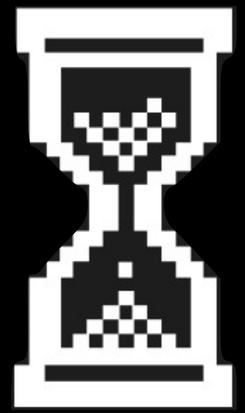
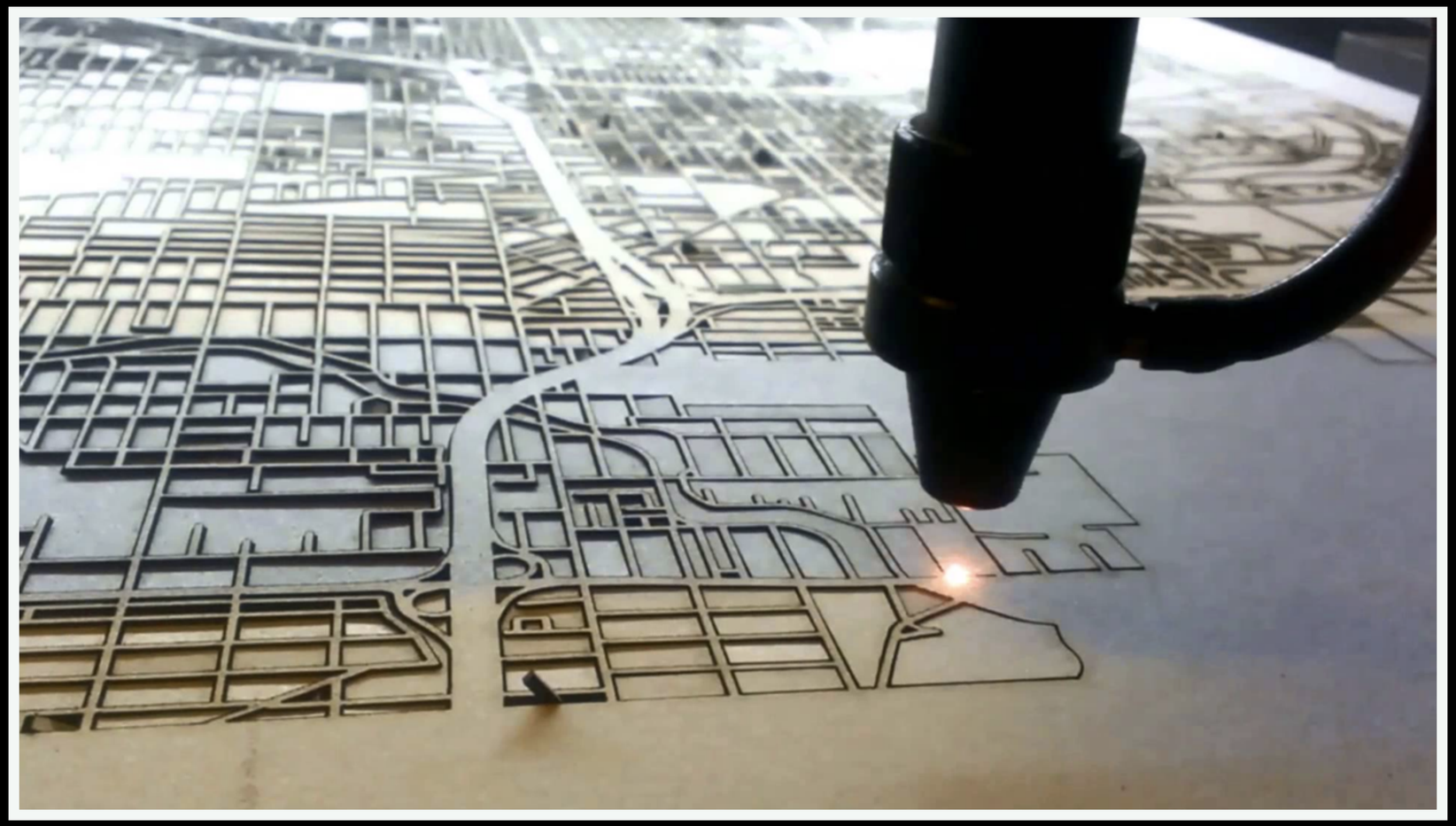


LOADING



Marvel of Annotation Preprocessing

by Alexey Buzdin

@AlexeyBuzdin

GDGRiga.lv

JUG.lv

Citadele.lv

RigaDevDay.lv

- What?
- Why?
- How?

- What?
- Why?
- How?
- Dagger
- MapStruct
- Lombok

“Developer loves to write code”

"Developer loves to write code"

- noone ever

“Developer loves complex tasks”

“Developer loves complex tasks”

- probably you

CREATE READ UPDATE DELETE

C R U D

CRUD

CRUD

- **api**
  - **type:** XML, JSON
  - **resources**
    - */clients*
      - access: R
      - entity
        - 🔑 clientId
        - full name
        - email
    - */payments*
      - entity
        - 🔑 paymentId
        - 🔑 clientId **l link**
        - amount
  - **db:** jdbc:mysql://localhost:3306/crudapp

- **api**
  - **type:** XML, JSON
  - **resources**
    - */clients*
      - access: R
      - entity
        - 🔑 clientId
        - full name
        - email
    - */payments*
      - entity
        - 🔑 paymentId
        - 🔑 clientId **| link**
        - amount
  - **db: jdbc:**mysql://localhost:3306/crudapp

CRUD

Authentication?

CRUD

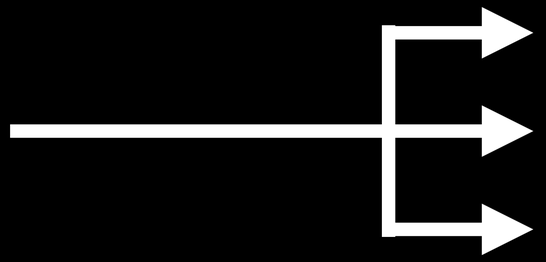
- **api**
  - **type:** XML, JSON
  - **resources**
    - */clients*
      - access: R
      - entity
        - 🔑 clientId
        - full name
        - email
    - */payments*
      - entity
        - 🔑 paymentId
        - 🔑 clientId **l link**
        - amount
  - **db: jdbc:**mysql://localhost:3306/crudapp
  - **ldap:** ldap://ldap.example.com/dc=example,dc=com


never let go of your
DREAMS

In Practice

In Practice

Routing
XML, JSON
Building Response



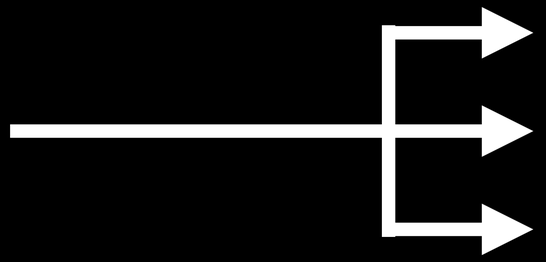
Generic Servlet

```
public class SimpleServlet extends GenericServlet {

    public void service(ServletRequest req, ServletResponse res)
            throws ServletException, IOException {
        // do something in here
    }
}
```

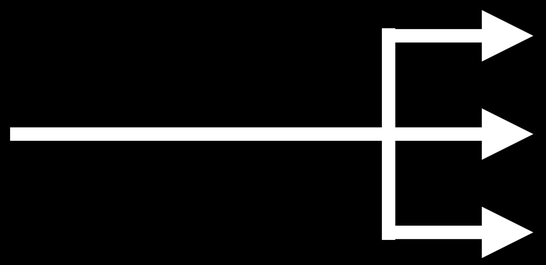
In Practice

Routing
XML, JSON
Building Response

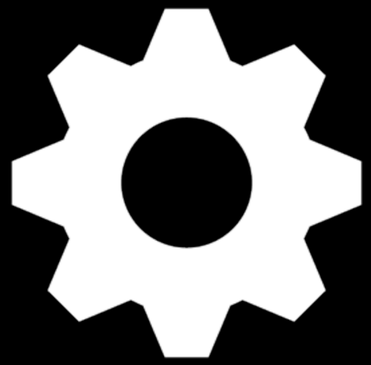


In Practice

Routing
XML, JSON
Building Response



Map
Validate

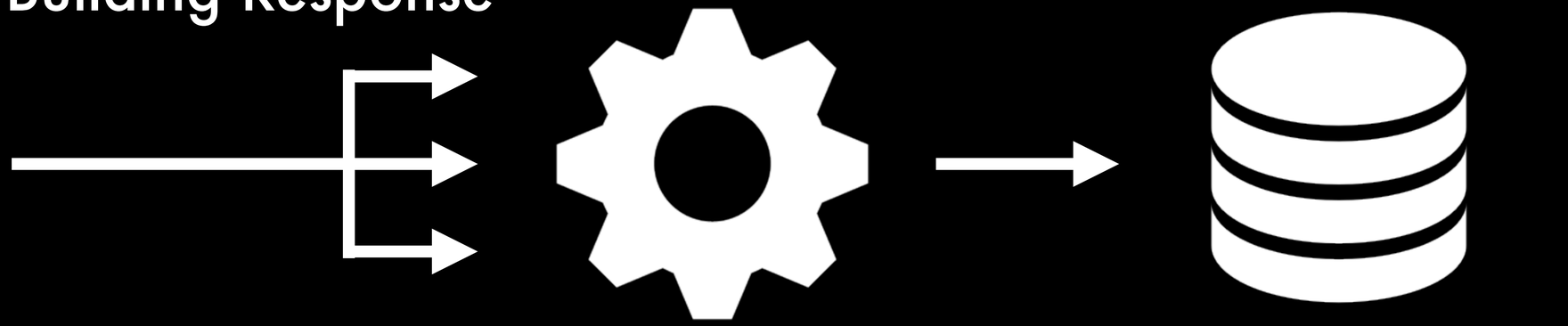


In Practice

Routing
XML, JSON
Building Response

Map
Validate

Construct Query



Reflection and Runtime Code Generation

Everywhere!

Spring, Hibernate, GSON, Jersy, Dozer, Guice, Weld, BeanValidation etc…

Reflection - Slow?

Reflection - Slow?

Performance Overhead
Because reflection involves types that are dynamically resolved, certain Java virtual machine optimizations can not be performed. Consequently, reflective operations have slower performance than their non-reflective counterparts, and should be avoided in sections of code which are called frequently in performance-sensitive applications.

http://docs.oracle.com/javase/tutorial/reflect/index.html

Reflection - Slow?

Performance Overhead

Because reflection involves types that are dynamically resolved, certain Java virtual machine optimizations can not be performed. Consequently, reflective operations have slower performance than their non-reflective counterparts, and should be avoided in sections of code which are called frequently in performance-sensitive applications.

http://docs.oracle.com/javase/tutorial/reflect/index.html

Reflection - Slow?

Performance Overhead

Because reflection involves types that are dynamically resolved, certain Java virtual machine optimizations can not be performed. Consequently, reflective operations have slower performance than their non-reflective counterparts, and should be avoided in sections of code which are called frequently in performance-sensitive applications.

http://docs.oracle.com/javase/tutorial/reflect/index.html

JOIN THE DARK SIDE:
WE HAVE OUR OWN VM!

“In NYTimes GSON costed - 700ms startup delay.”

Recommendation: understand what you're getting into when using reflection (or libraries that use reflection). In particular, do not use reflective type adapters to serialize or deserialize Java objects.

- Jake Wharton

http://blog.nimbledroid.com/2016/02/23/slow-Android-reflection.html

Code Generation?

https://zeroturnaround.com/rebellabs/how-to-make-java-more-dynamic-with-runtime-code-generation/

Code Generation?

Generates Java Bytecode at Runtime

https://zeroturnaround.com/rebellabs/how-to-make-java-more-dynamic-with-runtime-code-generation/

Code Generation?

Generates Java Bytecode at Runtime

Not always ok

https://zeroturnaround.com/rebellabs/how-to-make-java-more-dynamic-with-runtime-code-generation/

Code Generation?

GWT, Android, j2objc, RoboVM, code transpilers

Doesn't work

https://zeroturnaround.com/rebellabs/how-to-make-java-more-dynamic-with-runtime-code-generation/

Rafael Winterhalter
@rafaelcodes

How to make Java more dynamic with Runtime Code Generation

REBELLABS

https://zeroturnaround.com/rebellabs/how-to-make-java-more-dynamic-with-runtime-code-generation/

Annotation Processing

Annotation Processing

A tool build in javac for scanning and processing annotations at compile time

Annotation Processing

A tool build in javac for scanning and processing annotations at compile time

$ javac -cp target/apt-demo-1.0-SNAPSHOT.jar Test.java

How?

MyProcessor.jar
- com
  - example
    - **MyProcessor.class**
- META-INF
  - services
    - **javax.annotation.processing.Processor**

javax.annotation.processing.Processor

com.example.MyProcessor
com.foo.OtherProcessor
net.blabla.SpecialProcessor

javac will run the process in separate JVM

How?

```
public class MyProcessor extends AbstractProcessor {




}
```

http://hannesdorfmann.com/annotation-processing/annotationprocessing101/

How?

```
public class MyProcessor extends AbstractProcessor {

   public synchronized void init(ProcessingEnvironment env){ }



}
```

http://hannesdorfmann.com/annotation-processing/annotationprocessing101/

How?

```
public class MyProcessor extends AbstractProcessor {

   public synchronized void init(ProcessingEnvironment env){ }
   public boolean process(
      Set<? extends TypeElement> annotations, RoundEnvironment env) { }



}
```

http://hannesdorfmann.com/annotation-processing/annotationprocessing101/

How?

```
public class MyProcessor extends AbstractProcessor {

   public synchronized void init(ProcessingEnvironment env){ }
   public boolean process(
      Set<? extends TypeElement> annotations, RoundEnvironment env) { }
   public Set<String> getSupportedAnnotationTypes() { }


}
```

http://hannesdorfmann.com/annotation-processing/annotationprocessing101/

How?

```
public class MyProcessor extends AbstractProcessor {

   public synchronized void init(ProcessingEnvironment env){ }
   public boolean process(
      Set<? extends TypeElement> annotations, RoundEnvironment env) { }
   public Set<String> getSupportedAnnotationTypes() { }
   public SourceVersion getSupportedSourceVersion() { }

}
```

http://hannesdorfmann.com/annotation-processing/annotationprocessing101/

How?

MyProcessor.jar
- com
  - example
    - **MyProcessor.class**
- META-INF
  - services
    - **javax.annotation.processing.Processor**

javax.annotation.processing.Processor

com.example.MyProcessor
com.foo.OtherProcessor
net.blabla.SpecialProcessor

javac will run the process in separate JVM

AutoService

```
package foo.bar;

import javax.annotation.processing.Processor;

@AutoService(Processor.class)
final class MyProcessor extends Processor {
  // …
}
```

https://github.com/google/auto/tree/master/service

Example

```
public class PizzaStore {
 public Meal order(String mealName) {

    if ("Margherita".equals(mealName)) return new MargheritaPizza();
    if ("Calzone".equals(mealName)) return new CalzonePizza();
    if ("Tiramisu".equals(mealName)) return new Tiramisu();

    throw new IllegalArgumentException("Unknown meal '" + mealName + "'");
  }
}
```

What do we Want?

Automatically Detect All Meals
from ClassPath and Register them

What do we Want?

```
public class PizzaStore {

  private MealFactory factory = new MealFactory();

  public Meal order(String mealName) {
    return factory.create(mealName);
  }

}
```

```
@Target(ElementType.TYPE) @Retention(RetentionPolicy.CLASS)
public @interface Factory {
  Class type();
  String id();
}

@Factory(
    id = "Margherita",
    type = Meal.class
)
public class MargheritaPizza implements Meal {

  @Override public float getPrice() {
    return 6f;
  }
}
```

How?

```
public class MyProcessor extends AbstractProcessor {

   public synchronized void init(ProcessingEnvironment env){ }
   public boolean process(
      Set<? extends TypeElement> annotations, RoundEnvironment env) { }
   public Set<String> getSupportedAnnotationTypes() { }
   public SourceVersion getSupportedSourceVersion() { }

}
```

```java
@Override
public Set<String> getSupportedAnnotationTypes() {
  Set<String> annotataions = new LinkedHashSet<String>();
  annotataions.add(Factory.class.getCanonicalName());
  return annotataions;
}

@Override
public SourceVersion getSupportedSourceVersion() {
  return SourceVersion.latestSupported();
}
```

```java
@Override
public synchronized void init(ProcessingEnvironment env) {
  super.init(env);
  typeUtils = env.getTypeUtils();
  elementUtils = env.getElementUtils();
  filer = env.getFiler();
  messager = env.getMessager();
}
```

```
@Override
public boolean process(Set<? extends TypeElement> annotations,
                       RoundEnvironment env) {




}
```

javax.lang.model.TypeElement

```
package com.example; // PackageElement

public class Foo {    // TypeElement

   private int a;      // VariableElement
   private Foo other;  // VariableElement

   public Foo () {}    // ExecuteableElement

   public void setA (  // ExecuteableElement
          int newA     // TypeElement
   ) {}
}
```

Gilad Bracha and David Ungar. Mirrors

```
@Override
public boolean process(Set<? extends TypeElement> annotations,
                       RoundEnvironment env) {

  // Itearate over all @Factory annotated elements
  for (Element elem : env.getElementsAnnotatedWith(Factory.class)){
    ...
  }
}
```

```
@Override
public boolean process(Set<? extends TypeElement> annotations,
                       RoundEnvironment env) {

  // Itearate over all @Factory annotated elements
  for (Element elem : env.getElementsAnnotatedWith(Factory.class)){
    if (annotatedElement.getKind() != ElementKind.CLASS) {
      messager.printMessage(Diagnostic.Kind.ERROR, “Ooops!”, elem);
      return true;
    }
  ...
  }
}
```

What do we Want?

```
public class PizzaStore {
 public Meal order(String mealName) {

    if ("Margherita".equals(mealName)) return new MargheritaPizza();
    if ("Calzone".equals(mealName)) return new CalzonePizza();
    if ("Tiramisu".equals(mealName)) return new Tiramisu();

    throw new IllegalArgumentException("Unknown meal '" + mealName + "'");
  }
}
```

What do we Want?

```
if ("Margherita".equals(mealName)) return new MargheritaPizza();
```

What do we Want?

ID ClassType

```
if ("Margherita".equals(mealName)) return new MargheritaPizza();
```

FactoryType

```
@Override
public boolean process(Set<? extends TypeElement> annotations,
                       RoundEnvironment env) {

  // Itearate over all @Factory annotated elements
  for (Element elem : env.getElementsAnnotatedWith(Factory.class)){
    if (annotatedElement.getKind() != ElementKind.CLASS) {
      messager.printMessage(Diagnostic.Kind.ERROR, “Ooops!”, elem);
      return true;
    }
  ...
  }
}
```

Then Element is a TypeElement

```
public class FactoryAnnotatedClass {

  private TypeElement annotatedClassElement;
  private String qualifiedClassName;   // Class Canonical name
  private String simpleTypeName;       // Class name
  private String id;                   // “Margherita”

  public FactoryAnnotatedClass(TypeElement classElement) throws Exception {
    Factory annotation = classElement.getAnnotation(Factory.class);
    id = annotation.id();
    …
  }
}
```

Getting the Data

```
try {
  Class<?> clazz = annotation.type();
  qualifiedClassName = clazz.getCanonicalName();
  simpleClassName = clazz.getSimpleName();
} catch (MirroredTypeException mte) {
  DeclaredType classTypeMirror = (DeclaredType) mte.getTypeMirror();
  TypeElement classTypeElement = (TypeElement) classTypeMirror.asElement();
  qualifiedClassName = classTypeElement.getQualifiedName().toString();
  simpleClassName = classTypeElement.getSimpleName().toString();
}
```

Writing an Output

```
public class PizzaFactory {
  public Meal create(String id) {

    if ("Margherita".equals(id)) return new my.MargheritaPizza();
    if ("Calzone".equals(id)) return new my.CalzonePizza();
    if ("Tiramisu".equals(id)) return new my.Tiramisu();

    throw new IllegalArgumentException(“Unknown " + id);
  }
}
```

Writing an Output

```
String src =
 “public class PizzaFactory { \n
    public ”+T+“create(String id) { \n“;
    for (FactoryAnnotatedClass clazz : clazzes) {
      String id = clazz.getId();
      String clazzName = clazz.getQualifiedClassName();
src+=“if (“ + id + ”.equals(id)) return new “+clazzName+”(); \n”;
    }
src+=“throw new IllegalArgumentException(“Unknown " + id); \n”
+  “} \n”
+“} \n”;
```

```
public void generateCode(Elements elementUtils, Filer filer)throws Exception{

  MethodSpec.Builder method = MethodSpec.methodBuilder("create")
      .addModifiers(Modifier.PUBLIC)
      .addParameter(String.class, "id")
      .returns(TypeName.get(superClassName.asType()));

   // For Each Found @Factory
method.beginControlFlow("if ($S.equals(id))", item.getId())
          .addStatement("return new $L()”,
item.getTypeElement().getQualifiedName().toString())
          .endControlFlow();


  TypeSpec typeSpec = TypeSpec.classBuilder(factoryClassName)
                              .addMethod(method.build()).build();

  JavaFile.builder(packageName, typeSpec).build().writeTo(filer);
}
```

https://github.com/square/javapoet

```
MethodSpec.Builder method = MethodSpec.methodBuilder("create")
        .addModifiers(Modifier.PUBLIC)
        .addParameter(String.class, "id")
        .returns(TypeName.get(superClassName.asType()));
```

```
  public Meal create(String id) {

    …
  }
```

```
for (FactoryAnnotatedClass item : clazzes) {
  method.beginControlFlow("if ($S.equals(id))", item.getId())
            .addStatement("return new $L()”,
  item.getTypeElement().getQualifiedName().toString())
            .endControlFlow();
}
```

```
 public Meal create(String id) {

   if ("Margherita".equals(id)) return new my.MargheritaPizza();
   if ("Calzone".equals(id)) return new my.CalzonePizza();
   if ("Tiramisu".equals(id)) return new my.Tiramisu();

   …
 }
```

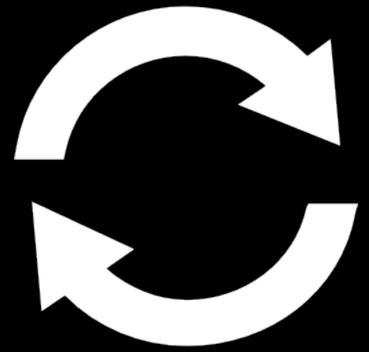
```
TypeSpec typeSpec = TypeSpec.classBuilder(factoryClassName)
                                        .addMethod(method.build()).build();
JavaFile.builder(packageName, typeSpec).build().writeTo(filer);
```

```
public class PizzaFactory {
  public Meal create(String id) {

    if ("Margherita".equals(id)) return new my.MargheritaPizza();
    if ("Calzone".equals(id)) return new my.CalzonePizza();
    if ("Tiramisu".equals(id)) return new my.Tiramisu();

    …
  }
}
```

https://github.com/square/javapoet



Round	Input	Output
1	CalzonePizza.java Tiramisu.java MargheritaPizza.java Meal.java PizzaStore.java	MealFactory.java
2	MealFactory.java	--- none ---
3	--- none ---	--- none ---

Processing Rounds

Google’s Compile Testing

```
Compiler.javac()
    .withProcessors(new MyProcessor())
    .compile(JavaFileObjects.forResource("com/example/MyClass"))
    .generatedSourceFile("MyGeneratedClass")
    .isPresent()
```

https://github.com/google/compile-testing

Annotation Processing

A tool build in javac for scanning and processing annotations at compile time

Dependency Injection

The fastest Java DI Framework!

https://google.github.io/dagger/

 Dagger 2 

```
import javax.inject.Inject;

class CoffeeMaker {
  @Inject Heater heater;
  @Inject Pump pump;
  ...
}
```

```java
@Component(modules = DripCoffeeModule.class)
interface CoffeeShop {
  CoffeeMaker maker();
}

@Module
class DripCoffeeModule {
  @Provides static Heater provideHeater() {
    return new ElectricHeater();
  }

  @Provides static Pump providePump(Thermosiphon pump) {
    return pump;
  }
}
```


```
import javax.inject.Inject;

class CoffeeMaker {
  @Inject Heater heater;
  @Inject Pump pump;
  ...
}
```

```
CoffeeShop coffeeShop = DaggerCoffeeShop.create();
```

 Dagger 2 

- Singletons and Scoped Bindings
- Lazy injections
- Provider injections
- Qualifiers

The fastest Java DI Framework!

https://google.github.io/dagger/

MapStruct

Bean Mapping at Compile Time

http://mapstruct.org/

```
public class Car {

    private String make;
    private int numberOfSeats;
    private CarType type;

    //constructor, getters, setters etc.
}
```

```java
public interface CarMapper {

    CarDto carToCarDto(Car car);
}
```

```java
@Mapper
public interface CarMapper {

    CarMapper INSTANCE = Mappers.getMapper( CarMapper.class );

    CarDto carToCarDto(Car car);
}
```

```java
@Mapper
public interface CarMapper {

    CarMapper INSTANCE = Mappers.getMapper( CarMapper.class );

    @Mappings({
        @Mapping(source = "make", target = "manufacturer"),
        @Mapping(source = "numberOfSeats", target = "seats")
    })
    CarDto carToCarDto(Car car);
}
```

MapStruct

- Nested mappings
- Updating existing bean instances

http://mapstruct.org/

Lombok

Spice up your Java

https://projectlombok.org/

@ToString

```
@ToString(exclude="id")
public class ToStringExample {
  private static final int STATIC_VAR = 10;
  private String name;
  private Shape shape = new Square(5, 10);
  private String[] tags;
  private int id;
}
```

@EqualsAndHashCode

```
@EqualsAndHashCode(exclude={"id", "shape"})
public class EqualsAndHashCodeExample {
  private transient int transientVar = 10;
  private String name;
  private double score;
  private Shape shape = new Square(5, 10);
  private String[] tags;
  private int id;
}
```

@RequiredArgsConstructor

```
@RequiredArgsConstructor(staticName = "of")
@AllArgsConstructor(access = AccessLevel.PROTECTED)
public class ConstructorExample<T> {
  private int x, y;
  @NonNull private T description;

   @NoArgsConstructor
   public static class NoArgsExample {
     @NonNull private String field;
   }
}
```

@Data & @Value

```java
@Data
public class PoJo {
  private String name;
  private double score;
  private String[] tags;
  private int id;
}
```

```java
@Value
public class PoJo {
  private String name;
  private double score;
  private String[] tags;
  private int id;
}
```

```
public String example() {
    val example = new ArrayList<String>();
    example.add("Hello, World!");
    val foo = example.get(0);
    return foo.toLowerCase();
}
```

val !!11one

```
public String example() {
    val example = new ArrayList<String>();
    example.add("Hello, World!");
    val foo = example.get(0);
    return foo.toLowerCase();
}
```

@ExtensionMethod

```
class Extensions {
    public static String toTitleCase(String in) {
      if (in.isEmpty()) return in;
      return "" + Character.toTitleCase(in.charAt(0)) +
          in.substring(1).toLowerCase();
    }
}

@ExtensionMethod({java.util.Arrays.class, Extensions.class})
public class ExtensionMethodExample {
  public String test() {return “hELlO,WORlD!”.toTitleCase();}
}
```

Lombok.config

lombok.fieldDefaults.defaultPrivate = true

Lombok

Spice up your Java

- @Builder
- @Log
- @Delegate

https://projectlombok.org/

- https://github.com/bluelinelabs/LoganSquare
- https://github.com/greenrobot/EventBus
- https://www.jooq.org/
- … and more

Conclusion
Annotation Processing

Conclusion
Annotation Processing

Powerful

Conclusion
Annotation Processing

Powerful
Fast

Conclusion
Annotation Processing

Powerful

Fast

Sometimes Brittle

Conclusion
Annotation Processing

Powerful

Fast

Sometimes Brittle

Sometimes a Life Saver


“Use it wisely we must!”

Thank You!
Q&A

Follow me at @AlexeyBuzdin

I Post Regular News and Updates about Java, Android and DevOps