

Functions

Doing One Thing!

One thing???



이미지 참조

<http://www.wired.com/threatlevel/2012/06/uncle-sam-piracy/>

한가지 일만 해야 한다 ?

Appropriate Size

```
Gentoo development discussion (... mostly) | Council elections now over; new cou  
a sec to check bug #148390 and let me know if that looks  
sane  
15:56 <+jeeves> strerror_work: https://bugs.gentoo.org/148390 nor, P2, All,  
brebs@sent.com->strerror@gentoo.org, NEW, pending,  
cryptsetup-luks incorrectly uses linux-info eclass  
15:56 <@Cardoe> got it  
15:56 * Cardoe sighs.  
15:56 <@Cardoe> I'm getting tired of everything changing so drastically  
15:56 <@Cardoe> and having to constantly adapt ebuilds and stuff  
15:57 <@Cardoe> and "arch" being more stable than "arch"  
15:57 <@dams> Attention ! please all smile, while I take a screenshot of my irssi  
- thank you !  
15:57 <@Christwhite> :(  
15:57 <@steev> ack  
15:57 <@steev> screenshots are evil!  
15:57 <@iggy> .1..  
15:57 <@DrEevil> :-o  
15:57 <@nightmorph> yo  
15:57 -!- fmccor is now known as fmccoraway  
15:57 <@steev> at least there is no one humping anyone else  
15:57 * nightmorph goes down in screenshot history  
15:57 * iggy humps steev  
15:58 <@nightmorph> steev: were saying?  
15:58 <@steev> oh noes!  
[15:58][@dams(+ei)][10:#gentoo-dev(cmnt)][Act: 2, 4, 5, 7, 8, 12, 13]  
[#gentoo-dev] []  
0 irc 1 zsh 2 root 3 admin @ryu - vendredi 22 septembre - 15:58
```



이미지 참조

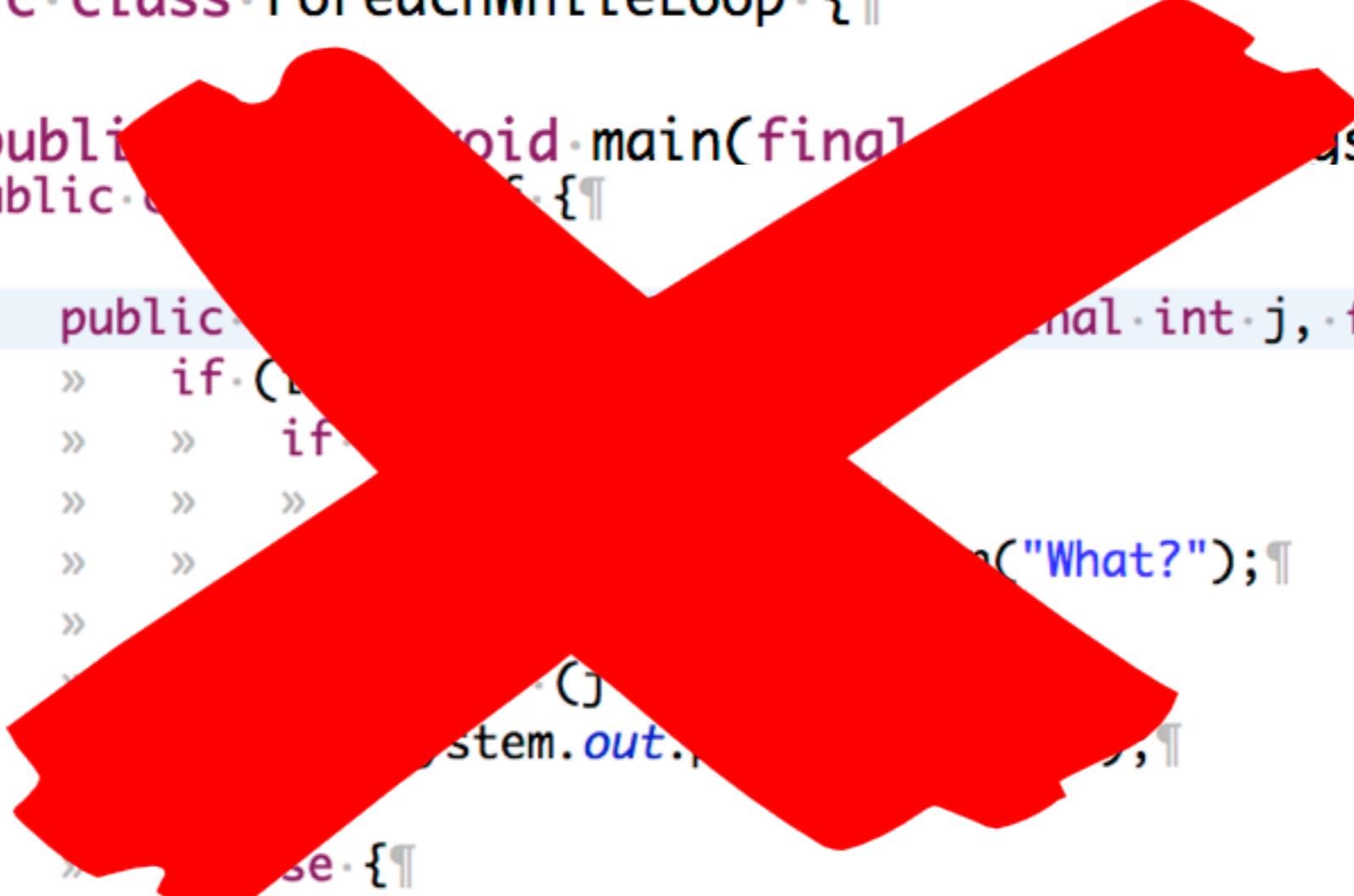
http://www.therandomshop.co.uk/housewarming-gifts-unique-funny-novelty-mugs-c-25_42.html

함수의 크기는 ?

- 80년대 vt100 터미널의 한 화면은 24줄, 80 컬럼을 표현할 수 있었다. 이 중 4줄은 에디터가 사용하므로 함수는 20줄 이내여야 한다고 했다.
- 요즘. big monitor. 100줄 이상, 150 컬럼 이상. 거기다 다중 모니터
- **Smaller than a Screenful**
- 4줄짜리 함수여야(4:Best, 5:Ok, 6~7:Not Bad, 8~:No!)



이미지 참조
<http://www.prosofteng.com/blog/drive-genius-3-how-to-guides/>



```
50 | <!-- middle content -->
51 | <div class="middle" style="padding:10px 0 0 0">
52 |   <div class="container_12">
53 |     <div class="grid_12 content">
54 |       <p style="padding-left: 18px"><b>Hate</b>
55 |       public class ForeachWhileLoop {
56 |
57 |       public void main(final String[] args) {
58 |         public {
59 |           public {
60 |             if (true) {
61 |               if (true) {
62 |                 final int j, final int k) {
63 |                   if (true) {
64 |                     if (true) {
65 |                       if (true) {
66 |                         if (true) {
67 |                           if (true) {
```

이미지 참조
<http://manoman12.wordpress.com/2011/10/>

indentation, while, nested if 등은 없어야



Small many functions

이미지 참조

<http://www.cosmospizzas.com/2012/05/cosmos-makes-the-cut/>

nice long descriptive name

function을 적절한 사이즈로 유지하려면 클래스가 추출된다. 클래스는 뒤에 큰 **function**을 감추고 있다.

The First Rule of Functions

- They should be small.
- They should be smaller than that.



이미지 참조
<http://berylloeb.wordpress.com/2011/01/04/just-one-thing/>

- they should be small. they should be smaller than that
- 큰 함수를 보면 클래스로 추출할 생각을 해야한다.
- 클래스는 일련의 변수들에 동작하는 오퍼레이션의 집합니다.

FitnessExample

```
public String testableHtml(PageData pageData, boolean includeSuiteSetup)
    throws Exception {
    WikiPage wikiPage = pageData.getWikiPage();
    StringBuffer buffer = new StringBuffer();
    if (pageData.hasAttribute("Test")) {
        if (includeSuiteSetup) {
            WikiPage suiteSetup = PageCrawlerImpl.getInheritedPage(SuiteResponder.SUITE_SETUP_NAME, wikiPage);
            if (suiteSetup != null) {
                WikiPagePath pagePath = suiteSetup.getPageCrawler().getFullPath(suiteSetup);
                String pagePathName = PathParser.render(pagePath);
                buffer.append("!include -setup .").append(pagePathName).append("\n");
            }
        }
        WikiPage setup = PageCrawlerImpl.getInheritedPage("SetUp", wikiPage);
        if (setup != null) {
            WikiPagePath setupPath = wikiPage.getPageCrawler().getFullPath(setup);
            String setupPathName = PathParser.render(setupPath);
            buffer.append("!include -setup .").append(setupPathName).append("\n");
        }
    }
    buffer.append(pageData.getContent());
    if (pageData.hasAttribute("Test")) {
        WikiPage teardown = PageCrawlerImpl.getInheritedPage("TearDown", wikiPage);
        if (teardown != null) {
            WikiPagePath tearDownPath = wikiPage.getPageCrawler().getFullPath(teardown);
            String tearDownPathName = PathParser.render(tearDownPath);
            buffer.append("\n").append("!include -teardown .").append(tearDownPathName).append("\n");
        }
        if (includeSuiteSetup) {
            WikiPage suiteTeardown = PageCrawlerImpl.getInheritedPage(SuiteResponder.SUITE_TEARDOWN_NAME, wikiPage);
            if (suiteTeardown != null) {
                WikiPagePath pagePath = suiteTeardown.getPageCrawler().getFullPath(suiteTeardown);
                String pagePathName = PathParser.render(pagePath);
                buffer.append("\n").append("!include -teardown .").append(pagePathName).append("\n");
            }
        }
    }
    pageData.setContent(buffer.toString());
    return pageData.getHtml();
}
```

- 3분만에 이 코드를 이해할 수 있나? 아닐 것이다.
 - 너무 많은 일이 수행되고 있다.
 - 너무 추상화 수준이 다양하다.
 - 이상한 문자열이 있다.
 - 수상한 함수 호출이 있다.
 - flag에 의해 제어되는 중첩 조건문이 있다.
- 약간의 리팩토링으로 함수의 의도를 포착할 수 있다.

Extract method object

```
public class FitnessExample {
    public String testableHtml(PageData pageData, boolean includeSuiteSetup)
        throws Exception {
        WikiPage wikiPage = pageData.getWikiPage();
        StringBuffer buffer = new StringBuffer();
        if (pageData.hasAttribute("Test")) {
            if (includeSuiteSetup) {
                WikiPage suiteSetup = PageCrawlerImpl.getInheritedPage(SuiteF
                if (suiteSetup != null) {
                    WikiPagePath pagePath = suiteSetup.getPageCrawler().getFu
                    String pagePathName = PathParser.render(pagePath);
                    buffer.append("!include -setup .").append(pagePathName).a
                }
            }
            WikiPage setup = PageCrawlerImpl.getInheritedPage("SetUp", wikiP
            if (setup != null) {
                WikiPagePath setupPath = wikiPage.getPageCrawler().getFullPa
                String setupPathName = PathParser.render(setupPath);
                buffer.append("!include -setup .").append(setupPathName).app
    }
}

public class FitnessExample {
    public String testableHtml(PageData pageData, boolean includeSuiteSetup)
        throws Exception {
        return new TestableHtmlMaker(pageData, includeSuiteSetup).invoke();
    }

    private class TestableHtmlMaker {
        private PageData pageData;
        private boolean includeSuiteSetup;

        public TestableHtmlMaker(PageData pageData, boolean includeSuiteSetup) {
            this.pageData = pageData;
            this.includeSuiteSetup = includeSuiteSetup;
        }

        public String invoke() throws Exception {
            WikiPage wikiPage = pageData.getWikiPage();
            StringBuffer buffer = new StringBuffer();
            if (pageData.hasAttribute("Test")) {
```

1. big function block을 extract method object한다.

- 큰 함수는 클래스를 숨기고 있는 것이다.
- 클래스란 일련의 변수들(함수의 인자, 지역변수)에 동작하는 함수들의 집합이다.
- TestableHtmlMaker로 Extract Method Object 한다.

Promote all local variables to fields

```
private class TestableHtmlMaker {  
    private PageData pageData;  
    private boolean includeSuiteSetup;  
  
    public TestableHtmlMaker(PageData pageData, boolean includeSuiteSetup)  
    {  
        this.pageData = pageData;  
        this.includeSuiteSetup = includeSuiteSetup;  
    }  
  
    public String invoke() throws Exception {  
        WikiPage wikiPage = pageData.getWikiPage();  
        StringBuffer buffer = new StringBuffer();  
        if (pageData.hasAttribute("Test")) {  
            if (includeSuiteSetup) {  
                WikiPage suiteSetup = PageCrawlerImpl.getInheritedPage(SuiteSetup.class);  
                if (suiteSetup != null) {  
                    buffer.append(suiteSetup.getWikiPage().getRawContent());  
                }  
            }  
            buffer.append(wikiPage.getWikiPage().getRawContent());  
        }  
        return buffer.toString();  
    }  
}
```

```
14 14  
15 15  
16 16  
17 17  
18 18  
19 19  
20 20  
21 21  
22 22  
23 23  
24 24  
25 25  
26 26  
27 27  
28 28  
29 29
```

```
private class TestableHtmlMaker {  
    private PageData pageData;  
    private boolean includeSuiteSetup;  
    private WikiPage wikiPage;  
    private StringBuffer buffer;  
  
    public TestableHtmlMaker(PageData pageData, boolean includeSuiteSetup)  
    {  
        this.pageData = pageData;  
        this.includeSuiteSetup = includeSuiteSetup;  
        wikiPage = pageData.getWikiPage();  
        buffer = new StringBuffer();  
    }  
  
    public String invoke() throws Exception {  
        if (pageData.hasAttribute("Test")) {  
            if (includeSuiteSetup) {  
                WikiPage suiteSetup = PageCrawlerImpl.getInheritedPage(SuiteSetup.class);  
                if (suiteSetup != null) {  
                    buffer.append(suiteSetup.getWikiPage().getRawContent());  
                }  
            }  
            buffer.append(wikiPage.getWikiPage().getRawContent());  
        }  
        return buffer.toString();  
    }  
}
```

2. promote all local variables to fields

- 생성자에서 field를 초기화
- 작은 함수로 추출할 때 인자 전달을 피할 수 있게한다.

Introduce Variable

```
public String invoke() throws Exception {
    if (pageData.hasAttribute("Test")) {
        if (includeSuiteSetup) {
            WikiPage suiteSetup = PageCrawlerImpl.getInheritedPage("SuiteSetup");
            if (suiteSetup != null) {
                WikiPagePath pagePath = suiteSetup.getPageCrawler();
                String pagePathName = PathParser.render(pagePath);
                buffer.append("!include -setup .").append(pagePathName);
            }
        }
        WikiPage setup = PageCrawlerImpl.getInheritedPage("SetUp", wikiPage);
        if (setup != null) {
            WikiPagePath setupPath = wikiPage.getPageCrawler().getFullPath();
            String setupPathName = PathParser.render(setupPath);
            buffer.append("!include -setup .").append(setupPathName);
        }
    }
    buffer.append(pageData.getContent());
    if (pageData.hasAttribute("Test")) {
        WikiPage teardown = PageCrawlerImpl.getInheritedPage("TearDown");
        if (teardown != null) {
            WikiPagePath tearDownPath = wikiPage.getPageCrawler().getFullPath();
            String tearDownPathName = PathParser.render(tearDownPath);
            buffer.append("!include -teardown .").append(tearDownPathName);
        }
        if (includeSuiteSetup) {
            WikiPage suiteTeardown = PageCrawlerImpl.getInheritedPage("SuiteTeardown");
            if (suiteTeardown != null) {
                WikiPagePath pagePath = suiteTeardown.getPageCrawler();
                String pagePathName = PathParser.render(pagePath);
                buffer.append("!include -teardown .").append(pagePathName);
            }
        }
    }
    pageData.setContent(buffer.toString());
    return pageData.getHtml();
}

if (pageData.hasAttribute("Test")) {
    if (includeSuiteSetup) {
        WikiPage suiteSetup = PageCrawlerImpl.getInheritedPage("SuiteSetup");
        if (suiteSetup != null) {
            String mode = "setup";
            WikiPagePath pagePath = wikiPage.getPageCrawler().getFullPath();
            String pagePathName = PathParser.render(pagePath);
            buffer.append("!include -" + mode + " .").append(pagePathName);
        }
    }
    WikiPage setup = PageCrawlerImpl.getInheritedPage("SetUp", wikiPage);
    if (setup != null) {
        String mode = "setup";
        WikiPagePath setupPath = wikiPage.getPageCrawler().getFullPath();
        String setupPathName = PathParser.render(setupPath);
        buffer.append("!include -" + mode + " .").append(setupPathName);
    }
}
buffer.append(pageData.getContent());
if (pageData.hasAttribute("Test")) {
    WikiPage teardown = PageCrawlerImpl.getInheritedPage("TearDown");
    if (teardown != null) {
        String mode = "teardown";
        WikiPagePath tearDownPath = wikiPage.getPageCrawler().getFullPath();
        String tearDownPathName = PathParser.render(tearDownPath);
        buffer.append("!include -" + mode + " .").append(tearDownPathName);
    }
}
if (includeSuiteSetup) {
    WikiPage suiteTeardown = PageCrawlerImpl.getInheritedPage("SuiteTeardown");
    if (suiteTeardown != null) {
        String mode = "teardown";
        WikiPagePath pagePath = wikiPage.getPageCrawler().getFullPath();
        String pagePathName = PathParser.render(pagePath);
        buffer.append("!include -" + mode + " .").append(pagePathName);
    }
}
```

3. Introduce Variable

반복되는 비슷한 코드 블록을 extract method해서 중복을 제거

- 먼저 비슷한 완전히 동일한 구조를 같도록 한다.
- 이를 위해 다른 점은 파라미터로 처리할 수 있도록 한다.
- 문자열에서 블록별로 다른 점을 변수로 추출

Extract Method

The diagram illustrates the process of extracting a repeating block of code into a new method. On the left, the original code is shown with several sections highlighted in blue, indicating the logic being moved. On the right, the refactored code shows the extracted logic in a new `includePage` method, which is highlighted in green.

```
    WikiPagePath pagePath = wikiPage.getPageCrawler().getFullPath();
    String pagePathName = PathParser.render(pagePath);
    buffer.append("!include -" + mode + " .").append(pagePathName);
}
WikiPage setup = PageCrawlerImpl.getInheritedPage("SetUp", wikiPage);
if (setup != null) {
    String mode = "setup";
    WikiPagePath setupPath = wikiPage.getPageCrawler().getFullPath();
    String setupPathName = PathParser.render(setupPath);
    buffer.append("!include -" + mode + " .").append(setupPathName);
}
buffer.append(pageData.getContent());
if (pageData.hasAttribute("Test")) {
    WikiPage teardown = PageCrawlerImpl.getInheritedPage("TearDown");
    if (teardown != null) {
        String mode = "teardown";
        WikiPagePath tearDownPath = wikiPage.getPageCrawler().getFullPath();
        String tearDownPathName = PathParser.render(tearDownPath);
        buffer.append("!include -" + mode + " .").append(tearDownPathName);
    }
    if (includeSuiteSetup) {
        WikiPage suiteTeardown = PageCrawlerImpl.getInheritedPage("SuiteTearDown");
        if (suiteTeardown != null) {
            String mode = "teardown";
            WikiPagePath pagePath = wikiPage.getPageCrawler().getFullPath();
            String pagePathName = PathParser.render(pagePath);
            buffer.append("!include -" + mode + " .").append(pagePathName);
        }
    }
}
pageData.setContent(buffer.toString());
return pageData.getHtml();
}

private void includePage(WikiPage page, String mode) throws Exception {
    WikiPagePath pagePath = wikiPage.getPageCrawler().getFullPath();
    String pagePathName = PathParser.render(pagePath);
    buffer.append("!include -" + mode + " .").append(pagePathName);
}
```

4. Extract Method

- 반복되는 코드를 새로운 메소드로 추출하여 중복 제거

Introduce Variable

```
public String invoke() throws Exception {
    if (pageData.hasAttribute("Test")) {
        if (includeSuiteSetup) {
            WikiPage suiteSetup = PageCrawlerImpl.getInheritedPage(SuiteResponder.SUITE_SETUP_NAME);
            if (suiteSetup != null) {
                String mode = "setup";
                includePage(suiteSetup, mode);
            }
        }
        WikiPage setup = PageCrawlerImpl.getInheritedPage("SetUp", pageName);
        if (setup != null) {
            String mode = "setup";
            includePage(setup, mode);
        }
    }
    buffer.append(pageData.getContent());
    if (pageData.hasAttribute("Test")) {
        WikiPage teardown = PageCrawlerImpl.getInheritedPage("TearDown");
        if (teardown != null) {
            String mode = "teardown";
            includePage(teardown, mode);
        }
        if (includeSuiteSetup) {
            WikiPage suiteTeardown = PageCrawlerImpl.getInheritedPage(SuiteResponder.SUITE_TEARDOWN_NAME);
            if (suiteTeardown != null) {
                String mode = "teardown";
                includePage(suiteTeardown, mode);
            }
        }
    }
    pageData.setContent(buffer.toString());
}

public String invoke() throws Exception {
    if (pageData.hasAttribute("Test")) {
        if (includeSuiteSetup) {
            String pageName = SuiteResponder.SUITE_SETUP_NAME;
            String mode = "setup";
            WikiPage suiteSetup = PageCrawlerImpl.getInheritedPage(pageName);
            if (suiteSetup != null)
                includePage(suiteSetup, mode);
            String pageName = "SetUp";
            String mode = "setup";
            WikiPage setup = PageCrawlerImpl.getInheritedPage(pageName, pageName);
            if (setup != null)
                includePage(setup, mode);
        }
        buffer.append(pageData.getContent());
        if (pageData.hasAttribute("Test")) {
            String pageName = "TearDown";
            String mode = "teardown";
            WikiPage teardown = PageCrawlerImpl.getInheritedPage(pageName);
            if (teardown != null)
                includePage(teardown, mode);
            if (includeSuiteSetup) {
                String pageName1 = SuiteResponder.SUITE_TEARDOWN_NAME;
                String mode1 = "teardown";
                WikiPage suiteTeardown = PageCrawlerImpl.getInheritedPage(pageName1);
                if (suiteTeardown != null)
                    includePage(suiteTeardown, mode1);
            }
        }
    }
}
```

5. Introduce Variable

- Extract Method를 위해서 반복 블록이 동일한 구조를 갖게하기 위해.

Extract Method

```
public String invoke() throws Exception {
    if (pageData.hasAttribute("Test")) {
        if (includeSuiteSetup) {
            String pageName = SuiteResponder.SUITE_SETUP_NAME;
            String mode = "setup";
            WikiPage suiteSetup = PageCrawlerImpl.getInheritedPage(pageName);
            if (suiteSetup != null)
                includePage(suiteSetup, mode);
        }
        String pageName = "SetUp";
        String mode = "setup";
        WikiPage setup = PageCrawlerImpl.getInheritedPage(pageName, mode);
        if (setup != null)
            includePage(setup, mode);
    }
    buffer.append(pageData.getContent());
    if (pageData.hasAttribute("Test")) {
        String pageName = "TearDown";
        String mode = "teardown";
        WikiPage teardown = PageCrawlerImpl.getInheritedPage(pageName);
        if (teardown != null)
            includePage(teardown, mode);
        if (includeSuiteSetup) {
            String pageName1 = SuiteResponder.SUITE_TEARDOWN_NAME;
            String mode1 = "teardown";
            WikiPage suiteTeardown = PageCrawlerImpl.getInheritedPage(pageName1);
            if (suiteTeardown != null)
                includePage(suiteTeardown, mode1);
        }
    }
    pageData.setContent(buffer.toString());
    return pageData.getHtml();
}

27   25
28   26
29   27
30   28
31   29
32   30
33   31
34   32
35   33
36   34
37   35
38   36
39   37
40   38
41   39
42   40
43   41
44   42
45   43
46   44
47   45
48   46
49   47
50   48
51   49
52   50
53   51
54   52
55   53
56   54
57   55
58   56
59   57
60   58
```

```
}

public String invoke() throws Exception {
    if (pageData.hasAttribute("Test")) {
        if (includeSuiteSetup) {
            String pageName = SuiteResponder.SUITE_SETUP_NAME;
            String mode = "setup";
            includeInherited(pageName, mode);
        }
        String pageName = "SetUp";
        String mode = "setup";
        includeInherited(pageName, mode);
    }
    buffer.append(pageData.getContent());
    if (pageData.hasAttribute("Test")) {
        String pageName = "TearDown";
        String mode = "teardown";
        includeInherited(pageName, mode);
        if (includeSuiteSetup) {
            String pageName1 = SuiteResponder.SUITE_TEARDOWN_NAME;
            String mode1 = "teardown";
            includeInherited(pageName1, mode1);
        }
    }
    pageData.setContent(buffer.toString());
    return pageData.getHtml();
}

private void includeInherited(String pageName, String mode) throws Exception {
    WikiPage suiteSetup = PageCrawlerImpl.getInheritedPage(pageName,
    if (suiteSetup != null)
        includePage(suiteSetup, mode);
}
```

6. Extract Method

- includeInherited로 Extract Method

Inline variable

```
public String invoke() throws Exception {
    if (pageData.hasAttribute("Test")) {
        if (includeSuiteSetup) {
            String pageName = SuiteResponder.SUITE_SETUP_NAME;
            String mode = "setup";
            includeInherited(pageName, mode);
        }
        String pageName = "SetUp";
        String mode = "setup";
        includeInherited(pageName, mode);
    }
    buffer.append(pageData.getContent());
    if (pageData.hasAttribute("Test")) {
        String pageName = "TearDown";
        String mode = "teardown";
        includeInherited(pageName, mode);
        if (includeSuiteSetup) {
            String pageName1 = SuiteResponder.SUITE_TEARDOWN_NAME;
            String mode1 = "teardown";
            includeInherited(pageName1, mode1);
        }
    }
    pageData.setContent(buffer.toString());
    return pageData.getHtml();
}

27   public String invoke() throws Exception {
28       if (pageData.hasAttribute("Test")) {
29           if (includeSuiteSetup)
30               includeInherited(SuiteResponder.SUITE_SETUP_NAME, "setup");
31           includeInherited("setUp", "setup");
32       }
33       buffer.append(pageData.getContent());
34       if (pageData.hasAttribute("Test")) {
35           includeInherited("TearDown", "teardown");
36           if (includeSuiteSetup)
37               includeInherited(SuiteResponder.SUITE_TEARDOWN_NAME, "tear");
38       }
39       pageData.setContent(buffer.toString());
40       return pageData.getHtml();
41   }

42   private void includeInherited(String pageName, String mode) throws E;
43       WikiPage suiteSetup = PageCrawlerImpl.getInheritedPage(pageName,
44           if (suiteSetup != null)
45               includePage(suiteSetup, mode);
46   }

47   private void includePage(WikiPage page, String mode) throws Exception {
48       WikiPagePath pagePath = wikiPage.getPageCrawler().getFullPath(pa;
49       String pagePathName = PathParser.render(pagePath);
50
51
```

7. Inline variable

- 불필요한 지역 변수를 제거하기 위해 inline한다.
- 불필요한 괄호를 제거한다
 - if, loop 등에서 사용하는 괄호가 없을 때까지 extract method
 - 이렇게 extract method하면 function은 한가지 일만하게 된다.

Extract Method

```
public String invoke() throws Exception {
    if (pageData.hasAttribute("Test")) {
        if (includeSuiteSetup)
            includeInherited(SuiteResponder.SUITE_SETUP_NAME, "setup");
        includeInherited("SetUp", "setup");
    }
    buffer.append(pageData.getContent());
    if (pageData.hasAttribute("Test")) {
        includeInherited("TearDown", "teardown");
        if (includeSuiteSetup)
            includeInherited(SuiteResponder.SUITE_TEARDOWN_NAME, "tear");
    }
    pageData.setContent(buffer.toString());
    return pageData.getHtml();
}

private void includeInherited(String pageName, String mode) throws E:
WikiPage suiteSetup = PageCrawlerImpl.getInheritedPage(pageName,
if (suiteSetup != null)
    includePage(suiteSetup, mode);
}
```



```
public String invoke() throws Exception {
    if (pageData.hasAttribute("Test"))
        includeSetups();
    buffer.append(pageData.getContent());
    if (pageData.hasAttribute("Test"))
        includeTeardowns();
    pageData.setContent(buffer.toString());
    return pageData.getHtml();
}

private void includeTeardowns() throws Exception {
    includeInherited("TearDown", "teardown");
    if (includeSuiteSetup)
        includeInherited(SuiteResponder.SUITE_TEARDOWN_NAME, "tear");
}

private void includeSetups() throws Exception {
    if (includeSuiteSetup)
        includeInherited(SuiteResponder.SUITE_SETUP_NAME, "setup");
    includeInherited("SetUp", "setup");
}
```

8. Extract Method

- includeSetups, includeTeardowns 2개의 메소드를 추출
- if 문장의 괄호를 없애기 위해

Merging If statement

```
public String invoke() throws Exception {  
    if (pageData.hasAttribute("Test"))  
        includeSetups();  
    buffer.append(pageData.getContent());  
    if (pageData.hasAttribute("Test"))  
        includeTeardowns();  
    pageData.setContent(buffer.toString());  
    return pageData.getHtml();  
}
```

```
27 27  
28 28  
29 29  
30 30  
31 31  
32 32  
33 33  
34 34  
35 35  
36 36
```

```
public String invoke() throws Exception {  
    if (pageData.hasAttribute("Test")) {  
        includeSetups();  
        buffer.append(pageData.getContent());  
        includeTeardowns();  
    }  
    pageData.setContent(buffer.toString());  
    return pageData.getHtml();  
}
```

StringBuffer -> String

```
private class TestableHtmlMaker {  
    private PageData pageData;  
    private boolean includeSuiteSetup;  
    private WikiPage wikiPage;  
    private StringBuffer buffer;  
  
    public TestableHtmlMaker(PageData pageData, boolean includeSuiteSetup) {  
        this.pageData = pageData;  
        this.includeSuiteSetup = includeSuiteSetup;  
        wikiPage = pageData.getWikiPage();  
        buffer = new StringBuffer();  
    }  
  
    public String invoke() throws Exception {  
        if (pageData.hasAttribute("Test")) {  
            includeSetups();  
            buffer.append(pageData.getContent());  
            includeTeardowns();  
            pageData.setContent(buffer.toString());  
        }  
        return pageData.getHtml();  
    }  
  
    private void includeTeardowns() throws Exception {  
        includeInherited("TearDown", "teardown");  
        if (includeSuiteSetup)  
            includeInherited(SuiteResponder.SUITE_TEARDOWN_NAME, "teardown");  
    }  
  
    private void includeSetups() throws Exception {  
        includeInherited("Setup", "setup");  
        if (includeSuiteSetup)  
            includeInherited(SuiteResponder.SUITE_SETUP_NAME, "setup");  
    }  
  
    private class TestableHtmlMaker {  
        private PageData pageData;  
        private boolean includeSuiteSetup;  
        private WikiPage wikiPage;  
        private String content;  
  
        public TestableHtmlMaker(PageData pageData, boolean includeSuiteSetup) {  
            this.pageData = pageData;  
            this.includeSuiteSetup = includeSuiteSetup;  
            wikiPage = pageData.getWikiPage();  
            content = new String();  
        }  
  
        public String invoke() throws Exception {  
            if (pageData.hasAttribute("Test")) {  
                content = includeSetups();  
                content += pageData.getContent();  
                content += includeTeardowns();  
                pageData.setContent(content);  
            }  
            return pageData.getHtml();  
        }  
  
        private String includeTeardowns() throws Exception {  
            String teardowns = "";  
            teardowns += includeInherited("TearDown", "teardown");  
            if (includeSuiteSetup)  
                teardowns += includeInherited(SuiteResponder.SUITE_TEARDOWN_NAME, "teardown");  
            return teardowns;  
        }  
    }  
}
```

10. StringBuffer를 String으로 변경하기

- 코드의 의도를 명확하게 하기 위해.

Extract Method

```
public String invoke() throws Exception {
    if (pageData.hasAttribute("Test")) {
        content = includeSetups();
        content += pageData.getContent();
        content += includeTeardowns();
        pageData.setContent(content);
    }
    return pageData.getHtml();
}

private String includeTeardowns() throws Exception {
    String teardowns = "";
    teardowns += includeInherited("TearDown", "teardown");
    if (includeSuiteSetup)
        teardowns += includeInherited(SuiteResponder.SUITE_TEARDOWN_N_
}

```

28	28	public String invoke() throws Exception {
29	29	if (ifTestPage())
30	30	surroundPageWithSetupsAndTeardowns();
31	31	return pageData.getHtml();
32	32	}
33	33	
34	34	private boolean ifTestPage() throws Exception {
35	35	return pageData.hasAttribute("Test");
36	36	}
37	37	
38	38	private void surroundPageWithSetupsAndTeardowns() throws Exception {
39	39	content = includeSetups();
40	40	content += pageData.getContent();
41	41	content += includeTeardowns();
42	42	pageData.setContent(content);
43	43	}
44	44	

11. Extract Method

- surroundPageWithSetupsAndTeardowns로 if body를 추출한다.
- pageData.hasAttribute("Test")를 ifTestPage로 extract method
- if 문장의 불필요한 괄호 제거

Rename

```
public class FitnessExample {  
    public String testableHtml(PageData pageData, boolean includeSuiteSetup)  
        throws Exception {  
        return new TestableHtmlMaker(pageData, includeSuiteSetup).invoke();  
    }  
  
    private class TestableHtmlMaker {  
        private PageData pageData;  
        private boolean includeSuiteSetup;  
        private WikiPage wikiPage;  
        private String content;  
  
        public TestableHtmlMaker(PageData pageData, boolean includeSuiteSetup)  
            throws Exception {  
            this.pageData = pageData;  
            this.includeSuiteSetup = includeSuiteSetup;  
            wikiPage = pageData.getWikiPage();  
            content = new String();  
        }  
  
        public String invoke() throws Exception {  
            if (ifTestPage())  
                surroundPageWithSetupsAndTeardowns();  
            return pageData.getHtml();  
        }  
    }  
}  
  
public class FitnessExample {  
    public String testableHtml(PageData pageData, boolean includeSuiteSetup)  
        throws Exception {  
        return new SetupTeardownSurrounder(pageData, includeSuiteSetup).surround();  
    }  
  
    private class SetupTeardownSurrounder {  
        private PageData pageData;  
        private boolean includeSuiteSetup;  
        private WikiPage wikiPage;  
        private String content;  
  
        public SetupTeardownSurrounder(PageData pageData, boolean includeSuiteSetup)  
            throws Exception {  
            this.pageData = pageData;  
            this.includeSuiteSetup = includeSuiteSetup;  
            wikiPage = pageData.getWikiPage();  
            content = new String();  
        }  
  
        public String surround() throws Exception {  
            if (ifTestPage())  
                surroundPageWithSetupsAndTeardowns();  
            return pageData.getHtml();  
        }  
    }  
}
```

12. Rename class/method

- SetupTeardownSurrounder로 rename
- invoke를 surround로 rename

Refactored

```

public class FitnessExample {
    public String testableHtml(PageData pageData, boolean includeSuiteSetup)
        throws Exception {
        return new SetupTeardownSurrounder(pageData, includeSuiteSetup).surround();
    }

    private class SetupTeardownSurrounder {
        private PageData pageData;
        private boolean includeSuiteSetup;
        private WikiPage wikiPage;
        private String content;
        private PageCrawler crawler;

        public SetupTeardownSurrounder(PageData pageData, boolean includeSuiteSetup) {
            this.pageData = pageData;
            this.includeSuiteSetup = includeSuiteSetup;
            wikiPage = pageData.getWikiPage();
            content = "";
            crawler = wikiPage.getPageCrawler();
        }

        public String surround() throws Exception {
            if (isTestPage())
                surroundPageWithSetupsAndTeardowns();

            return pageData.getHtml();
        }

        private void surroundPageWithSetupsAndTeardowns() throws Exception {
            content += includeSetups();
            content += pageData.getContent();
            content += includeTeardowns();
            pageData.setContent(content);
        }
    }
}

```

```

private boolean isTestPage() throws Exception {
    return pageData.hasAttribute("Test");
}

private String includeTeardowns() throws Exception {
    String teardowns = "";
    teardowns += includeIfInherited("teardown", "TearDown");
    if (includeSuiteSetup)
        teardowns += includeIfInherited("teardown", SuiteResponder.SUITE_TEARDOWN_NAME);
    return teardowns;
}

private String includeSetups() throws Exception {
    String setups = "";
    if (includeSuiteSetup)
        setups += includeIfInherited("setup", SuiteResponder.SUITE_SETUP_NAME);
    setups += includeIfInherited("setup", "SetUp");
    return setups;
}

private String includeIfInherited(String mode, String pageName) throws Exception {
    WikiPage page = PageCrawlerImpl.getInheritedPage(pageName, wikiPage);
    if (page != null)
        return includePage(mode, page);
    return "";
}

private String includePage(String mode, WikiPage page) throws Exception {
    WikiPagePath pagePath = crawler.getFullPath(page);
    String pagePathName = PathParser.render(pagePath);
    return String.format("!include -ts %s\n", mode, pagePathName);
}
}

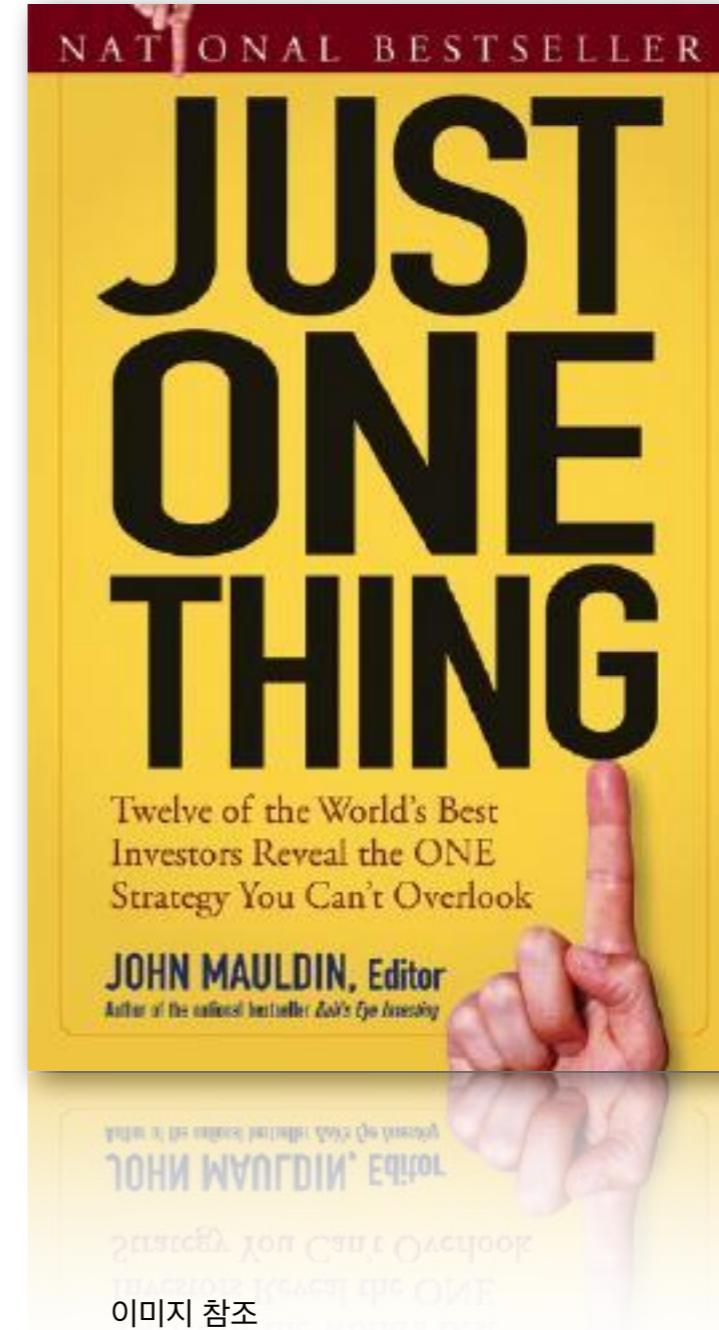
```

결론

- 리팩토링된 코드가 왜 읽고, 이해하기 쉬워졌는가 ?
- 어떻게 함수가 자신의 의도를 전달할 수 있게되었는가 ?
- Small – 함수의 첫번째 규칙이다.
- The Rules of functions
 - The first rule: They should be small.
 - The second rule: The should be smaller than that.
- Block이 적다는 것의 의미
 - if, else, while 문장 등의 내부 블록은 한줄이여야 한다(괄호가 필요 없어야한다).
 - 이 라인은 함수 호출일 것이다.
 - 이러한 방식이 함수를 작게 할 뿐 아니라 문서화 가치도 추가적으로 제공한다.
- Indenting 적다는 것의 의미
 - 함수는 중첩 구조를 갖을 만큼 크면 안된다.
 - 함수에서 들어쓰기는 한두단계까지만 사용해야 한다.
 - 이런 규칙은 함수를 읽고, 이해하기 쉽게한다.

Function should do one thing

- Doing One Thing!



이미지 참조

<http://www.amazon.com/Just-One-Thing-Investors-Strategy/dp/0470081813>

- 22
- 리팩토링 전의 함수는 하나 이상의 일을 한다.
 - creating buffers,
 - fetching pages,
 - searching for inherited pages,
 - rendering paths,
 - appending arcane strings,
 - and generating HTML
 - 리팩토링 후의 함수는 한가지 단순한 일만 한다.
 - including setups and teardowns into test pages.
 - 결과물이 3가지 일을 하는 것은 아닌가 ?
 - Determine whether the page is a test page.
 - If so, include setups and teardowns.
 - Render the page in HTML.
 - 그래서. 함수가 하나의 일을 하는 것인가 ? 아니면 3가지 일을 하는 것인가 ?
 - 함수가 수행하는 모든 일의 추상화 수준이 같다(All At Same Level...)
 - 함수의 각 스텝들은 함수 이름보다 한수준 낮은 추상화 수준을 갖는다.
 - 간략하게 표현하면
 - TO RenderPageWithSetupsAndTeardowns we:
 - check if the page is a test page
 - if so include the setups and teardowns.
 - else render the page in HTML.
 - 함수의 각 스텝들이 함수 이름이 갖는 추상화 수준보다 한 단계 낮은 것으로만 이뤄졌다면 함수는 한가지 일만 하는 것이다.
- Doing One Thing!**
- 원래 코드는 추상화 수준이 다른 많은 단계들을 포함하므로 한가지 이상의 일을 한다는 것이 명확하다.
 - 하지만 의미있게 최종적으로 줄이는 것은 어렵다(But it would be very hard to meaningfully shrink the final).
 - surround 메소드의 3줄을 includeSetupsAndTeardownsIfTestPage 메소드로 추출할 수 있다(We could extract the if statement into a function named includeSetupsAndTeardownsIfTestPage).
 - 하지만 이 작업은 추상화 수준의 변경 없이 코드를 단순히 재인용하게된다(but that simply restates the code without changing the level of abstraction).
 - 함수가 하나 이상의 일을 한다고 말할 수 있는 경우
 - 단순한 구현의 재인용이 아닌 이름으로 함수를 추출할 수 있을 때(if you can extract a function from it with a name that is not merely a restatement of its implementation).

Read code from top to bottom

- Read code from top to bottom



이미지 참조
<http://www.sellingskillsempowerment.com/blog/top-down-selling/>

Reading code from top to bottom.

- 우리는 코드를 top-down 이야기체로 읽기를 원한다.
- 현재 함수 바로 밑에 현재 함수 다음의 추상화 수준을 갖는 함수들이 있기를 원한다.
 - 한번에 한 수준씩 추상화 수준을 내려가면서 프로그램을 읽을 수 있다.
- 프로그램을 일련의 TO 문단으로 읽기를 원한다.
 - 각 TO 문단은 현재의 추상화 수준을 기술하고, 바로 다음 추상화 수준의 TO 문단을 참조한다.
- **To Paragraphs** 예
 - To include the setups and teardowns we
 - include setups,
 - then include the test page content,
 - then include the teardowns.
 - To include the setups we
 - include the suite setup if this is a suite,
 - then include the regular setup.
 - To include the suite setup we
 - search the parent hierarchy for the "SuiteSetUp" page
 - add an !include with the path of that page.

Are you out of mind?



이미지 참조

<http://www.time.com/time/arts/article/0,8599,1585726,00.html>

24

■ 우려

- 작은 함수들의 바다에서 길을 잃을 것이다.
- 작은 함수들이 많아서 function call overhead가 있을 것이다.
- 이렇게 작은 함수들을 많이 만드는 것은 많은 시간이 들 것이다.

■ 전혀 안 그렇다.

- sea of functions에서 길을 잃지 않을 것이다.
- 그런 functions들은 이름이 있기 때문이다.
- 당신이 부여한 이름
- 잘 이름 지어진 클래스에 위치한 함수들
- 이런 이름(함수, 클래스)들이 길을 잃지 않도록 하는 안내표시가 된다(좋은 이름을 사용한다면).

Efficiency

- Too Slow?????



이미지 참조
[http://news.kemco.or.kr/publish/php/articleview.php?
diaryDate=2008-03-26&idx=529§ion=11](http://news.kemco.or.kr/publish/php/articleview.php?diaryDate=2008-03-26&idx=529§ion=11)

Efficiency

- 옛날에는 function call overhead를 걱정했었다.
 - stack을 조작하는 것
 - 인자를 전달하는 것이 비용이 많이 발생하는 작업이었다.
 - 1970년대 C. microseconds 단위의 function call overhead가 중요했던 때.
- 지금 우리의 컴퓨터는 너무 빠르다. 컴파일러는 너무 좋고, 최적화되었다. 함수 호출은 nano seconds보다도 적게 걸린다.
- 이런 overhead를 걱정하는 것은 바람직하지 않다. 대신 우리의 코드를 잘 파티션해서 읽기 쉽게 해야한다.

Conclusion



conclusion

1. 1st rule: function should be small
2. 2nd rule: smaller than that
3. 이름을 잘 지으면 당신뿐 아니라 모든 사람들의 시간을 절약해 준다. 이정표 역할을 하기 때문에. 당신의 코드를 이해하기 위한 네비게이션 역할을 한다.
4. function call overhead의 efficiency를 걱정할 필요없다.
5. 함수를 작게 만들면 당신 뿐 아니라 모두의 시간을 절약해 준다.
6. 클래스는 큰 함수를 감춘다. 함수를 여러 클래스들에 잘 배분하려면 함수를 작게 만들어야 한다.
7. 함수는 한가지 일만 해야 하고, 한가지만 하는지 확신할 수 있는 유일한 방법은 extract till you drop이다.

참조

- <http://www.cleancoders.com/codecast/clean-code-episode-3/show>
- <http://book.daum.net/detail/book.do?bookid=BOK00009909239IN>