编写干净代码有助于其他开发人员阅读、理解和维护您编写的代码。但是，并不是所有人都赞成 “漂亮”、“精密” 或 “干净” 等定义。不同的开发人员拥有不同的风格和审美鉴赏力。到现在为止，Eclipse 通过少量修饰以一种简单的功能方式设定了导入代码的格式。Eclipse V3.3 中对这些操作进行了扩展，从而提供了更宽泛的清理功能级别。Eclipse V3.3 允许您清除代码、添加缺少的代码并应用某种编码样式。向导将帮助您配置清理设置并将其存储起来以供稍后使用。

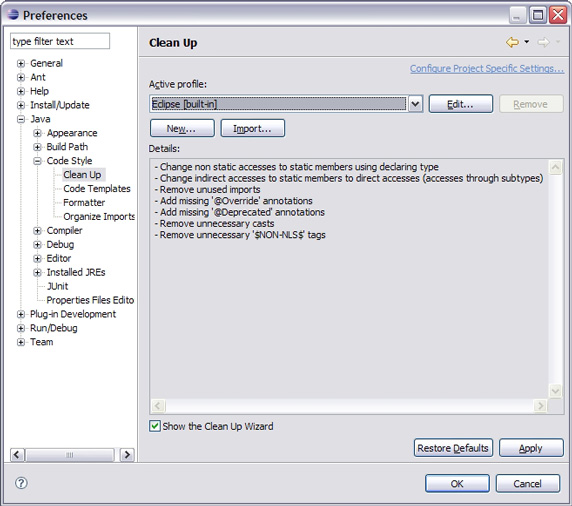
我们将讨论清理的基本概念并提供有助于保持代码干净的工具的概览。

**用配置文件管理清理配置**

某个具体的清理配置被称为一个 配置文件。配置文件可以保存，这样您就可以将设置提供给其他人或把来自早期项目和其他人的设置应用到当前代码中。根据组织的编码约定，配置文件可以应用于所有的 Eclipse 项目，这样便可以在所有开发团队中获得相同的代码样式。

Eclipse 首选项为配置文件提供了管理功能。配置文件可被创建、编辑和删除。您可以指定在工作区中全局使用的配置文件。当您第一次打开工作区首选项并浏览到 Java > Code Style > Clean Up 时，您将看到活动配置文件 Eclipse [built-in]。此内置配置文件已经过预先配置并且是随 Eclipse 一起交付的。内置配置文件有两个：Eclipse 和 Save Participant。这两个配置文件定义了两个最小限度的清理配置，可基本上删除不必要代码。您可以通过将它们设为活动配置文件来查看这些内置配置文件的设置。所有详细信息如详细信息区域所示。

**图 1. 内置的详细信息**



现有配置文件可作为各类模板使用，并且可以扩展和定制这些模板。因此，从下拉式菜单中选择现有配置文件作为活动配置文件并单击 edit。内置配置文件不能更改。可以使用内置配置文件作为您自己的配置文件的基础或简单地将内置配置文件按原样应用到代码中。

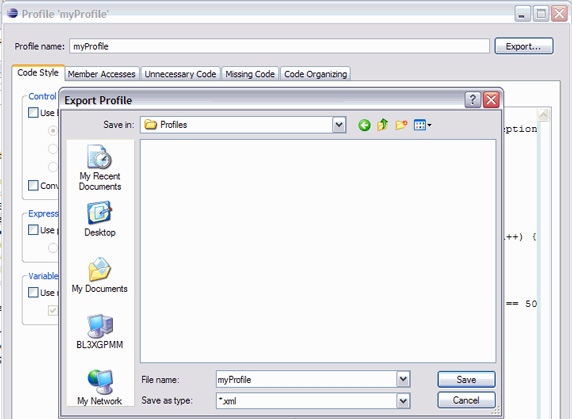
要创建您自己的配置文件，请单击 New。为配置文件命名并从下拉式菜单中选择现有配置文件进行初始化。取消选中 Open the edit dialog now 并单击 OK。

**图 2. 新建配置文件**



要共享配置文件，请使用导出功能。要打开配置文件，请单击 Edit，然后单击 Export。为配置文件命名，然后单击 OK，接下来就可以共享配置文件了。

**图 3. 导出配置文件**



要将 XML 文件中的外部配置应用到项目中，必须先将其导入。在主清理首选项中单击 Import，选择文件，然后单击 OK。

[**回页首**](http://www.ibm.com/developerworks/cn/opensource/os-eclipse-clean/#ibm-pcon)

**清理类型**

清理设置分为五个主要类别。每个类别都显示在相应选项卡中，选项卡由设置部分和预览部分构成。预览用于立即显示设置对代码的影响。尝试使用这些设置并观察预览中的代码发生怎样的变化，以了解每种更改将怎样影响您的代码。当您在主清理首选项中单击 Edit，或在创建新配置文件时选择 Open the edit dialog now，将弹出编辑所有这些设置的对话框。

下面将讨论可用的设置及其优缺点。由于许多设置可以风格各异，因此将不提供任何推荐。注意，如果不选择具体选项，您的代码将保持键入时的样子。

**Code Style**

第一个选项卡用于处理编码样式并定义应当如何显示块、表达式和变量声明。

**Control statements**

选择　Use blocks in if, while, for, and do statements　可定义使用大括号的位置。大括号有助于提高代码可读性。使用大括号时更易于看出哪些内容属于同一个单元。在添加属于 if 或 else 条件的另一条语句时，还可以帮助避免错误。另一方面，大括号太多会使代码变得臃肿并可能使代码难于处理。  
  
选择 Convert for loops to enhanced 将使用 for 循环符号，这是由 Java™ V5.0 引入的，用于减少代码。注意，此转换不具有向后兼容性。

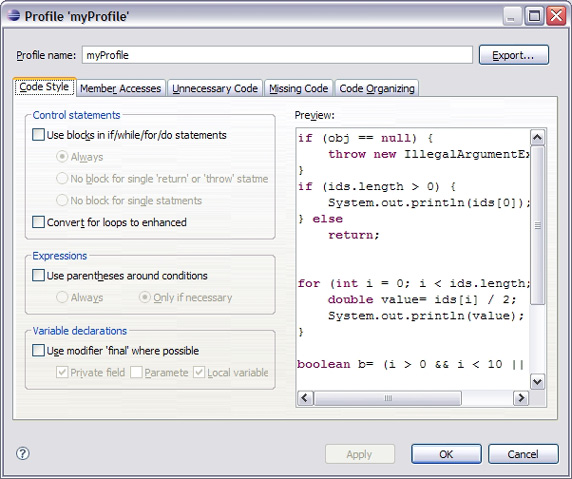
**Expressions**

选择 Use parentheses around conditions 可定义使用圆括号的位置。对于圆括号，请参阅以上关于大括号的讨论。

**Variable declarations**

选择 Use modifier 'final' where possible 可定义使用 final 关键字的位置。final 修饰符不但可用于声明不能更改的变量，而且还是强制永远设定私有字段的优秀选择。final 修饰符对于性能、健壮性和正确性至关重要。

**图 4. 编码样式**



**Member Accesses**

第二个选项卡允许您定义应当怎样访问类型成员。

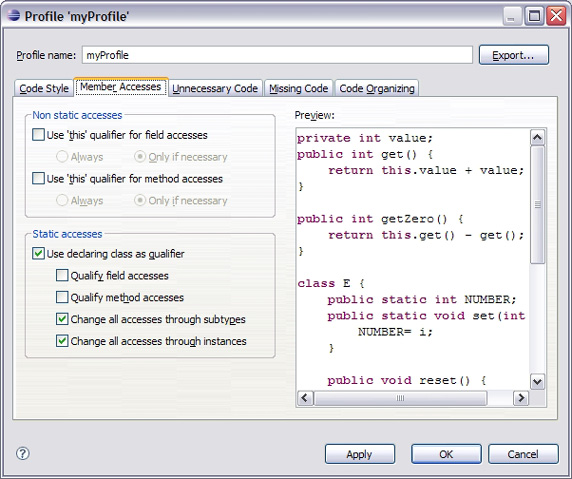
**Non static accesses**

需要使用 this 限定词访问字段或方法时选中该选项。this 限定词帮助您快速查看哪些字段或方法是正在编写的当前类的成员。它可以帮助您区分使用同一名称的字段和局部变量。

**Static accesses**

使用复选框来定义限定设置。可以通过声明类限定静态成员访问以更好地识别定义该成员的类型。另一方面，长类名可能使简单的成员访问变得臃肿并且使成员访问看上去不具有可读性或跨度多行代码。

**图 5. 成员访问**



**Unnecessary Code**

第三个选项卡允许您为删除未使用的代码和多余代码指定设置。

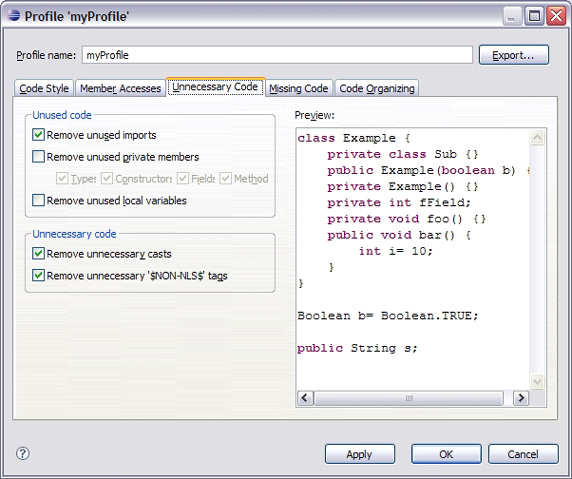
**Unused code**

使用第一个复选框可以删除未使用的导入。如果不使用 Organize imports 或 Strg+Shift+o 组合键，则这样自动删除不使用的导入将帮助您使项目保持尽可能地小且没有任何未使用的库。  
  
使用第二个复选框可以删除未使用的私有成员。私有成员只能在保存类中访问。如果不使用私有成员，则不需要它们。私有成员只会增加编译器的开销。重构代码后如果拥有大量未使用的遗留代码，则删除会十分高效。另一方面，这样做会十分危险。假定您在设计尚未使用但可能在将来使用的新方法的原型。此清理选项如果处于激活状态将删除这些方法，并且可能会错过重要的工作。

**Unnecessary code**

使用第一个复选框来删除多余代码。多余代码的存在会导致运行时的额外开销，这取决于您使用的编译器。  
  
使用第二个复选框可以去掉多余的 $NON-NLS$ 标记。这些标记仅由 Eclipse 使用来识别不应当具体化的字符串。

**图 6. 多余代码**



**Missing Code**

第四个选项卡允许您添加缺少代码。

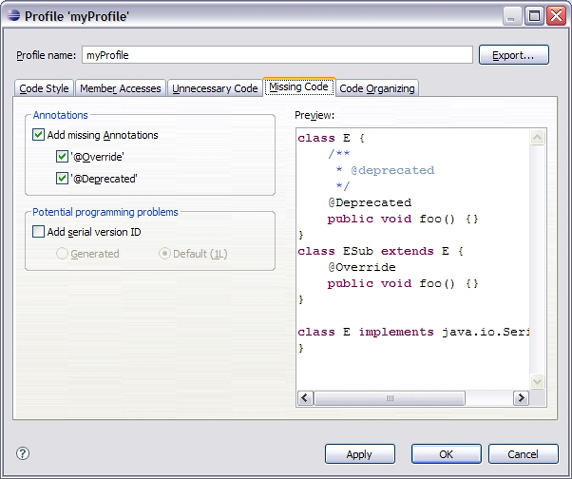
**Annotations**

定义要将哪些注释添加到代码中。因为当使用不赞成使用的方法或者覆盖已标记方法未能正确覆盖某一个超类中的方法时，Java V5.0@Override 或 @Deprecated 注释将帮助编译器生成错误。注意，这些注释不具有向后兼容性。

**Potential programming problems**

如果需要添加序列版本 ID，则定义此设置。对于实现 Serializable 接口的类，建议这些类使用私有静态 final 变量定义一个序列版本 UID。此变量可以自动生成。它用于在反序列化过程中检查兼容性。

**图 7. 缺失的代码**



**Code Organizing**

第五个选项卡虽然位于最后但不可忽略其重要性，它用于帮助您组织代码。

**Formatter**

定义在代码清理内是否应当使用格式程序。查阅格式程序首选项：Preferences > Java > Code Style > Formatter。

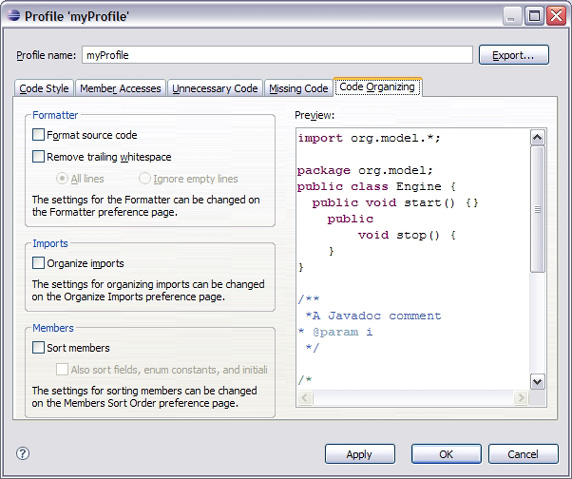
**Imports**

定义是否应当使用 Organize Imports。查阅组织导入首选项：Preferences > Java > Code Style > Organize Imports。

**Members**

定义是否需要按字母序把成员分类。有时，将成员按字母序分类非常好可以更好地浏览代码。不过，也可能有人会反对这样做。假定您将组织您的代码，使彼此调用的方法靠近放置在一起以便于进行代码浏览。排序可能会重新组织这些方法，并且它们可能不是所需的顺序。概览视图提供了一项优秀的功能用于给视图中的成员排序，但是不能给代码中的成员排序。具体配置和给成员排序的方法可以在 Preferences > Java > Appearance > Members Sort Order 中找到。

**图 8. 代码组织**

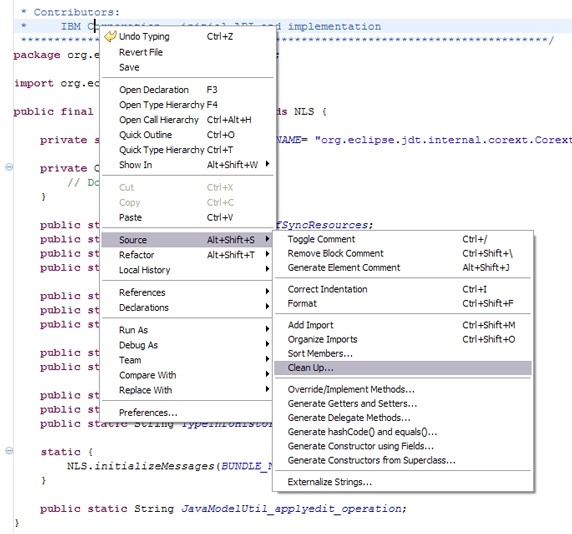


[**回页首**](http://www.ibm.com/developerworks/cn/opensource/os-eclipse-clean/#ibm-pcon)

**如何应用配置文件**

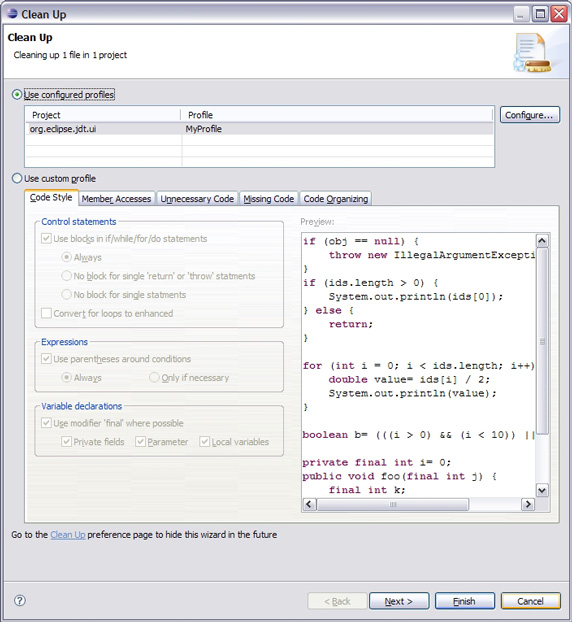
在创建了清理配置文件后，可以通过多种方法将其应用到代码中。最简单的方法是在 Java 编辑器中打开上下文菜单并选择 Source > Clean Up。

**图 9. 打开清理向导**



此操作将打开清理向导，如下所示：

**图 10. 清理向导**

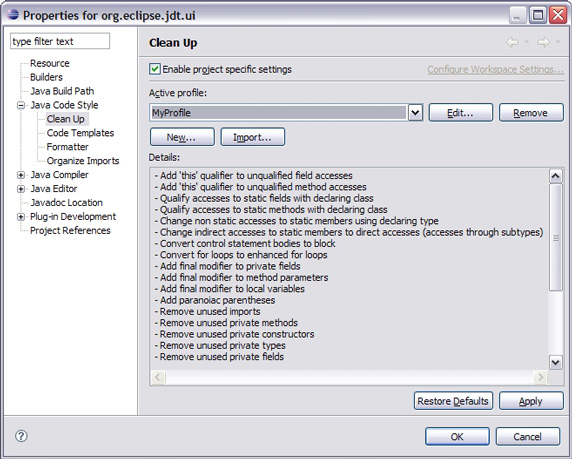


向导将引导您完成清理选定源代码的过程。左上的描述将显示将被清理的项目及编译单元的数目。通常，我们会将已配置的配置文件应用到编译单元中。但是，可以在应用代码清理之前先进行自定义。如果需要查看某个设置影响代码的结果，这可能十分有帮助。

只要任何 Java 项目、软件包或 Java 文件至少包含一个编译单元，清理向导就可以在其中启动。例如，您可以选择工作区中的所有 Java 项目并启动向导。执行清理将影响工作区中可以用选定配置文件重构的所有编译单元。

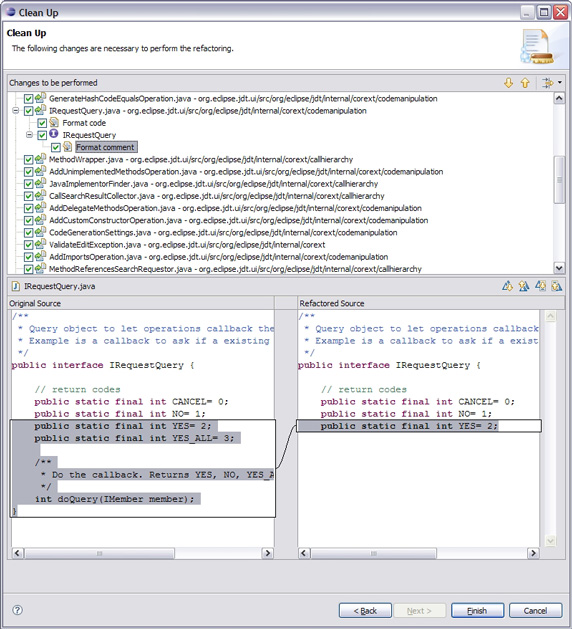
默认情况下，有一个用于整个工作区的全局清理配置文件。但是，也可以在项目属性中启用特定于项目的清理。每个项目都可以有自己的清理配置文件。要启用这种清理操作，只需打开项目属性并浏览到 Java Code Style > Clean Up，如下所示：

**图 11. 应用配置文件**



要预览清理结果，请在清理向导中单击 Next。这时向导将计算代码更改数目。根据选定的编译单元数目，完成此过程可能需要花费一段时间。在下一个页面中，将为您呈现将要应用的更改。

**图 12. 预览结果**



该树列出了将受清理影响的所有编译单元。您可以进入到树中去选择编译单元的不同更改。选择更改将在比较视图中显示初始源代码和经过重构的源代码。在查看了更改后，您可能不希望应用所有更改。在这种情况下，您可以简单地取消选中不必执行的更改。单击 Finish 将执行整个清理操作。

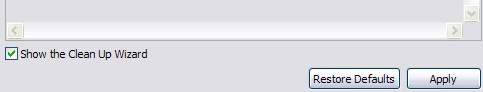
[**回页首**](http://www.ibm.com/developerworks/cn/opensource/os-eclipse-clean/#ibm-pcon)

**注意事项**

诸如 Convert for loops to enhanced 或 Add missing annotations 之类的某些清理重构操作是与 Java 代码依从性 5.0 或 6.0 绑定的，并且仅当源代码是根据要求的 Java 版本编译的才能应用。清理向导允许您选择那些重构选项，而不论使用的是哪一个 Java 版本。因此如果您想知道没有转换 for 循环或者未能正确注释掉不赞成使用的方法的原因，请在 Preferences > Java > Compiler 下查看工作区或项目的编译器依从级别。

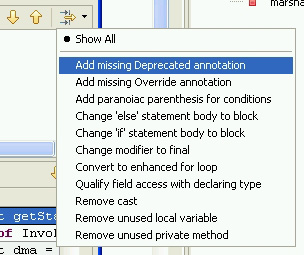
在多次运行清理向导并且配置文件已经正确设置后，您可能不希望每次执行清理操作时都在向导中多次进行单击。在这种情况下，您只需在Preferences > Java > Code Style > Clean Up 下的配置文件设置页面中禁用向导。

**图 13. 隐藏/显示清理向导**



在各种资源上执行代码清理通常会导致很多更改。向导允许您在应用之前预览那些更改。但是，如果数百计文件都受到影响则会使预览不方便，尤其是当您需要在预览过程中查找某个更改时。使用过滤器选项可以缩小预览页面中显示的更改列表。通常，几乎所有文件都会受到源代码格式操作的影响，但是添加缺失的不赞成注释影响的文件却不多。在这种情况下，过滤器将通过过滤其他更改来帮助您找到那些文件。过滤器位于预览页面的右上角。

**图 14. 应用过滤器**



清理操作不但可以手动执行，而且还可以在执行 Java 文件的保存操作期间执行。要启用此功能，请转到 Window > Preferences > Java > Editor > Save Actions 并选择附加操作。请按前述配置清理操作，然后就会在每次保存 Java 文件时都执行这些清理操作。请注意，执行那些操作有时会加大开销并且降低工作台的速度。同时，如果没有想到清理操作，您可能会在保存刚刚编写的代码后觉得很困惑为什么代码不太一样。

**图 15. 保存时执行清理**

