

避免复制与粘贴

张逸

www.agiledon.com

在开发过程中，当你发现代码可以 Copy-paste 时，就意味着代码出现了重复。这是一种典型的反模式。William J. Brown 等在著作 *AntiPatterns-Refactoring Software, Architecture, and Projects in Crisis* (即《反模式—危机中软件、架构和项目的重构》) 中认为这种形式的复用让开发的代码行数量虚假地增加，但是不能像其他形式的复用一样降低成本。Copy-Paste 代码的方式违背了 DRY (即不要重复你自己) 原则，使得多处地方出现了同样或者相似的代码。这是一种征兆，一旦在方法中或方法之间开始 Copy-Paste 操作，就意味着需要采用 Extract Method 重构手法。在提取方法之后，还可以根据情况利用 Move Method 重构手法，将其搬移到一个类中，然后在原来的调用处转为对该类方法的调用。或者利用 Replace Method with Method Object，将这些职责封装为专有的类。

在我的编程生涯中，碰到类似 Copy-Paste 的情况简直不胜枚举。在一次项目中，我们对开源项目 Jasper Report 进行了扩展。我们加入了对新报表类型 (CJT_REPORT) 的支持。在 ReportParameterAction 类中，我们需要对报表对象 ReportUnit 判断报表类型。于是，我在 ReportParameterAction 类中定义了如下的私有方法：

```
private void setReportUnitTypeByFileResource(ReportUnit reportUnit)
{
    final String JS_FILE_TYPE = "jrxml";
    ResourceReference reference = reportUnit.getMainReport();
    if (reference.isLocal()) {
        FileResource resource =
            (FileResource)reference.getLocalResource();
        String fileType = resource.getFileType();
        if (fileType.toLowerCase().equals(JS_FILE_TYPE)) {
            reportUnit.setReportType(ReportUnit.JS_REPORT);
        } else {
            reportUnit.setReportType(ReportUnit.CJT_REPORT);
        }
    }
}
```

```
}
```

该方法根据 `FileResource` 中的文件类型获得对应的报表类型。这一方法在 `ReportParameterAction` 类的 `createWrappers()` 方法中被调用。

```
protected Event createWrappers(RequestContext context) {
    ReportUnit reportUnit = loadReportUnit(context);
    setReportUnitTypeByFileResource(reportUnit);
    InitialRequestContext(context, reportUnit);

    int reportType = reportUnit.getReportType();
    if (reportType == ReportUnit.CJT_REPORT) { //..... }
    //.....
}
```

后来，我发现在 `EngineServiceImpl` 类中同样需要判断报表的类型。然而，`setReportUnitTypeByFileResource()` 方法却被定义为 `ReportParameterAction` 的私有方法，无法被 `EngineServiceImpl` 对象调用。最简单的做法是采用复制的方式，将这段代码复制到 `EngineServiceImpl` 中。好了，如果此时我们不能忍住 `copy-paste` 的简便所带来的诱惑，或许就会陷入重复的泥沼了。Copy 的动作绝对应该鸣起刺耳的警声。我们需要对这一做法保持足够的警惕。

通过分析 `setReportUnitTypeByFileResource()` 方法，我发现该方法需要操作的对象均与 `ReportUnit` 有关，而本身该方法的职责就是要获得 `ReportUnit` 的类型，并对其字段 `reportType` 进行设置，因此，完全可以将它搬移到 `ReportUnit` 中（Move Method 重构，不是吗？）。由于 `ReportUnit` 已经增加了对 `reportType` 字段的 `get` 和 `set` 访问方法，我发现它们与我要重构的方法实有异曲同工之处，因而决定将其搬移到 `getReportType()` 方法中：

```
public int getReportType() {
    if (hasReportTypeBeenSet()) {
        return reportType;
    }

    return getReportTypeByFileResource();
}

private int getReportTypeByFileResource() {
```

```
final String CJT_FILE_TYPE = "cjtxml";
int reportType = ReportUnit.JS_REPORT;
ResourceReference reference = this.getMainReport();
if (reference != null) {
    if (reference.isLocal()) {
        FileResource resource = (FileResource) reference
            .getLocalResource();
        if (resource != null) {
            String fileType = resource.getFileType();
            if (fileType.toLowerCase().equals(CJT_FILE_TYPE))
{
                reportType = ReportUnit.CJT_REPORT;
            }
        }
    }
}

return reportType;
}

private boolean hasReportTypeBeenSet(){
    if (reportType != ReportUnit.JS_REPORT
        && reportType != ReportUnit.CJT_REPORT) {
        return false;
    } else {
        return true;
    }
}
```

注意，在以上过程中，除了使用 Move Method 之外，我还运用了 Rename Method 以及 Extract Method 等重构手法。

现在，对报表类型的获取与调用就变得简单了，在 ReportParameterAction 与 EngineServiceImpl 类中，也规避了重复代码的产生。例如，在 createWrappers() 方法中的调用被更改为：

```
protected Event createWrappers(RequestContext context) {
```

```
ReportUnit reportUnit = loadReportUnit(context);
setReportUnitTypeByFileResource(reportUnit);
InitialRequestContext(context, reportUnit);

int reportType = reportUnit.getReportType();
if (reportType == ReportUnit.CJT_REPORT) { //.....}
//.....
}
```