

# ¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.  
 Ese apoyo que siempre quiso tener...

## 1. Desarrollo de componentes y proyectos a medida



## 2. Auditoría de código y recomendaciones de mejora

## 3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



## 4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,  
 HTML5, CSS3, JavaScript-jQuery

Control de autenticación y  
 acceso (Spring Security)  
 UDDI

JPA-Hibernate, MyBatis  
 Motor de búsqueda empresarial (Solr)  
 ETL (Talend)

Gestor portales (Liferay)  
 Gestor de contenidos (Alfresco)  
 Aplicaciones híbridas

Web Services  
 Rest Services  
 Social SSO  
 SSO (Cas)

Dirección de Proyectos Informáticos.  
 Metodologías ágiles  
 Patrones de diseño  
 TDD

Tareas programadas (Quartz)  
 Gestor documental (Alfresco)  
 Inversión de control (Spring)

BPM (jBPM o Bonita)  
 Generación de informes (JasperReport)  
 ESB (Open ESB)



E-mail

Contraseña

Entrar

[Regístrate](#)  
[Olvidé mi contraseña](#)
» Estás en: [Inicio](#) [Tutoriales](#) Experimenta con tu código en Eclipse utilizando Scrapbooks

Alberto Moratilla Ocaña

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero en Informática, especialidad en Ingeniería del Software. DEA en doctorado en Información, Documentación y Conocimiento. MBA en empresas de Nuevas Tecnologías. CQF.

Puedes encontrarme en [Autentia](#): Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

[Ver todos los tutoriales del autor](#)

Fecha de publicación del tutorial: 2015-02-19

Tutorial visitado 6 veces [Descargar en PDF](#)

## Experimenta con tu código en Eclipse utilizando Scrapbooks

### 1. Introducción

Seguro que más de una vez te has visto en la necesidad de probar o experimentar con algún fragmento de código a la hora de programar. Suele pasar, por ejemplo, con las cadenas de texto: sabes lo que quieres hacer, pero no sabes si ese método que aparece en el autocompletar con un nombre que te gusta hará lo que crees que su nombre sugiere. O también con unos algoritmos aparentemente básicos pero que no recordamos cómo afrontar.

Cada uno tenemos nuestra forma de afrontar estas situaciones. A mí se me suelen pasar por la cabeza:

- Programarlo e irlo probando con el programa completo, lo que es lento.
- Programarlo y recurrir a un test (si está hecho antes mejor -TDD-) para probar sólo esa funcionalidad.
- Crear la típica clase "Borrame.java" con un "main" para probar ahí sólo el código.
- Hacer un "main" en la clase en la que estamos y programar ahí nuestro experimento.
- Hacer un script con Groovy usando sintaxis de Java, pero la facilidad de este lenguaje interpretado para acelerar las pruebas

...y seguro que tú también tienes tu forma casera de probar los fragmentos de código sin tener que levantar la aplicación completa...

Para resolver esta problemática, el entorno de desarrollo Eclipse nos brinda la funcionalidad "ScrapBook", que nos permite probar fragmentos de código de forma aislada para que luego podamos incorporarlos a nuestros programas completos.

### 2. Entorno

Para realizar este tutorial se ha empleado el siguiente entorno de desarrollo:

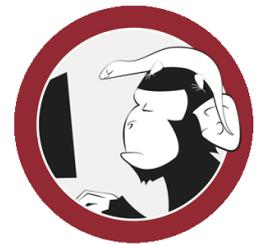
- **Hardware:** Mac Book Pro 15" Intel Core i7 2,8 GHz, 16 GB RAM.
- **Sistema Operativo:** Mac OS X Yosemite.
- Eclipse Kepler SR2 (aunque vale cualquier Eclipse).

### 3. Creación y uso de un ScrapBook

Para poder utilizar los ScrapBook de Eclipse deberemos seguir los pasos que se indican a continuación

1. En primer lugar deberemos estar en un proyecto Java en Eclipse.
2. Vamos a File>New>Other. En el Wizard deberemos buscar "Scrapbook Page". Podemos escribir en la caja de texto si no queremos buscar por el árbol de posibilidades.

### Catálogo de servicios Autentia



### Síguenos a través de:



### Últimas Noticias

» 2015: ¡Volvemos a la oficina!

» [Curso JBoss de Red Hat](#)

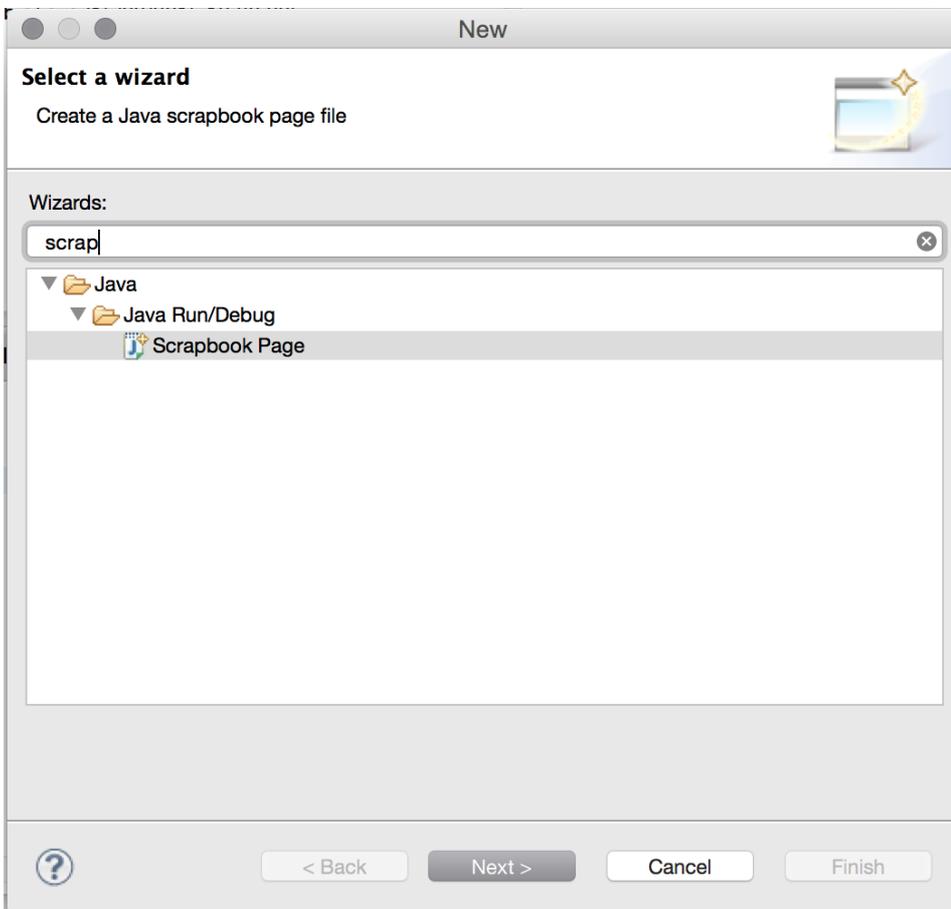
» Si eres el responsable o líder técnico, considérate desafortunado. No puedes culpar a nadie por ser gris

» Portales, gestores de contenidos documentales y desarrollos a medida

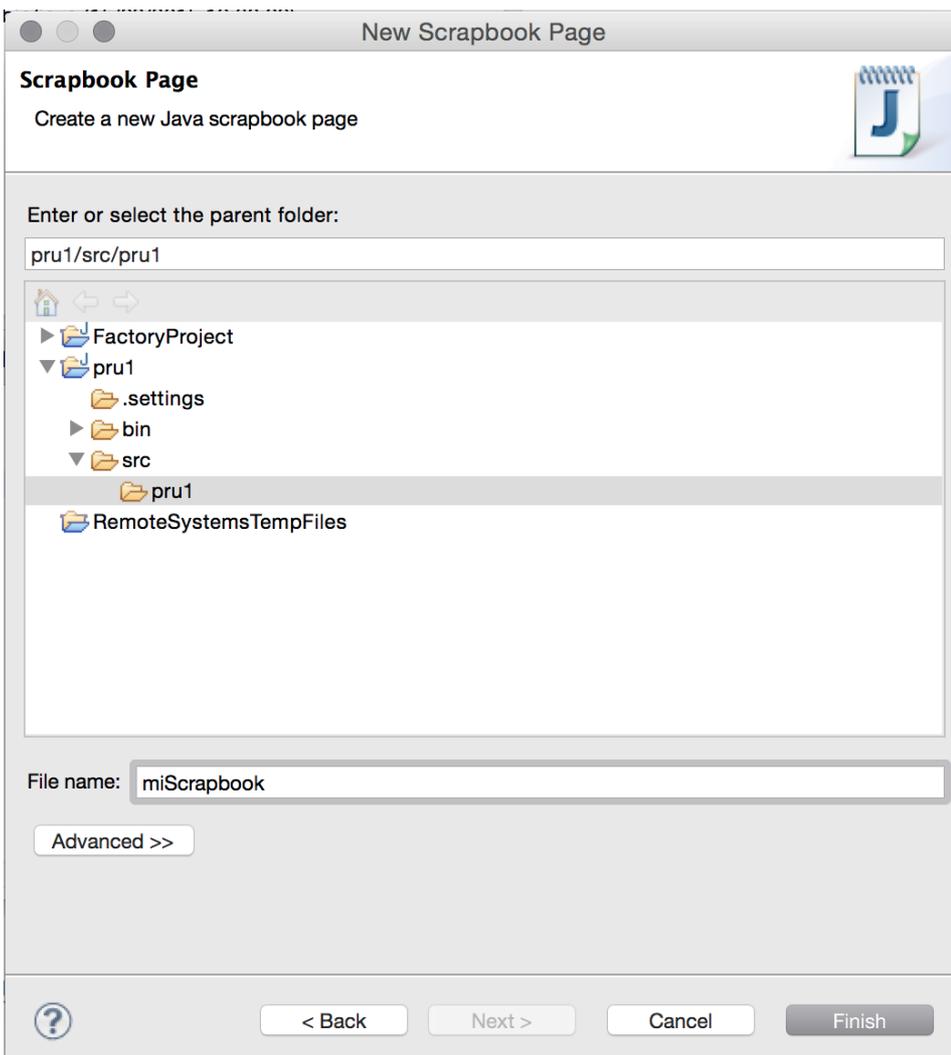
» [Comentando el libro Start-up Nation, La historia del milagro económico de Israel, de Dan Senor & Salu Singer](#)[Histórico de noticias](#)

### Últimos Tutoriales

» [Curso de WatchKit ¡ahora sólo 9 dólares!](#)» [Cómo implementar una nube de etiquetas con D3.js](#)» [Test de servicios REST con Spring MVC y Spring Test](#)» [Introducción a React](#)» [Tutorial Apple Watch](#)



3. Elegimos el nombre del fichero de Scrapbook que queremos utilizar y la ubicación de este. Es posible que quieras en un futuro incluir en la lista de ficheros ignorados de Git (u otro control de versión), ya que no debería formar parte del producto final.



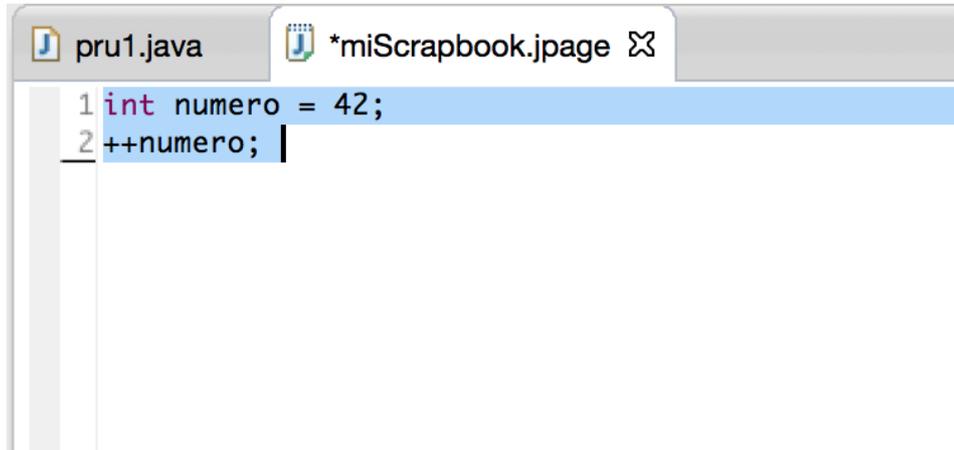
El resultado que obtendremos será un fichero llamado miScrapbok.jpape (o el nombre que hayamos utilizado .jpape). Ya podremos comenzar a escribir nuestro código de prueba.

Comenzaremos con algo sencillo para ver cómo ejecutarlo. Escribiremos:

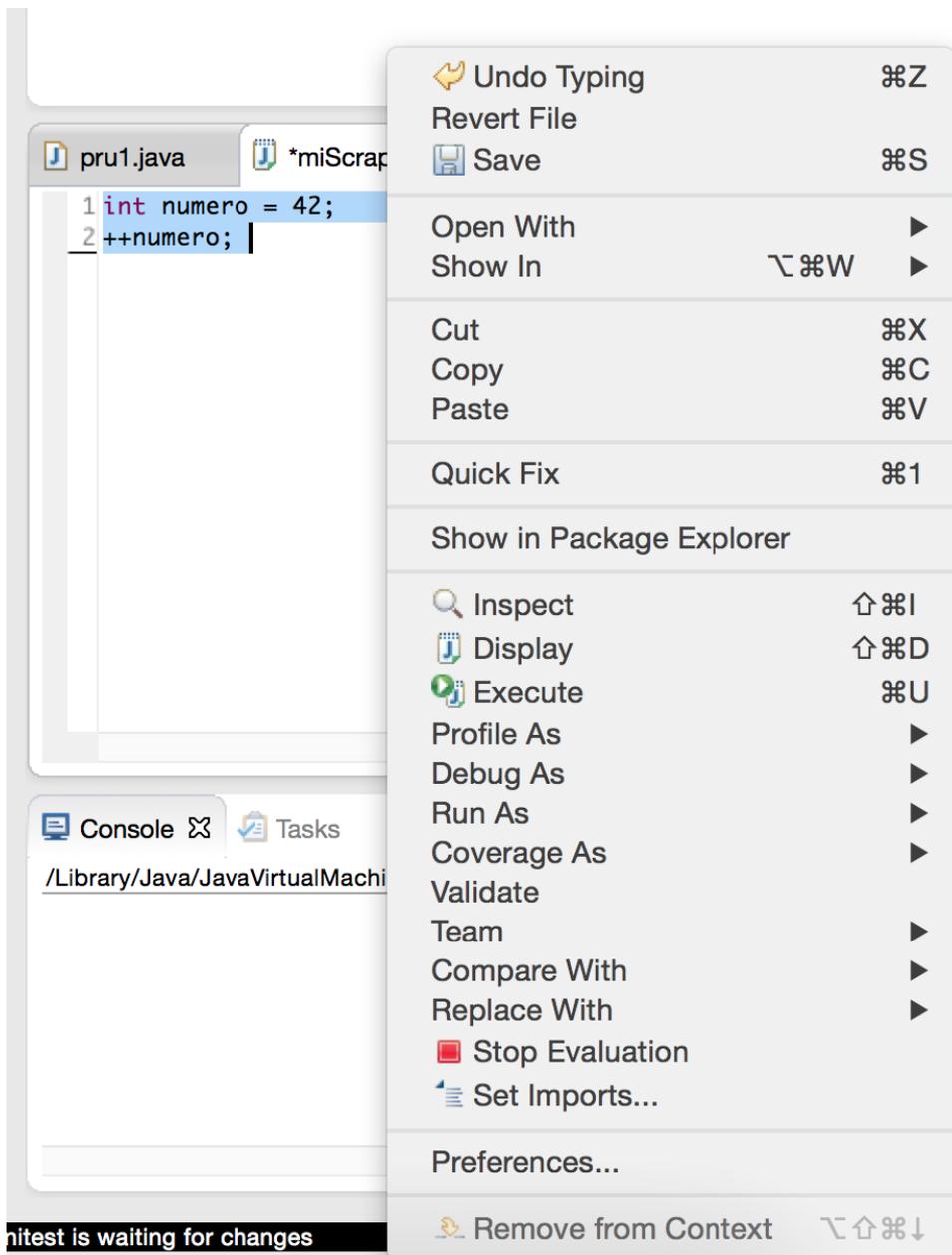
```
int numero = 42; numero++;
```

¿Cómo ejecutamos este código? Es algo diferente a cómo lo hacemos para una clase normal de Java:

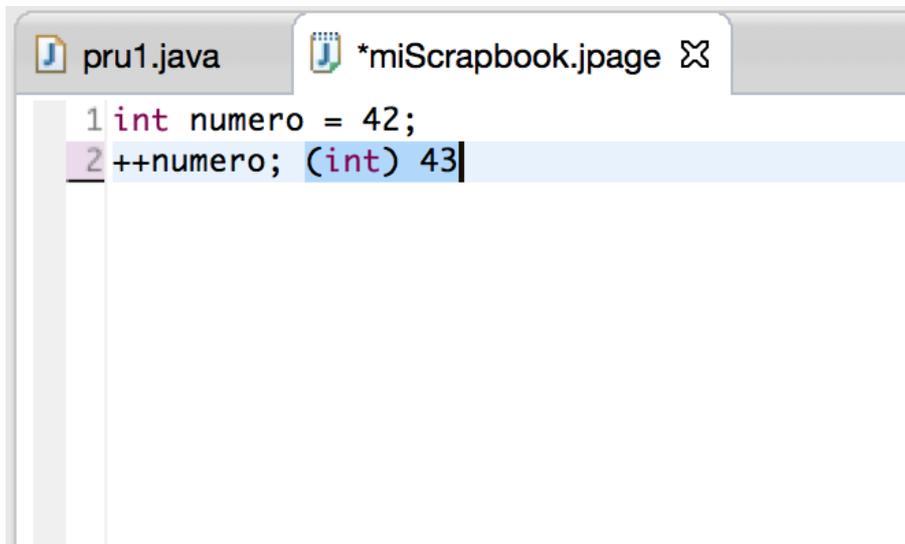
1. Seleccionamos el código que queremos ejecutar (Comando+a para seleccionar todo - en Mac, en tu SO será otro-).



2. Pulsamos con el botón secundario del ratón para obtener el menú secundario. Ahí podremos ver las opciones para ejecutar el código



3. Pulsamos sobre la opción "Display", y veremos el resultado a continuación de la última instrucciones. Como puedes imaginar, el resultado, (int) 43 nos indica el número obtenido y el tipo. Aparece seleccionado por si queremos borrarlo o copiarlo.

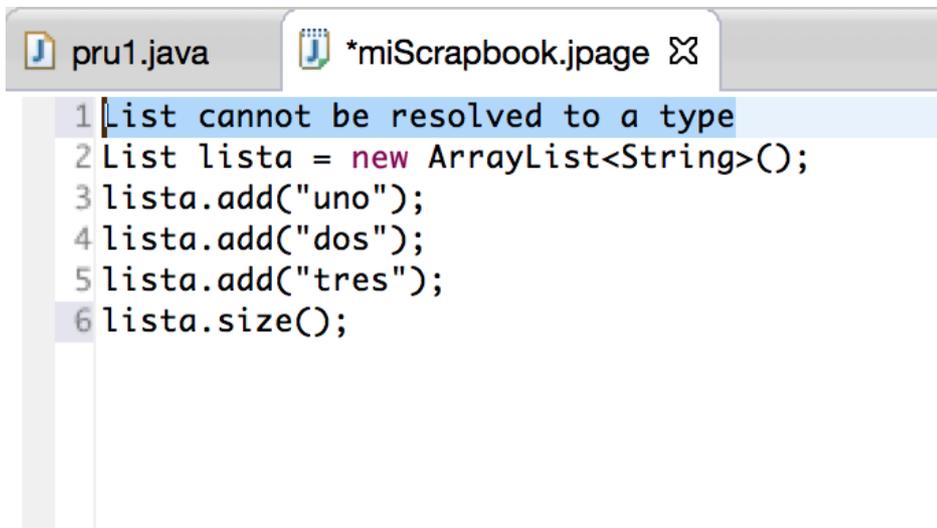


```
pru1.java *miScrapbook.jpape ✕  
1 int numero = 42;  
2 ++numero; (int) 43
```

Vamos a ver ahora un ejemplo algo más complejo que implique el uso de clases. Vamos a contar los elementos de un ArrayList

```
1 List lista = new ArrayList<String>();  
2 lista.add("uno");  
3 lista.add("dos");  
4 lista.add("tres");  
5 lista.size();
```

Si seguimos los pasos del primer ejemplo, veremos que al ejecutar el código seleccionado (todo) con el comando Display obtenemos un error:



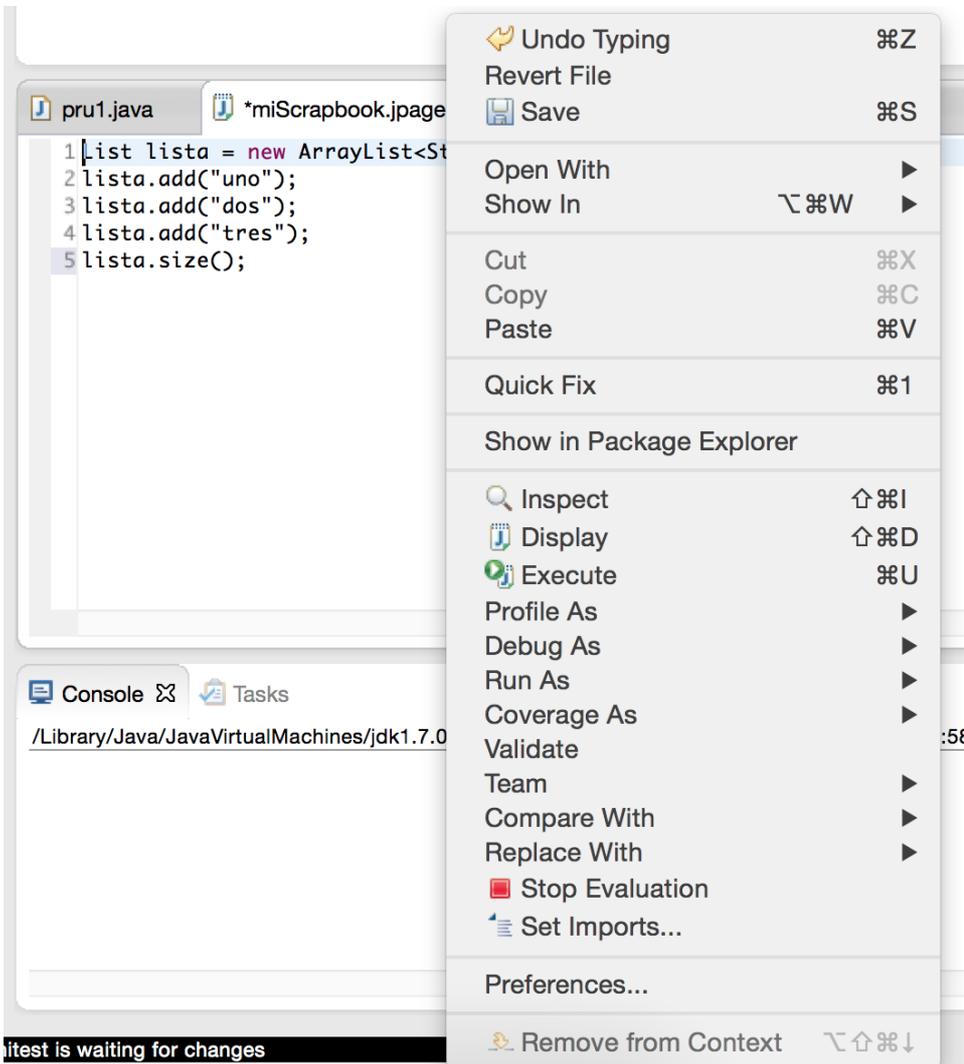
```
pru1.java *miScrapbook.jpape ✕  
1 List cannot be resolved to a type  
2 List lista = new ArrayList<String>();  
3 lista.add("uno");  
4 lista.add("dos");  
5 lista.add("tres");  
6 lista.size();
```

Puedes imaginar de dónde viene el problema: tenemos que tener en cuenta que en los Scrapbook no funcionan las instrucciones de tipo import de una clase normal de Java, y por tanto el error proviene de que no reconoce el tipo List (ni ArrayList).

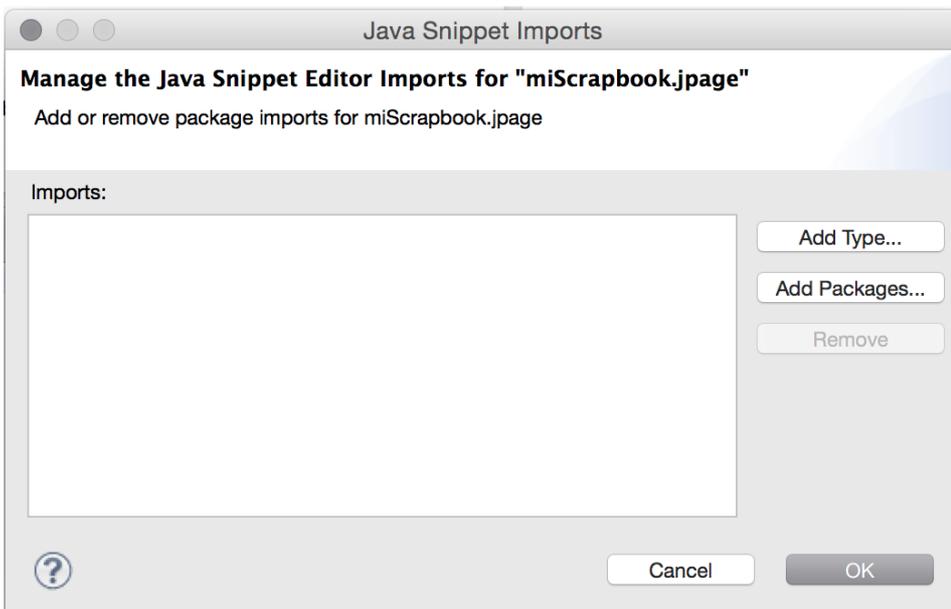
Esto no quiere decir que no podamos utilizar las librerías necesarias. Para ello debemos configurar el entorno mediante la opción "Set Imports" que aparece en el menú contextual. Ahí podremos establecer los imports necesarios, tanto de librerías de Java del JDK, como de cualquiera que estemos empleando en el proyecto, incluyendo claro está, nuestras clases.

Vamos a ver cómo incluir StringBuffer para poder utilizarlo en el ScrapBook:

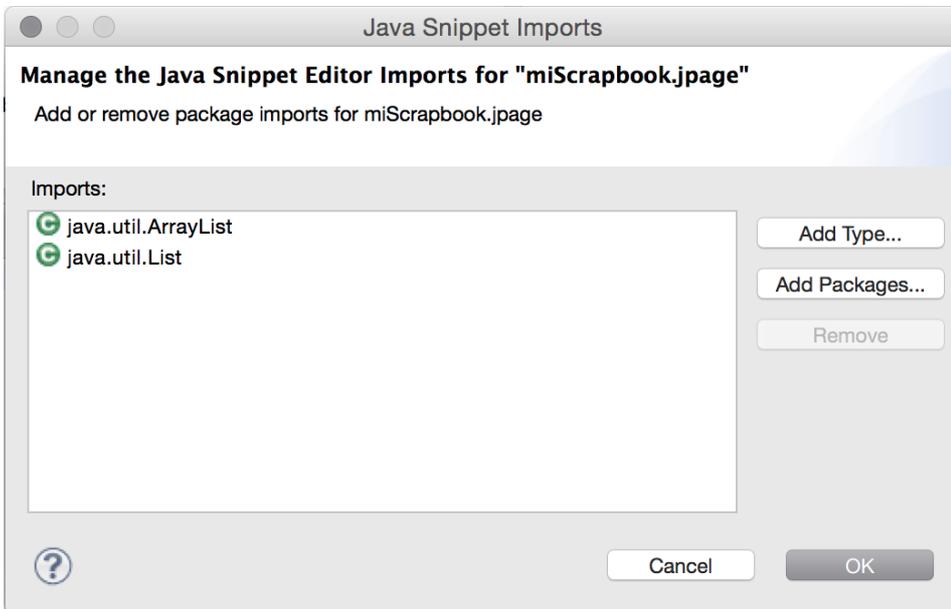
1. Pulsamos con el botón secundario sobre el Scrapbook para obtener el menú contextual.
2. Seleccionamos la opción "Set Imports"



3. En la ventana que aparece, gestionaremos las dependencias, incluyendo o borrando las librerías que queramos, pulsando Add Type o Add Packages. Simplemente incluimos los tipos List y ArrayList:



Aquí podemos ver el resultado



Ahora dentro del Scrapbook, Eclipse sabrá a qué nos referimos. Por supuesto, como en la ventana de edición de clases Java de Eclipse, permite completar código pulsando Control+Espacio (o poniendo el ".") de modo que aparecerá el clásico menú contextual para elegir método o código que podemos poner.

Volvemos a ejecutar el código anterior seleccionándolo y pulsando sobre "Display":



Podemos ver el resultado, que es de tipo entero y es 3, tal y como cabe esperar de las instrucciones que hemos puesto en el Scrapbook: un Objeto de tipo List que se ha completado con 3 elementos y se le ha pedido el resultado.

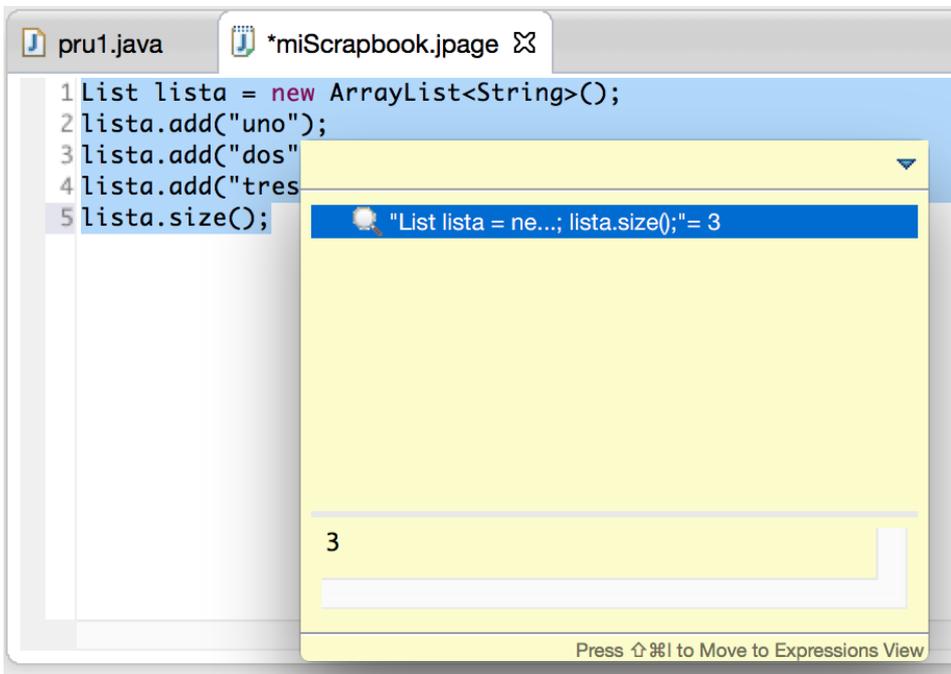
#### 4. Otros modos de ejecución

Hasta ahora solo hemos visto cómo ejecutar con la opción "Display" nuestro código. Podemos utilizar otros dos modos: "Inspect" y "Execute". Vamos a ver las características de cada uno basándonos en el último ejemplo.

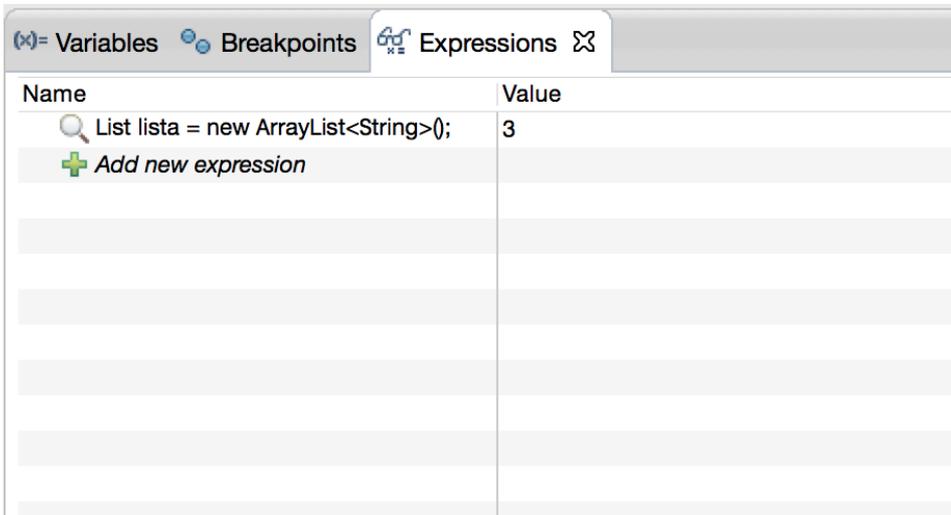
##### Opción 1: usando Inspect

En este modo de ejecución podremos ver el resultado mediante el popup de inspección que aparece cuando nos situamos sobre una variable cuando estamos haciendo debug con Eclipse de una clase normal de Java.

De este modo podremos inspeccionar no solo el resultado, sino las variables que componen el código.



Podemos analizar la información con más detenimiento si llevamos estos datos a la ventana "Expressions View", pulsando la combinación de teclas que nos indica en la parte inferior de la ventana que aparece. En el caso de Mac es: Shift+Comando+I



### Opción 2: Usando Display

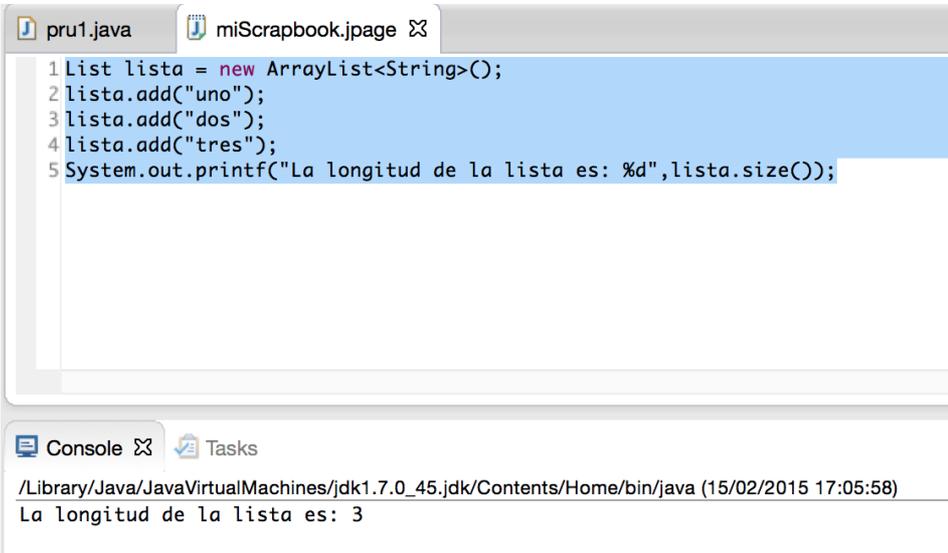
Es el modo que ya hemos visto en los ejemplos iniciales. El resultado es una cadena de texto con el tipo y el valor del resultado de las líneas seleccionadas. En esta ocasión solamente vamos a seleccionar las dos primeras líneas del ejemplo de la lista: la definición y la asignación del primer elemento. Vemos que el resultado se pinta justo detrás del último carácter seleccionado:



Más adelante veremos qué sucede con los errores en la ejecución.

### Opción 3: Ejecutando la expresión

Es el modo normal en el que lo ejecutaríamos en una clase de Java con Eclipse. Es decir, no obtendremos ni la ventana de Inspección ni se escribirá el mensaje con el resultado a partir del lugar en el que esté el cursor como con Display. Para conocer el resultado deberemos hacer un print a la consola por ejemplo.



pru1.java miScrapbook.jpape

```

1 List lista = new ArrayList<String>();
2 lista.add("uno");
3 lista.add("dos");
4 lista.add("tres");
5 System.out.printf("La longitud de la lista es: %d", lista.size());

```

Console Tasks

/Library/Java/JavaVirtualMachines/jdk1.7.0\_45.jdk/Contents/Home/bin/java (15/02/2015 17:05:58)

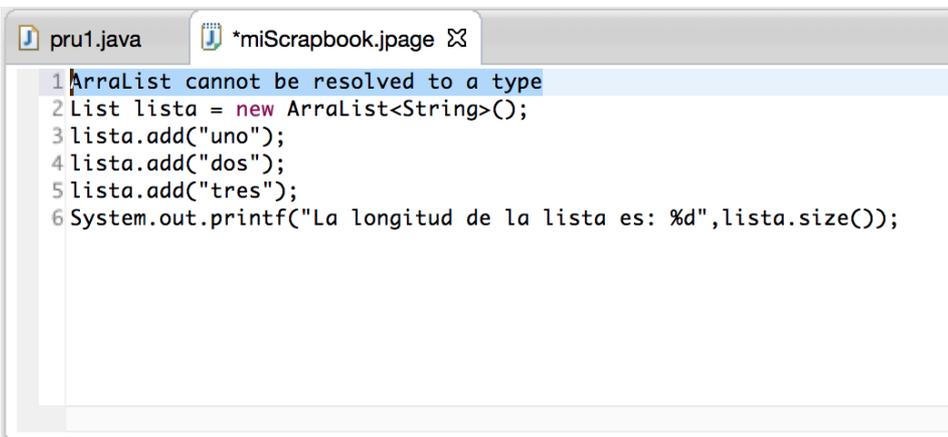
La longitud de la lista es: 3

#### 4. Cómo ver los errores en un ScrapBook

También es posible recibir la información relativa a los errores en el código que estamos probando en los ScrapBook, aunque desgraciadamente no es tan explícita e informativa como la que obtendríamos en una clase normal, donde podremos ver la traza con detenimiento en la consola. Tenemos dos formas de verlo:

##### Errores de compilación

Los típicos errores que no permiten compilar el código: métodos mal escritos, dependencias erróneas, instrucciones mal especificadas... El error aparecerá al comienzo del fragmento de código escrito en una cadena de texto. Para ilustrarlos hemos cambiado el constructor de la clase ArrayList por el de la clase inexistente ArraList



pru1.java \*miScrapbook.jpape

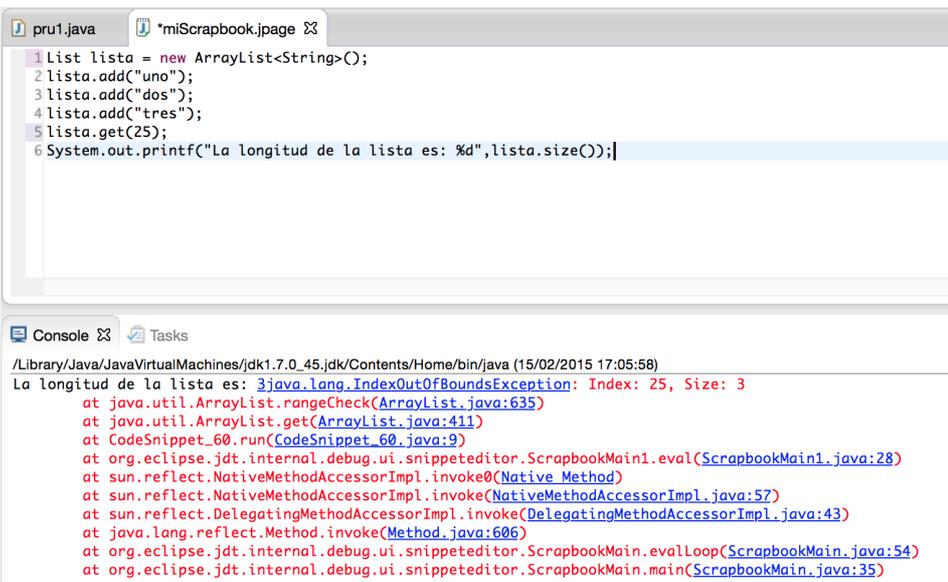
```

1 ArraList cannot be resolved to a type
2 List lista = new ArraList<String>();
3 lista.add("uno");
4 lista.add("dos");
5 lista.add("tres");
6 System.out.printf("La longitud de la lista es: %d", lista.size());

```

##### Errores en tiempo de ejecución

Son los errores no detectados en tiempo de compilación fruto de los valores que pueden tomar las variables del programa. En este caso aparecerá el error al final del código seleccionado y también suele aparecer una traza en la consola que nos facilitará el proceso de depuración. En este ejemplo hemos introducido una petición para el elemento 25 de la lista (que no existe).



pru1.java \*miScrapbook.jpape

```

1 List lista = new ArrayList<String>();
2 lista.add("uno");
3 lista.add("dos");
4 lista.add("tres");
5 lista.get(25);
6 System.out.printf("La longitud de la lista es: %d", lista.size());

```

Console Tasks

/Library/Java/JavaVirtualMachines/jdk1.7.0\_45.jdk/Contents/Home/bin/java (15/02/2015 17:05:58)

La longitud de la lista es: 3 java.lang.IndexOutOfBoundsException: Index: 25, Size: 3  
 at java.util.ArrayList.rangeCheck(ArrayList.java:635)  
 at java.util.ArrayList.get(ArrayList.java:411)  
 at CodeSnippet\_60.run(CodeSnippet\_60.java:9)  
 at org.eclipse.jdt.internal.debug.ui.snippeteditor.ScrapbookMain1.eval(ScrapbookMain1.java:28)  
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
 at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)  
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  
 at java.lang.reflect.Method.invoke(Method.java:606)  
 at org.eclipse.jdt.internal.debug.ui.snippeteditor.ScrapbookMain.evalLoop(ScrapbookMain.java:54)  
 at org.eclipse.jdt.internal.debug.ui.snippeteditor.ScrapbookMain.main(ScrapbookMain.java:35)

Sea cual sea el tipo de error o de resultado obtenido, no olvides que Eclipse escribe los mensajes en el propio Scrapbook, así

que ¡tendrás que borrarlos para volver a ejecutarlo!

### 5. Conclusiones

En este tutorial hemos visto cómo utilizar una herramienta que nos propone Eclipse para realizar pruebas o prototipos de código sin la necesidad de tener que manipular el código de nuestro proyecto haciendo clases o métodos de prueba, o acudiendo a lenguajes de scripting de la JVM. Además dispone de la posibilidad de ejecutar solamente fragmentos seleccionados de código de una forma rápida.

### A continuación puedes evaluarlo:

[Regístrate para evaluarlo](#)

### Por favor, vota +1 o compártelo si te pareció interesante

Share |  0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL**:

» **Regístrate** y accede a esta y otras ventajas «



Esta obra está licenciada bajo licencia [Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

**IMPULSA**    Impulsores    Comunidad    [¿Ayuda?](#)

----  
sin clicks

0 personas han traído clicks a esta página

+ + + + + + + +

powered by [karmacrazy](#)

Copyright 2003-2015 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

