canoo

Java Boilerplate Busters

Hamlet D'Arcy – Canoo Engineering AG
@HamletDRC
http://hamletdarcy.blogspot.com

Agenda

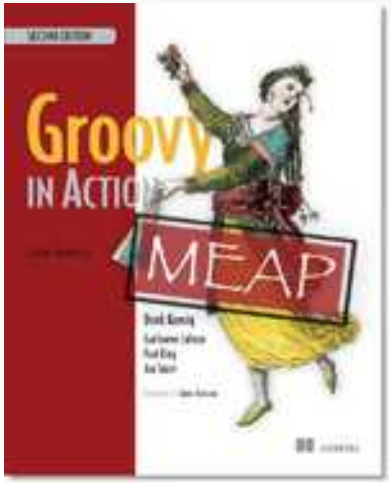
- Google Guava / Commons Lang
- Builders / Fluent APIs
- Function Objects / Proxies / Interceptors
- Java 7
- Lombok / Groovy

My Confess-ion

I used the wrong template

About Me

JETBRAINS
DEVELOPMENT ACADEMY
MEMBER



Groovy

CODENARC
Less Bugs Better Code

canoo
› your provider for business web solutions ›

Boilerplate – noun

Code you shouldn't be writing yourself

Agenda

- **Google Guava / Commons Lang**
- Builders / Fluent APIs
- Function Objects / Proxies / Interceptors
- Java 7
- Lombok / Groovy

```
import static org.apache.commons.lang.StringUtils.*;
```

```groovy
import static org.apache.commons.lang.StringUtils.*;

assert isAlpha("abcde");
assert isNumeric("12345");
assert isAlphanumeric("12345abc");
```

```groovy
import static org.apache.commons.lang.StringUtils.*;

assert isBlank("");
assert isBlank(null);
```

```
import static org.apache.commons.lang.StringUtils.*;

assert isAllLowerCase("abcdef");
assert isAllUpperCase("ABCDEF");
```

```groovy
import static org.apache.commons.lang.StringUtils.*;

assert "abc" == defaultString("abc");
assert "" == defaultString("");
assert "" == defaultString(null);
```

```
import static org.apache.commons.lang.StringUtils.*;

assert "abc".equals(left("abcdef", 3));
assert "def".equals(right("abcdef", 3));
assert null == right(null, 3);
```

```
import static org.apache.commons.lang.StringUtils.*;

Object[] array = new Object[]{1, 2, 3};

assert "1, 2, 3".equals(join(array, ", "));

List list = Arrays.asList(1, 2, 3);

assert "1, 2, 3".equals(join(list, ", "));
```

```
import org.apache.commons.lang.StringEscapeUtils;
```

```
import org.apache.commons.lang.StringEscapeUtils;

String xmlText = "\"Hello\" & \"GoodBye\"";

String escapedXml = "&quot;Hello&quot; &amp; "
        + "&quot;GoodBye&quot;";

assert escapedXml.equals(
        StringEscapeUtils.escapeXml(xmlText)
);

assert xmlText.equals(
        StringEscapeUtils.unescapeXml(escapedXml)
);
```

```
import org.apache.commons.lang.StringEscapeUtils;

String htmlText = "<body>text</body>";

String escapedHtml = "&lt;body&gt;" +
        "text&lt;/body&gt;";

assert escapedHtml.equals(
        StringEscapeUtils.escapeHtml(htmlText)
);
assert htmlText.equals(
        StringEscapeUtils.unescapeHtml(escapedHtml)
);
```

```
import org.apache.commons.lang.StringEscapeUtils;

String csvText = "red, beans, and rice";

String escapedCsv = "\"red, beans, and rice\"";

assert escapedCsv.equals(
        StringEscapeUtils.escapeCsv(csvText)
);

assert csvText.equals(
        StringEscapeUtils.unescapeCsv(escapedCsv)
);
```

```
import static org.apache.commons.lang.ArrayUtils.*;
```

```
import static org.apache.commons.lang.ArrayUtils.*;

int[] evens = new int[]{2, 4, 6};
int[] odds = new int[]{1, 3, 5};


assert contains(evens, 2);
assert contains(odds, 3);
```

```
import static org.apache.commons.lang.ArrayUtils.*;

int[] evens = new int[]{2, 4, 6};
int[] odds = new int[]{1, 3, 5};

assert Arrays.equals(
        new int[]{2, 4, 6, 8},
        add(evens, 8)
);

assert Arrays.equals(
        new int[]{2, 4, 6, 1, 3, 5},
        addAll(evens, odds)
);
```

```
import static org.apache.commons.lang.ArrayUtils.*;

int[] evens = new int[]{2, 4, 6};
int[] odds = new int[]{1, 3, 5};


assert Arrays.equals(
        new int[]{2, 6},
        remove(evens, 1)
);

assert Arrays.equals(
        new int[]{1, 3},
        remove(odds, 2)
);
```

```
import static org.apache.commons.lang.BooleanUtils.*;
```

```groovy
import static org.apache.commons.lang.BooleanUtils.*;

assert toBoolean("true");
assert toBoolean("TRUE");
assert toBoolean("tRUe");
assert toBoolean("on");
assert toBoolean("ON");
assert toBoolean("yes");
assert toBoolean("YES");
```

```
import org.apache.commons.lang.builder.ToStringBuilder;
```

```java
import org.apache.commons.lang.builder.ToStringBuilder;

public class Person {
    String name;
    Date timestamp = new Date();

    @Override
    public String toString() {
        return new ToStringBuilder(this)
                .append("name", name)
                .append("timestamp", timestamp)
                .toString();
    }
}
```

```
import org.apache.commons.lang.builder.EqualsBuilder;
```

```java
import org.apache.commons.lang.builder.EqualsBuilder;

public class Person {
    String name;
    Date timestamp = new Date();

    @Override
    public boolean equals(Object obj) {
        if (obj == null) { return false; }
        if (obj == this) { return true; }
        if (obj.getClass() != getClass()) { return false; }

        Person rhs = (Person) obj;
        return new EqualsBuilder()
                .append(name, rhs.name)
                .append(timestamp, rhs.timestamp)
                .isEquals();
    }
}
```

```
import org.apache.commons.lang.builder.HashCodeBuilder;
```

```java
import org.apache.commons.lang.builder.HashCodeBuilder;

public class Person {
  String name;
  Date timestamp = new Date();

  @Override
  public int hashCode() {
    return new HashCodeBuilder(99, 33)
        .append(name)
        .append(timestamp)
        .toHashCode();
  }
}
```

```
import org.apache.commons.lang.builder.CompareToBuilder;
```

```java
import org.apache.commons.lang.builder.CompareToBuilder;

public class Person implements Comparable<Person> {
    String name;
    Date timestamp = new Date();

    public int compareTo(Person other) {
        return new CompareToBuilder()
                .append(this.name, other.name)
                .toComparison();
    }
}
```

Apache Commons Lang

+ Tons of other Helper Objects

- Version 2.6 Stable
  - JDK 1.3 Compatible
  - No Generics
  - Many functions obsoleted by JDK 5
- Version 3.0 Beta
  - JDK 1.5 Compatible
  - Not backwards compatible with 2.6

```
new HashMap<T1, Set<T2>>
```

```
import com.google.common.collect.Multimap;
```

```
import com.google.common.collect.Multimap;

Multimap<String, String> multiMap =
                              HashMultimap.create();
multiMap.put("Canoo", "Hamlet");
multiMap.put("Canoo", "Dierk");
multiMap.put("Canoo", "Andres");
multiMap.put("EngineYard", "Charles");
multiMap.put("EngineYard", "Thomas");
multiMap.put("EngineYard", "Nick");
```

```groovy
import com.google.common.collect.Multimap;

assert multiMap.containsKey("Canoo");
assert multiMap.containsKey("EngineYard");
assert multiMap.containsValue("Hamlet");
assert multiMap.containsValue("Charles");
```

```
import com.google.common.collect.Multimap;

Collection<String> canooies = multiMap.get("Canoo");

Collection<String> c = Lists.newArrayList(
           "Dierk","Andres", "Hamlet"
);

assert canooies.containsAll(c);
```

```
import com.google.common.collect.BiMap;
```

```
import com.google.common.collect.BiMap;

BiMap<String, String> biMap = HashBiMap.create();
biMap.put("Switzerland", "die Schweiz");
biMap.put("Poland", "Polska");
biMap.put("Austria", "Österreich");

assert "Polska" == biMap.get("Poland");
assert "Switzerland" == biMap.inverse()
                             .get("die Schweiz");
```

```
new HashMap<K1, Map<K2, V>>
```

```
import com.google.common.collect.Table;
```

```
import com.google.common.collect.Table;

Table<Integer, Integer, String> table =
                          HashBasedTable.create();

table.put(1, 1, "Hamlet");
table.put(1, 2, "Dierk");
table.put(2, 1, "Andres");
table.put(2, 2, "Matthius");
```

```
import com.google.common.collect.Table;

assert table.contains(1, 1);
assert table.containsRow(1);
assert table.containsColumn(2);

assert !table.contains(3, 1);
assert !table.containsRow(3);
assert !table.containsColumn(3);

assert "Hamlet" == table.get(1, 1);
assert table.containsValue("Hamlet");
```

```
import com.google.common.base.Joiner;
```

```java
import com.google.common.base.Joiner;

Set<Integer> set = Sets.newHashSet(1, 2, 3);

assert "1, 2, 3".equals(Joiner.on(", ").join(set));
```

```
import com.google.common.base.Splitter;
```

```java
import com.google.common.base.Splitter;

String string = "1, 2, , 3";

Iterable<String> i = Splitter.on(",")
        .omitEmptyStrings()
        .trimResults()
        .split(string);
```

```
import static com.google.common.collect.Iterables.*;
import static com.google.common.collect.Lists.*;

Iterable<Integer> setAsInts = transform(i,
        new Function<String, Integer>() {
            @Override
            public Integer apply(String input) {
                return Integer.valueOf(input);
            }
        });

assert newArrayList(1, 2, 3)
            .containsAll(newArrayList(setAsInts));
```

```
import static com.google.common.base.Strings.*;
```

```
import static com.google.common.base.Strings.*;

assert emptyToNull("") == null;
assert "".equals(nullToEmpty(null));
assert isNullOrEmpty("");
assert "abcabcabc".equals(repeat("abc", 3));
```

```
import static com.google.common.base.Preconditions.*;
```

```
import static com.google.common.base.Preconditions.*;

int userID = 1;
String userName = "some name";

checkArgument(userID > 0,
              "userid is negative: %s", userID);

checkNotNull(userName,
              "user %s missing name", userID);


checkState(userName.length() > 0,
              "user %s missing name", userID);
```

```
import static com.google.common.base.Objects.*;
```

```java
import static com.google.common.base.Objects.*;

class Person {
    String name;
    Date timestamp = new Date();

    @Override
    public String toString() {
        return toStringHelper(this)
                .add("name", name)
                .add("timestamp", timestamp).toString();
    }
    @Override
    public int hashCode() {
        return hashCode(name, timestamp);
    }
}
```

```
import static com.google.common.base.Equivalences.*;
```

```java
import static com.google.common.base.Equivalences.*;

class Person implements Comparable<Person> {
    String name;
    Date timestamp = new Date();

    @Override
    public boolean equals(Object obj) {
        if (obj == null) { return false; }
        if (obj == this) { return true; }
        if (obj.getClass() != getClass()) {
            return false;
        }
        Person rhs = (Person) obj;

        return equals().equivalent(name, rhs.name) &&
                equals().equivalent(timestamp, rhs.timestamp);
    }
}
```

```
import static com.google.common.collect.ComparisonChain.*;
```

```
import static com.google.common.collect.ComparisonChain.*;

class Person implements Comparable<Person> {
    String name;
    Date timestamp = new Date();

    public int compareTo(Person other) {
        return start()
                .compare(name, other.name)
                .compare(timestamp, other.timestamp)
                .result();
    }
}
```

```
import com.google.common.base.Throwables;
```

```java
import com.google.common.base.Throwables;

Exception ex = new Exception();

Throwables.propagateIfPossible(ex);

Throwables.propagate(ex);

// throws Exception
Throwables.throwCause(ex, false);
```

```
import com.google.common.base.Function;
```

```
import com.google.common.base.Function;

Person alice = new Person("Alice");
Person bob = new Person("Bob");
Person carol = new Person("Carol");
List<Person> people = newArrayList(alice, bob, carol);
```

```
import com.google.common.base.Function;

Function<Person, String> getName =
        new Function<Person, String>() {
            public String apply(Person p) {
                return p.getName();
            }
        };

assert "Alice" == getName.apply(alice);
Collection<String> names = Collections2
            .transform(people, getName);

assert names.containsAll(
            newArrayList("Alice", "Bob", "Carol")
);
```

```
import com.google.common.base.Predicate;
```

```
import com.google.common.base.Predicate;

Predicate<Person> predicate = new Predicate<Person>() {
    public boolean apply(Person input) {
        return input.getName().contains("o");
    }
};

Iterable<Person> filteredList = Iterables
                .filter(people, predicate);

assert Iterables.contains(filteredList, bob);
assert Iterables.contains(filteredList, carol);
assert !Iterables.contains(filteredList, alice);
```

Google Guava

+ Tons of other Helper Objects

- Modern (Fluent APIs & Generics)
- Well-Designed
- Consistent
- Active

Agenda

- Google Guava / Commons Lang
- **Builders / Fluent APIs**
- Function Objects / Proxies / Interceptors
- Java 7
- Lombok / Groovy

“Traditional API”

```
Order order = new Order();
order.setItemName("lattes");
order.setQuantity(2);
order.pay(Currency.getInstance("USD"));
```

“Modern API”

```
new FluentOrder()
        .withItem("lattes")
        .withQuantity(2)
        .pay(Currency.getInstance("USD"));
```

“Modern API”

```
class FluentOrder {
    private String item;
    private int quantity;

    public FluentOrder withItem(String item) {
        this.item = item;
        return this;
    }

    public FluentOrder withQuantity(int quantity) {
        this.quantity = quantity;
        return this;
    }

    boolean pay(Currency currency) {
        // pay order
        return true;
    }
}
```

```
class OrderBuilder {
  QuantityHolder withItem(String item) {
    return new QuantityHolder(item);
  }
}
class QuantityHolder {
    private final String item;
    public QuantityHolder(String item) {
        this.item = item;
    }
    OrderFiller withQuantity(int quantity) {
        return new OrderFiller(item, quantity);
    }
}
class OrderFiller {

    private final String item;
    private final int quantity;

    OrderFiller(String item, int quantity) {
        this.item = item; this.quantity = quantity;
    }


    boolean pay(Currency currency) {
        /* pay order... */ return true;
    }
}
```

Builders & Fluent API

+ More expressive
+ Type-safer
+ Easier to use

- Harder to write
- Harder to learn
- Some reformatting tools choke

Agenda

- Google Guava / Commons Lang
- Builders / Fluent APIs
- **Function Objects / Proxies / Interceptors**
- Java 7
- Lombok / Groovy

“Traditional API”

```
public class ResourceProvider {


  private final ReadWriteLock lock = new ReentrantReadWriteLock();
  private final Map<String, String> data = new HashMap<String, String>();


  public String getResource(String key) throws Exception {
    lock.readLock().lock();
    try {
      return data.get(key);
    } finally {
      lock.readLock().unlock();
    }
  }


  public void refresh() throws Exception {
    lock.writeLock().lock();
    try {
        // reload settings...
    } finally {
      lock.writeLock().unlock();
    }
  }
}
```

“Modern API” -- Function Objects

```
public class ResourceProvider {

    private final ResourceController controller = new ResourceController();
    private final Map<String, String> data = new HashMap<String, String>();

    public String getResource(final String key) throws Exception {
        return controller.withReadLock(new Callable<String>() {
            public String call() throws Exception {
                return data.get(key);
            }
        });
    }

    public void refresh() throws Exception {
        controller.withWriteLock(new Callable<Void>() {
            public Void call() throws Exception {
                // reload settings
                return null;
            }
        });
    }
}
```

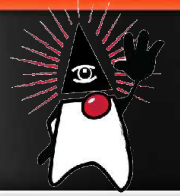
“Modern API” con't

```
public class ResourceController {
    private final ReadWriteLock lock = new ReentrantReadWriteLock();

    public <T> T withReadLock(Callable<T> block) throws Exception {
        lock.readLock().lock();
        try {
            return block.call();
        } finally {
            lock.readLock().unlock();
        }
    }

    public <T> T withWriteLock(Callable<T> block) throws Exception {
        lock.writeLock().lock();
        try {
            return block.call();
        } finally {
            lock.writeLock().unlock();
        }
    }
}
```

“Modern API” -- Groovy Function Objects



```
class ResourceProvider {

    private final def controller = new ResourceController()
    private final def data = [:]

    String getResource(String key) {
        controller.withReadLock({ data.get(key) } as Callable)
    }

    void refresh() {
        controller.withWriteLock( { /* reload settings */} as Callable)
    }
}
```

“Modern API” -- Proxy Class

```
public class DefaultResourceProvider implements ResourceProvider {

    private final Map<String, String> data = new HashMap<String, String>();

    @WithReadLock
    public String getResource(String key) throws Exception {
        return data.get(key);
    }

    @WithWriteLock
    public void refresh() throws Exception {
        System.out.println("Reloading the settings...");
    }
}
```

```
public class ResourceController implements InvocationHandler {

    private final Object target;
    private final ReadWriteLock lock = new ReentrantReadWriteLock();

    ResourceController(Object target) { this.target = target; }

    public Object invoke (Object proxy, Method method, Object[] args) {
        try {
            Method targetMethod = target.getClass().getMethod(
                method.getName(),
                method.getParameterTypes());

            if (targetMethod.isAnnotationPresent(WithReadLock.class)) {
                lock.readLock().lock();
                try {
                    return targetMethod.invoke(target, args);
                } finally {
                    lock.readLock().unlock();
                }
            } else if (targetMethod.isAnnotationPresent(WithWriteLock.class)){
                lock.writeLock().lock();
                try {
                    return targetMethod.invoke(target, args);
                } finally {
                    lock.writeLock().unlock();
                }
            } else {
                return targetMethod.invoke(target, args);
            }
        } catch (Exception ex) {
            throw new RuntimeException(ex);
        }
    }
}
```

“Modern API” -- Proxy Class con't

```
InvocationHandler handler = new ResourceController(
        new DefaultResourceProvider()
);

ResourceProvider provider = (ResourceProvider) Proxy.newProxyInstance(
        DefaultResourceProvider.class.getClassLoader(),
        new Class[]{ResourceProvider.class},
        handler);

provider.refresh();
provider.getResource("property");
```

“Modern API” -- Proxy Class con't

```groovy
class ResourceProvider {

    private final def data = [:]

    @WithReadLock
    String getResource(String key) {
        return data.get(key);
    }

    @WithWriteLock
    void refresh() {
        // Reload settings...
    }
}
```

```java
public class WithLockInterceptor  {

    // This is shared!!!
    private final ReadWriteLock lock = new ReentrantReadWriteLock();

    @AroundInvoke
    public Object enforceLock(InvocationContext context) throws Exception{
        try {
            Method targetMethod = context.getMethod();

            if (targetMethod.isAnnotationPresent(WithReadLock.class)) {
                lock.readLock().lock();
                try {
                    return context.proceed();
                } finally {
                    lock.readLock().unlock();
                }
            } else if (targetMethod.isAnnotationPresent(WithWriteLock.class)){
                lock.writeLock().lock();
                try {
                    //reload the resources into memory
                    return context.proceed();
                } finally {
                    lock.writeLock().unlock();
                }
            }
            return context.proceed();
        } catch (Exception ex) {
            throw new RuntimeException(ex);
        }
    }
}
```

Function Objects

+ Safe
+ Boilerplate is compiler enforced
+ Easily replaces finally & catch blocks
+ Easier in Java 8 and Groovy

- Verbose
- Different

Proxies & Interceptors

+ Declarative
+ Expressive
+ Easily replaces finally & catch blocks
+ Many implementation options

- Proxy requires interface
- Proxy hard to instantiate
- Unproxied class may not be “valid”

Agenda

- Google Guava / Commons Lang
- Builders / Fluent APIs
- Function Objects / Proxies / Interceptors
- **Java 7**
- Lombok / Groovy

“Traditional Error Handling”

```
try {
    doSomething();
} catch (UnsupportedOperationException e) {
  handleError(e);
} catch (IllegalStateException e) {
    handleError(e);
} catch (IllegalArgumentException e) {
    handleError(e);
}
```

“Modern Error Handling”

```
try {
    doSomething();
} catch (UnsupportedOperationException
         | IllegalStateException
         | IllegalArgumentException e) {
    handleError(e);
}
```

“Traditional Resource Management”

```
FileReader reader = new FileReader(path);
try {
    return reader.read();
} finally {
    reader.close();
}
```

“Modern Resource Management”

```
try (FileReader reader = new FileReader(path)) {
    return reader.read();
}
```

“Modern Resource Management”

```
class MyResource implements AutoCloseable {

    String getResource() {
        return "some resource";
    }

    @Override
    public void close() throws Exception {
        // closing
    }
}


try (MyResource resource = new MyResource()) {
    return resource.getResource();
}
```

Java 7 Features

+ Soon to be widely used
+ Terser code

- No Lambdas
- Not widely used

Agenda

- Google Guava / Commons Lang
- Builders / Fluent APIs
- Function Objects / Proxies / Interceptors
- Java 7
- **Lombok / Groovy**

“Traditional Java Bean”

```
public class Person {
    private String firstName;
    private String lastName;

    public String getFirstName() { return firstName; }
    public void setFirstName(String firstName) { this.firstName = firstName; }

    public String getLastName() { return lastName; }
    public void setLastName(String lastName) { this.lastName = lastName; }

    @Override
    public boolean equals(Object o) {
        ...
    }

    @Override
    public int hashCode() {
        ...
    }
}
```

“Modern Java Bean” -- Lombok

```
@EqualsAndHashCode
public class Person {
    @Setter @Getter private String firstName;
    @Setter @Getter private String lastName;
}
```

“Modern Java Bean” -- Lombok

```
@Data
public class Person {
    private String firstName;
    private String lastName;
}
```

Lombok

@Getter / @Setter

@ToString

@EqualsAndHashCode

@NoArgsConstructor

@RequiredArgsConstructor

@AllArgsConstructor

@Data

@Cleanup

@Synchronized

@SneakyThrows

@Log

val

Lombok

+ Generates boilerplate into Class file
+ Java-only
+ delombok option

- Only works on Eclipse and javac
- Hinders some Eclipse features
- Not all features are loved by all

“Modern Java Bean” -- Groovy

```groovy
@EqualsAndHashCode
public class Person {
    String firstName
    String lastName
}
```

Groovy

@ToString

@EqualsAndHashCode

@Canonical

@Lazy

@InheritConstructors

@TupleConstructor

@AutoClone

@AutoExternalize

@Delegate

@Singleton

@Immutable

@Bindable

@IndexedProperties

@ListenerList

@Vetoable

Groovy

@Synchronized

@WithReadLock

@WithWriteLock

@Log

@Log4j

@Slf4j

@Commons

... and many more as libraries

Groovy

+ Generates boilerplate into Class file
+ Great Java interop & IDE support
+ Gateway to many other features

- Requires a new language

? Performs worse than Java
? Hinders some IDE features

Agenda

- Google Guava / Commons Lang
- Builders / Fluent APIs
- Function Objects / Proxies / Interceptors
- Java 7
- Lombok / Groovy

Thanks

Twitter: @HamletDRC

Blog: http://hamletdarcy.blogspot.com

Work Blog: http://www.canoo.com/blog

JetBrains.tv Screencasts:
http://tv.jetbrains.net/tags/hamlet

YouTube Screencasts:
http://www.youtube.com/user/HamletDRC

Share-a-Canooie – http://people.canoo.com/share/

Hackergarten – http://www.hackergarten.net/