二、中斷點類型及中斷點窗口

**可以設置的中斷點類型有五種**

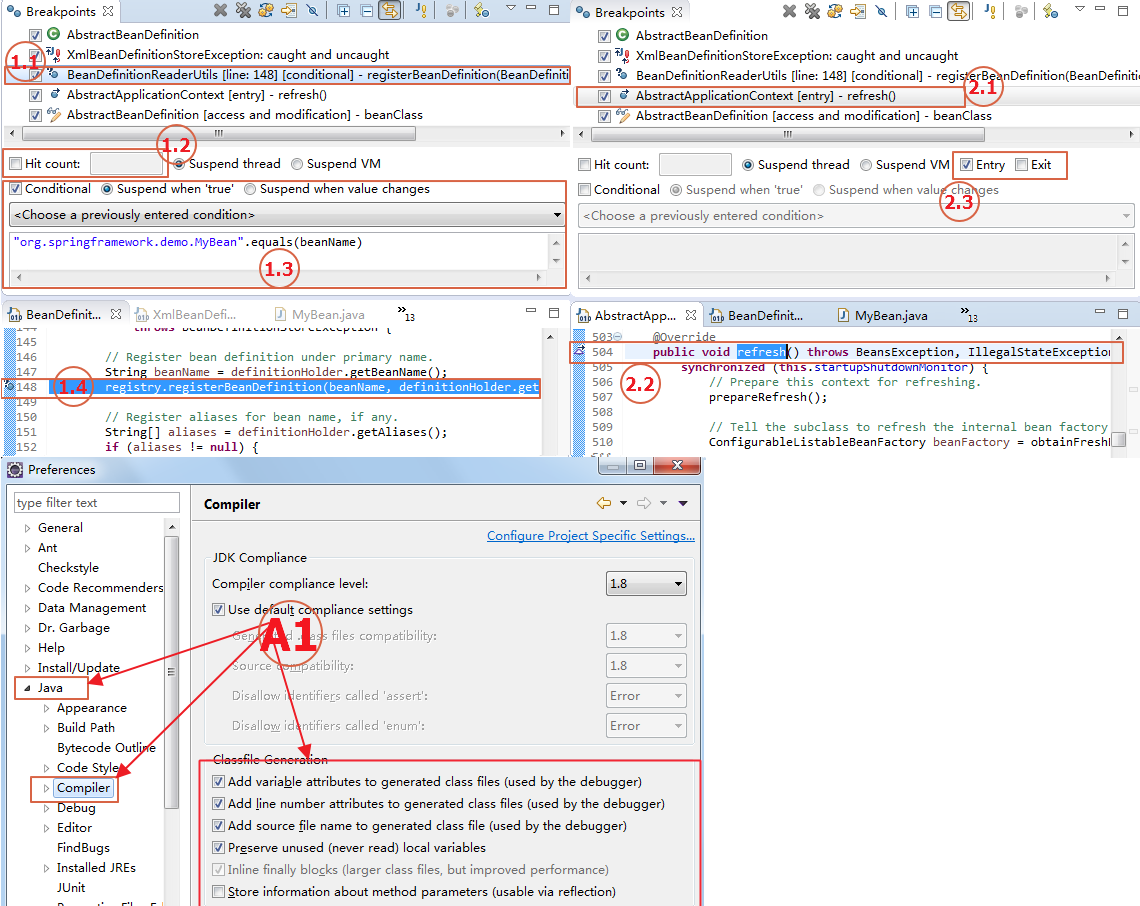
行中斷點(line breakpoints)，方法中斷點(method breakpoints )，觀察中斷點(watch breakpoints-成員變數訪問變更)，異常中斷點(exception breakpoints), 類載入中斷點(class load breakpoints)。每種中斷點的設置有些許不一樣，可以在中斷點上右鍵->Breakpoint properties進行設置，但一般在中斷點視窗有快速設置的介面，Breakpoint properties中多了filter, 其實比較雞肋，用處不大。

**中斷點相關的快速鍵**：

| **快速鍵** | **說明** |
| --- | --- |
| ctrl+shift+b | 在游標處大中斷點/取消中斷點 |
| ctrl+alt+b | 忽略所有中斷點 |
| Alt+shift+q, b | 啟動中斷點窗口 |

**1、行中斷點**： 在方法中的某一行上打斷點1.1、1.4。行中斷點可以設置掛起執行緒/VM的條件1.3，訪問次數1.2。  
1.3中的條件是，spring在註冊Bean定義(registerBeanDefinition)時，如果是org.springframework.demo.MyBean，就掛起執行緒，可以開始單步調試了。  
對於命中次數(hit count)1.2的使用，一般是在迴圈中，第N個物件的處理有問題，設置hit count = N， 重調試時，可以方便到達需要調試的迴圈次數時，停下來調試。

**2、方法中斷點**：在方法上打斷點2.1、2.2。方法中斷點的好處是可以從方法方法進入或者退出時2.3，停下來調試，類似行中斷點，而且只有行中斷點和方法中斷點有條件和訪問次數的設置功能。  
但是方法中斷點還有另外一個好處，如果代碼編譯時，指定不攜帶調試資訊，行中斷點是不起作用的，只能打方法中斷點。  
有興趣的可以通過A1將Add line number… 前的勾去掉， 調試下看看。

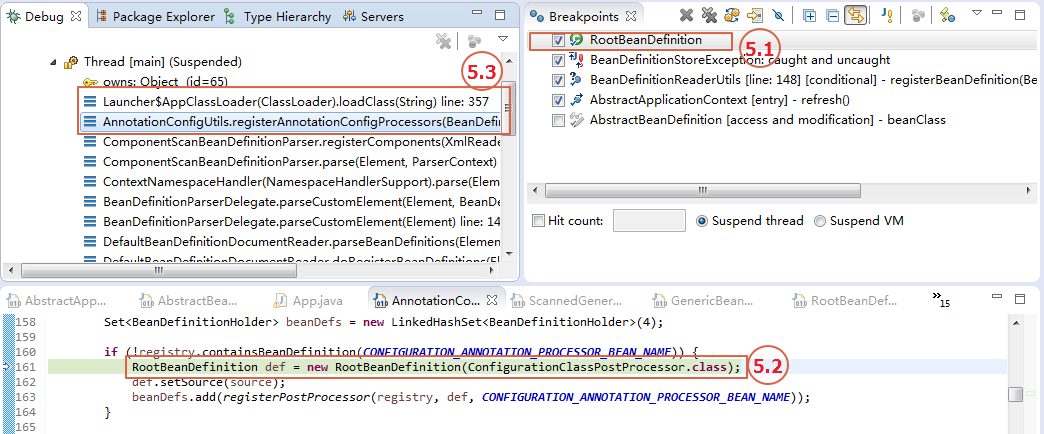


**3、觀察中斷點**： 在成員變數上打的中斷點3.1、3.3。只有物件成員變數有效果，靜態成員變數不起作用。  
可以設置變數被訪問或者設置的時候掛起執行緒/VM 3.2，也就是類似3.4的所有對成員變數的訪問或者設置的方法都會被監控到  
**4、異常中斷點**： 異常中斷點可以通過4.6添加，或者點擊日誌資訊中輸出的異常類資訊添加。  
異常中斷點4.1，系統發生異常時，在被捕獲異常的拋出位置處或者程式未捕獲的異常拋出處 4.2、4.4, 掛起執行緒/VM， 也可以指定是否包括異常的子類也被檢測4.3、4.5。  
另外除了以上正常設置的異常掛起，從java->debug中可以設置掛起執行，主要有下面兩個：

1、是否在發生全域未捕獲時掛起(suspend execution on uncaught exceptions)，調試時，老是有異常掛起影響調試，但是沒有設置異常中斷點的情況，就可以勾選掉這個選項；

2、是否在編譯錯誤時掛起，一般在邊調試邊改代碼時會發生這種情況；

**5、類載入中斷點**: 在類名上打的中斷點5.1。介面上是打不了類載入中斷點的，但是抽象類別是可以的，只是在調試的時候，中斷點不會明顯進入classloader中，單步進入知會進入到子類的構造方法中，非抽象類別在掛起執行緒後單步進入就會到classloader中（如果沒有filter過濾掉的話）5.3。類載入中斷點不管是打在抽象或者非抽象類別上，都會在類第一次載入或者第一個子類第一次被載入時，掛起執行緒5.2。



## 三、Debug視窗、代碼視窗，快速鍵

打完中斷點進入調試階段，調試過程中最主要的就是調試視窗和代碼視窗， 調試視窗和代碼視窗的右鍵功能表、使用的快速鍵以及達到的效果都值得深入瞭解；

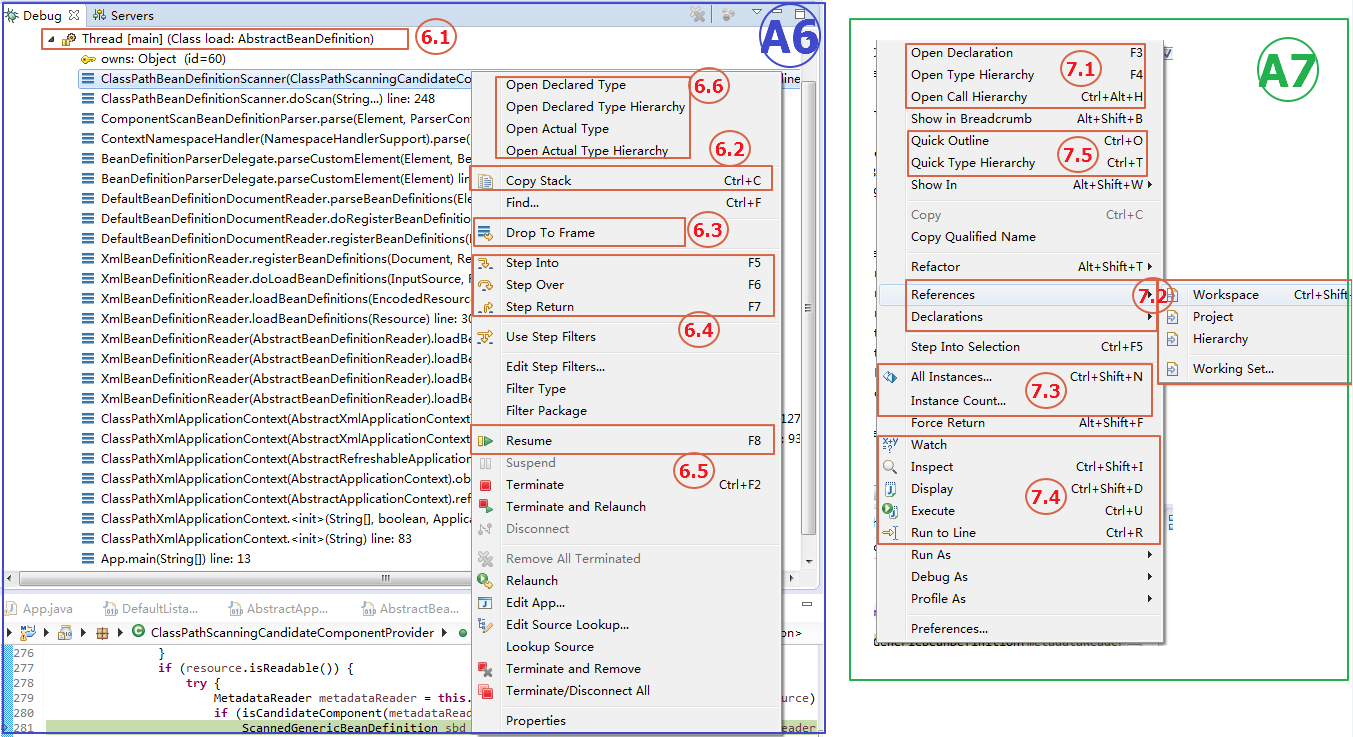
**Debug視窗右鍵功能表及快速鍵**：如A6所示。  
**代碼視窗右鍵功能表及快速鍵**：如A7所示。

下面將按照功能，結合A6、A7一起說明：

**1、調試執行**：

| **標記** | **功能** | **快速鍵** | **描述** |
| --- | --- | --- | --- |
| 6.4 | Step Info | F5 | -> 單步進入(如果有方法調用，將進入調用方法中進行調試); |
| 6.4 | Step Over | F6 | -> 單步跳過(不進入行的任何方法調用中，直接執行完當前代碼行，並跳到下一行); |
| 6.4 | Step Return | F7 | -> 單步返回(執行完當前方法，並從調用棧中彈出當前方法，返回當前方法被調用處)； |
| 6.5 | Resume | F8 | -> 恢復正常執行(直到遇到下一個中斷點); |
| 7.4 | Run to Line | Ctrl+R | -> 執行到當前行(將忽略中間所有中斷點，執行到當前游標所在行); |
| 6.3 | Drop To Frame | 無 | -> 回退到指定方法開始處執行，這個功能相當贊。 在方法調用棧上的某個方法右鍵，選擇Drop To Frame就可以從該方法的開始處執行 比如重新執行本方法，可以在本方法上用Drop To Frame，將從本方法的第一行重新執行。 當然對於有副作用的方法，比如[資料庫](http://www.2cto.com/database/)操作，更改傳入參數的物件內容等操作可能重新執行就再是你想要的內容了。 |
| 6.1+6.2 | Copy Stack | 無 | -> 拷貝當前執行緒棧資訊 |

如果在調試時，需要排除一些類、包等不需要進入調試，可以使用Edit Step Filters設置。  
A6的properties： [Java](http://www.2cto.com/kf/ware/Java/)進程啟動相關資訊，包括控制台啟動參數，環境參數等。如果參數啟動參數有問題可以先看看這裡的實際啟動參數是不是有誤。另外還可以查看[虛擬機器](http://www.2cto.com/os/xuniji/" \t "_blank)支援調試的相關選項。



**2、數據查看**

| **標記** | **功能** | **快速鍵** | **描述** |
| --- | --- | --- | --- |
| 7.4 | Inspect | ctrl+shift+i | -> 察看選擇的變數、運算式的值或執行結果, 再次按ctrl+shift+i可以將當前運算式或值添加到Expressions視窗中查看; |
| 7.4 | Display | ctrl+shift+d | -> 顯示選擇的變數、運算式的值或執行結果, 再次按ctrl+shift+d可以將當前運算式或值添加到Display視窗中顯示; |
| 7.4 | Execute | ctrl+u | -> 執行選擇運算式; |
| 7.4 | Run to Line | ctrl+r | -> 執行到當前行(將忽略中間所有中斷點，執行到當前游標所在行); |
| 7.3 | All Instances | ctrl+shift+n | -> 查看選擇的類的所有物件，這個功能超贊； |
| 7.3 | Instance Count | 無 | -> 查看選擇的類的所有物件個數； |
| 7.4 | Watch | 無 | -> 添加當前變數、運算式到Expressions視窗中； |

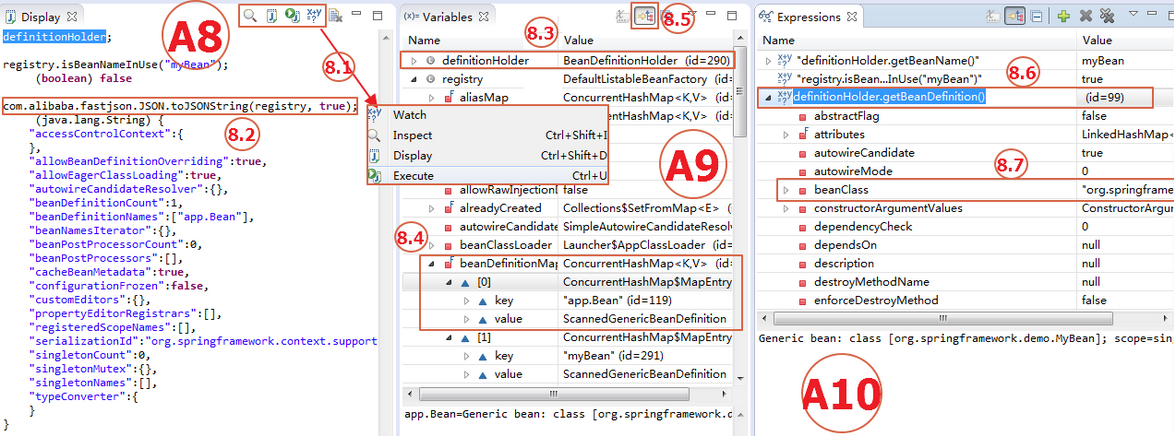
## 四、調試時窗口

**1、Display窗口A8**  
在Display視窗中，可以執行代碼之外的任何合理的語句，也可以使用Inspect， Display, Execute這些方式查看或者執行語句——8.1。  
比如，

**a**、輸入com.alibaba.fastjson.JSON.toJSONString(registry, true); 並選擇這一行代碼，ctrl+shift+d，可以執行這行代碼並在Display視窗中輸出結果——8.2。  
**b**、當然也可以改變當前代碼的變數值，比如輸入beanName = “myBeanX”; 選擇這行代碼，用Inspect， Display, Execute這幾種方式都可以執行這行代碼。  
**c**、也可以輸入System.out.println(com.alibaba.fastjson.JSON.toJSONString(registry, true)); 執行就會在console窗口輸出結果。

**2、Variables窗口A9**  
在Variable視窗中可以看到當前方法的本地變數值8.3，非靜態方法會有this表示物件自己，也可以查看本地變數相應的屬性8.4， 8.5的功能可以讓集合類型物件中的資料顯示容易看點。  
這個視窗除了察看變數的功能之外，還可以直接在Value列修改變數值或物件屬性。

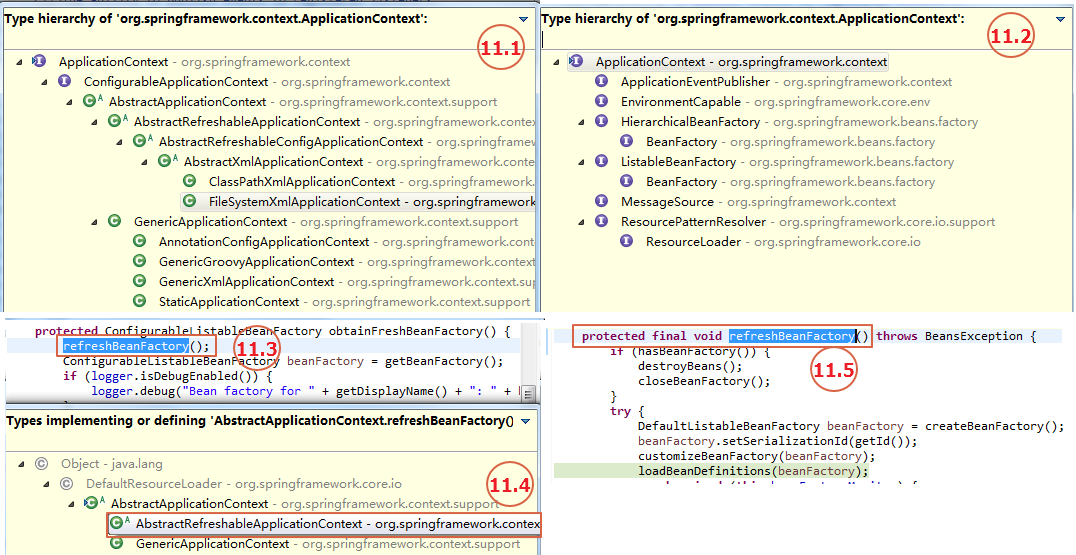
**3、Expression窗口A10**  
Expression視窗的功能主要是用來察看變數之外的運算式的結果，執行固定運算式等功能。比如察看變數definitionHolder中的屬性beanDefinitionHolder, 直接將definitionHolder.getBeanDefinition()加入到Expressions窗口8.6， 8.7。



## 五、代碼查看輔助視窗

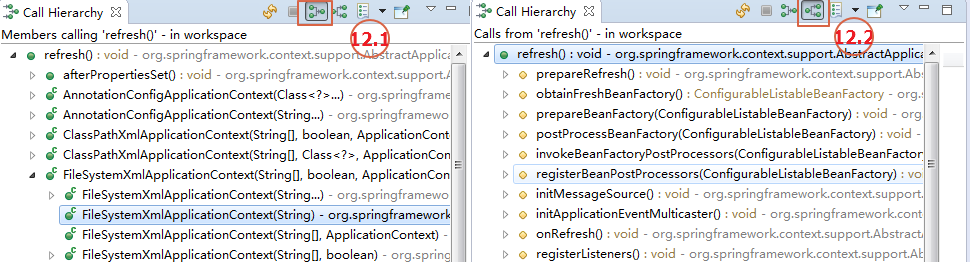
###### 1、代碼視窗：

| **標記** | **功能** | **快速鍵** | **描述** |
| --- | --- | --- | --- |
| 11.1~11.5 | quick type hierarchy | ctrl+t | 查看當前類、介面的繼承層次， 預設進入時，顯示繼承/實現當前類/方法的子類，子介面11.1；再次ctrl+t，將顯示當前類、介面繼承/實現的超類/介面11.2；調試時，經常用該功能，在介面或抽象類別的方法調用處11.3，ctrl+t察看實現類11.4，直接導航到對應的實現方法中11.5。 |
|  | quick outline | ctrl+o | 查看當前類的大綱，包括方法，屬性等內容； 用處不大； |
|  | open declarations | F3 | 查看變數、屬性、方法定義的地方 |



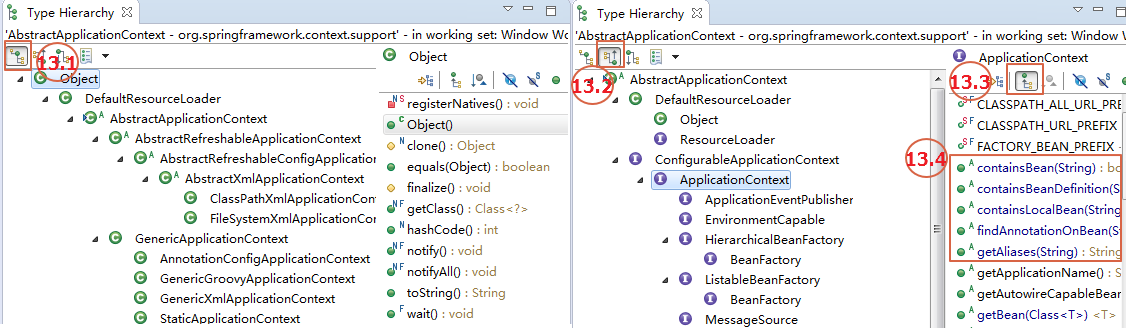
###### 2、Call Hierarchy窗口:

| **標記** | **功能** | **快速鍵** | **描述** |
| --- | --- | --- | --- |
| 12.1~12.2 | open call hierarchy | ctrl+alt+h | 查看方法被調用層次， 可以看當前方法被調用的地方12.1，或者當前方法調用了其他類的方法12.2 |



###### 3、Type Hierarchy窗口：

| **標記** | **功能** | **快速鍵** | **描述** |
| --- | --- | --- | --- |
| 13.1~13.4 | open type hierarchy | F4 | 查看繼承層次， 可以查看類的繼承層次，包括子類父類13.1， 或者類實現的介面繼承層次13.2，還會根據選擇的類/介面，在右邊顯示該類的大綱13.3、 13.4； 13.3可以選擇是否顯示父類/父介面的屬性、方法等 |



###### 4、Search 窗口：

| **標記** | **功能** | **快速鍵** | **描述** |
| --- | --- | --- | --- |
| 14.1 | declarations | ctrl+g | 相同的方法簽名在工作空間中及協力廠商jar包中被定義的位置14.1 |
| 14.2 | references | ctrl+shif+g | 當前選中的變數、屬性、方法在工作空間中及協力廠商jar包中被引用的位置14.2 |
| 14.3 |  | ctrl+shift+u | 查看變數、屬性 、方法在當前類中出現的地方14.3 |
| 14.4 | implements |  | 查看實現當前介面的類14.8 |
| 14.4~14.7 | 顯示方式 |  | 可以選擇不同的顯示方式，具體看圖 |

