



10 Effortless tricks to speed up your Java development in Eclipse

Ctrl
+ space

Code Completion

The screenshot shows the Eclipse IDE interface with the Java editor open. A code completion dropdown menu is displayed over the word 'RSet'. The menu lists several Java classes and interfaces, including 'ResourceSet', 'ResultSet', and 'RowSet'. To the right of the dropdown, a detailed tooltip provides information about 'ResourceSet': it defines it as a collection of related persistent documents, explains its role in managing resources, and describes how it can be created or loaded from a registry. The tooltip also mentions 'load options', 'URI converter', and the need to extend the default implementation.

```
package org.eclipse.jdt;
public class Main {
    public static void main(String[] args) {
        RSet
    }
}
```

A collection of related persistent documents.
A resource set manages a collection of related [resources](#) and produces notification for changes to that collection. It provides a [tree](#) of contents. A collection of [adapter factories](#) supports [adapter lookup](#) via registered adapter factory.
A resource can be [created](#) or [demand loaded](#) into the collection. The [registry](#) of resource factories can be configured to create resources of the appropriate type. A proxy can be [resolved](#) by the resource set, and may cause the demand load of a resource. Default [load options](#) are used during demand load. A [URI converter](#) can be configured to normalize URIs for comparison and to monitor access to the backing store. Clients must extend the default [implementation](#), since methods can
Press 'Tab' from proposal table or click for focus

Never type 100% of your code anymore by using the power of camel case

Problems x @ Javadoc Declaration

0 items

Description	Resource	Path	Location	Type
-------------	----------	------	----------	------

The screenshot shows the Eclipse IDE interface. The central part is the Java editor with a file named `Main.java` open. The code is as follows:

```
package org.eclipse.jdt;
import org.eclipse.emf.ecore.resource.ResourceSet;
public class Main {
    public static void main(String[] args) {
        ResourceSet
    }
}
```

A code completion dropdown menu is displayed over the word `ResourceSet`, listing two suggestions:

- resourceSet : ResourceSet
- set : ResourceSet

At the bottom of the dropdown, there is a message: "Press 'Ctrl+Space' to show Template Proposals".

The status bar at the bottom of the IDE shows the tabs: "Problems", "@ Javadoc", and "Declaration".

The name of your variables is automatically computed for you

The screenshot shows the Eclipse IDE interface with the Java editor open. The code being edited is:

```
package org.eclipse.jdt;
import java.io.File;
import org.eclipse.emf.ecore.resource.ResourceSet;
import org.eclipse.emf.ecore.resource.impl.ResourceSetImpl;

public class Main {
    public static void main(String[] args) {
        ResourceSet resourceSet = new ResourceSetImpl();
        File file = new File("");
        resourceSet.getR
```

A code completion tooltip is displayed over the method call 'resourceSet.getR'. The tooltip contains three suggestions:

- getResource(URI uri, boolean loadOnDemand) : Resource - Returns the resource resolved by the URI.
- getResourceFactoryRegistry() : Registry - ResourceSet
- getResources() : EList<Resource> - ResourceSet

The tooltip also includes a note about the implementation strategy and a keybinding 'Press 'Ctrl+Space' to show Template Proposals'.

Use camel case to select quickly the method to call among all the methods available

0 errors, 1 warning, 0 others

Description

Resource

Path

Location

Type

Warnings (1 item)

Quick Fix

Ctrl + 1

The screenshot shows the Eclipse Java Development Tool (JDT) interface. In the center, there is a code editor window with two tabs: *Main.java* and ResourceSet.class. The code in Main.java is:

```
package org.eclipse.jdt;
import java.io.File;
import org.eclipse.emf.ecore.resource.ResourceSet;
import org.eclipse.emf.ecore.resource.impl.ResourceSetImpl;

public class Main {
    public static void main(String[] args) {
        ResourceSet resourceSet = new ResourceSetImpl();
        File file = new File("");
        resourceSet.getResource(uri, [LoadOnDemand]);
    }
}
```

A code completion pop-up is displayed over the line `resourceSet.getResource(uri, [LoadOnDemand]);`. The pop-up contains the following options:

- Create local variable 'loadOnDemand'
- Create field 'loadOnDemand'
- Create parameter 'loadOnDemand'
- Create constant 'loadOnDemand'

Below the pop-up, a status bar message says "Press 'Ctrl+1' to go to original position". To the right of the code editor, the Outline view shows the current file structure.

At the bottom of the screen, a large text overlay reads:

Let Eclipse create local variables, choose their name and realize the necessary import

0 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
▶ Warnings (1 item)				

The screenshot shows the Eclipse IDE interface with the Java editor open. The code being edited is:

```
package org.eclipse.jdt;
import java.io.File;
import org.eclipse.emf.common.util.URI;
import org.eclipse.emf.ecore.resource.ResourceSet;
import org.eclipse.emf.ecore.resource.impl.ResourceSetImpl;

public class Main {
    public static void main(String[] args) {
        ResourceSet resourceSet = new ResourceSetImpl();
        File file = new File("");
        boolean loadOnDemand;
        URI uri;
        resourceSet.getResource(uri, loadOnDemand);
    }
}
```

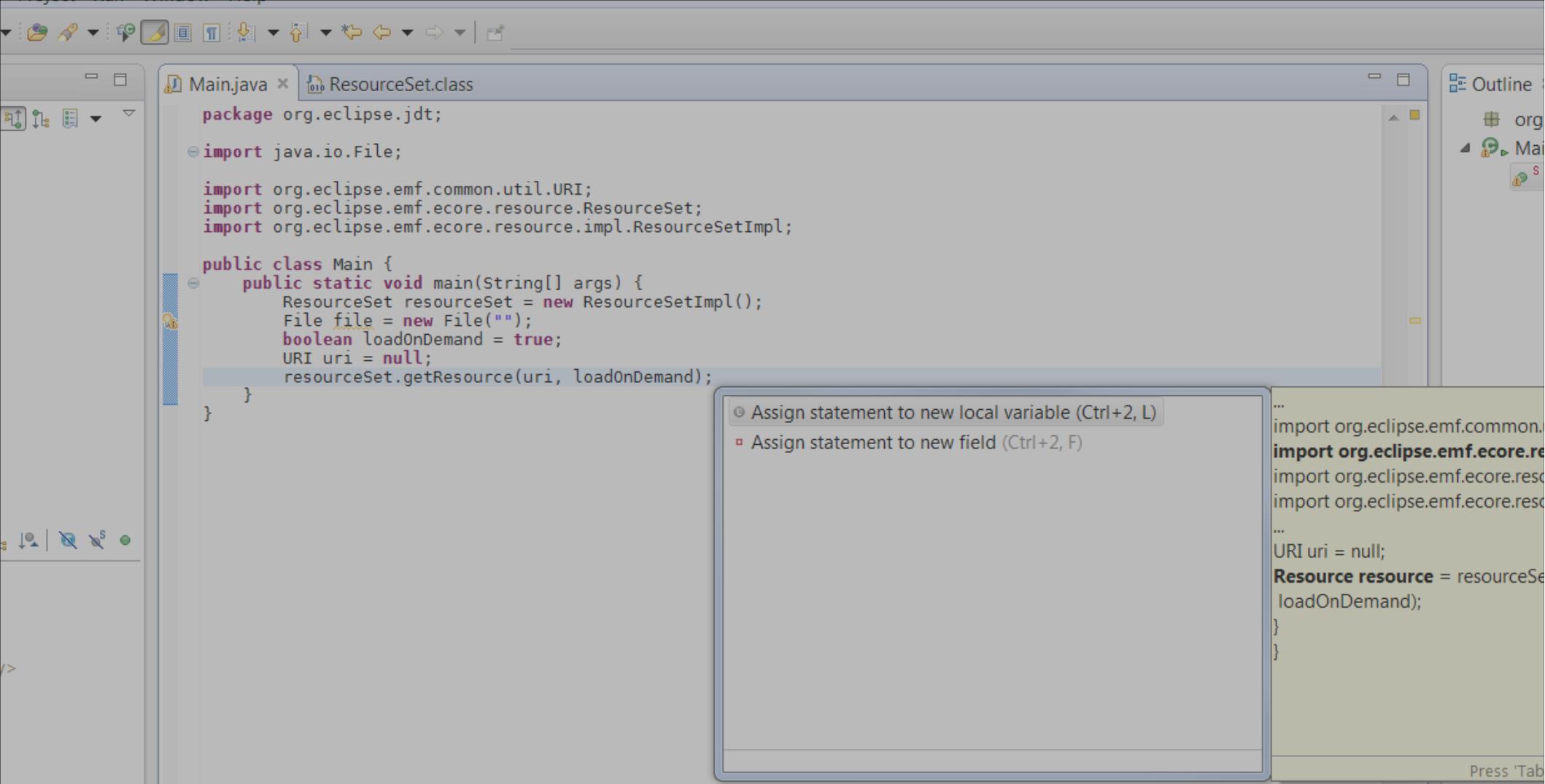
A code completion proposal is displayed in a floating window titled "Initialize variable". The proposal contains the following code:

```
File file = new File("");
boolean loadOnDemand = false;
URI uri;
```

Below the proposal window, there is a message: "Press 'Ctrl+1' to go to original position".

Initialize local variables

Description	Resource	Path	Location	Type
Errors (1 item)				
uri cannot be resolved to a variable	Main.java	/org.eclipse.jdt/src/m...	line 13	Java Problem



The screenshot shows the Eclipse Java Development Tool (JDT) interface. In the center-left, there's a code editor with tabs for `Main.java` and `ResourceSet.class`. The code in `Main.java` is:

```
package org.eclipse.jdt;
import java.io.File;
import org.eclipse.emf.common.util.URI;
import org.eclipse.emf.ecore.resource.ResourceSet;
import org.eclipse.emf.ecore.resource.impl.ResourceSetImpl;

public class Main {
    public static void main(String[] args) {
        ResourceSet resourceSet = new ResourceSetImpl();
        File file = new File("");
        boolean loadOnDemand = true;
        URI uri = null;
        resourceSet.getResource(uri, loadOnDemand);
    }
}
```

A code completion tooltip is open over the line `resourceSet.getResource(uri, loadOnDemand);`, listing two options:

- Assign statement to new local variable (Ctrl+2, L)
- Assign statement to new field (Ctrl+2, F)

In the bottom right corner of the editor area, there's a message: "Press 'Tab'".

Assign the result of a method to a new local variable

Problems x @ Javadoc Declaration

0 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
▶  Warnings (1 item)				

Code Templates

The screenshot shows the Eclipse IDE interface with the Java perspective. A code editor window is open, displaying a file named `Main.java`. The code contains a `main` method that creates a `ResourceSet`, a `File`, and a `URI`. It then retrieves a `Resource` and its contents. A cursor is positioned at the end of the `foreach` keyword. A tooltip box is displayed, containing the text "foreach - iterate over an array or Iterable". Below the tooltip, a message says "Press 'Ctrl+Space' to show Template Proposals". To the right of the tooltip, a portion of the code is shown with a cursor at the start of a `for` loop. A message below it says "Press 'Tab' from proposal table or click for focus". The status bar at the bottom of the IDE displays the text "Type a complex piece of code thanks to a small template".

```
package org.eclipse.jdt;
import java.io.File;

public class Main {
    public static void main(String[] args) {
        ResourceSet resourceSet = new ResourceSetImpl();
        File file = new File("");
        boolean loadOnDemand = true;
        URI uri = URI.createFileURI(file.getAbsolutePath());
        Resource resource = resourceSet.getResource(uri, loadOnDemand);
        List<EObject> contents = resource.getContents();
        foreach
    }
}
```

foreach - iterate over an array or Iterable

Press 'Ctrl+Space' to show Template Proposals

for (EObject eObject : contents) {
}

Press 'Tab' from proposal table or click for focus

Type a complex piece of code thanks to a small template

The screenshot shows the Eclipse IDE interface with the Java perspective. A code editor window is open, showing the same `Main.java` file. The cursor is now at the end of the `URI` constructor call in the `createFileURI` method. A call hierarchy view is displayed, titled "Call Hierarchy". It lists the methods that call the current one: `URI(boolean, String, String, String, String, boolean, String[], String, String)` from `org.eclipse.emf.common.util.URI`. This method has three entries: `appendFileExtension(String)`, `appendFragment(String)`, and `createURIWithCache(String)`. The status bar at the bottom of the IDE displays the text "Line 845" and "850".

Call Hierarchy

Members calling 'URI(boolean, String, String, String, String, boolean, String[], String, String)' - in workspace

- URI(boolean, String, String, String, String, boolean, String[], String, String) - org.eclipse.emf.common.util.URI
 - appendFileExtension(String) : URI - org.eclipse.emf.common.util.URI
 - appendFragment(String) : URI - org.eclipse.emf.common.util.URI
 - createURIWithCache(String) : URI - org.eclipse.emf.common.util.URI (2 matches)

Line 845

850

The screenshot shows the Eclipse IDE's Preferences dialog box. The left sidebar lists various preference categories like General, Java, and Editor. The 'Editor' category is expanded, and 'Templates' is selected. The main area is titled 'Templates' and contains a table with columns: Name, Context, Description, and Auto Insert. The table lists several Java code snippets with their descriptions and checkboxes for auto-insertion. A preview window shows the generated code for a 'while' loop template. At the bottom, there are buttons for 'OK', 'Cancel', 'Apply', and 'Restore Defaults'.

Name	Context	Description	Auto Insert
Text	SWT statements	new Text	
toarray	Java	convert collection to ...	
ToolBar	SWT statements	new ToolBar	
ToolItem	SWT statements	new ToolItem for a T...	
Tree	SWT statements	new Tree	
TreeColu...	SWT statements	new TreeColumn for ...	
TreeItem	SWT statements	new TreeItem for a T...	
true	Javadoc	<code>true</code>	on
try	Java statements	try catch block	
while	Java statements	iterate with enumera...	
while	Java statements	iterate with iterator	
while	Java statements	while loop with cond...	

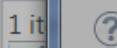
Preview:

```
while (${en:var(java.util.Enumeration)}.hasMoreElements()) {  
    ${type:argType(en)} ${elem:newName(type)} = (${type}) ${en}.nextElement();  
    ${cursor}  
}
```

Use code formatter

OK Cancel Apply Restore Defaults

Create your own code templates in a few seconds



PERF resolving the whole resource is time consuming!

Main.java

/org.eclipse.jdt/src/main/java/org/eclipse/jdt

Location
line 40

Type
Ja

Open Type

Ctrl
+ Shift
+ 

The screenshot shows the Eclipse IDE interface. In the center, a modal dialog titled "Open Type" is displayed. The input field contains the prefix "RSet". Below it, a list of matching items is shown, starting with "ResourceSet - org.eclipse.emf.ecore.resource". Other items include "ResourceSetImpl", "ResultSet", "ResultSetMetaData", "RowSet", "RowSetEvent", "RowSetFactory", "RowSetFactoryImpl", "RowSetInternal", and "RowSetListener". At the bottom of the list, the fully qualified name "org.eclipse.emf.ecore.resource - org.eclipse.emf.ecore_2.8.1.v20120911-0500.jar" is visible. At the bottom right of the dialog are "OK" and "Cancel" buttons.

Find the type that you were looking for and filter the result with camel case

Call Hierarchy

Ctrl
+ **Alt** + **H**

```

/*
public static URI createFileURI(String pathName)
{
    File file = new File(pathName);
    String uri = File.separatorChar != '/' ? pathName.replace(File.separatorChar, SEGMENT_SEPARATOR) : pathName;
    uri = encode(uri, PATH_CHAR_HI, PATH_CHAR_LO, false);
    if (file.isAbsolute())
    {
        URI result = createURI((uri.charAt(0) == SEGMENT_SEPARATOR ? "file:" : "file:/") + uri);
        return result;
    }
    else
    {
        URI result = createURI(uri);
        if (result.scheme() != null)
        {
            throw new IllegalArgumentException("invalid relative pathName: " + pathName);
        }
        return result;
    }
}

/**
 * Static factory method based on parsing a workspace-relative path string.
 *
 * <p>The <code>pathName</code> must be of the form:
 * <pre>
 *   /project-name/path</pre>
 */

```

```

parseIntoURI(String)
segmentsRemainder()
find(String, int, int)
createFileURI(String)
createPlatformURI(String, String, String, boolean)
createPlatformURI(String, String, String)
createPlatformURI(String, String)
URI(boolean, String)
validateURI(boolean)
validScheme(String)
validOpaquePart(String)
validAuthority(String)
validArchiveAuthority(String)
validJarAuthority(String)
validDevice(String)
validSegment(String)
validSegments(String[])
firstInvalidSegment(String)
... (many more methods listed)

```

Problems @ Javadoc Declaration Call Hierarchy

Members calling 'URI(boolean, String, String, String, boolean, String[], String, String)' - in workspace

- URI(boolean, String, String, String, boolean, String[], String, String) - org.eclipse.emf.common.util.URI
 - appendFileExtension(String) : URI - org.eclipse.emf.common.util.URI
- appendFragment(String) : URI - org.eclipse.emf.common.util.URI
- createURIWithCache(String) : URI - org.eclipse.emf.common.util.URI (2 matches)
 - createDeviceURI(String) : URI - org.eclipse.emf.common.util.URI
 - createURI(String, boolean, int) : URI - org.eclipse.emf.common.util.URI
- createURI(String, boolean) : URI - org.eclipse.emf.common.util.URI
- createURI(int, int) : URI - org.eclipse.emf.common.util.URI
- createFileURI(String) : URI - org.eclipse.emf.common.util.URI (2 matches)
 - createInputStream(String) : InputStream - org.eclipse.emf.ecore.resource.impl.ArchiveURIHandlerImpl.Archive
 - createInputStream(String) : InputStream - org.eclipse.emf.ecore.resource.impl.URIConverterImpl.Archive
 - createOutputStream(String) : OutputStream - org.eclipse.emf.ecore.resource.impl.ArchiveURIHandlerImpl.Archive
 - createOutputStream(String) : OutputStream - org.eclipse.emf.ecore.resource.impl.URIConverterImpl.Archive
- createPlatformURI(String, String, String, boolean) : URI - org.eclipse.emf.common.util.URI
- createResource(String) : Resource - org.eclipse.emf.ecore.impl.EPackageImpl

Find out where your code is called from!

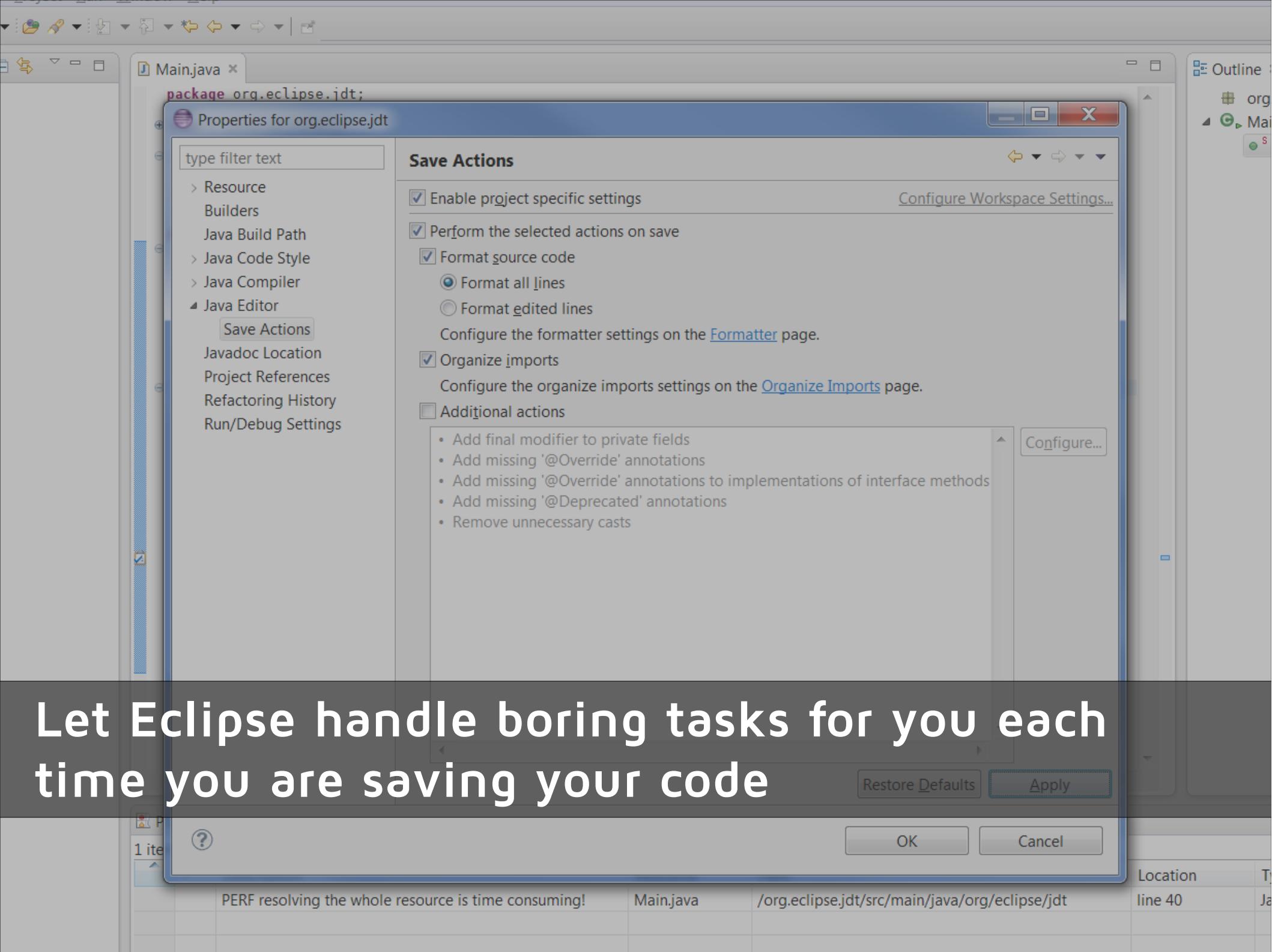
Ctrl +
Shift +
O

Organize
Imports

The screenshot shows the Eclipse IDE interface. In the center, a modal dialog titled "Organize Imports" is open, displaying the message "Choose type to import:" and "Page 1 of 2". It lists two options: "@ javax.annotation.Resource" and "org.eclipse.emf.ecore.resource.Resource", with the second option selected. At the bottom of the dialog are buttons for "?", "< Back", "Next >" (which is highlighted in blue), "Finish", and "Cancel". In the background, the Java code editor shows a file named "Main.java" with imports for "java.io.File" and "org.eclipse.jdt.ResourceSet". The "Resource" import is currently being organized, as indicated by the red squiggly underline. The status bar at the bottom shows "4 errors, 0 warnings, 0 others".

Ask Eclipse to import the necessary types. In case of conflicts, Eclipse will ask you what to import

Save Actions



Let Eclipse handle boring tasks for you each time you are saving your code

Debug
Debug !
Debug

Debug

- Main [Java Application]
- org.eclipse.jdt.Main at localhost:54982
- Thread [main] (Suspended (breakpoint at line 39 in Main))
 - Main.main(String[]) line: 39
- C:\Program Files\Java\jdk1.7.0_02\bin\javaw.exe (11 févr. 2013 15:44:51)

Variables

Name
args
resourceSet
file

Main.java

```
if (args.length == 0) {
    throw new IllegalArgumentException(
        "At least one parameter is required, the absolute path of a file."); //NON-NLS-N$
}
ResourceSet resourceSet = new ResourceSetImpl();
resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap()
    .put("ecore", new EcoreResourceFactoryImpl());
File file = new File(args[0]);
if (!file.exists()) {
    System.out.println();
}
boolean loadOnDemand = true;
URI uri = URI.createFileURI(file.getAbsolutePath());
Resource loadedResource = resourceSet.getResource(uri, loadOnDemand);

// PERF resolving the whole resource is time consuming!
try {
    resolveAll(loadedResource);
} catch (Exception exception) {
    // TODO Auto-generated catch block
    exception.printStackTrace();
}
```

Console

Main [Java Application] C:\Program Files\Java\jdk1.7.0_02\bin\javaw.exe (11 févr. 2013 15:44:51)

Stop your application on a specific instruction to have a look at the state of your variables

The screenshot shows the Eclipse IDE interface during a debugging session. The top window is the 'Variables' view, which lists the current values of variables:

Name	Value
args	String[1] (id=16)
resourceSet	ResourceSetImpl (id=18)
file	File (id=25)

The bottom window is the 'Expressions' view, which displays the value of the expression "file.getName()".

Name	Value
file.getName()	Model.ecore
count	11
hash	0
offset	68
value	(id=32)

A tooltip for the 'value' entry indicates it is of type 'Object'. A button labeled '+ Add new expression' is visible at the bottom of the 'Expressions' view.

Use the Expressions view to track the value of complex expressions

Display

View

The screenshot shows the Eclipse IDE interface with several open windows:

- Top Left:** A stack trace window showing the current thread is suspended at line 37 in Main.main.
- Top Right:** A file tree view showing a directory structure under F:\Developpement\Workspaces\Java\.
- Middle Left:** A code editor window titled "Main.java" containing Java code. A tooltip is visible over the line `System.out.println();`.
- Bottom Left:** A "Console" tab showing the output of the expression `file.getName()` which is `Model.ecore`.
- Bottom Center:** An "Expressions" view dialog box showing the result of the expression `file.getName()`. It displays a search result for "file.getName() = Model.ecore" with an ID of 684. The result object has attributes: count= 11, hash= 0, offset= 68, and value= (id=685). The value attribute is expanded to show its internal structure: [0]= F.

Text from the screenshot:

```
throw new IllegalArgumentException(  
        "At least one parameter is required, the absolute path of a file."); //NON-NLS-N$  
}  
ResourceSet resourceSet = new ResourceSetImpl();  
File file = new File(args[0]);  
if (!file.exists()) {  
    System.out.println();  
}  
boolean loadOnDemand = true;  
URI uri = URI.createFileURI(file.getAbsolutePath());  
Resource loadedResource = resourceSet.getResource(uri, loadOnDemand);
```

Press Ctrl+Shift+I to Move to Expressions View

Run expressions against the current state of your application to fix bugs easily

Scrapbook
Page

The screenshot shows the Eclipse IDE interface with two code editors open. The top editor, titled 'Main.java', contains Java code for reading an Ecore file from the command line. The bottom editor, titled '*scrapbook.jpage', contains a single line of code that runs the Main class with a specific argument.

```
* file from the file system as an EMF resource.  
*  
* @param args  
*         The arguments of the Java program. At least one argument  
*         should be provided, a string representing the absolute path of  
*         a file located on the file system.  
*/  
public static void main(String[] args) {  
    if (args.length == 0) {  
        throw new IllegalArgumentException(  
            "At least one parameter is required, the absolute path of a file."); //$/NON-NLS-N$  
    }  
    ResourceSet resourceSet = new ResourceSetImpl();  
    resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap()  
        .put("ecore", new EcoreResourceFactoryImpl());  
    File file = new File(args[0]);  
    if (!file.exists()) {  
        System.out.println();  
    }  
    boolean loadOnDemand = true;
```

```
*scrapbook.jpage  
org.eclipse.jdt.Main.main(new String[]{"F:\\Developpement\\Workspaces\\Java\\org.eclipse.jdt\\src\\test\\resources\\Model.ecore", });  
(No explicit return value)
```

Test your code without having to run your complete application

Problems Javadoc Declaration Console Cairn Hierarchy Tasks

C:\Program Files\Java\jdk1.7.0_02\bin\javaw.exe (11 févr. 2013 15:51:29)

org.eclipse.emf.ecore.impl.EPackageImpl@129e76f3 (name: ecoretoscala) (nsURI: http://ecore-to-scala.obeonetwork.org, nsPrefix: ecoretoscala)

Thanks for watching!

More tips and tricks:

Blog: stephanebegaudeau.tumblr.com

Twitter: @sbegaudeau