

# **Fundamentals: The Don't Repeat Yourself Principle Part 1**

Steve Smith

<http://pluralsight.com/>



# Outline

- DRY Defined
- Demo: Repetition in Code
- Analysis
- Demos: Refactoring to apply DRY
- Code Generation
- Repetition in Process
- Demo: Automation to apply DRY
- Summary and Related Fundamentals

I will not repeat myself  
I will not repeat myself  
I will not repeat myself  
I will not repeat myself  
I will not repeat myself  
I will not repeat myself  
I will not repeat myself  
I will not repeat myself  
I will not repeat myself

# DON'T REPEAT YOURSELF

Repetition is the root of all software evil

# Don't Repeat Yourself

*“Every piece of knowledge must have a single, unambiguous representation in the system.”*

The Pragmatic Programmer

*“Repetition in logic calls for abstraction. Repetition in process calls for automation.”*

97 Things Every Programmer Should Know

Variations include:

- Once and Only Once
- Duplication Is Evil (DIE)

# Demo

Violating DRY in a simple Data Warehouse Sample  
Overview



# Analysis

- Magic Strings/Values
- Duplicate logic in multiple locations
- Repeated if-then logic
- Conditionals instead of polymorphism
- Repeated Execution Patterns
- Lots of duplicate, probably copy-pasted, code
- Only manual tests
- Static methods everywhere

# Magic Strings / Values

```
34 using (var myConnection = new SqlConnection())
35 {
36     myConnection.ConnectionString =
37         "Data Source=localhost;Initial Catalog=Northwind;Integrated Security=True";
38
39
49     foreach (DataRow row in _invoiceTable.Rows)
50     {
51         Console.WriteLine("{0}-{1}-{2}", row[0], row[1], row[2]);
52     }
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93 if (_freightByShipperList.Count > 1)
94 {
95     _freightByShipperList[1].Freight =
96         (decimal)
97         _invoiceTable.Compute("sum(freight)", "shippername='" + _freightByShipperList[1].ShipperName + "'");
98     Console.WriteLine("{0}:{1:###.##}", _freightByShipperList[1].ShipperName, _freightByShipperList[1].Freight);
99 }
100
```

# Demo

Applying DRY to Remove Magic Strings





## Duplicate Logic in Multiple Locations

```

49         foreach (DataRow row in _invoiceTable.Rows)
50         {
51             Console.WriteLine("{0}-{1}-{2}", row[0], row[1], row[2]);
52         }
53     }
54
55     private void ShowEmployeeTable()
56     {
57         foreach (DataRow row in _employeeTable.Rows)
58         {
59             Console.WriteLine("{0}-{1}", row[0], row[1]);
60         }
61     }
62 }

```

```

34      using (var myConnection = new SqlConnection())
35      {
36          myConnection.ConnectionString =
55      using (var myConnection = new SqlConnection())
56      {
57          myConnection.ConnectionString =
143      using (var myConnection = new SqlConnection())
144      {
145          myConnection.ConnectionString =
177      using (var myConnection = new SqlConnection())
178      {
179          myConnection.ConnectionString =

```

# Demo

Applying DRY to Duplicate Logic in Multiple Locations



# Repeated if-then Logic

```
85     if (_freightByShipperList.Count > 0)
86     {
87         _freightByShipperList[0].Freight =
88             (decimal)
89             _invoiceTable.Compute("sum(fre
90             Console.WriteLine("{0}:{1:#.##}",
91         }
92
93     if (_freightByShipperList.Count > 1)
94     {
95         _freightByShipperList[1].Freight =
96             (decimal)
97             _invoiceTable.Compute("sum(fre
98             Console.WriteLine("{0}:{1:#.##}",
99         }
100
101     if (_freightByShipperList.Count > 2)
102     {
103         _freightByShipperList[2].Freight =
104             (decimal)
105             _invoiceTable.Compute("sum(fre
106             Console.WriteLine("{0}:{1:#.##}",
107         }
```

# Demo


Applying DRY to Repeated if-then Logic



# Conditional Instead of Polymorphism

- Example of Flags Over Objects anti-pattern
- Violates the Tell, Don't Ask principle (aka DIP)

```
126      foreach (Employee employee in _employees)
127      {
128          if (employee.IsManager)
129          {
130              employee.Bonus = totalFreight/10m;
131          }
132          else
133          {
134              employee.Bonus = totalFreight/1000m;
135          }
136      }
```



# Demo

Applying DRY to use of Conditional Instead of  
Polymorphism



# Summary

- Repetition breeds errors and waste
- Refactor code to remove repetition
- Recommended Reading:
  - The Pragmatic Programmer: From Journeyman to Master  
<http://amzn.to/b2gJdK>
  - 97 Things Every Programmer Should Know  
<http://amzn.to/cAse1Y>

For more in-depth **online** developer **training** visit



**on-demand** content from authors you **trust**