Tutorial base su Eclipse

Jug Marche

Relatore: Andrea Del Bene

www.jugancona.it

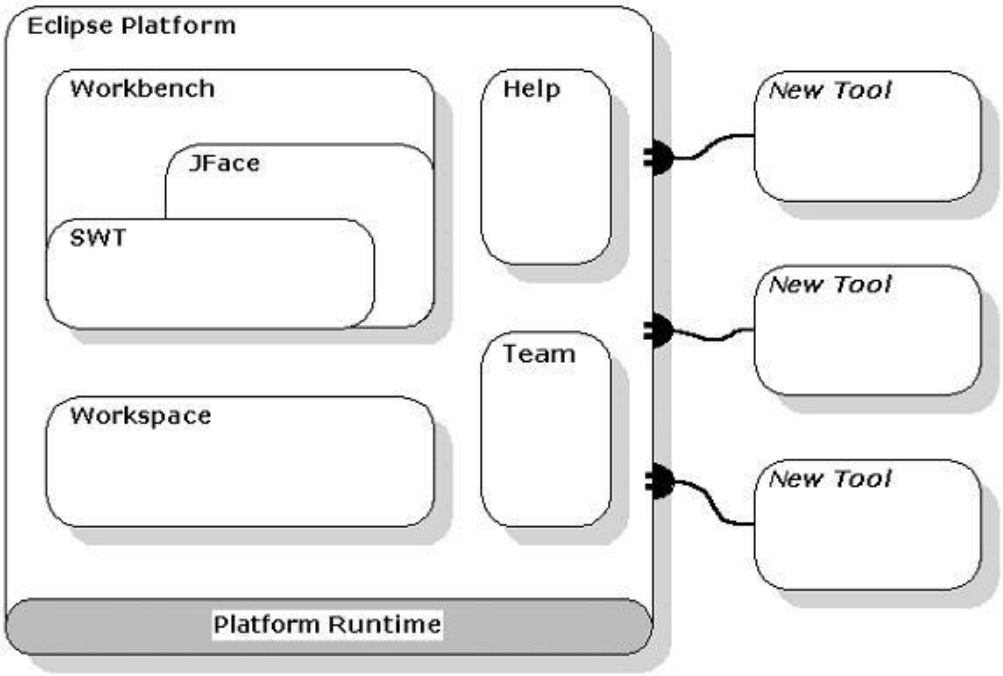
03/03/2010

Eclipse in 2 punti.

- Che cos'è Eclipse?
  - Eclipse è un IDE multilinguaggio e multipiattaforma scritto in Java. E'gratuito e Open Source. Nato come evoluzione dell'ambiente di sviluppo per Java *Visual Age di IBM*, è diventato Open Source nel 2001 e dal 2003 è controllato dalla Eclipse Foundation, organiz-zazione *no profit* indipendente (anche da IBM!).

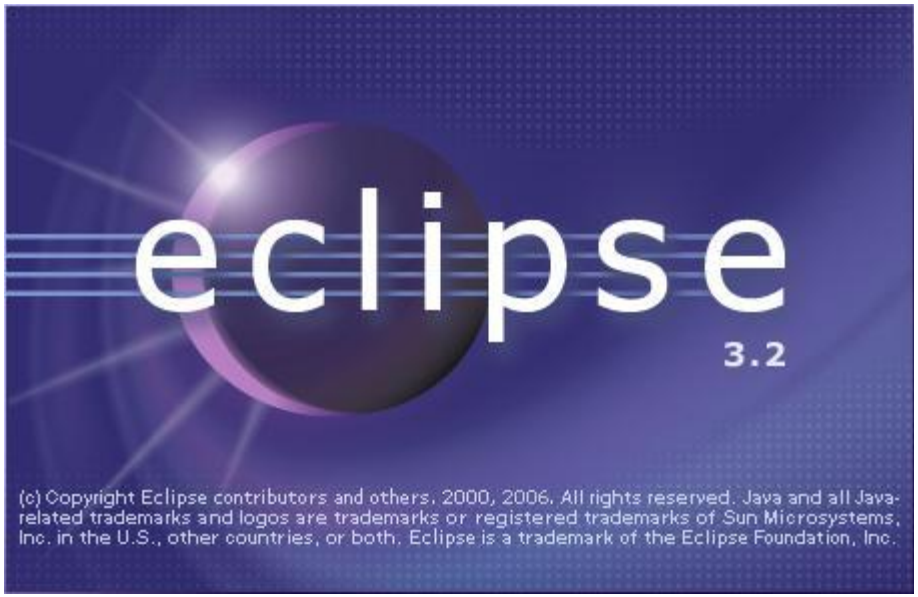
- Perché ha avuto così tanto successo?
  - La sua natura aperta lo anno reso velocemente adat-tabile alle esigenze dei suoi utenti. Eclipse è inoltre basato su un'architettura a plugin che lo rende fortemente modulare e liberamente espandibile *da chiunque* per venire incontro alle esigenze più particolari e disparate.

Architettura



- La struttura di Eclipse è composta da una serie di tecnologie *core* e da plugin che si agganciano alle funzionalità di base per espandere l'intero applicativo. Le componenti *core* possono essere utilizzate per costruire nuove applicazioni indipendenti da Eclipse (es: SWT come toolkit grafico al posto di Swing).

1, 2, 3…via!

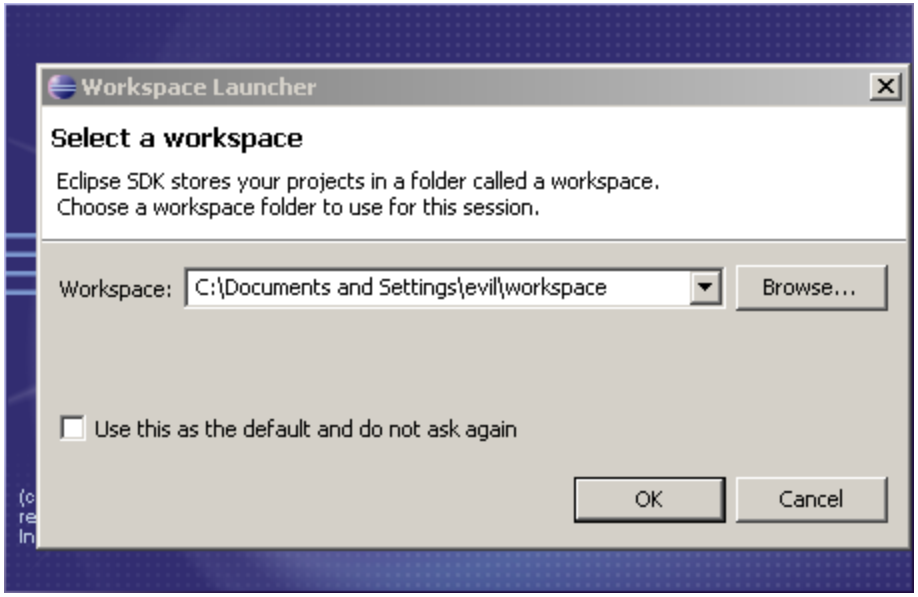


Splashscreen di Eclipse

- Essendo scritto in Java Eclipse è disponibile per tutti i sistemi più diffusi (Windows, Mac, Linux, Solaris, ecc…). Per Windows è disponibile come file zip (non necessita di installazione ☺) al sito www.eclipse.org.

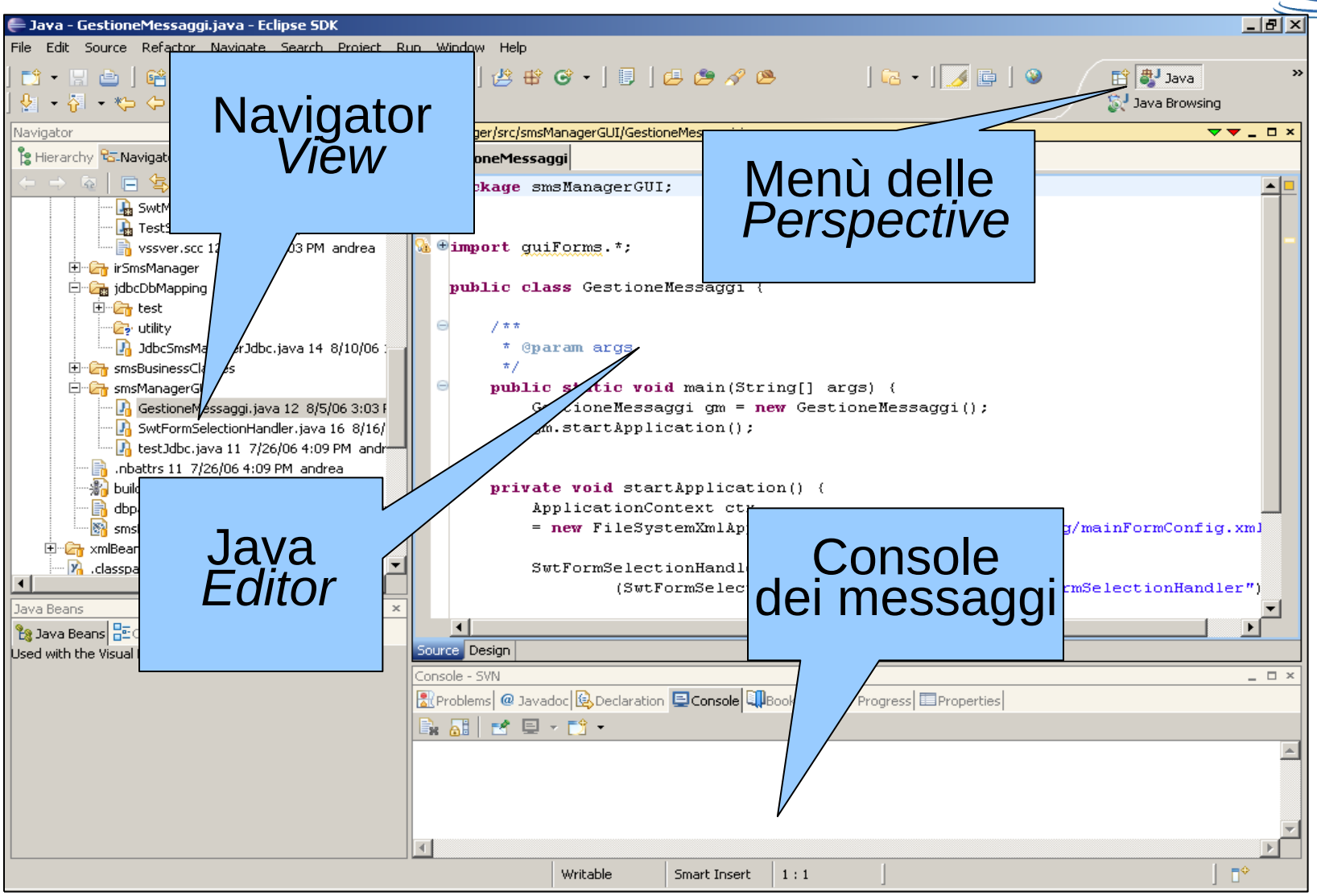
Impostazione Workspace

- Al primo avvio Eclipse vi chiederà quale cartella usare per memorizzare i suoi progetti. In gergo tale cartella è chiamata Workspace. Le impostazioni di default proposte di solito vanno più che bene…



Schermata di benvenuto

Il workspace



Le perspectives

- L'interfaccia grafica di Eclipse è organizzata a *perspectives*. Le *perspectives* non sono altro che dei raggruppamenti di funzionalità dell'IDE, fatti in base ad una specifica operazione di sviluppo. La Java perspective ad esempio riunisce strumenti di stesura e organizzazione del codice mentre la Debug perspective fornisce strumenti in fase di debug.

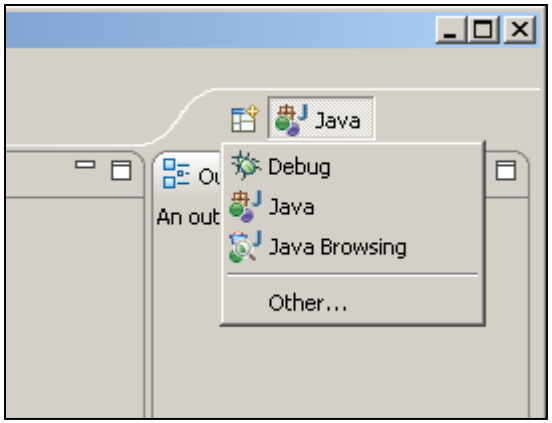
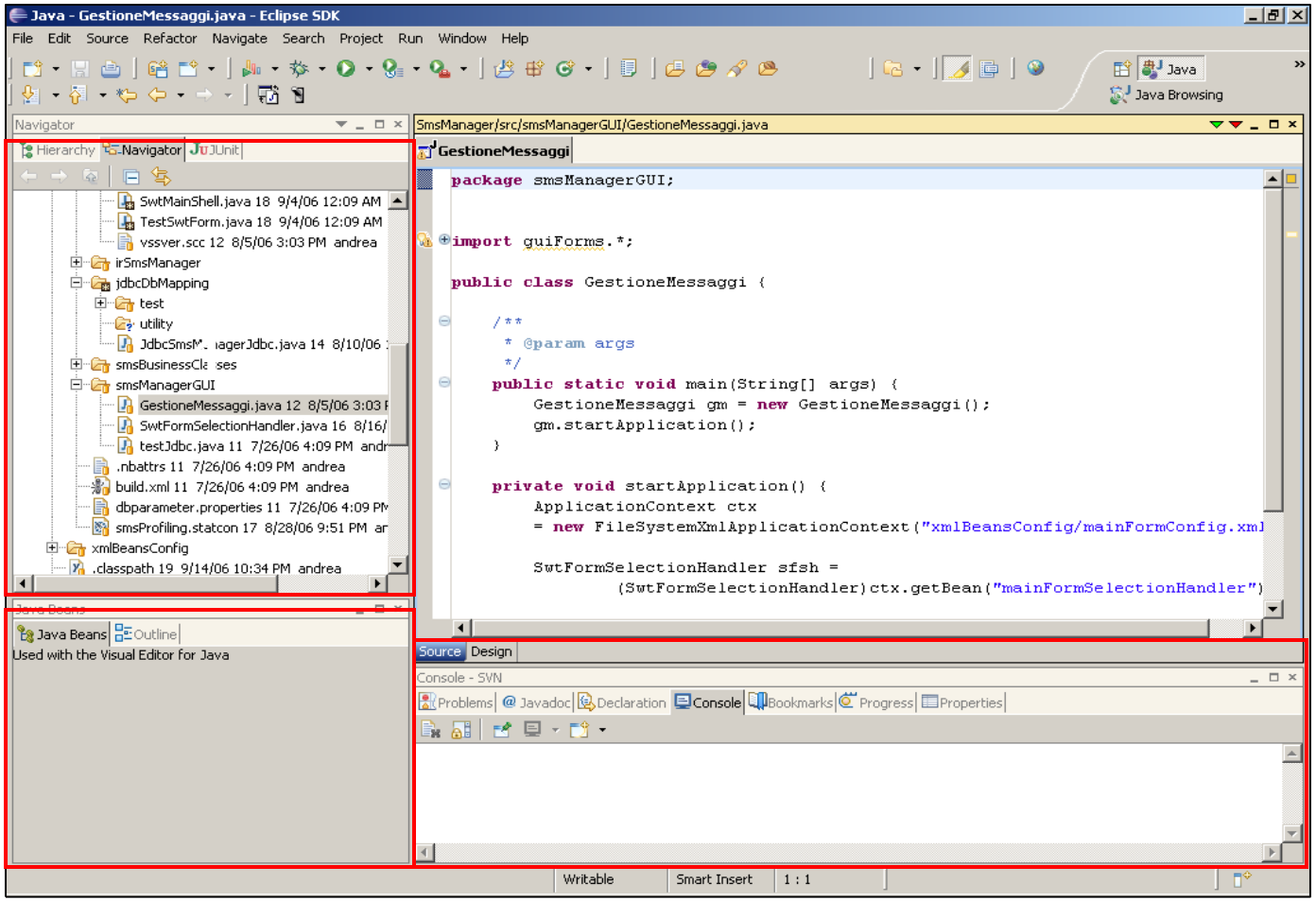


Figura 2 la ceppa

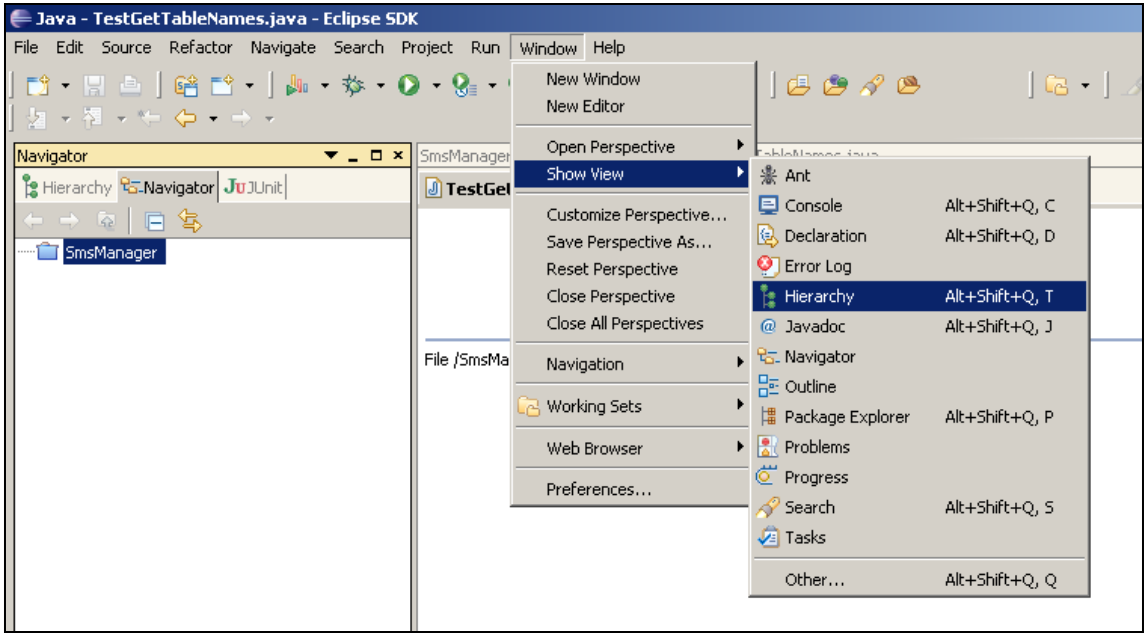
Le viste (1/3)

- Le viste sono “riquadri” organizzati in schede che offrono supporto per organizzare e stendere il codice. Ci sono tre gruppi schede che possono ospitare le viste:



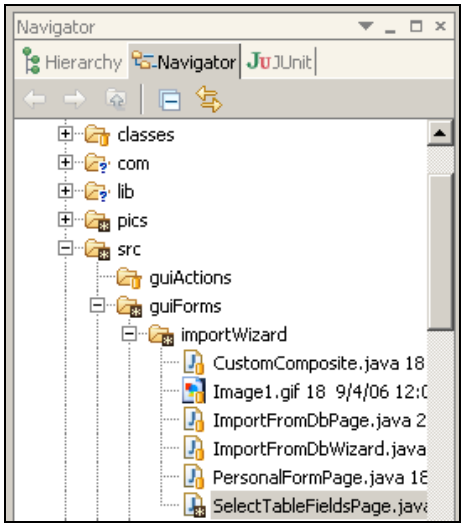
Le viste (2/3)

- Le varie viste possono essere spostate da un gruppo all'altro mediante semplice trascinamento della scheda sul nuovo gruppo.
- Le viste sono molto numerose. Per aggiungerne di nuove occorre accedere al menù "Window→Show Views".

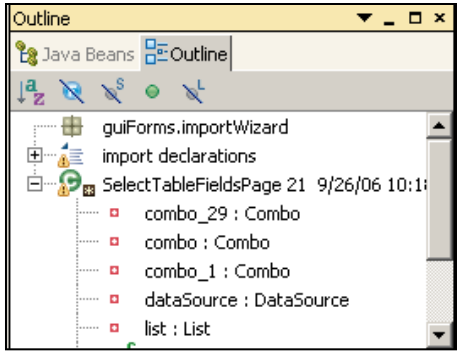


Le viste (3/3)

- Tra le viste più utili c'è la "*Navigator*":

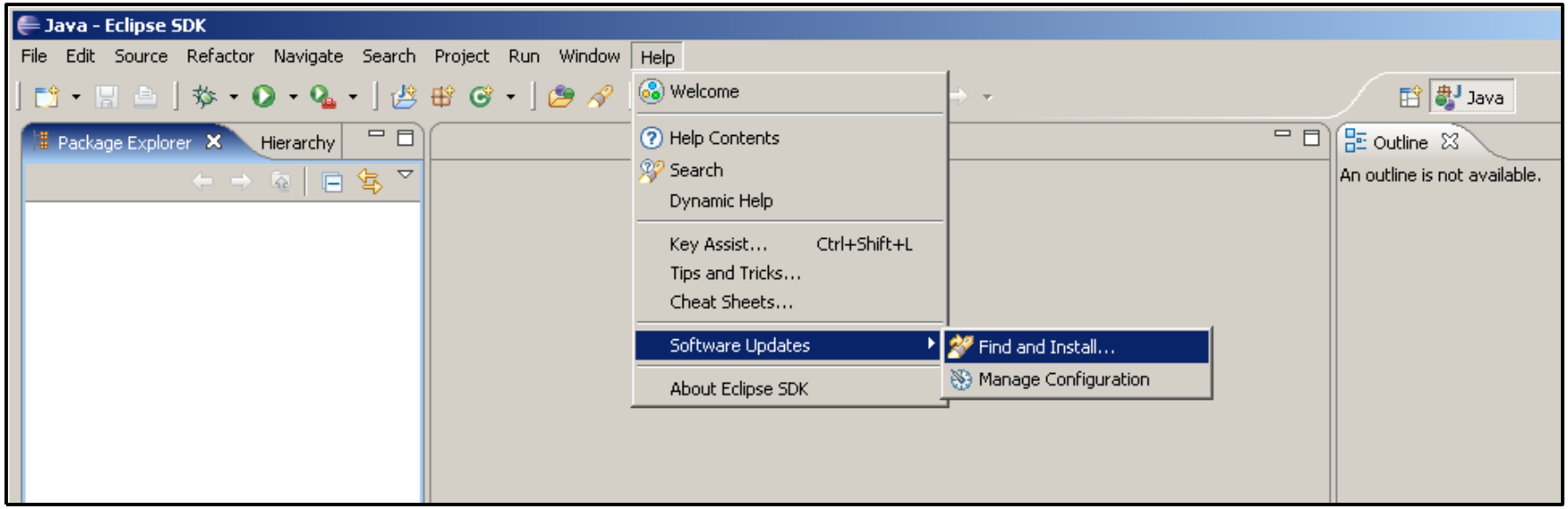


- Anche la vista *"Outline"* risulta molto comoda:

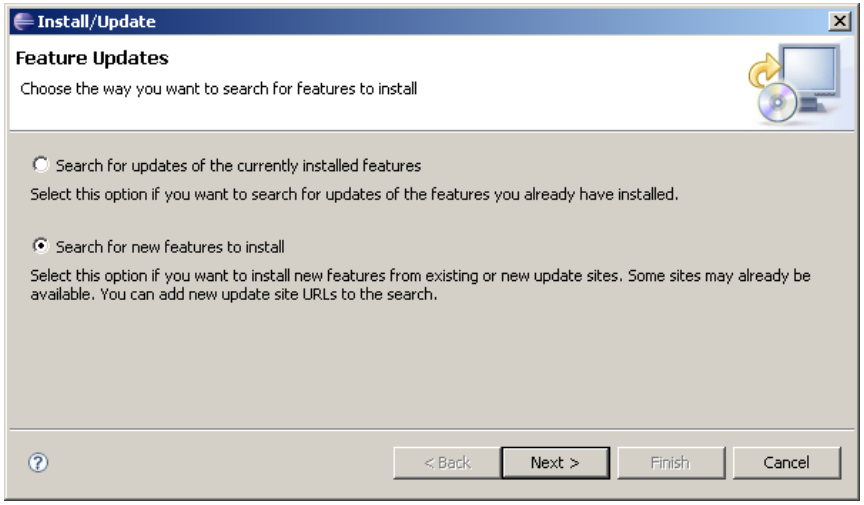


Una finestra sul mondo…

- In Eclipse è fondamentale, prima ancora di scrivere codice, imparare a gestire i plugin e a scaricarne di nuovi. Per accedere al *plugin manager* di Eclipse basta selezionare la voce di menù alla posizione *'Tool→Software Updates→Find and Install…'*

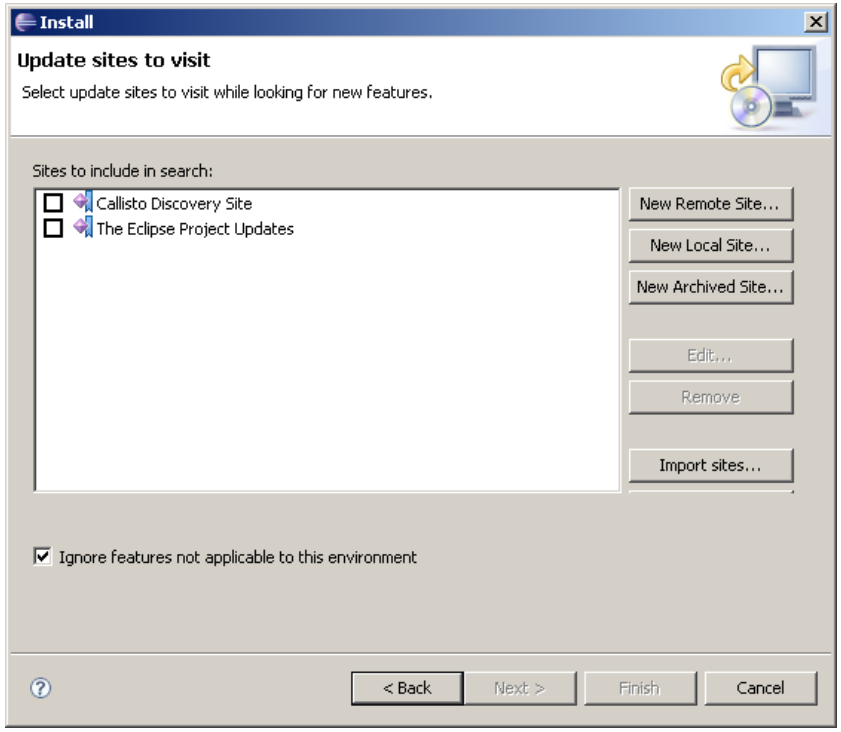


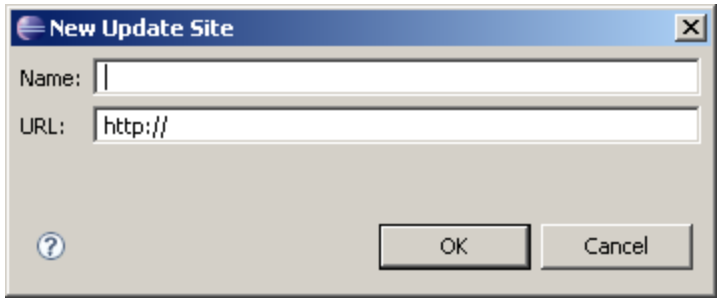
Ricerca e aggiornamento plugin 1/2



- Il Wizard ci chiede se aggiornare i plugin presenti o installare nuovi plugin. Se si sceglie di installare nuovi plug –in verrà chiesto di compilare una lista dei siti in cui cercare i nuovi componenti. Alcune voci sono già presenti di default.

Ricerca e aggiornamento plugin 2/2





- Tramite il pulsante ‘New Remote Site’ si possono ovviamente aggiungere nuovi siti che ospitano nuovi plugin.

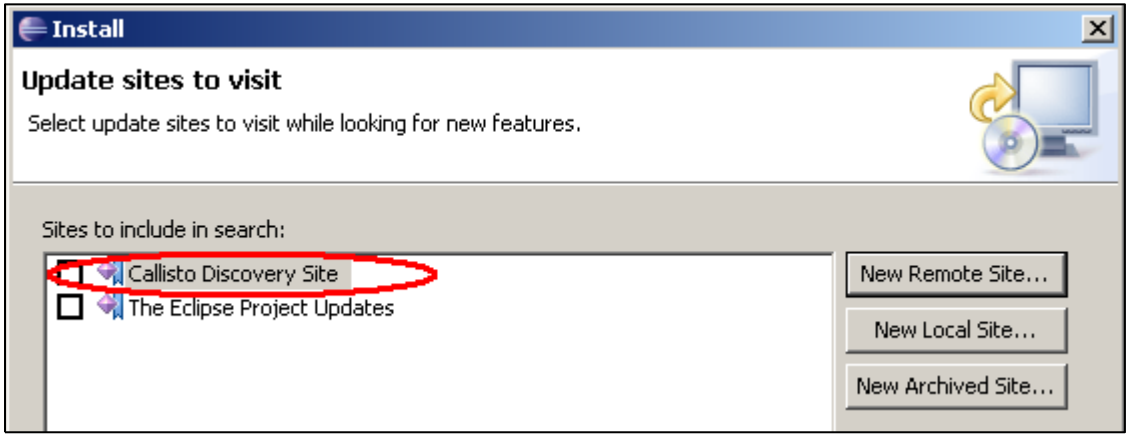
Pseudo esercizio…

- Molte delle figure riportate mostrano Eclipse con un “look” diverso da quello standard. Questo perché sulla mia installazione è presente il plug in “**Extended VS Presentation**” di Andrei Loskutov.
- Vi invito a scaricarlo e provarlo per fare pratica con il plugin manager di Eclipse.

  Url update manager:
  http://andrei.gmxhome.de/eclipse/

Cenni su Callisto

- Callisto è un progetto che raggruppa 10 plugin in unica voce nel plugin manager. I 10 plugin possono quindi essere scaricati e aggiornati in contemporanea e i loro rilasci avvengono simultaneamente. Callisto è disponibile solo dalla versione 3.2 di Eclipse.



- NOTA: per differenziare le versioni, Callisto dalla versione 3.3 di Eclipse si chiamerà Europa.

Primi passi con Eclipse

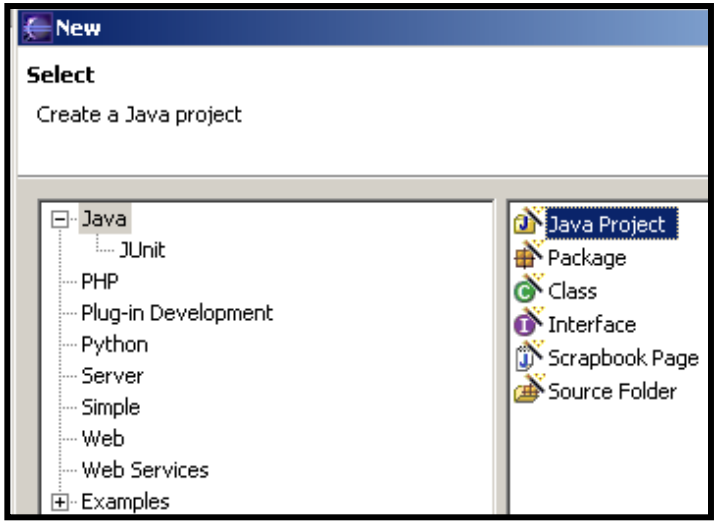
Jug Marche

Relatore: Andrea Del Bene

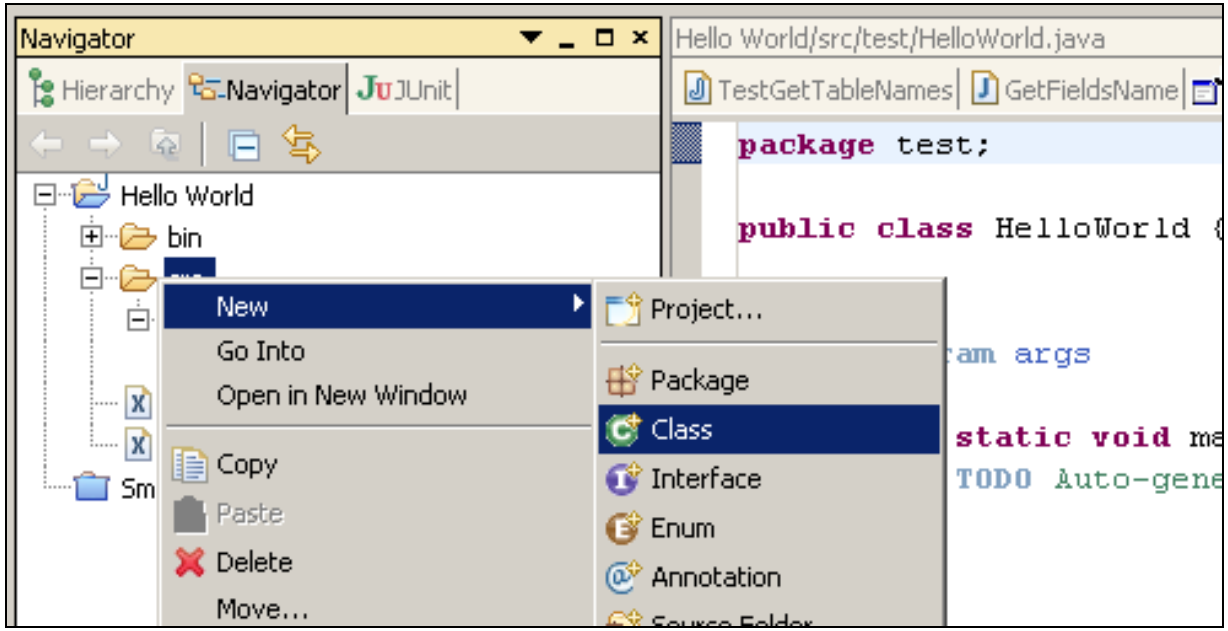
www.jugancona.it

Nuovo progetto Java in Eclipse

- Cliccare su **File→New→ Project…**
- Scegliere Java Project, dargli un nome e cliccare su ‘Finish’.



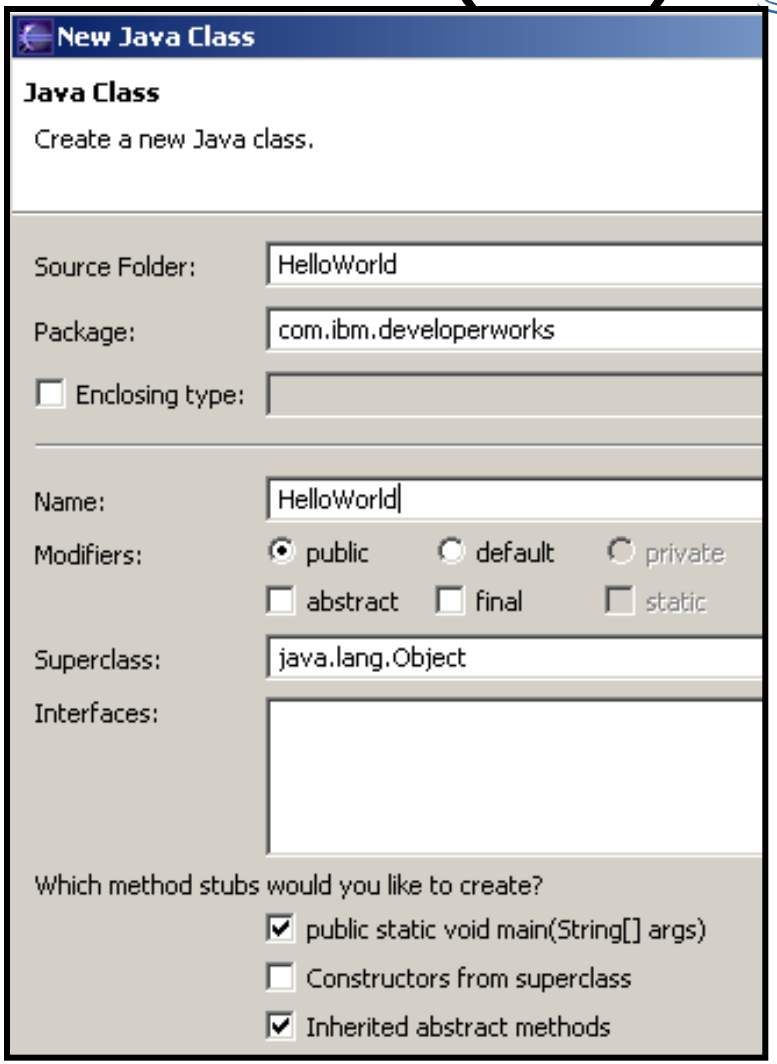
Creare una nuova classe (1/3).



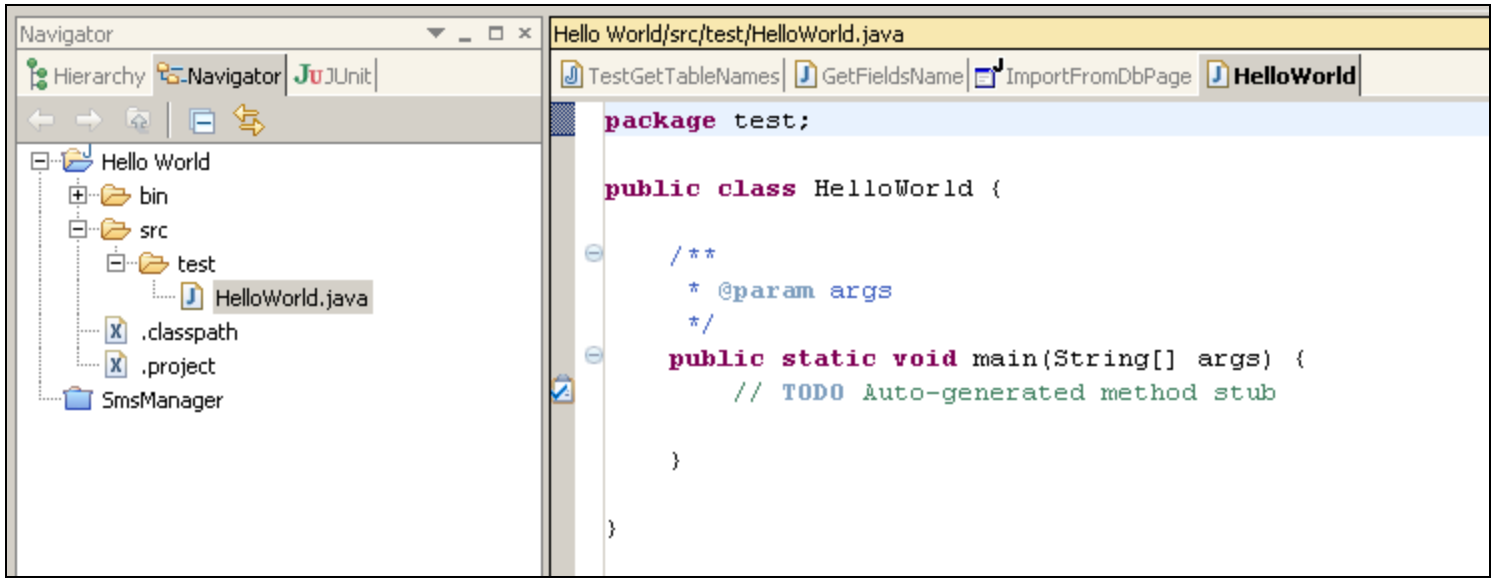
- Per creare una nuova classe fare click con il tasto destro su una package nella vista “Package Explorer”, poi si scelga dal menù “**New→Class**”.

Creare una nuova classe (2/3).

- Inserire un nome per la classe.
- Eclipse controlla mentre digitiamo se il nome è sintatticamente corretto.
- Si possono impostare diversi attributi della classe che stiamo creando, come la visibilità, la superclasse, eventuali interfacce implementate, ecc….



Creare una nuova classe (3/3).



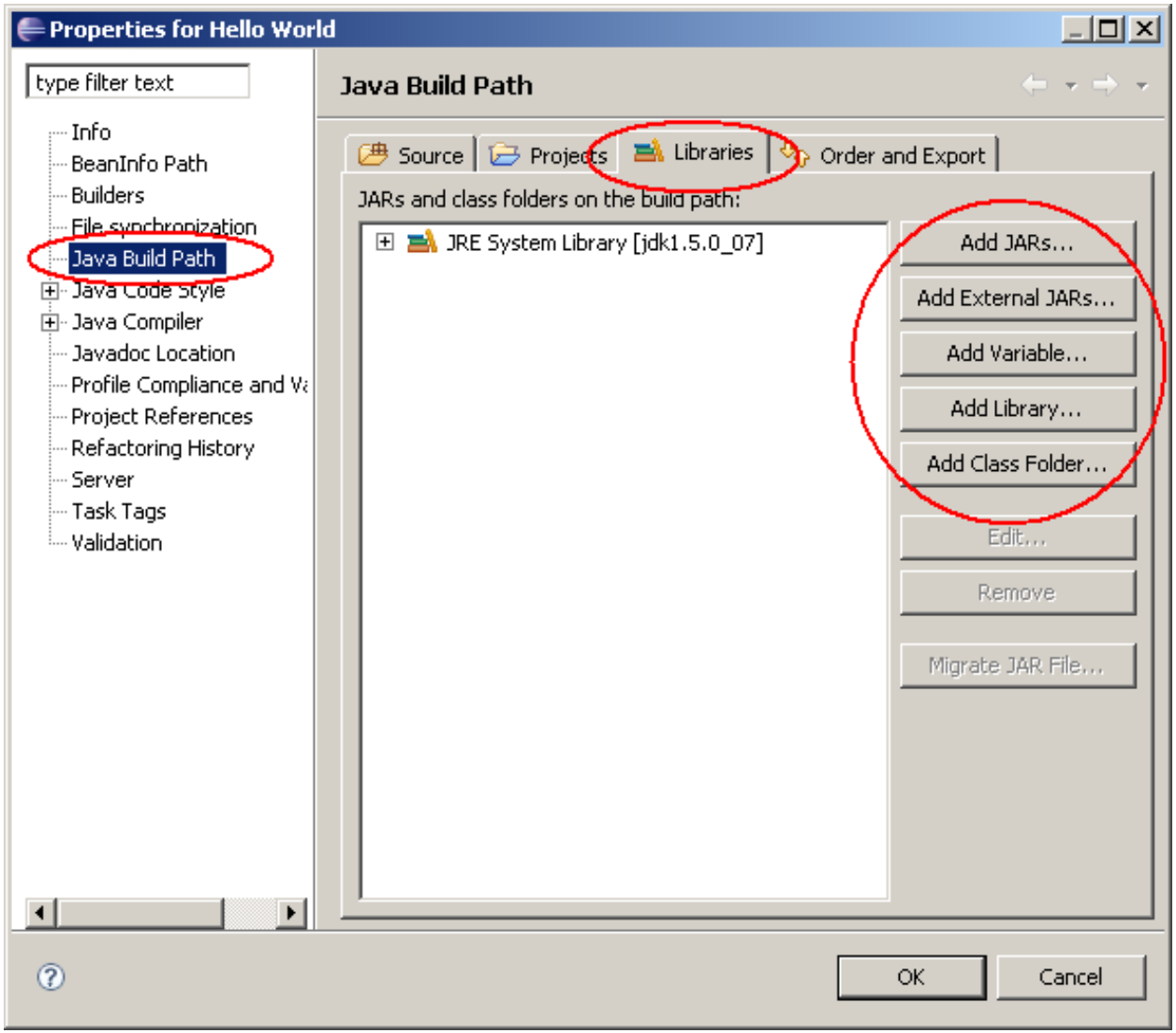
- La nuova classe appare nella vista sulla sinistra e l'editor mostra il suo codice generato.

Gestione del Classpath (1/2)

- Per compilare ed eseguire un programma Java occorre preoccuparsi che le librerie usate al suo interno siano incluse nel classpath. Cliccando la voce di menù alla posizione “Project→Properties”

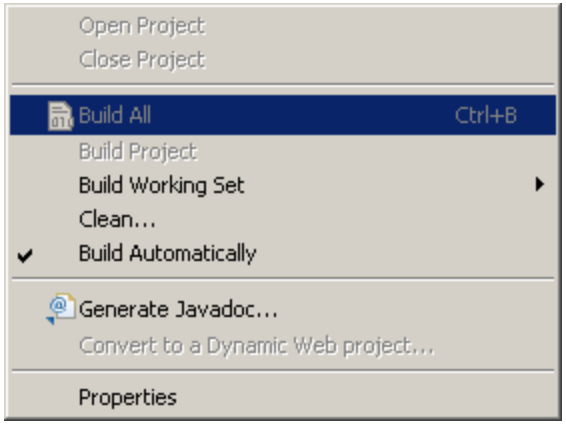
- Nel menù di sinstra si scelga “Java Build Path” e nella scheda “Libraries” si possono aggiungere nel classpath jar, directory, ecc…

Gestione del Classpath (2/2)



Compilazione sorgenti

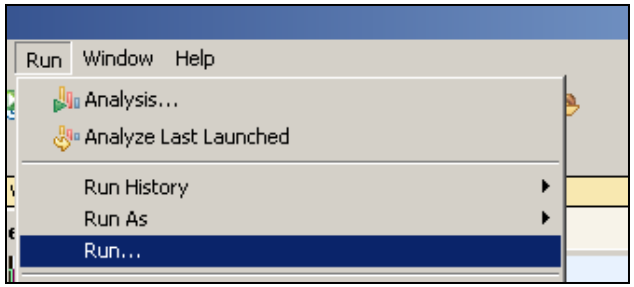
- Il menù “Project” fornisce accesso alle funzioni di compilazione del sorgente. In eclipse è possibile compileare “a comando” (“Build All”) o attivare la compilazione automatica (“Build Automatically”), altro punto forte di Eclipse.



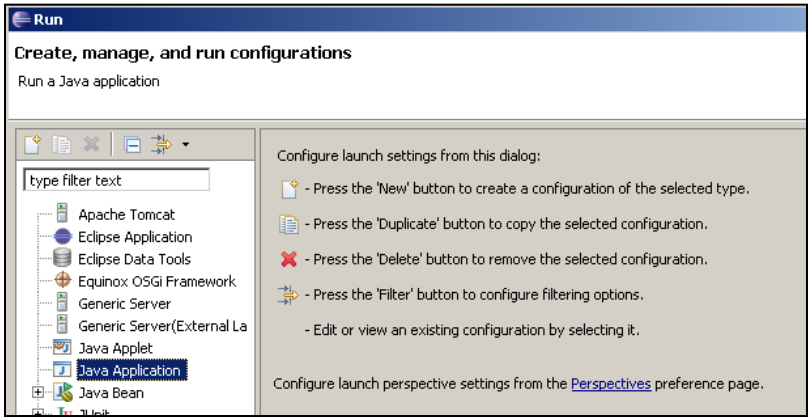
- Con la compilazione automatica l’editor segnala in tempo reale gli errori di compilazione del codice…

Esecuzione di un programma (1/3)

- Per eseguire il progetto corrente, si clicchi sul menù “Run” e si scelga “Run…”

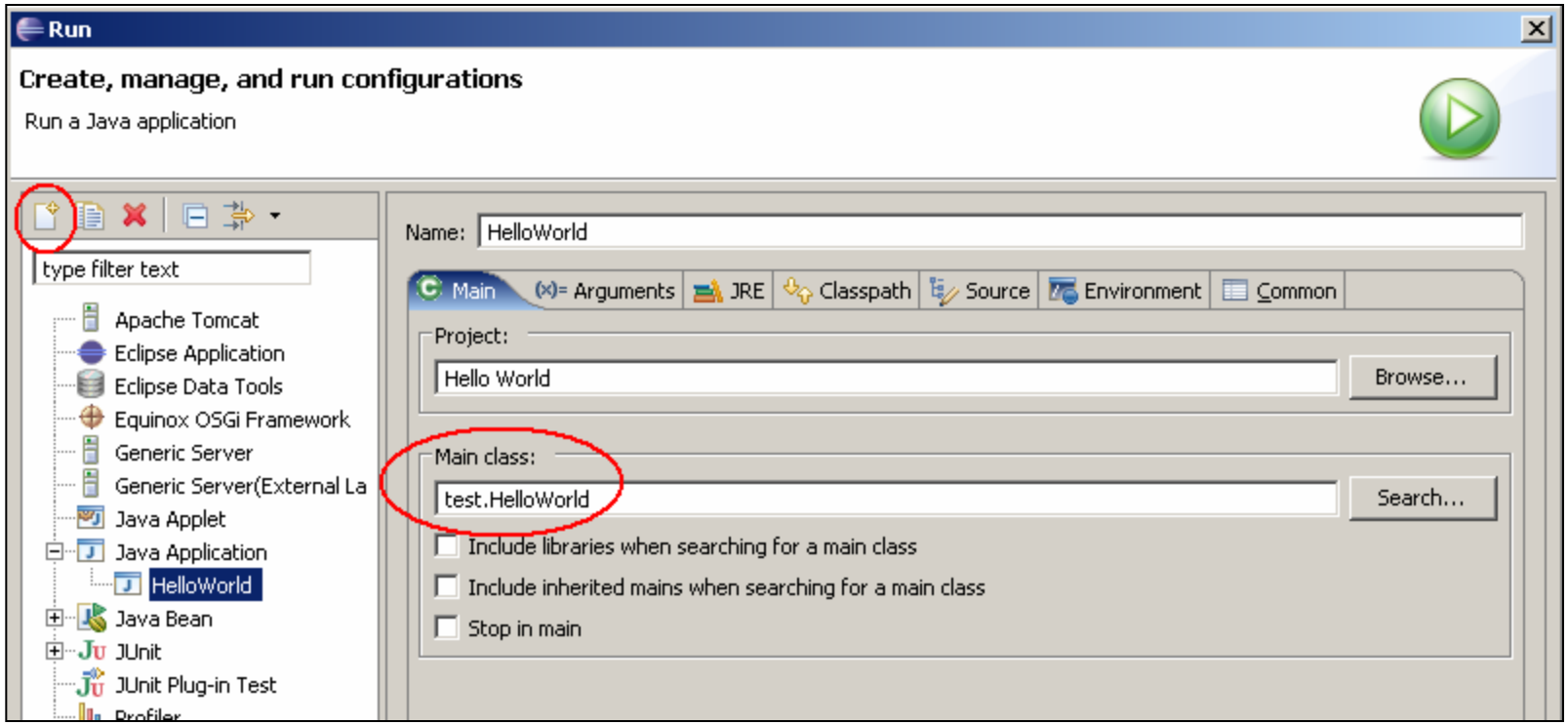


- In questo modo si avvia il Run configurations manager. Qui si crea la “configurazione” che indica all’ambiente qual è la classe principale del progetto da eseguire.



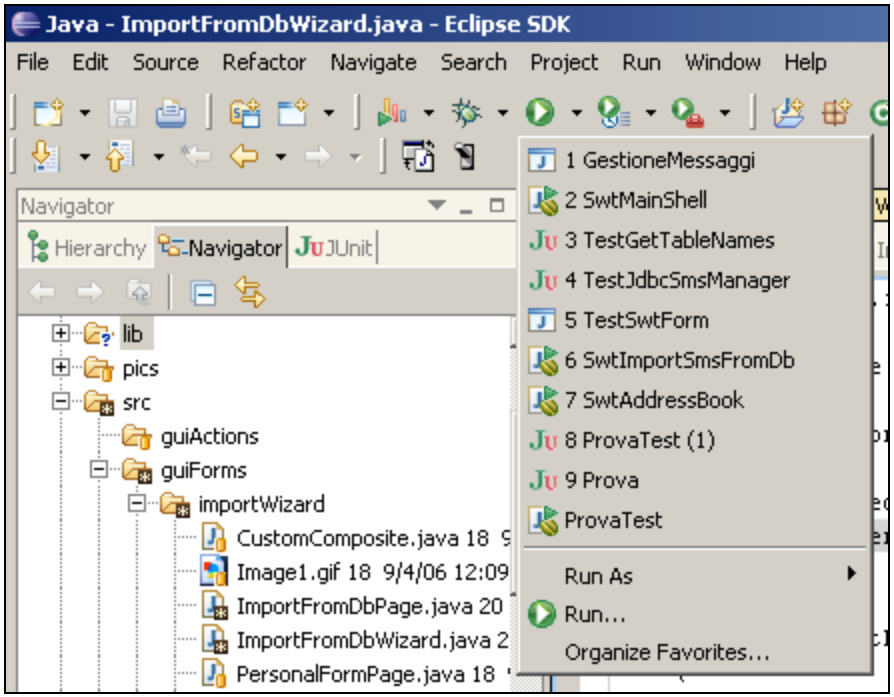
Esecuzione di un programma (2/3)

- Il manager è abbastanza “furbo” e di solito, quando si crea una nuova configurazione “Java Application”, individua automaticamente nel nostro progetto la classe con il main da eseguire.



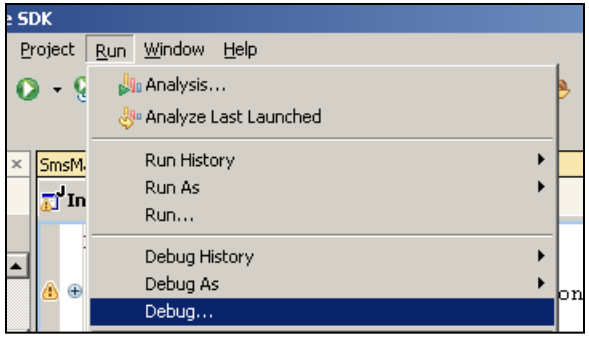
Esecuzione di un programma (3/3)

- Per rieseguire l'ultima configurazione è sufficiente premere Ctrl + F11. Le configurazioni create possono essere richiamate dal menù "Run" 

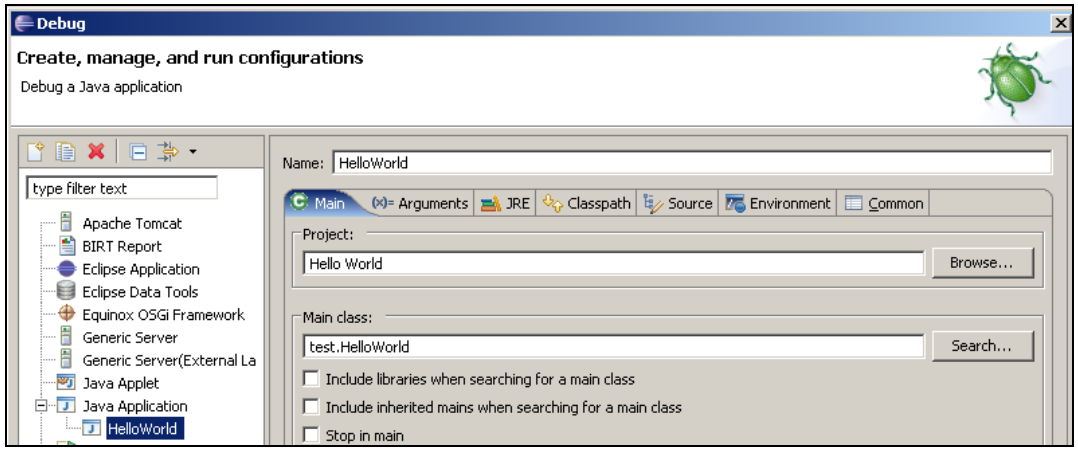


Debug di un programma (1/4)

- Non è molto diverso rispetto all'esecuzione… occorre cliccare sul menù "Run" e si scelga "Debug…"

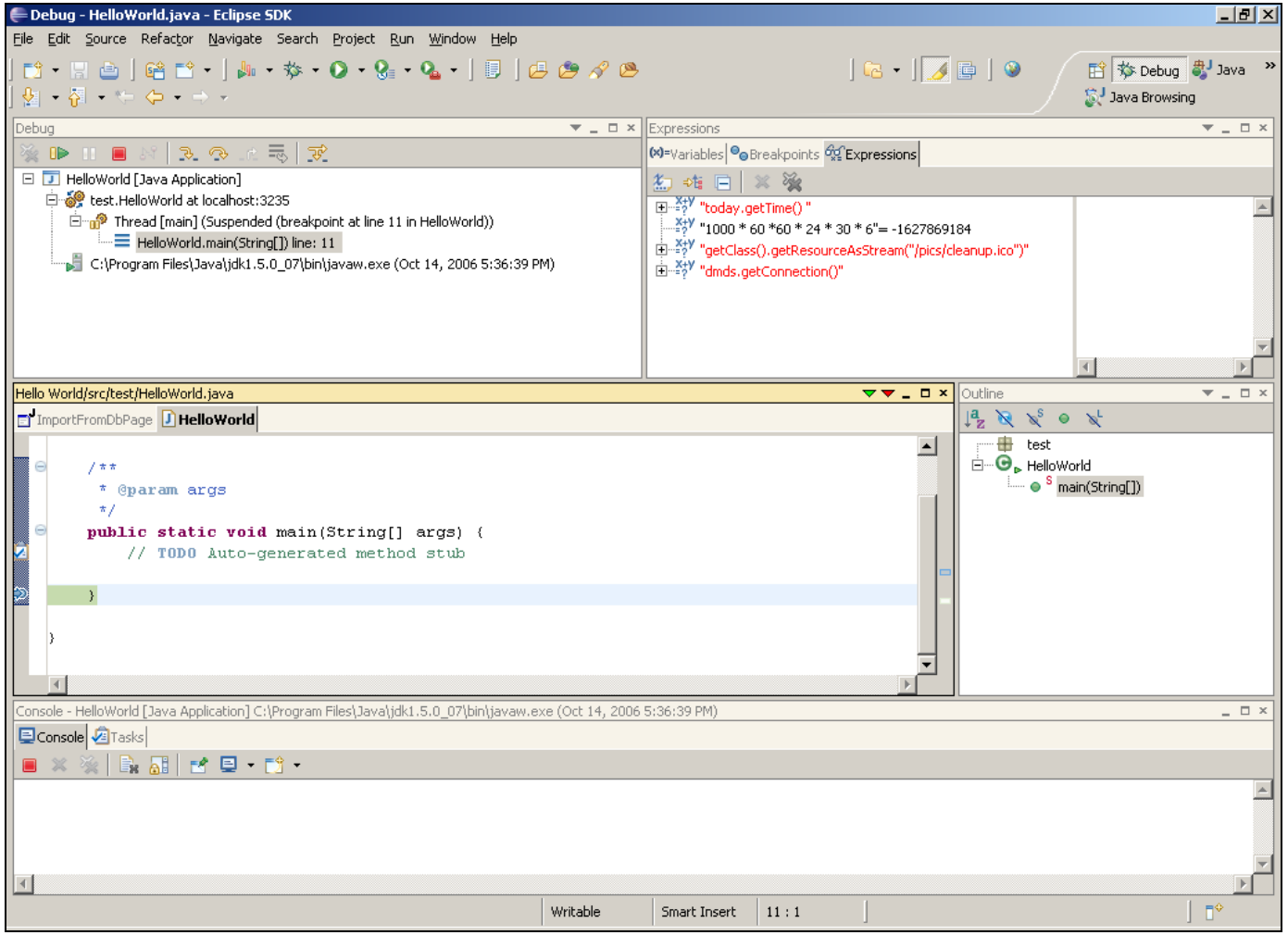


- In debug si sceglie una configurazione già esistente e si procede con il Debug.



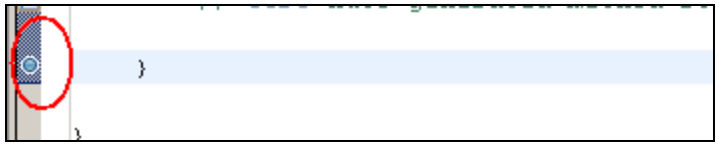
Debug di un programma (2/4)

- L'operazione di debug avviene in una perspective apposita.

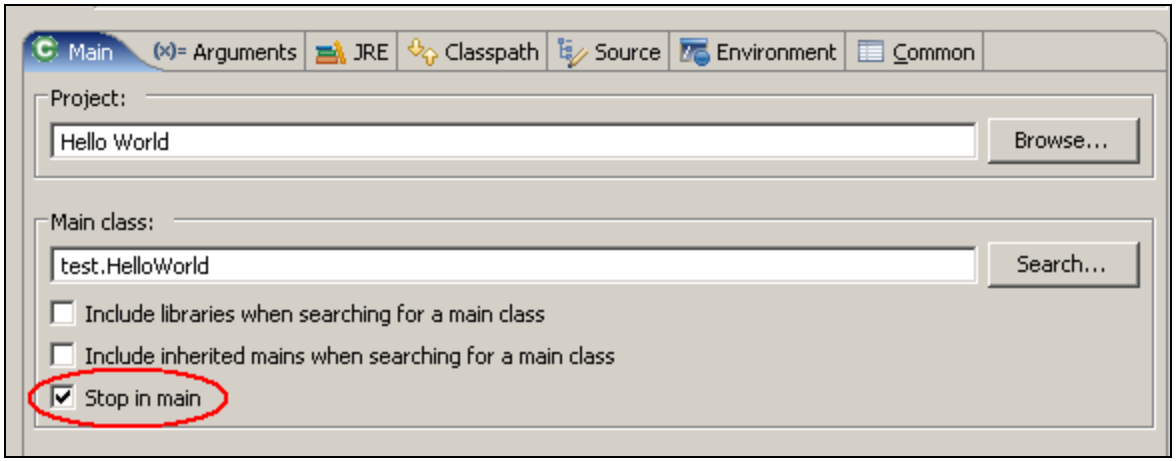


Debug di un programma (3/4)

- Come ci si può aspettare dall'editor possiamo inserire i breakpoint per il debug, basta fare doppio click a bordo riga.



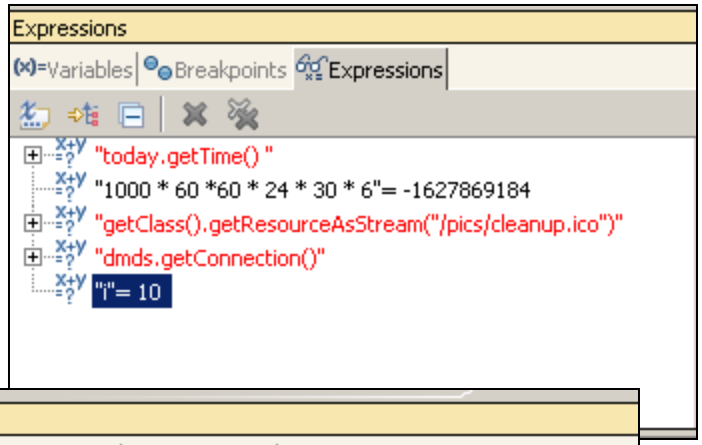
- Dal configurations manager possiamo scegliere se fermar e il debug all'inizio del metodo main.



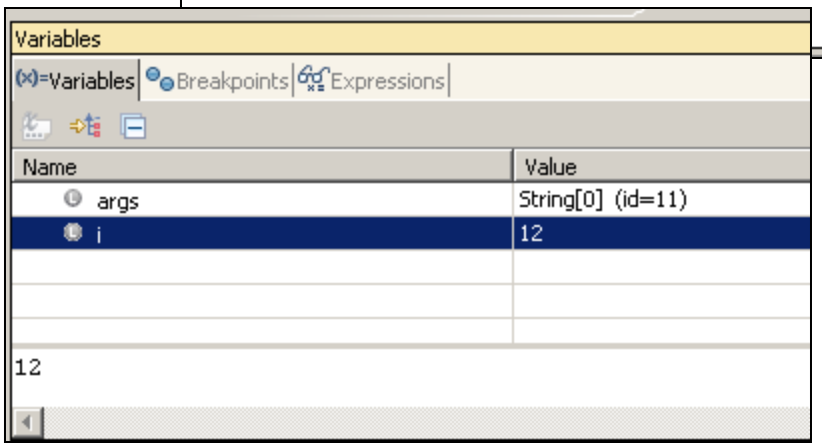
Debug di un programma (4/4)

- Dalla debug perspective possiamo vedere (e modificare volendo) i valori di variabili ed espressioni.

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int i = 10;
    System.out.println("Hello!");
}
```



- Es: nel codice ho una variabile intera chiamata i di valore 10. Posso vedere e modificare il valore
- Nella scheda “Varia - bles” ho cambiato il valore a 12.



L’editor dei sorgenti

Jug Marche

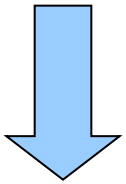
Relatore: Andrea Del Bene

Tasti di scelta rapida (1/2)

- Sono il vero punto di forza dell'ambiente e richiamano funzioni sofisticate dell'editor. Sono diverse decine ma vale la pena tenerne a mente qualcuno...
- Ctrl + Space: il più “gettonato” in assoluto. Attiva l' *intellisense in ogni punto del codice, completando i nomi delle variabili o parole chiave. Utilissimo dopo le new...*

  Esempio:

  - StringBuffer sb = new

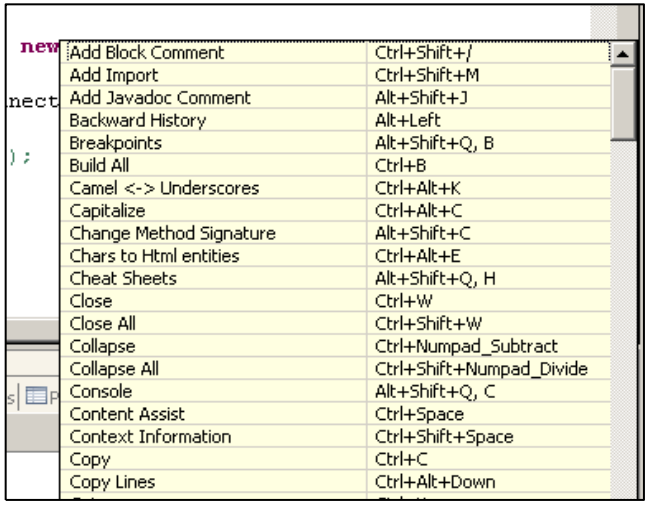


(premendo Ctrl + Space)

  - StringBuffer sb = new StringBuffer

Tasti di scelta rapida (2/2)

- Ctrl + Shift + Space: in prossimità di un metodo restituisce informazioni sui parametri di invocazione
- Ctrl + Shift + X: converte il testo selezionato in maiuscolo. Con la Y al posto della X il testo è convertito in minuscolo.
- Ctrl + Shift + L: il più importante. Richiama la lista con tutte le combinazioni dei tasti di scelta rapida:

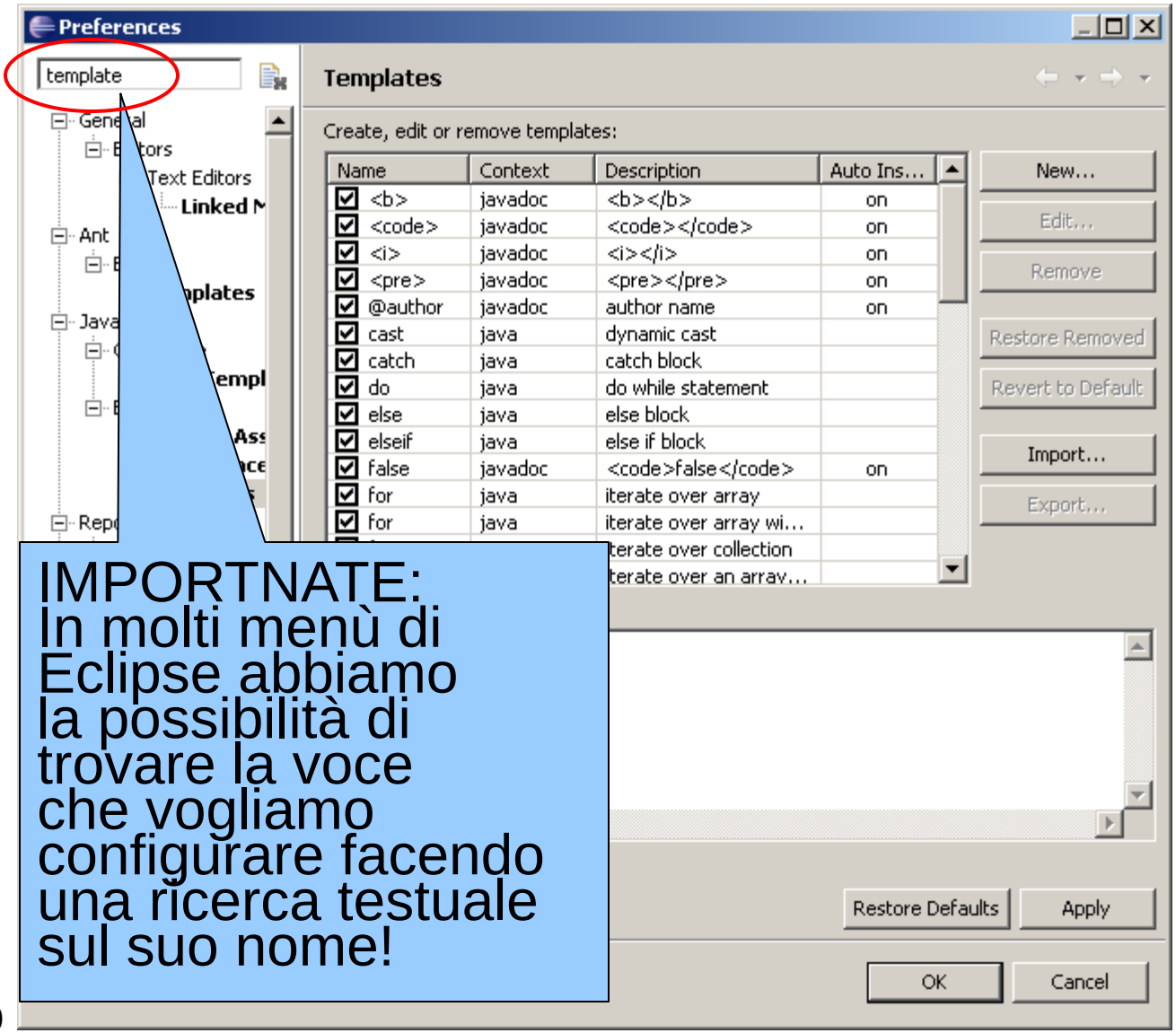


Add Block Comment	Ctrl+Shift+/
Add Import	Ctrl+Shift+M
Add Javadoc Comment	Alt+Shift+J
Backward History	Alt+Left
Breakpoints	Alt+Shift+Q, B
Build All	Ctrl+B
Camel <-> Underscores	Ctrl+Alt+K
Capitalize	Ctrl+Alt+C
Change Method Signature	Alt+Shift+C
Chars to Html entities	Ctrl+Alt+E
Cheat Sheets	Alt+Shift+Q, H
Close	Ctrl+W
Close All	Ctrl+Shift+W
Collapse	Ctrl+Numpad_Subtract
Collapse All	Ctrl+Shift+Numpad_Divide
Console	Alt+Shift+Q, C
Content Assist	Ctrl+Space
Context Information	Ctrl+Shift+Space
Copy	Ctrl+C
Copy Lines	Ctrl+Alt+Down

Template (1/2)

- I template sono modelli di codice che ricorrono nell'uso di certi costrutti sintattici di Java. Tramite I template quando questi costrutti si presentano possiamo fare in modo che Eclipse li costruisca in automatico risparmiando tempo (ed errori!).
- Es: in Eclipse se voglio usare un ciclo *for each* è sufficiente scrivere *foreach* e premere ctrl + space. Eclipse si occuperà di creare un modello del costrutto for each al quale andremo ad aggiungere solo la variabile di iterazione e la collezione di oggetti da iterare.
- Es:scrivo *if* premo ctrl + space: mi ritrovo il costrutto if pronto all'uso.
- I template si possono creare e modificare dalla voce "Window→Preference".

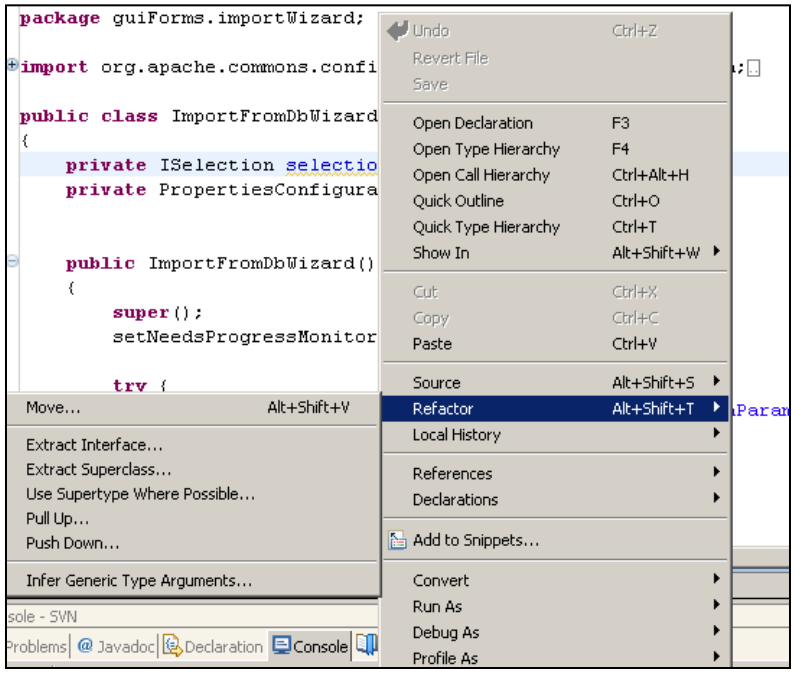
Template (2/2)



IMPORTNATE:
In molti menù di Eclipse abbiamo la possibilità di trovare la voce che vogliamo configurare facendo una ricerca testuale sul suo nome!

Refactoring del codice (1/3)

- Spiegare cos'è il refactoring in "due parole" è difficile… Da Wikipedia: "Il refactoring è il processo di riscrittura di un programma che ne migliora la struttura e la leggibilità preservandone il comportamento".
- Mediante click del tasto destro sull'editor il menù di refactor può essere raggiunto come mostrato in figura:



Refactoring del codice (2/3)

- L'esempio più classico di refactoring è cambiare nome ad un variabile di una classe. Portandosi con il cursore sulla variabile che volgiamo rinominare premiamo Alt + Shift +R:

ImportFromDbWizard | ImportFromDbPage

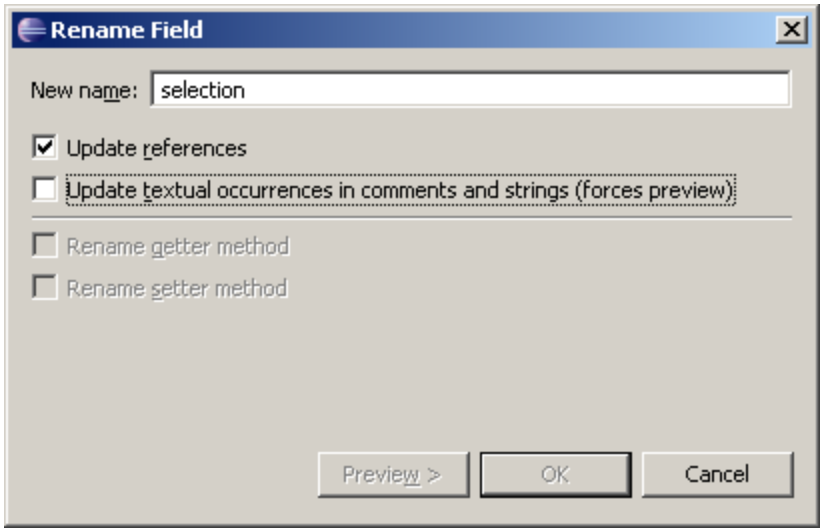
```
package guiForms.importWizard;

import org.apache.commons.configuration.ConfigurationException;

public class ImportFromDbWizard extends Wizard
{
    private ISelection selection;
    private PropertiesConfiguration propFile;
```

- Il processo di refactoring è guidato da un wizard abbastanza intuitivo…

Refactoring del codice (3/3)



- Scelto il nuovo nome possiamo vedere una preview degli efffetti che la modifica avrà sul codice. Da notare che si possono modificare (se già esistono ☺) anche i metodi di get e set dell'attributo.

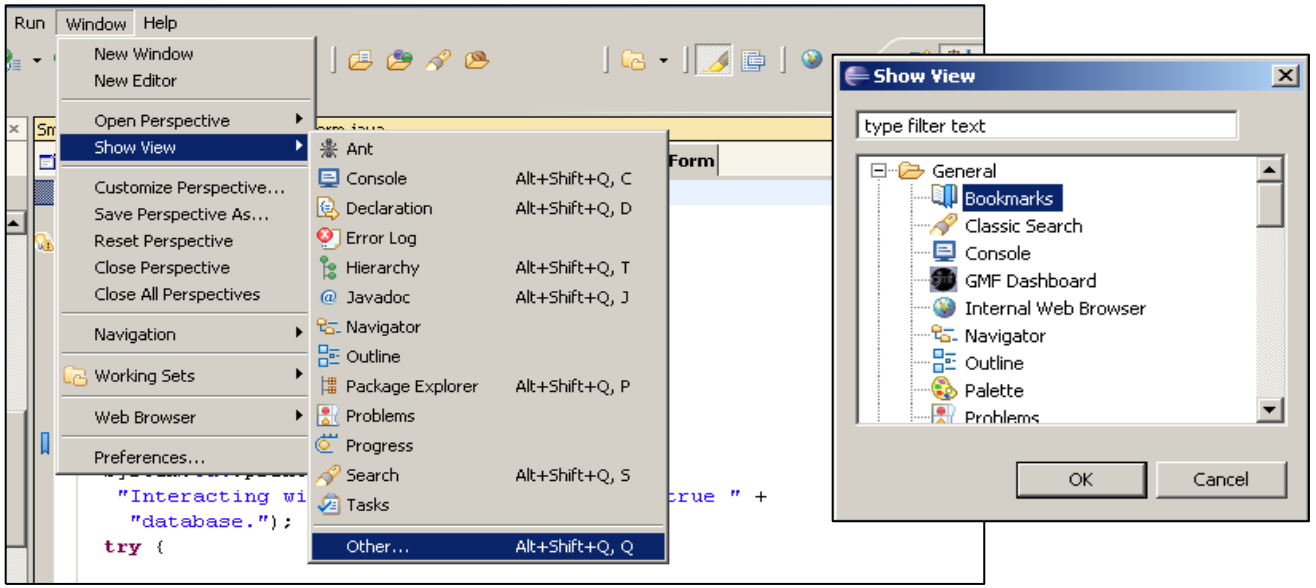
Tools di sviluppo

Jug Marche

Relatore: Andrea Del Bene

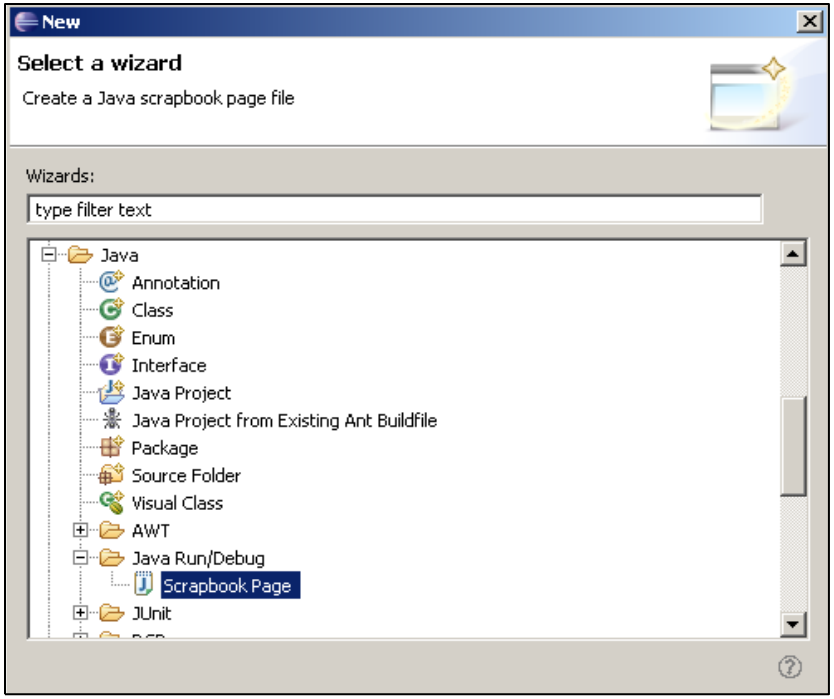
Bookmarks

- Un IDE moderno non può non avere un sistema di Bookmarks che consenta di muoversi agilmente “su e giù” tra le righe del nostro codice evidenziando i punti di maggior interesse.
- Per aggiungere la riga corrente è sufficiente andare nel menù 'Edit' e selezionare 'Add Bookmark'. Per vedere e usare i Bookmarks occorre attivare la View relativa

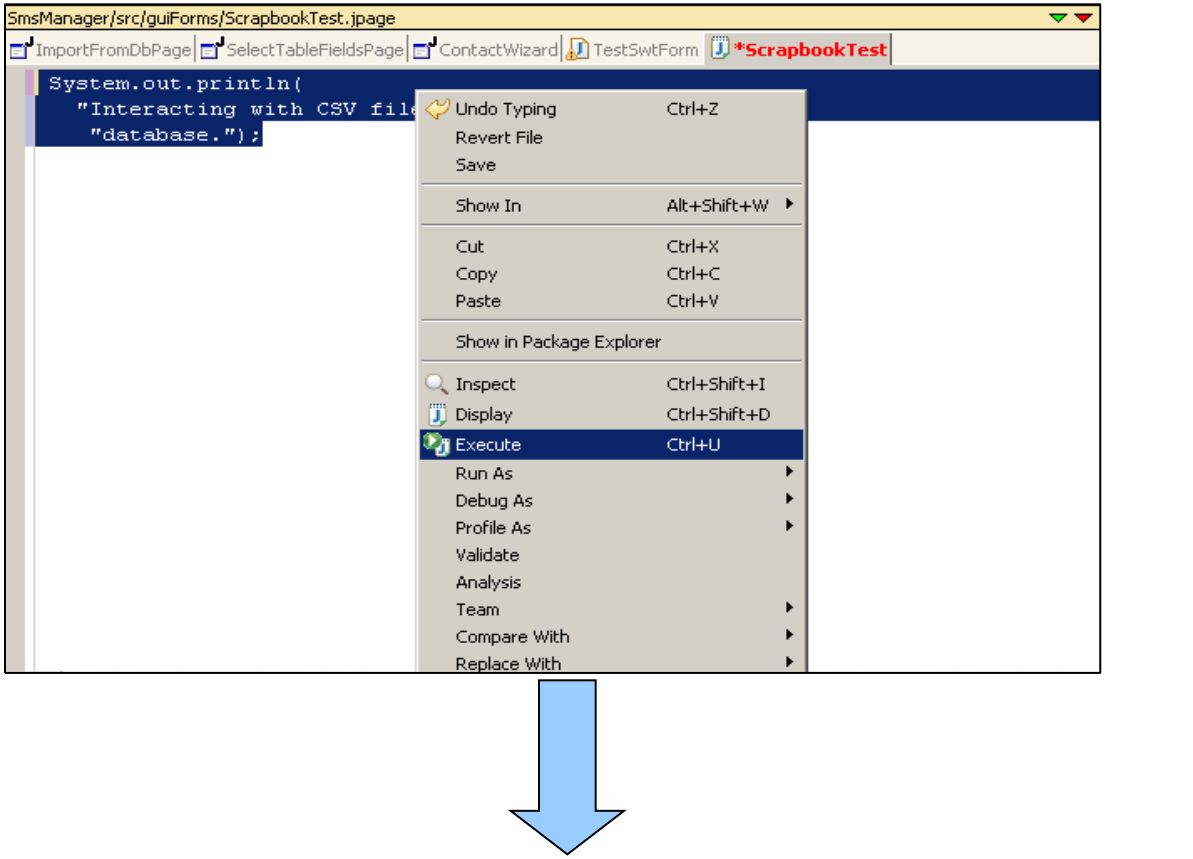


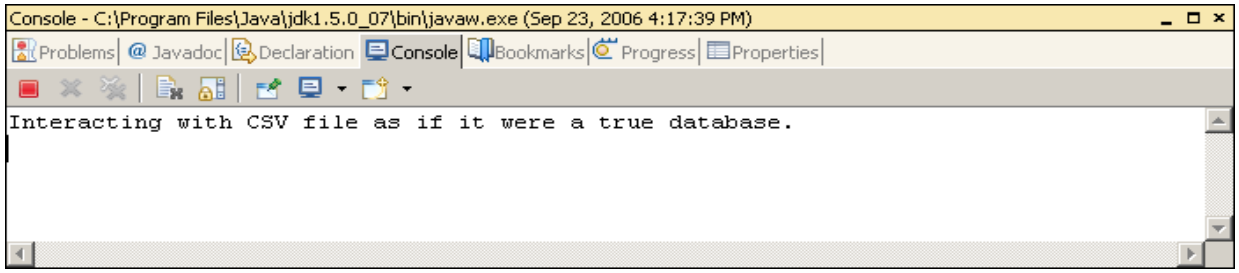
Java Scarpbook (1/2)

- Si tratta di funzione molto interessante che consente di eseguire il codice selezionato nell'editor. Uno Scarpbook si crea nel menù alla posizione 'File --> New --> Other'. La voce Scarpbook è presente alla posizione mostrata in figura.



Java Scarpbook (2/2)





Refactoring:uno sguardo al futuro (1/3).

- NOTA: ciò che segue è una libera traduzione dell'articolo in inglese all'indirizzo:

  http://www.eclipsezone.orgclipsezone.org/eclipse/forums/t78609.html

- Il refactoring è un'operazione talmente fondamentale e ricorrente che si è pensato, nell'imminente rilascio di Eclipse 3.3, di semplificarne ulteriormente l'uso. Il risultato è il nuovo *refactoring light* che rende la procedura più diretta ed interattiva.
- Temporaneamente (Eclipse 3.3 M2) il nuovo refactoring è accessibile nelle preferenze di Eclipse.

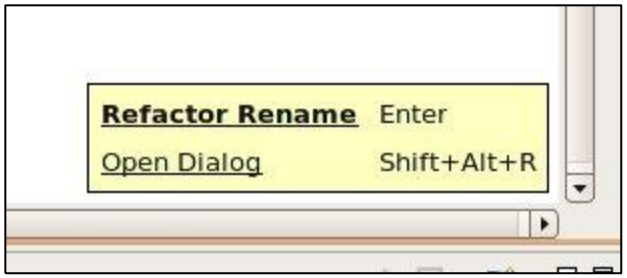
Refactoring:uno sguardo al futuro (2/3).

- Avviando il refactoring in presenza di una variabile o del nome di una classe (Alt+Shift+R) l'elemento che si sta rinominando viene incorniciato, così come ogni riferimento ad esso:

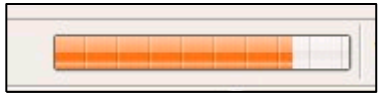
- La procedura ricorda molto l'uso dei template di codice...

Refactoring:uno sguardo al futuro (3/3).

- Durante l'esecuzione del refactoring un popup in basso a destra ci assiste descrivendo i passi da compiere.



- Premendo Enter si avvia il refactoring che coinvolge l'intero progetto corrente. Lo stato di avanzamento dell'operazione è indicato da una scrollbar.



Lavorare con CVS

Jug Marche

Relatore: Andrea Del Bene

www.jugancona.it

Cos'è il cvs?

- Il CVS è un *sistema di controllo di versione*.
- Cos'è un sistema di controllo di versione ? Wikipedia: "…???".
- La definizione è davvero troppo verbosa ed astratta…..meglio partire con un'altra domanda.

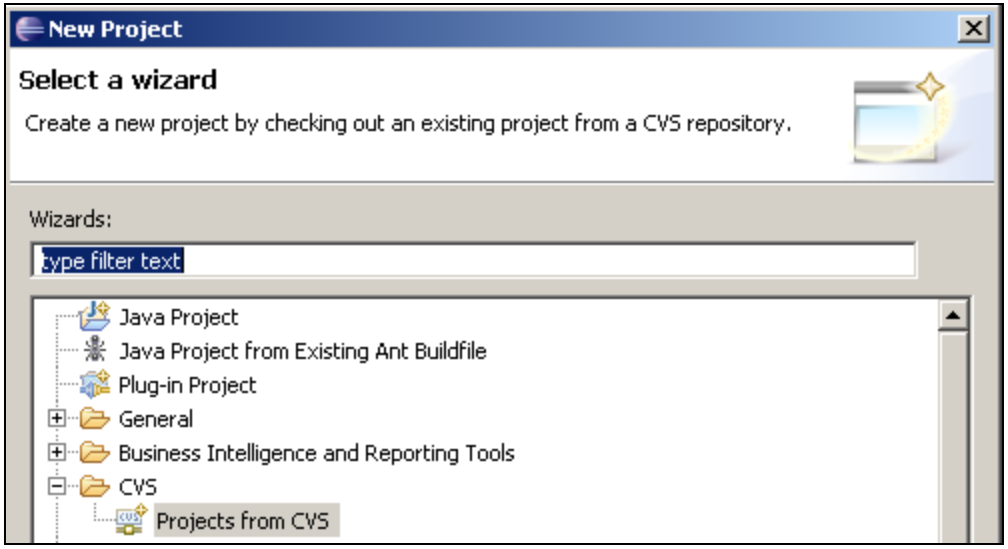
Come si fa a lavorare in gruppo sullo stesso sorgente contemporaneamente?

Cosa fa cvs

- Per lavorare in gruppo su uno stesso progetto occorre un sistema che:
  - Sia condiviso e accessibile a tutti alla stessa maniera (magari via rete…).
  - Impedisca di “pestarci i piedi” a vicenda: non voglio che due persone modifichino contemporaneamente lo stesso codice e uno salvando il lavoro cancelli le modifiche dell’altro…
  - Tenga traccia delle versioni del sorgente, delle modifiche fatte e dare la possibilità di navigare tra lo storico delle versioni. Es: che differenze ci sono tra la versione x.1 e la x.2?

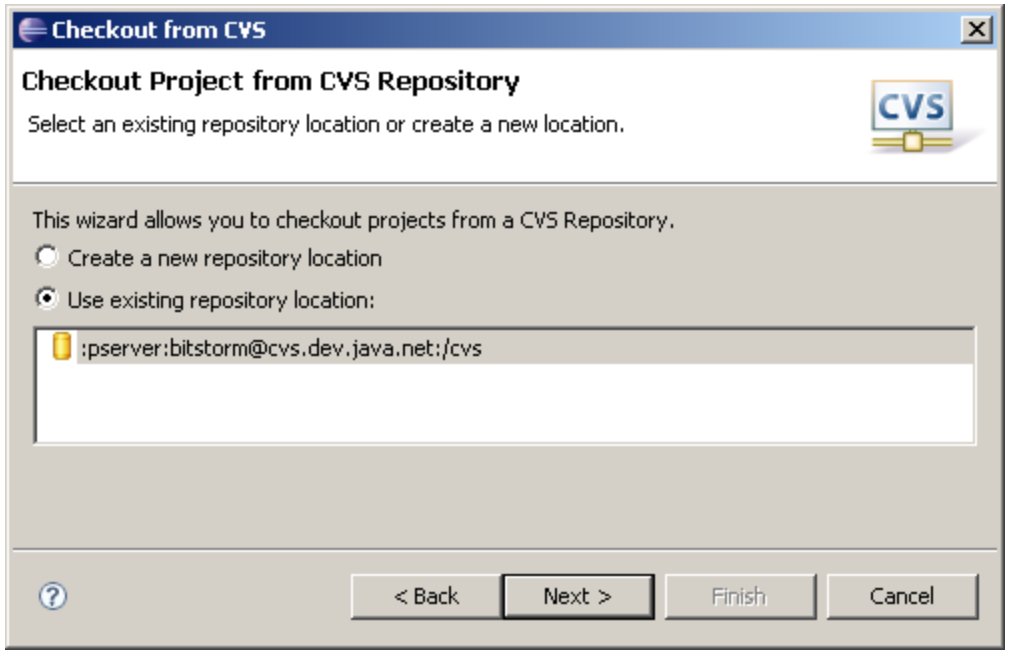
Cvs, Eclipse ed i nostri progetti

- Eclipse offre ampio supporto per lavorare su un repository CVS. Offre addirittura una perspective e una view per lavorare in gruppo con cvs. Per imparare tutto quello che ha da offrire ci vorrebbero giorni…
- Per contribuire ad un progetto su Cvs si avvi il wizard 'New project'(a menù "New→Project…") e si scelga 'Project from CVS'



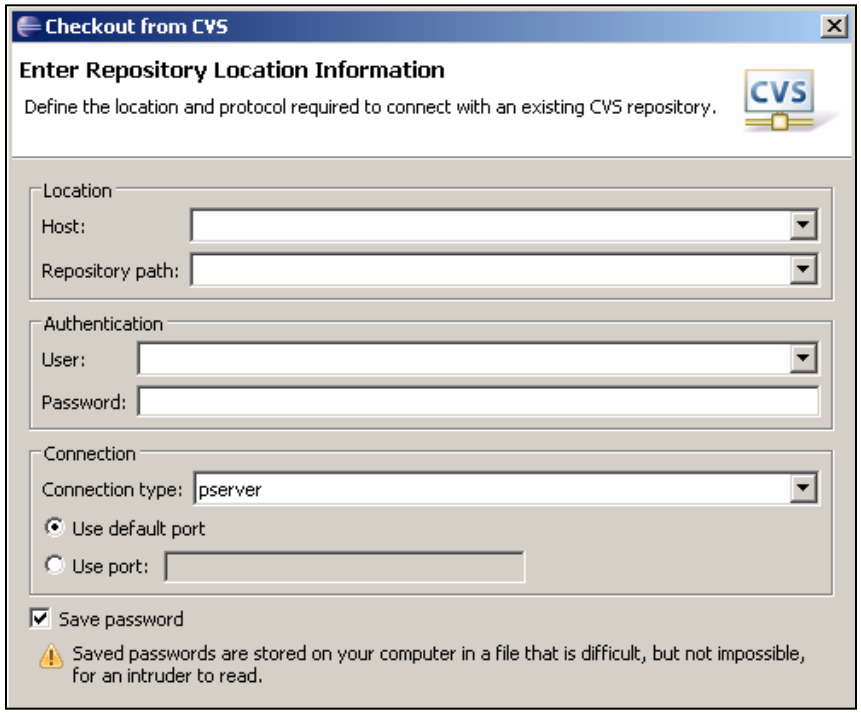
Connessione al repository

- Per connettersi al repository occorre fornire alla prima connessione I parametri per identificare il repository ed autenticarsi. Nel wizard si sceglie quindi “Create a new repository location”.



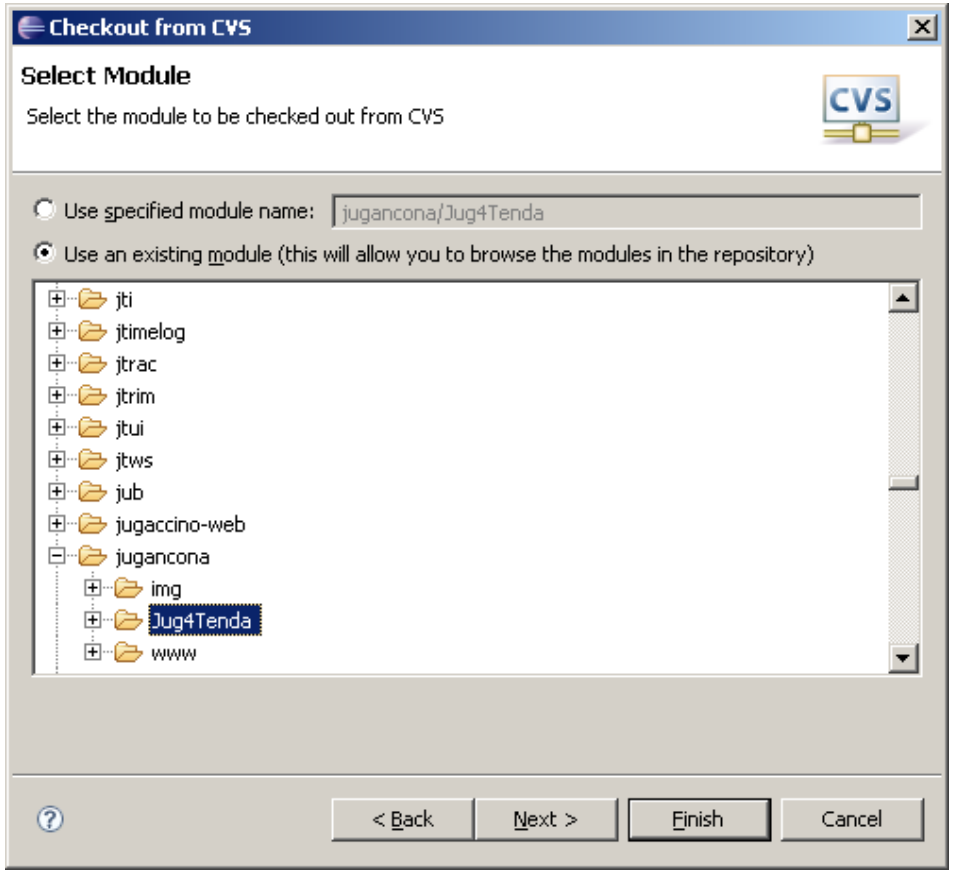
Creazione della conenssione

- I dati per la connessione sono disponibili nel sito dello Jug all'indirizzo: https://jugancona.dev.java.net/servlets/ProjectSource
- Il tipo di connessione da usare è ***pserver***.



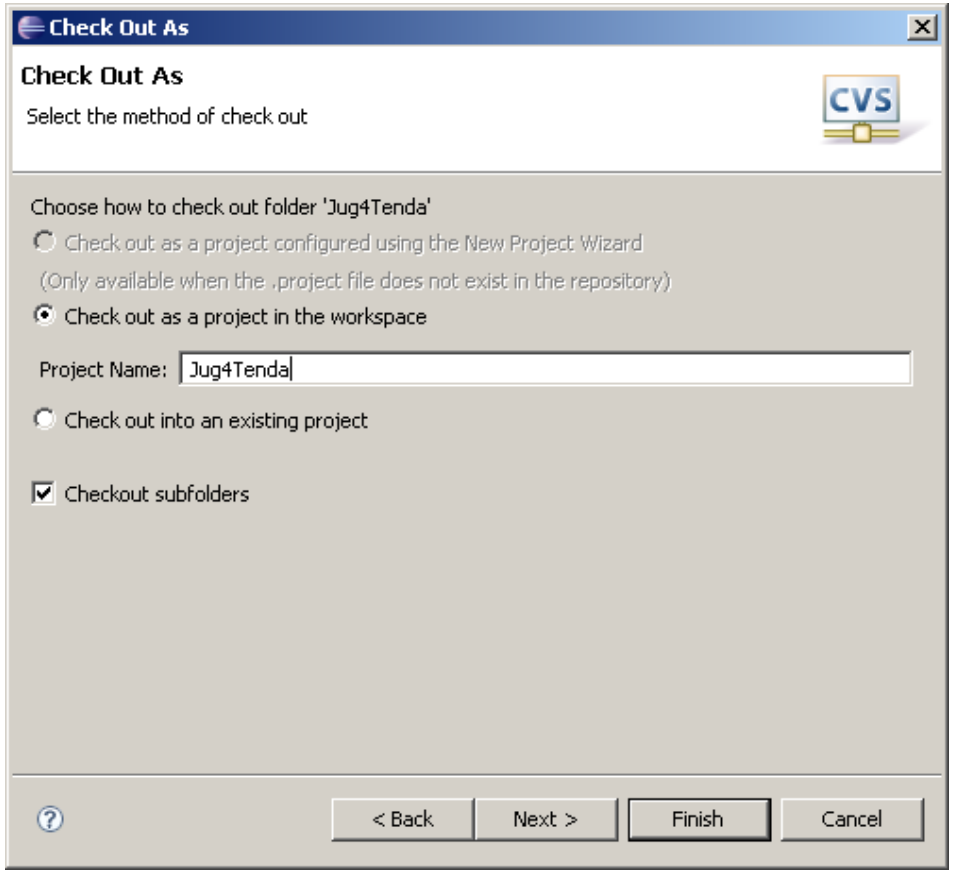
Check-out del progetto

- Se la connessione è andata a buon fine sarà possibile vedere tutti I progetti ospitati dal sito.Supponiamo di volere prelevare il progetto *Jug4Tenda* sotto la cartella *jugancona.*



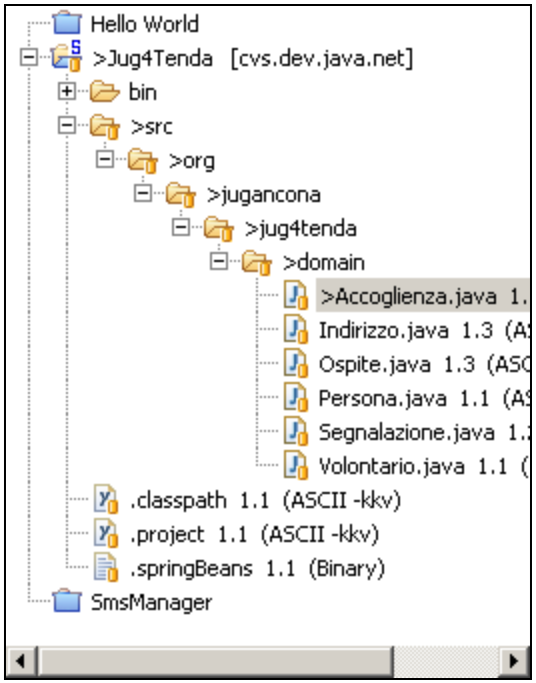
Dare un nome al progetto

- L'ultima schermata ci chiede di immettere un nome per il progetto. Per default viene proposto quello del repository (e di solito va benissimo!).



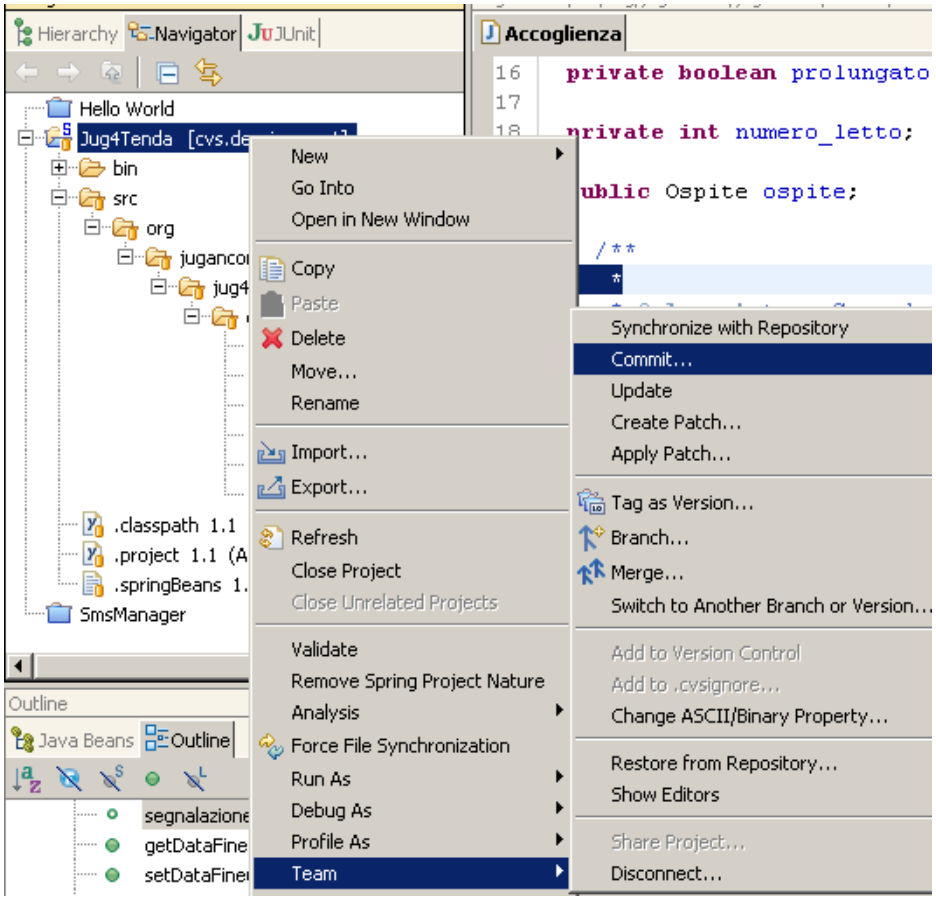
Lavorare sul nuovo progetto

- Il progetto importato e I suoi file saranno affiancati da un cilindro arancione, il simbolo del repository. Quando si modifica il sorgente appare un simbolo di maggiore > a sinistra della risorsa modificata (nella figura questo vale per il file accoglienza.java).



Salvare le modifiche sul repository

- Una volta soddisfatti delle proprie modifiche si deve fare la *commit* del progetto.
- Es: tasto destro sul progetto, menù team, voce Commit… Le modifiche verranno salvate sul repository remoto e saranno visibili a tutti.



Fine!

- Questa è solo la punta dell'iceberg!!!
- Sul discorso cvs e repository vi consiglio di leggere l'ottimo tutorial raggiungibile all'indirizzo:

  https://eclipse-tutorial.dev.java.net/eclipse-tutorial/part2.html