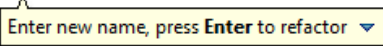


## 8 Eclipse Shortcut Keys for Code Refactoring

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;
22
```



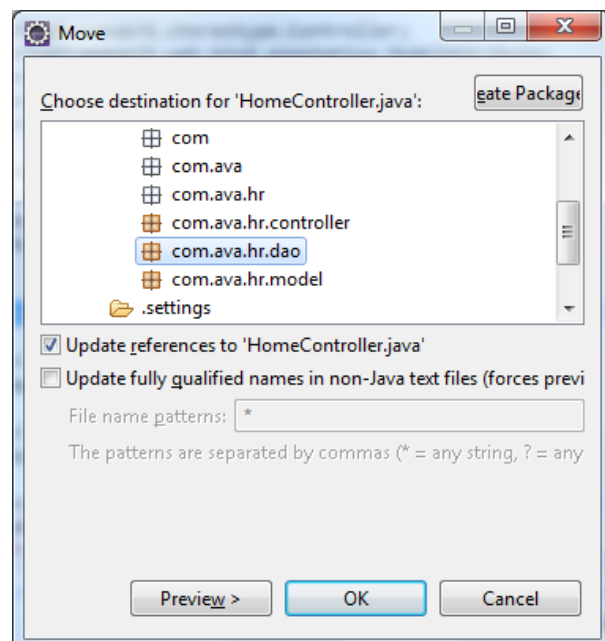
Type new name and press **Enter** when done, Eclipse automatically updates all related references for the new name, including ones found in other classes.

**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:

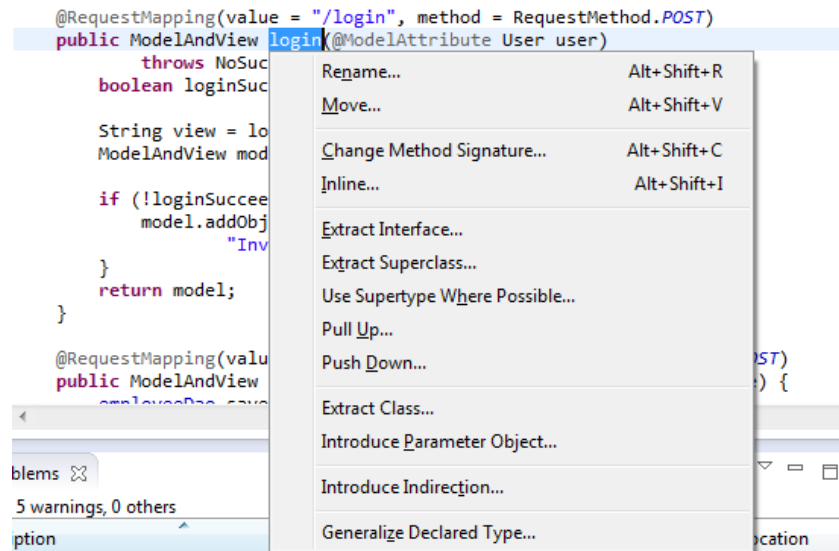
```
15
16 @Transactional
17 public void save(Employee employee) {
18     Session session = sessionFactory.getCurrentSession();
19
20     session.save(employee);
21 }
22
```

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

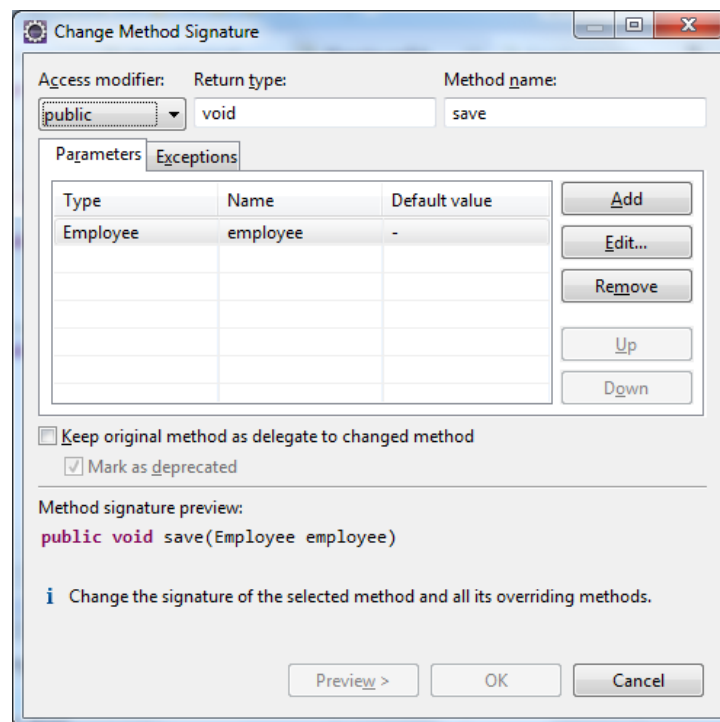
**3. Alt + Shift + V:** Moves a class, method to another destination. For example, select a class name and press this shortcut, the **Move** dialog appears. Choose a destination and then click **OK**:



**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:

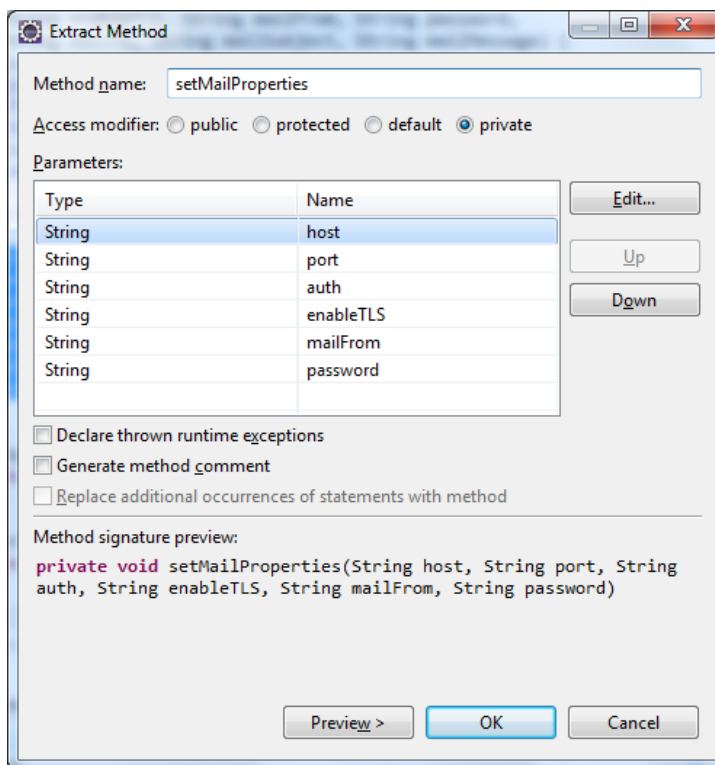
```

19
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 }
40

```

**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:

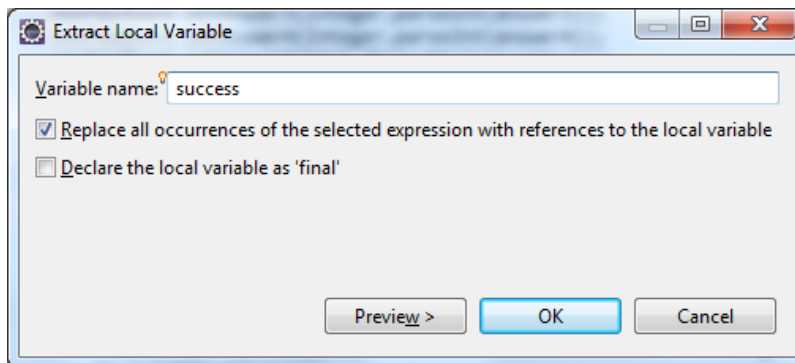
```

169
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 }
183

```

**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:

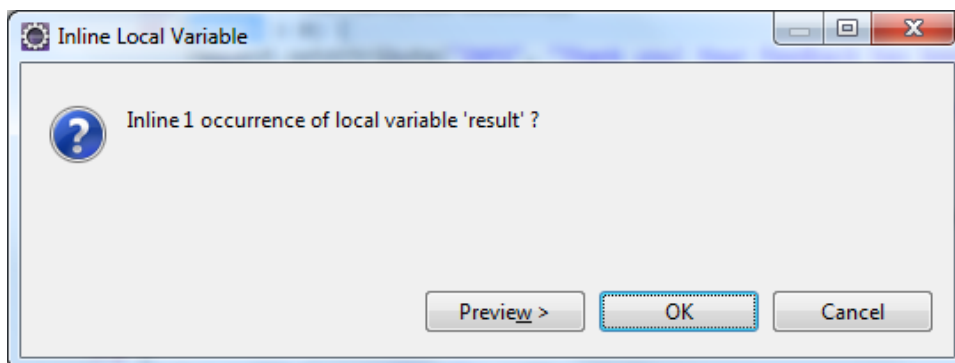
```
169
170     FeedbackDAO dao = new FeedbackDAO(dbURL, dbUser, dbPass);
171     try {
172         int result = dao.save(newFeedback);
173         boolean success = result > 0;
174         if (success) {
175             request.setAttribute("INFO", "Thank you! Your feedback has been sent.");
176         } else {
177             request.setAttribute("ERROR", "Cannot save feedback. Please try again.");
178         }
179         request.setAttribute("Feedback", newFeedback);
180     } catch (Exception ex) {
181         request.setAttribute("ERROR", "Cannot save feedback: " + ex.getMessage());
182         ex.printStackTrace();
183     }
```

**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:

```
169
170     FeedbackDAO dao = new FeedbackDAO(dbURL, dbUser, dbPass);
171     try {
172         int result = dao.save(newFeedback);
173         if (result > 0) {
174             request.setAttribute("INFO", "Thank you! Your feedback has been sent.");
175         } else {
176             request.setAttribute("ERROR", "Cannot save feedback. Please try again.");
177         }
178         request.setAttribute("Feedback", newFeedback);
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:

```
169 FeedbackDAO dao = new FeedbackDAO(dbURL, dbUser, dbPass);
170 try {
171     if (dao.save(newFeedback) > 0) {
172         request.setAttribute("INFO", "Thank you! Your feedback has been sent.");
173     } else {
174         request.setAttribute("ERROR", "Cannot save feedback. Please try again.");
175     }
176     request.setAttribute("Feedback", newFeedback);
177 } catch (Exception ex) {
178     request.setAttribute("ERROR", "Cannot save feedback: " + ex.getMessage());
179     ex.printStackTrace();
180 }
181
182
```