



# INDEX


## Eclipse Shortcuts

---

- |   |                           |
|---|---------------------------|
| 1. 8 Eclipse Shortcut Keys for Code Refactoring       | <a href="#"><u>04</u></a> |
| 2. 16 Eclipse Shortcut Keys for Workspace and Project | <a href="#"><u>08</u></a> |
| 3. 27 Eclipse Shortcut Keys for Code Editing          | <a href="#"><u>11</u></a> |

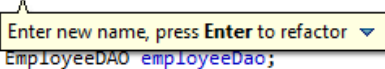
# Eclipse Shortcuts

## 8 Eclipse Shortcut Keys for Code Refactoring

 We do refactor most of the time when writing code. Thus, using shortcut keys can boost your productivity. Here, we round up a list of shortcut keys used for code refactoring Java code in Eclipse IDE.

**1. Alt + Shift + R:** Renames a variable, a method, a class or even a package name. This is the most frequently used shortcut in code refactoring. Select whole name of the class, method or variable you want to rename, and then press this shortcut:

```
17 @Controller
18 public class HomeController {
19
20     @Autowired
21     private EmployeeDAO employeeDao;
```



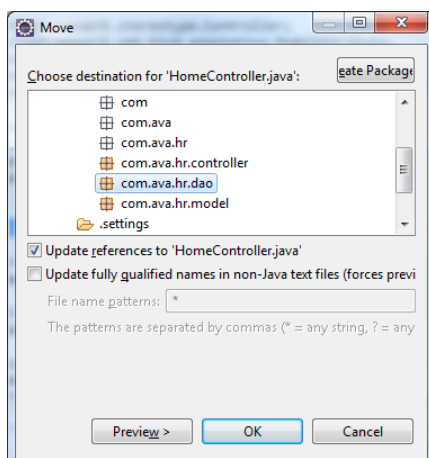
Type new name and press **Enter** when done, Eclipse automatically updates all related references for the new name, including ones found in other classes.

**2. Ctrl + 2, R:** Renames a variable, a method or a class name locally in the current file. Eclipse doesn't search outside references hence this renaming is faster than the **Alt + Shift + R** shortcut. However, use this shortcut with care: only for names used locally in the current file:

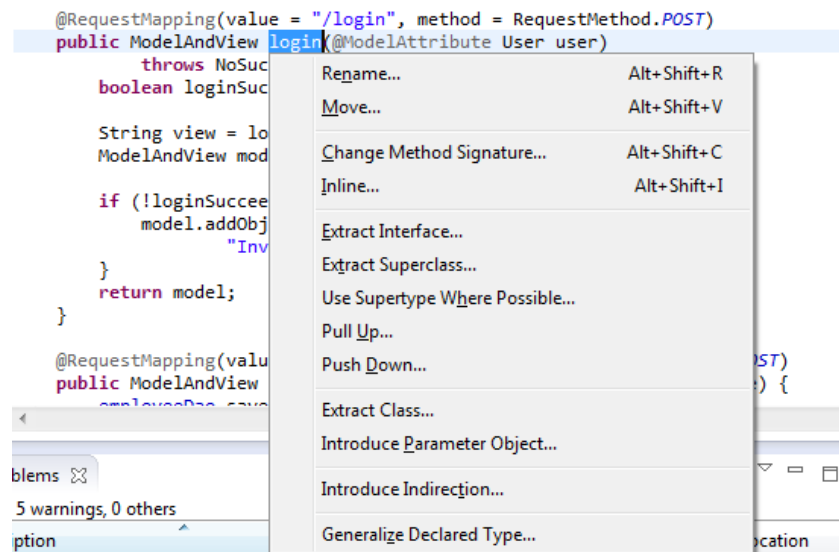
```
16 @Transactional
17 public void save(Employee employee) {
18     Session session = sessionFactory.getCurrentSession();
19
20     session.save(employee);
21 }
```

**Ctrl + 2, R to rename locally**

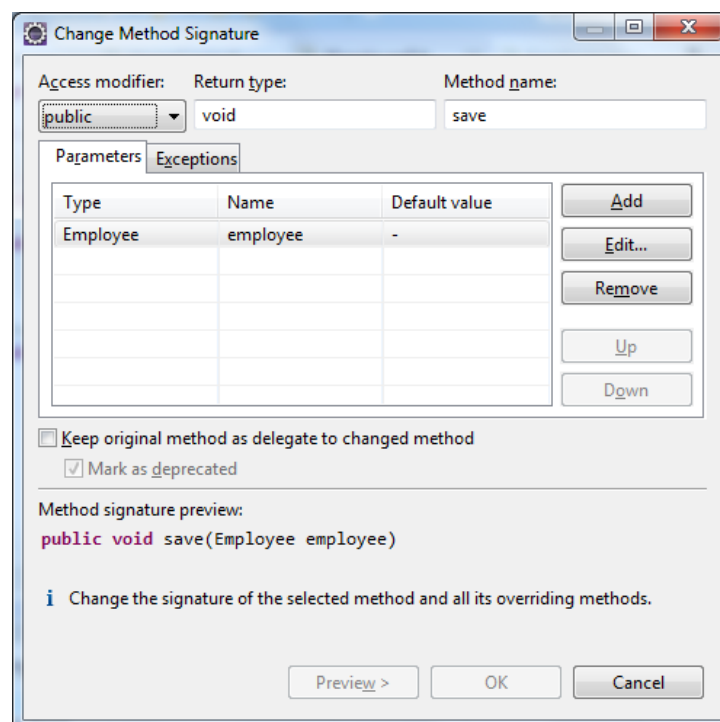
**3. Alt + Shift + V:** Moves a class, method to another destination. For example, select a class name and press this shortcut, the **Move** dialog appears. Choose a destination and then click **OK**:



**4. Alt + Shift + T:** Shows refactor context menu. This shortcut allows you to access a full list of refactoring operations which are possible for current context:



**5. Alt + Shift + C:** Changes signature of a method. Place the cursor inside a method or select method name, and then press this shortcut. The **Change Method Signature** dialog appears. You can change various elements of method signature such as access modifier, return type, parameters, exceptions, etc.




**6. Alt + Shift + M:** Extracts a selection to a method. This helps you move a

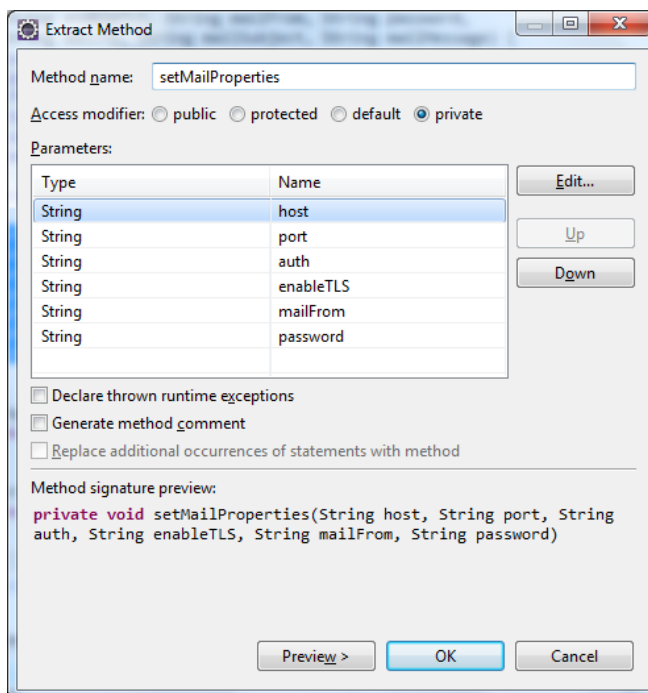
selected block of code to a separate method with ease. For example:

```
20 public EmailConfig(String host, String port, String auth,  
21 String enableTLS, String mailFrom, String password,  
22 String mailTo, String mailSubject, String mailMessage) {  
23     this.user_nm = mailFrom;  
24     this.password = password;  
25     this.from = mailFrom;  
26     this.to = mailTo;  
27     this.subject = mailSubject;  
28     this.message = mailMessage;  
29  
30     authenticated = Boolean.getBoolean(auth);  
31  
32     properties = new Properties();  
33     properties.put("mail.smtp.host", host);  
34     properties.put("mail.smtp.port", port);  
35     properties.put("mail.smtp.auth", auth);  
36     properties.put("mail.smtp.starttls.enable", enableTLS);  
37     properties.put("mail.user", mailFrom);  
38     properties.put("mail.password", password);  
39 }
```

**Alt + Shift + M**  
to extract this  
selection  
to a method



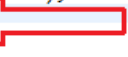
The **Extract Method** dialog appears. Enter new method name and specify access modifier, parameters list, and then click **OK** to do the refactoring:



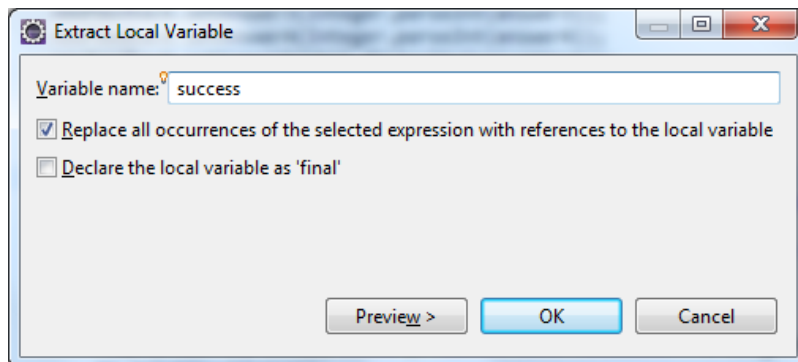
**7. Alt + Shift + L:** Extracts local variable from an expression. For example:

```
170 FeedbackDAO dao = new FeedbackDAO(dbURL, dbUser, dbPass);  
171 try {  
172     int result = dao.save(newFeedback);  
173     if (result > 0) {  
174         request.setAttribute("IN");  
175     } else {  
176         request.setAttribute("ER");  
177     }  
178     request.setAttribute("Feedba  
179 } catch (Exception ex) {  
180     request.setAttribute("ERROR"  
181     ex.printStackTrace();  
182 }
```

**Alt + Shift + L**  
to extract this  
expression to a local  
variable



Press this shortcut key brings the **Extract Local Variable** dialog which allows you to name the local variable:



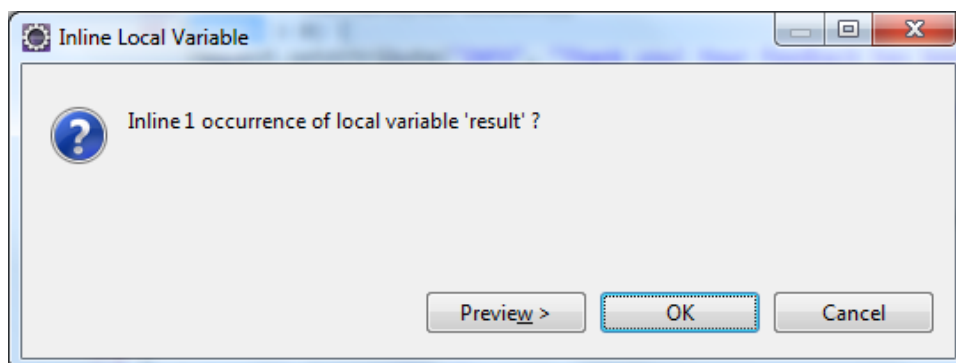
Click **OK** to do the refactoring, and here's the result:

**8. Alt + Shift + I:** Inlines a selected local variable, method or constant if possible. Eclipse replaces the selection with its declaration and puts it directly into the statement. For example:

```
170 FeedbackDAO dao = new FeedbackDAO(dbURL, dbUser, dbPass);
171 try {
172     int result = dao.save(newFi
173     if (result > 0) {
174         request.setAttribute(":
175     } else {
176         request.setAttribute("I
177     }
178     request.setAttribute("Feedl
179 } catch (Exception ex) {
180     request.setAttribute("ERROR", "Cannot save feedback: " + ex.getMessage());
181     ex.printStackTrace();
182 }
```

**Alt + Shift + I**  
to replace this variable with its  
declaration (inline)

If the selection is possible to inline, Eclipse will ask to confirm:

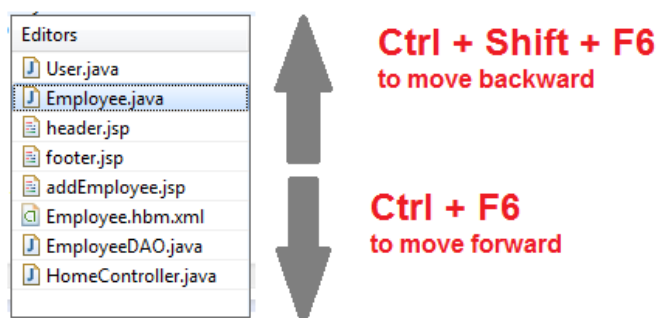


Click **OK** to do proceed, here's the result:

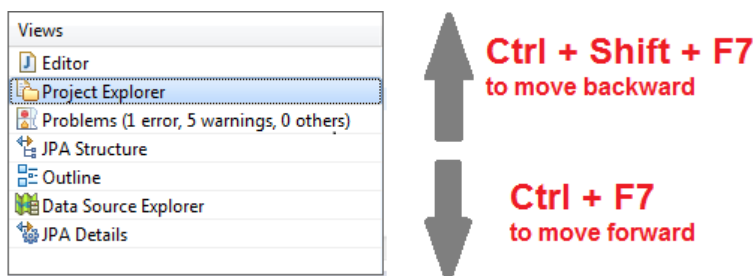
## 16 Eclipse Shortcut Keys for Workspace and Project

- In continuation with a series of articles about Eclipse shortcut keys, today we compile a list of useful shortcuts for working with workspace and projects using Eclipse IDE.
- Let's start with the shortcuts related to workspace first.

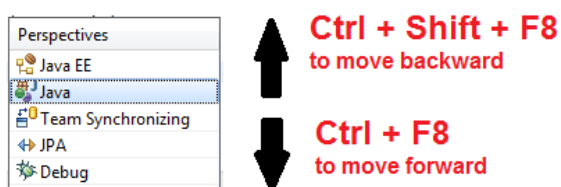
**1. Ctrl + F6/Ctrl + Shift + F6:** Switches between editors. It's very usual that we open as many code editors as many source files we want to work on. Press **Ctrl + F6** helps us moving forward between them. And press **Ctrl + Shift + F6** to move backward between the opening editors:



**2. Ctrl + F7/Ctrl + Shift + F7:** Switches between views. This allows you to move between open views in the current perspective:



**3. Ctrl + F8/Ctrl + Shift + F8:** Switches between perspectives. This allows you to move back and forth among open perspectives. For example, move from Java EE perspective to Team Synchronization perspective to update/commit your code. Hold down the **Ctrl** key and press the **F8** key to move around the list of open perspective. Release the keys when you decide to move to the selected one. Press **Ctrl + Shift + F8** lets you move reversely:



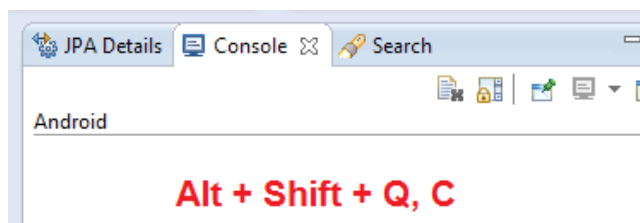


**4. Ctrl + M:** Toggles maximize/restore the current view. This shortcut is very useful when you want to quickly maximize the current view, e.g., the currently editing code editor. Press **Ctrl + M** again will restore the maximized view. This is equivalent to double clicking on tab header of the view.

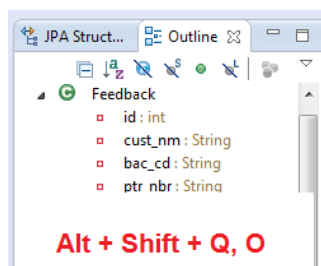
**5. F12:** Activates Editor. Let say, if you are currently at the Project explorer view and you want to jump to the editor view, press F12 will activate the currently selected code editor.

Here are other shortcuts that let you quickly move to frequently-used views:

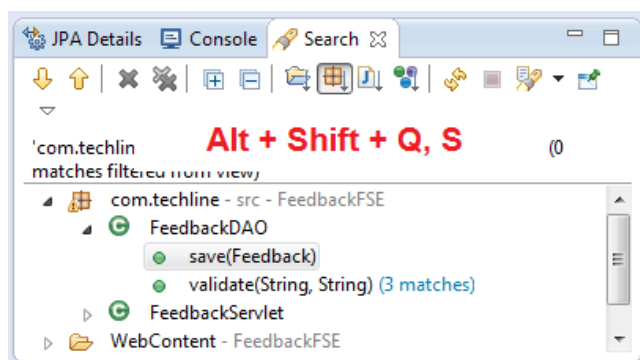
**6. Alt + Shift + Q, C:** Jumps to **Console** view (Hold down **Alt**, **Shift** and **Q**; then release all and press **C**):



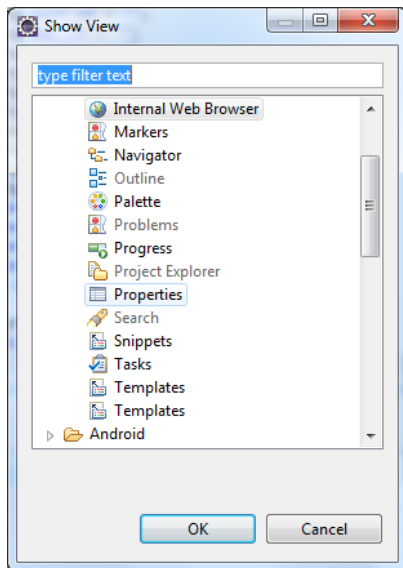
**7. Alt + Shift + Q, O:** Jumps to **Outline** view:



**8. Alt + Shift + Q, S:** Jumps to **Search** view:

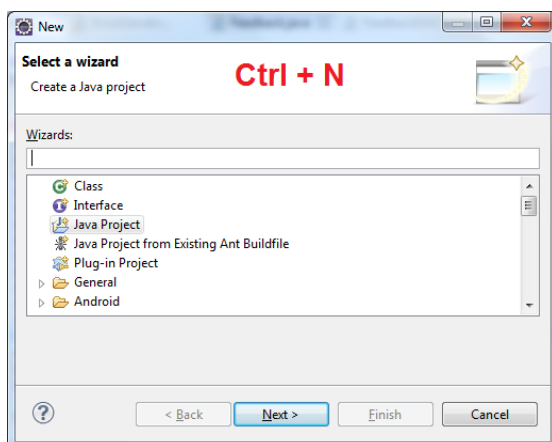


**9. Alt + Shift + Q, Q:** Brings Show View selection dialog that lets you to choose a specific view:

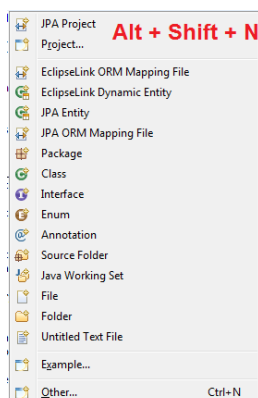


Now, let's go to other shortcuts related to working with projects in Eclipse.

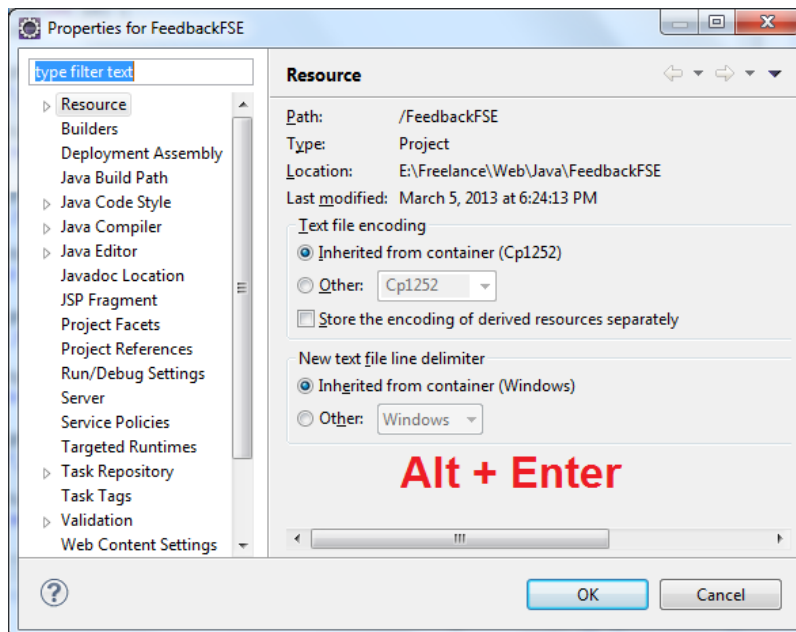
**10. Ctrl + N:** Opens the New wizard dialog that lets you create new project or file:



**11. Alt + Shift + N:** Shows context menu for creating new objects. This gives you quick access to the frequently used items:



**12. Alt + Enter:** Shows Properties dialog of the current selected project, or the currently active code editor.



**13. Ctrl + W:** Closes the currently active code editor.

**14. Ctrl + Shift + W:** Closes all opening code editors.

**15. Ctrl + S:** Saves currently editing file.

**16. Ctrl + Shift + S:** Saves all editing files.

## 27 Eclipse Shortcut Keys for Code Editing

✚ When using an IDE, you cannot be more productive without using its shortcut keys frequently as your habit. In this article, we summarize a list of shortcut keys which are useful for editing Java code in Eclipse IDE.

**1. Ctrl + D:** Deletes current line.

**2. Ctrl + Delete:** Deletes next word after the cursor.

**3. Ctrl + Shift + Delete:** Deletes from the cursor until end of line.


**4. Ctrl + Backspace:** Deletes previous word before the cursor.

**5. Shift + Ctrl + y:** Changes a selection to lowercase.

**6. Shift + Ctrl + x:** Changes a selection to uppercase.

**7. Alt + Up Arrow:** Moves up current line (or a selected code block) by one line:


```
11 public Employee(int id, String name, String address) {  
12     this.id = id;  
13     this.address = address;  
14     this.name = name;  
15 }
```



**Alt + Up**

**8. Alt + Down Arrow:** Moves down current line (or a selected code block) by one line:


```
11 public Employee(int id, String name, String address) {  
12     this.id = id;  
13     this.name = name;  
14     this.address = address;  
15 }
```



**Alt + Down**

**9. Ctrl + Alt + Up Arrow:** Copies and moves up current line (or a selected code block) by one line:


```
11 public Employee(int id, String name, String address) {  
12     this.id = id;  
13     this.name = name;  
14     this.address = address;  
15     this.address = address;  
16 }
```



**Ctrl + Alt + Up**

**10. Ctrl + Alt + Down Arrow:** Copies and moves down current line (or a selected code block) by one line:


```
11 public Employee(int id, String name, String address) {  
12     this.id = id;  
13     this.name = name;  
14     this.address = address;  
15     this.address = address;  
16 }
```



**Ctrl + Alt + Down**

**11. Shift + Enter:** Inserts a blank line after current line, regardless where the cursor is at the current line (very different from press **Enter** key alone):

```
11 public Employee(int id, String name, String address) {  
12     this.id = id;  
13     this.name = name;  
14     this.address = address;  
15  
16 }
```



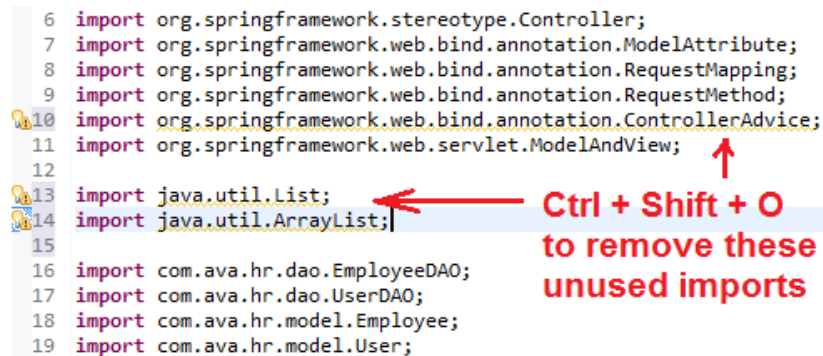
**Shift + Enter**

**12. Ctrl + Shift + Enter:** works similar to the **Shift + Enter**, but inserts a blank line just before the current line.

**13. Ctrl + Shift + O:** Organizes import statements by removing unused imports

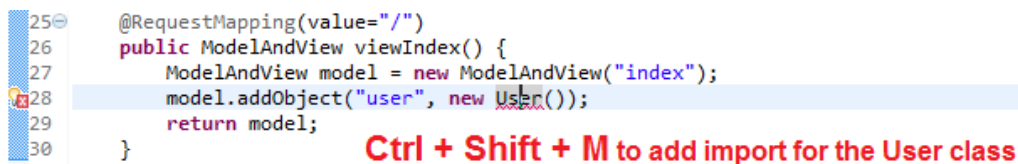
and sorts the used ones alphabetically. This shortcut also adds missing imports.

```
6 import org.springframework.stereotype.Controller;
7 import org.springframework.web.bind.annotation.ModelAttribute;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RequestMethod;
10 import org.springframework.web.bind.annotation.ControllerAdvice;
11 import org.springframework.web.servlet.ModelAndView;
12
13 import java.util.List;
14 import java.util.ArrayList;
15
16 import com.ava.hr.dao.EmployeeDAO;
17 import com.ava.hr.dao.UserDAO;
18 import com.ava.hr.model.Employee;
19 import com.ava.hr.model.User;
```



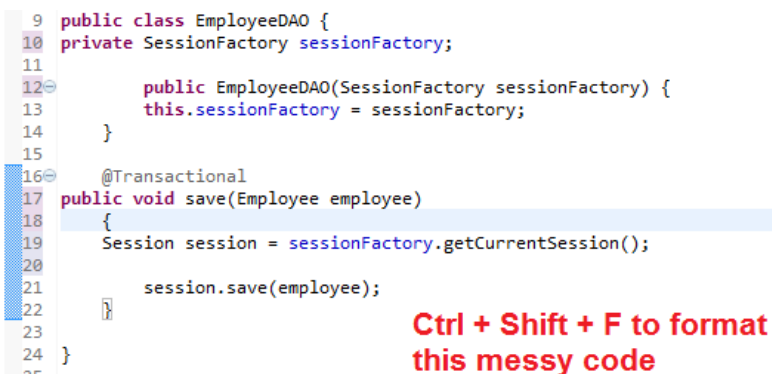
**14. Ctrl + Shift + M:** Adds a single import statement for the current error due to missing import. You need to place the cursor inside the error and press this shortcut:

```
25 @RequestMapping(value="/")
26 public ModelAndView viewIndex() {
27     ModelAndView model = new ModelAndView("index");
28     model.addObject("user", new User());
29     return model;
30 }
```



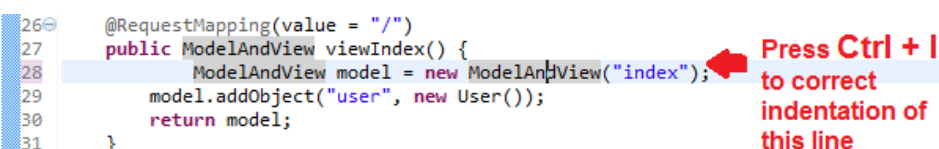
**15. Ctrl + Shift + F:** Formats a selected block of code or a whole source file. This shortcut is very useful when you want to format messy code to Java-standard code. Note that, if nothing is selected in the editor, Eclipse applies formatting for the whole file:

```
9 public class EmployeeDAO {
10     private SessionFactory sessionFactory;
11
12     public EmployeeDAO(SessionFactory sessionFactory) {
13         this.sessionFactory = sessionFactory;
14     }
15
16     @Transactional
17     public void save(Employee employee)
18     {
19         Session session = sessionFactory.getCurrentSession();
20
21         session.save(employee);
22     }
23 }
24
25
```



**16. Ctrl + I:** Corrects indentation for current line or a selected code block. This is useful as it helps you avoid manually using Tab key to correct the indentation:

```
26 @RequestMapping(value = "/")
27 public ModelAndView viewIndex() {
28     ModelAndView model = new ModelAndView("index");
29     model.addObject("user", new User());
30     return model;
31 }
```



**17. Ctrl + A, Ctrl + I:** Corrects indentation for hold page.

**18. Ctrl + / or Ctrl + 7:** Toggle single line comment. This shortcut adds single-line comment to current line or a block of code. Press again to remove comment. For example:

```
15
16 // @Transactional
17 // public void save(Employee employee) {
18 //     Session session = sessionFactory.getCurrentSession();
19 //
20 //     session.save(employee);
21 // }
22
```

**Ctrl + / or Ctrl + 7**

**19. Ctrl + /:** again, for remove the Single line comment.

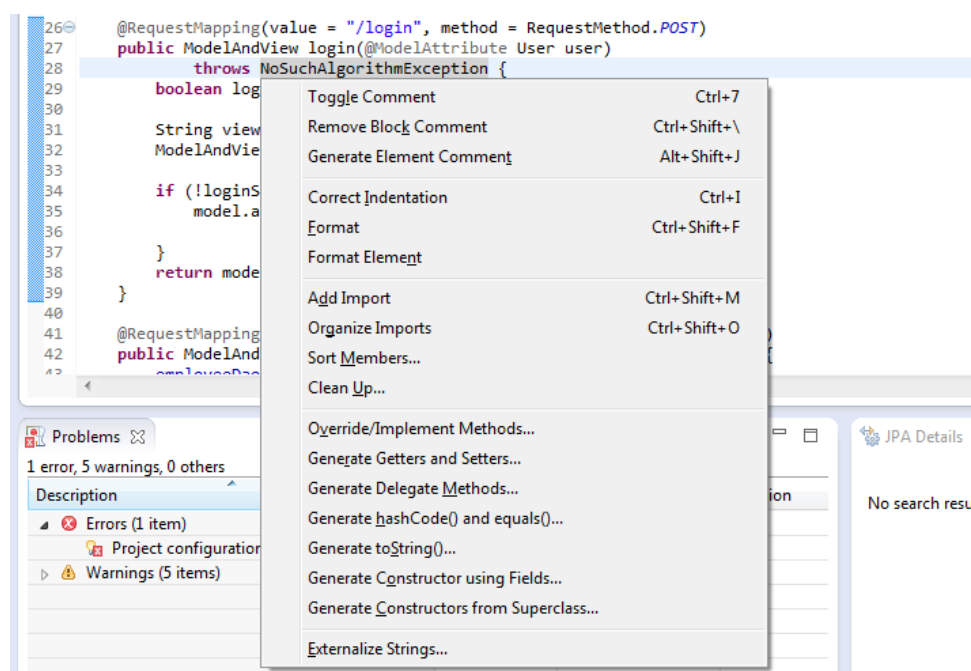
**20. Ctrl + Shift + /:** Adds block comment to a selection.

```
16 /* @Transactional
17     public void save(Employee employee) {
18         Session session = sessionFactory.getCurrentSession();
19
20         session.save(employee);
21     }*/
22
23 }
```

**Ctrl + Shift + / to add a block comment**

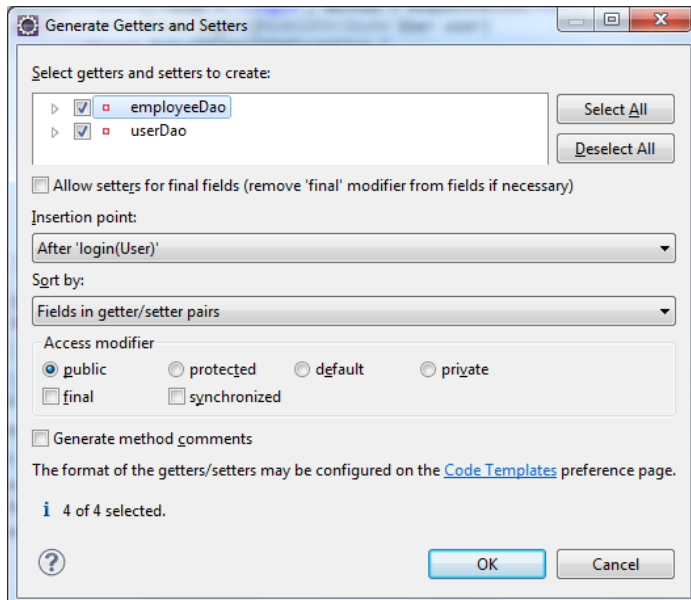
**21. Ctrl + Shift + \:** Removes block comment.

**22. Alt + Shift + S:** Shows context menu that lists possible actions for editing code:

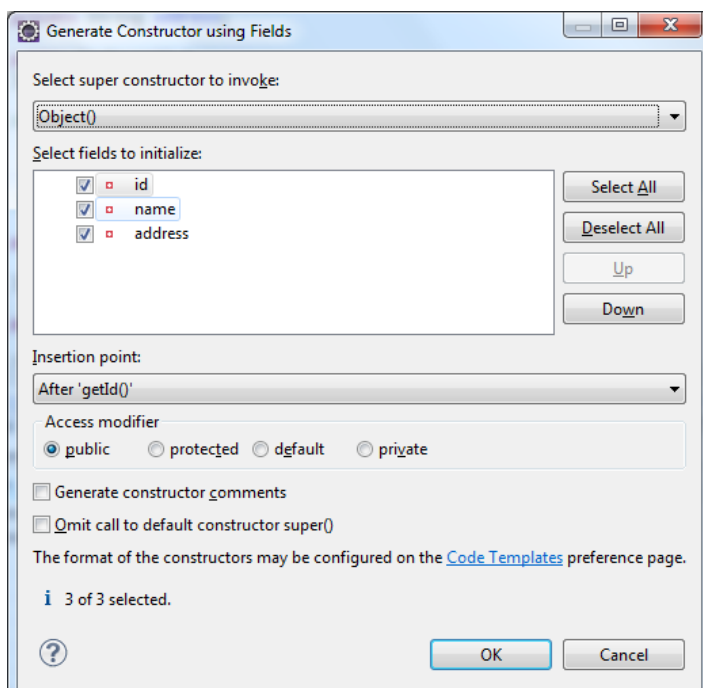


From this context menu, you can press another letter (according to the underscore letters in the names) to access the desired functions.

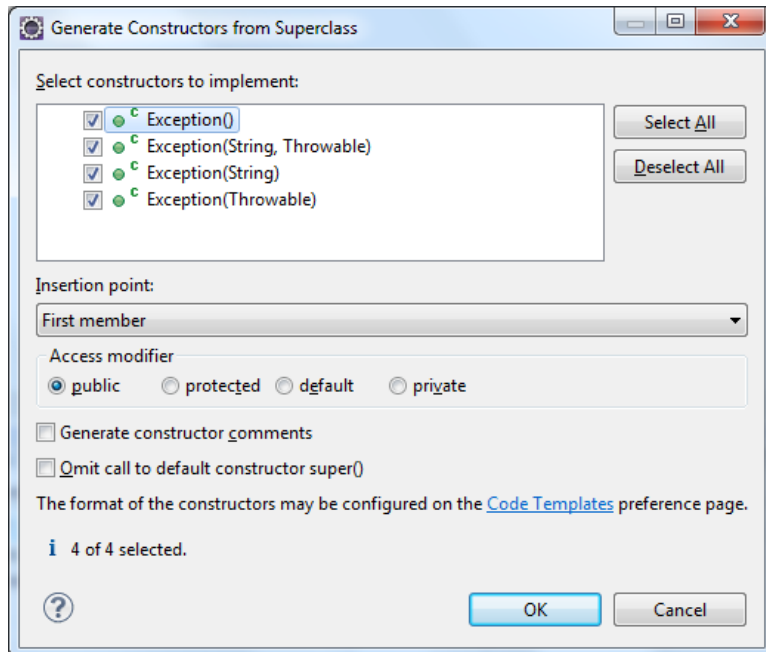
**23. Alt + Shift + S, R:** Generates getters and setters for fields of a class. This is a very handy shortcut that helps us generate getter and setter methods quickly. The following dialog appears:



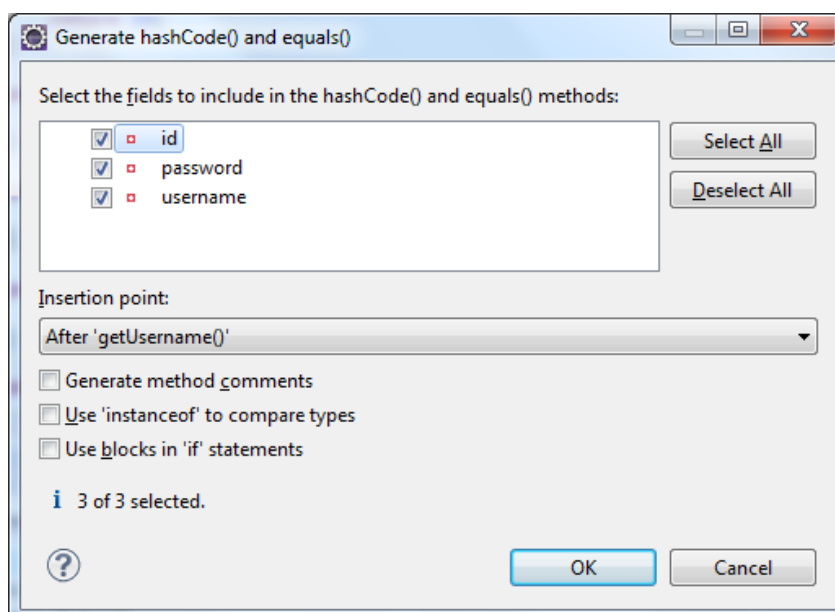
**24. Alt + Shift + S, O:** Generates constructor using fields. This shortcut is very useful when you want to generate code for a constructor that takes class's fields as its parameters. The following dialog appears:



**25. Alt + Shift + S, C:** Generates Constructors from Superclass. A common example for using this shortcut is when creating a custom exception class. In this case, we need to write some constructors similar to the Exception superclass. This shortcut brings the **Generate Constructors from Superclass** dialog which allows us to choose the constructors to be implemented in the subclass:



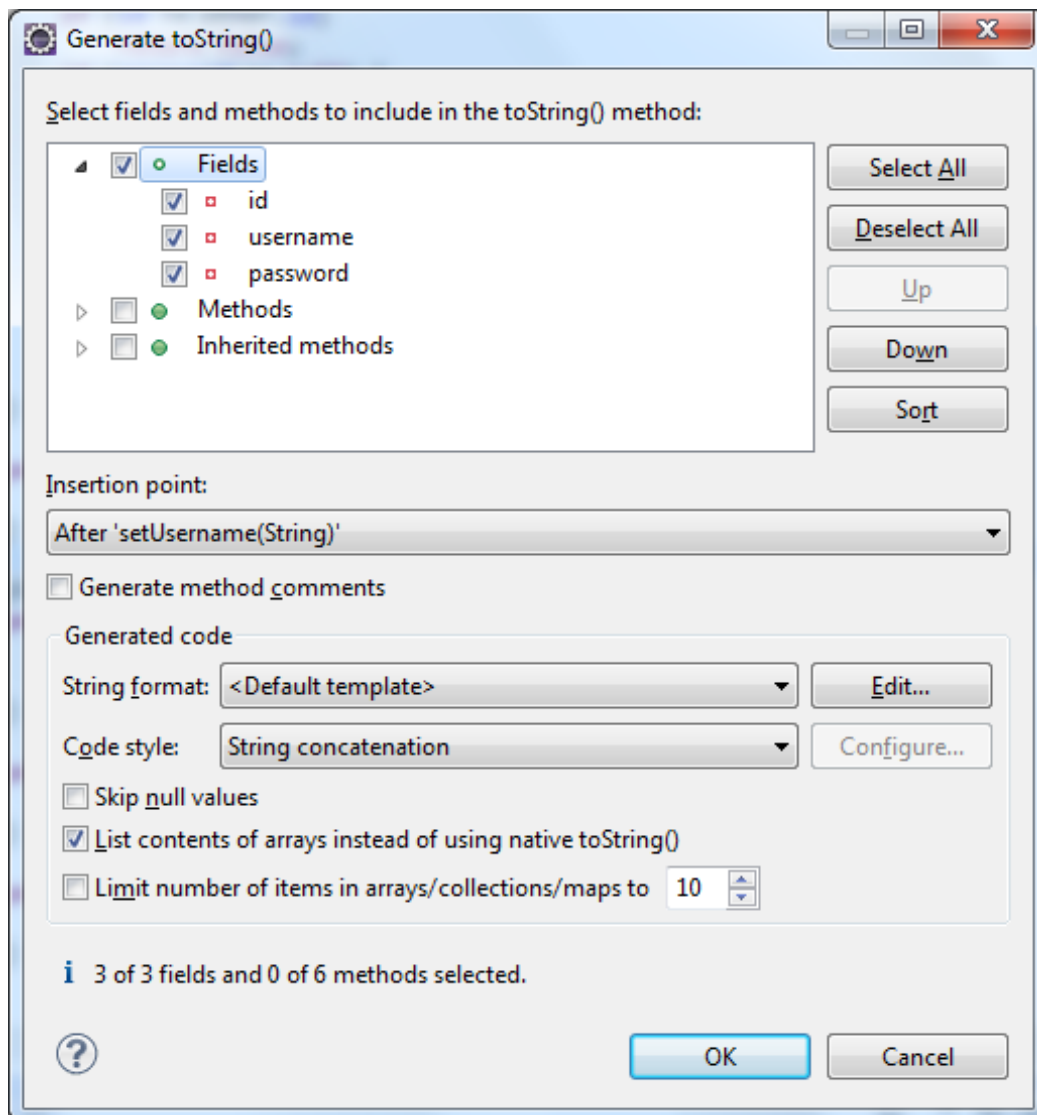
**26. Alt + Shift + S, H:** Generates hashCode() and equals() methods, typically for a JavaBean/POJO class. The class must have non-static fields. This shortcut brings the **Generate hashCode() and equals()** dialog as below:





Select the fields to be used in hashCode() and equals() method, and then click **OK**.

**27. Alt + Shift + S, S:** Generates toString() method. This shortcut comes in handy if we want to override toString() method that returns information of relevant fields of the class. This brings the **Generate toString()** dialogas below:



----- The END -----