

8 Lines of Code

Greg Young

Simplicity

I am stupid to work otherwise.
Fancy code befuddles me.

```
public class DeactivateInventoryItem
{
    private readonly IRepository repository;

    public DeactivateInventoryItem(IRepository repository)
    {
        this.repository = repository;
    }

    public void Deactivate(Guid id, string reason)
    {
        var item = repository.GetById(id);
        item.Deactivate(reason);
        repository.Save(item);
    }
}
```

```
[Transactional]
[RequiresPermission("admin")]
[Logged]
[EatsExceptions]
[DoesBadThingsWhenYouArentWatching]
public class DeactivateInventoryItem
{
    private readonly IRepository repository;

    public DeactivateInventoryItem(IRepository repository)
    {
        this.repository = repository;
    }

    public void Deactivate(Guid id, string reason)
    {
        var item = repository.GetById(id);
        item.Deactivate(reason);
    }
}
```

Simplicity?

```
<bean id="moreComplexObject" class="example.ComplexObject">
  <!-- results in a setAdminEmails(java.util.Properties) call -->
  <property name="adminEmails">
    <props>
      <prop key="administrator">administrator@example.org</prop>
      <prop key="support">support@example.org</prop>
      <prop key="development">development@example.org</prop>
    </props>
  </property>
  <!-- results in a setSomeList(java.util.List) call -->
  <property name="someList">
    <list>
      <value>a list element followed by a reference</value>
      <ref bean="myDataSource" />
    </list>
  </property>
  <!-- results in a setSomeMap(java.util.Map) call -->
  <property name="someMap">
    <map>
      <entry key="an entry" value="just some string"/>
      <entry key="a ref" value-ref="myDataSource"/>
    </map>
  </property>
</bean>
```



```
[Transaction 1]
[RequiresAdmin("admin")]
[Log 1]
[EventArgs]
[DoesBad]
[ParentWatching]
public DeactivateInventoryItem
{
    private repository IRepository repository;

    public DeactivateInventoryItem(IRepository repository)
    {
        this.repository = repository;
    }

    public void Deactivate(Guid id, string reason)
    {
        var item = repository.GetById(id);
        item.Deactivate(reason);
    }
}
```



```
[Transactional]
[RequiresPermission("admin")]
[Logged]
[EatsExceptions]
[DoesBadThingsWhenYouArentWatching]
public class DeactivateInventoryItem
{
    private readonly IRepository repository;

    public DeactivateInventoryItem(IRepository repository)
    {
        this.repository = repository;
    }

    public virtual void Deactivate(Guid id, string reason)
    {
        var item = repository.GetById(id);
        item.Deactivate(reason);
    }
}
```

```
public object Foo()  
{  
    return this;  
}
```

The title "Runes of Magic" is rendered in a highly stylized, gothic font. The letters are filled with a metallic, silver-blue texture and outlined in a bright gold. The text is set against a dark background. Behind the word "Runes" is a large, glowing blue sphere with a grid of white lines, resembling a globe or a magical orb. This sphere is surrounded by a circular border containing various runic symbols. To the right of the main title, there is a smaller, glowing blue orb with a similar grid pattern, from which several bright blue lightning bolts emanate, striking towards the left. The overall aesthetic is magical and fantastical.

Runes of Magic

www.RunesofMagic.com

© "Radiant Aroana" is the copyright and trademark of Runewaker Entertainment Corp.. All rights reserved. © "Runes of Magic" published by Frogster Interactive Pictures AG. All rights reserved.

If you find you need an extension to your ide to understand what's going on. Its probably not simple.

What's the root of the problem?

```
public void Deactivate(Guid id, string reason)
{
    var item = repository.GetById(id);
    item.Deactivate();
}
```

```
public void Reactivate(Guid id, DateTime effective,
                      string reason)
{
    var item = repository.GetById(id);
    item.Deactivate();
}
```

```
public void CheckIn(Guid id, int count)
{
    var item = repository.GetById(id);
    item.Deactivate();
}
```

No common interface!

```
public void Log(PointCut calledOn)
{
    logger.Log(calledOn.Name + ":" + calledOn.Parameters);
}
```



```
public void Handle(DeactivateCommand c)
{
    var item = repository.GetById(c.id);
    item.Deactivate();
}
```

```
public void Handle(ReactivateCommand c)
{
    var item = repository.GetById(c.id);
    item.Reactivate();
}
```

```
public void Handle(CheckInCommand c)
{
    var item = repository.GetById(c.id);
    item.CheckIn(c.quantity);
}
```

```
interface Handles<T> where T:Command
{
    void Handle(T command);
}
```

```
class LoggingHandler<T> : Handles<T> where T:Command
{
    private readonly Handles<T> next;

    public LoggingHandler(Handles<T> next)
    {
        this.next = next;
    }

    public void Handle(T command)
    {
        myLoggingFramework.Log(command);
        next.Handle(command);
    }
}
```

```
var handler = new LoggingHandler<DeactivateCommand>(
    new DeactivateCommandHandler(...)
);
```

The title "Runes of Magic" is rendered in a highly stylized, gothic font. The letters are filled with a metallic, silver-blue texture and outlined in a bright gold. The word "Runes" is on the top line, "of" is smaller and positioned below the first few letters of "Runes", and "Magic" is on the bottom line. The background features a glowing blue orb with a cracked surface, emitting bright white lightning bolts. A circular frame containing ancient-looking runic symbols surrounds the orb. The entire scene is set against a dark, almost black background.

Runes of Magic

www.RunesofMagic.com

© "Radiant Aroana" is the copyright and trademark of Runewaker Entertainment Corp.. All rights reserved. © "Runes of Magic" published by Frogster Interactive Pictures AG. All rights reserved.

```
public class DeactivateInventoryItem : Handles<DeactivateCommand>
{
    private readonly IRepository repository;

    public DeactivateInventoryItem(IRepository repository)
    {
        this.repository = repository;
    }

    public void Handle(DeactivateCommand command)
    {
        var item = repository.GetById(command.id);
        item.Deactivate(cmd.Reason);
    }
}
```

```
public class DeactivateInventoryItem :Handles<DeactivateCommand>
{
    private readonly IRepository repository;

    public DeactivateInventoryItem(IRepository repository)
    {
        this.repository = repository;
    }

    public void Handle(DeactivateCommand command)
    {
        var item = repository.GetById(command.id);
        item.Deactivate(cmd.Reason);
    }
}
```

```
class Handlers
{
    public static void Handle(ItemRepository repository,
                              DeactivateCommand c)
    {
        var item = repository.GetById(c.id);
        item.Deactivate();
    }

    public static void Handle(ItemRepository repository,
                              ReactivateCommand c)
    {
        var item = repository.GetById(c.id);
        item.Deactivate();
    }

    public static void Handle(ItemRepository repository,
                              CheckInCommand c)
    {
        var item = repository.GetById(c.id);
        item.Deactivate();
    }
}
```

Back to the same problem!


```
public static int Add(int a, int b)
{
    return a + b;
}
```

```
public static int Add(int a, int b)
{
    return a + b;
}
```

```
var add5 = x => Add(5, x);
```

```
public static void Deactivate(ItemRepository repository,  
                             DeactivateCommand c)  
{  
    var item = repository.GetById(c.id);  
    item.Deactivate();  
}
```

```
public static void Deactivate(ItemRepository repository,  
                             DeactivateCommand c)  
{  
    var item = repository.GetById(c.id);  
    item.Deactivate();  
}
```

```
var nodepends = x => Deactivate(new ItemRepository(), x);
```

```
void Bootstrap()  
{  
    handlers.Add(x => Deactivate(new ItemRepository(), x));  
    handlers.Add(x => Reactivate(new ItemRepository(), x));  
    handlers.Add(x => CheckIn(new ItemRepository(),  
                               new BarService(),  
                               x));  
}
```

```
void Bootstrap()
{
    handlers.Add(x => Deactivate(() => new ItemRepository(), x));
    handlers.Add(x => Reactivate(() => new ItemRepository(), x));
    handlers.Add(x => CheckIn(() => new ItemRepository(),
                                new BarService(),
                                x));
}
```

```
public static void Log<T>(T command, Action<T> next)
    where T:Command
    {
        myLoggingFramework.Log(command);
        next(command);
    }
```

```
class LoggingHandler<T> : Handles<T> where T:Command
{
    private readonly Handles<T> next;

    public LoggingHandler(Handles<T> next)
    {
        this.next = next;
    }

    public void Handle(T command)
    {
        myLoggingFramework.Log(command);
        next.Handle(command);
    }
}
```

```
var handler = new LoggingHandler<DeactivateCommand>(
    new DeactivateCommandHandler(...)
);
```



```
<bean id="moreComplexObject" class="example.ComplexObject">
  <!-- results in a setAdminEmails(java.util.Properties) call -->
  <property name="adminEmails">
    <props>
      <prop key="administrator">administrator@example.org</prop>
      <prop key="support">support@example.org</prop>
      <prop key="development">development@example.org</prop>
    </props>
  </property>
  <!-- results in a setSomeList(java.util.List) call -->
  <property name="someList">
    <list>
      <value>a list element followed by a reference</value>
      <ref bean="myDataSource" />
    </list>
  </property>
  <!-- results in a setSomeMap(java.util.Map) call -->
  <property name="someMap">
    <map>
      <entry key="an entry" value="just some string"/>
      <entry key="a ref" value-ref="myDataSource"/>
    </map>
  </property>
</bean>
```

```
public static void Log<T>(T command, Action<T> next)
    where T:Command
    {
        myLoggingFramework.Log(command);
        next(command);
    }
```

Understand the problem a tool
or idea solves well

If you need to add stuff to your IDE you are probably on the wrong path.

You own all code in your project.

Your boss doesn't care if the bug happened in someone else's library!