АВТОМАТИЗАЦИЯ РАЗРАБОТКИ
КОДОГЕНЕРАЦИЯ

https://github.com/AKurtukov/CodeGen

АВТОМАТИЗАЦИЯ РАЗРАБОТКИ КОДОГЕНЕРАЦИЯ

- Dagger 2
- DataBinding
- Butterknife
- Moxy
- …

КОДОГЕНЕРАЦИЯ

\+

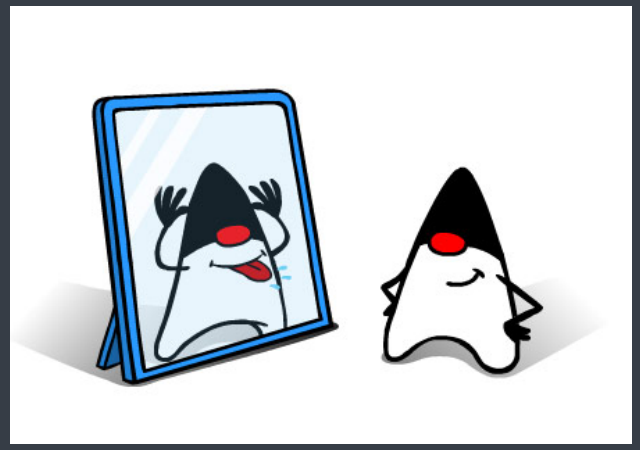
- Можно посмотреть сгенерированные классы
- Не замедляет работу классов
- Позволяет сокращать код
- Ускоряет разработку

\-

- Сложно тестируем
- Отсутствует stacktrace
- Отсутствует Debug
- Увеличивает время сборки
- Отсутствуют классы при первой компиляции
- Увеличение количества кода

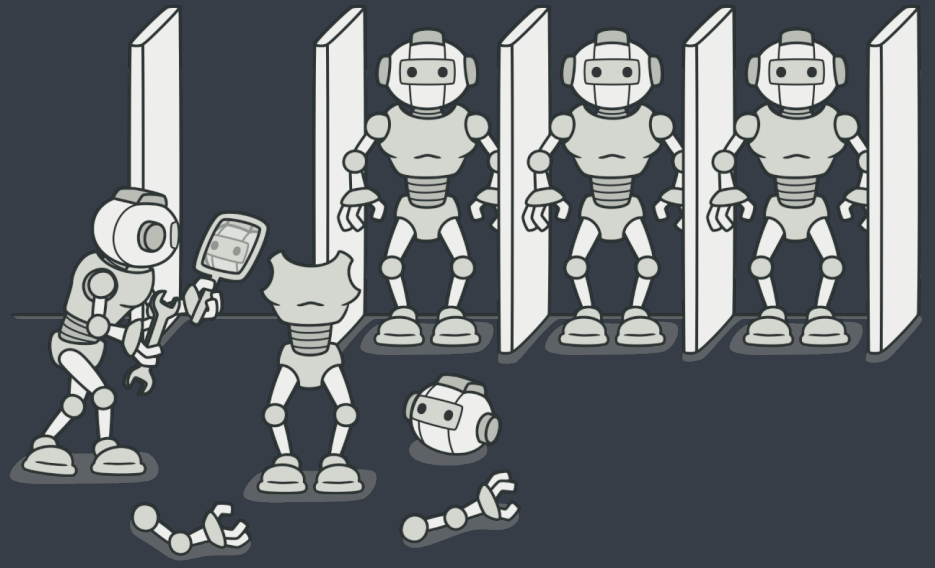
ИНТРОСПЕКЦИЯ

- **Интроспекция** — возможность языка анализировать самого себя во время выполнения



- Рефлексия — это механизм, с помощью которого можно вносить изменения и получать информацию о классах, интерфейсах, полях и методах во время выполнения, при этом не зная имен этих классов, методов и полей.

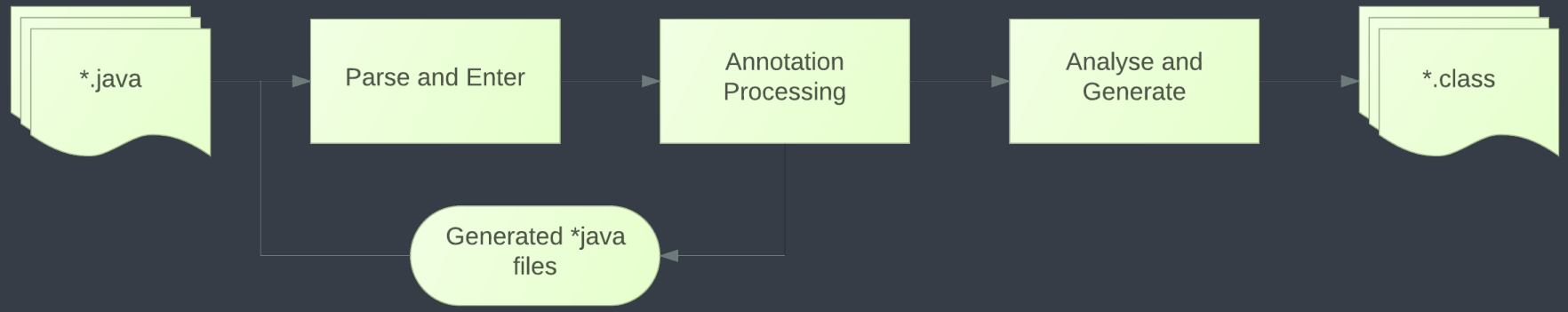
МЕТАПРОГРАММИРОВАНИЕ



- **МЕТАПРОГРАММИРОВАНИЕ** — ВИД ПРОГРАММИРОВАНИЯ, СВЯЗАННЫЙ С СОЗДАНИЕМ ПРОГРАММ, КОТОРЫЕ ПОРОЖДАЮТ ДРУГИЕ ПРОГРАММЫ КАК РЕЗУЛЬТАТ СВОЕЙ РАБОТЫ

2004 г. в Java 5 появился инструментарий Annotation Processing Tool

Весь процесс компиляции контролируется инструментарием Javac



СОЗДАЕМ ФАБРИКУ

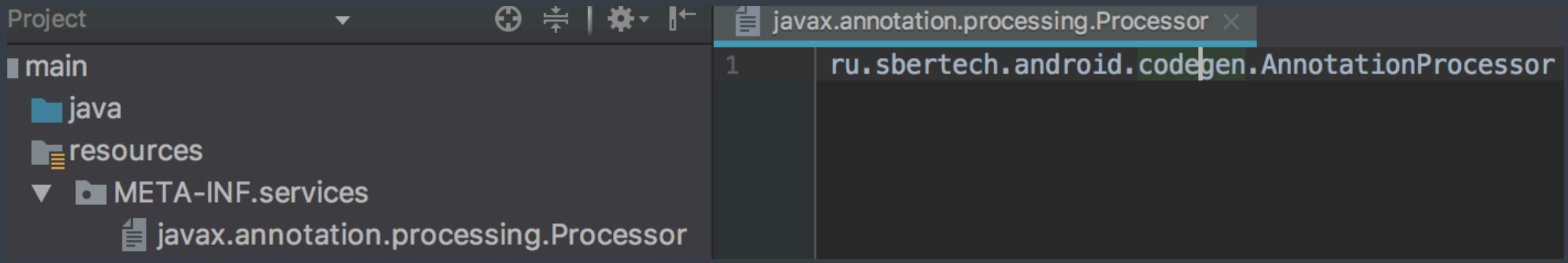
```java
public class USA implements Country {

    @Override
    public String getCapital() { return "Вашингтон"; }
}
```

```java
public interface Country {
    String getCapital();
}
```

```java
public class FactoryHandmade {

    private static final Map<String, Country> factoryMap = Collections.unmodifiableMap(new HashMap<String, Country
        {
            put("USA", new USA());
            put("Russia", new Russia());
            put("Germany", new Germany());
        }
    });

    public static Country create(String countryName) { return factoryMap.get(countryName); }
}
```

СОЗДАЕМ СВОЮ АННОТАЦИЮ

```java
@Retention(RetentionPolicy.SOURCE)
@Target(ElementType.TYPE)
public @interface CountryCodegenFactory {
    String value();
}
```



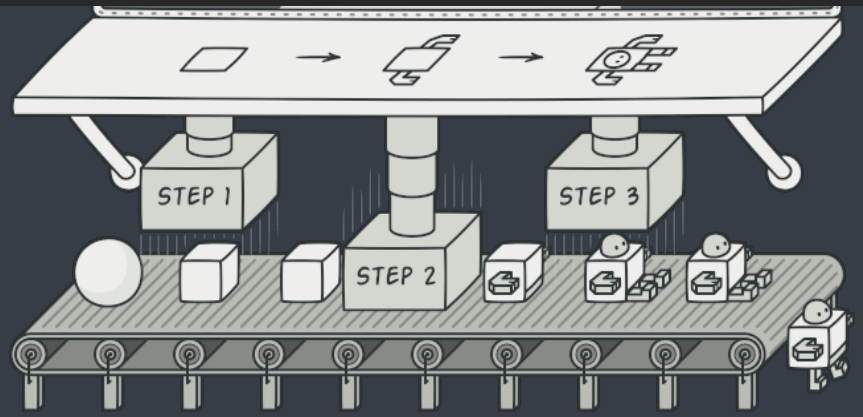

```java
@SupportedAnnotationTypes("ru.sbertech.android.codegen.CountryCodegenFactory")
@SupportedSourceVersion(SourceVersion.RELEASE_8)
public class AnnotationProcessor extends AbstractProcessor {
    @Override
    public boolean process(Set<? extends TypeElement> annotations, RoundEnvironment roundEnvironment) {

        return true;
    }
}
```

```
String putEntityString = "";
for (Element element : roundEnvironment.getElementsAnnotatedWith(CountryCodegenFactory.class)) {
    putEntityString = putEntityString + "put(\"" + element.getAnnotation(CountryCodegenFactory.class).value(
            + "\", new " + element.toString() + "()); \n        ";
}

try {
    String pack = clazz.getQualifiedName().toString();
    PrintWriter pw = new PrintWriter(w);
    pw.println("package "
            + pack.substring( i: 0, pack.lastIndexOf( i: '.')) + ";");
    pw.println("\nimport java.lang.String;");
    pw.println("\nimport java.util.Collections;");
    pw.println("\nimport java.util.HashMap;");
    pw.println("\npublic class Factory$$$Autogenerate {")
    pw.println("\n");
    pw.println("\n    private static final Map<String, Country> factoryMap = " +
            "Collections.unmodifiableMap(new HashMap<String, Country>() { ");
    pw.println("\n        {");
    pw.println(putEntityString);
    pw.println("\n        }");
    pw.println("\n    });");
    pw.println("\n    public static Country create(String countryName) {");
    pw.println("\n        return factoryMap.get(countryName);");
    pw.println("\n    }");
    pw.println("\n}");
    pw.flush();
```

ГЕНЕРИРУЕМ КОД

```
Factory$$$Autogenerate.create("Russia");
```

```
public class Factory$$$Autogenerate {

    private static final Map<String, Country> factoryMap = Collections.unmodifiableMap(new HashMap<String, Country>
        {
            put("USA", new USA());
            put("Russia", new Russia());
            put("Germany", new Germany());
        }
    });

    public static Country create(String countryName) { return factoryMap.get(countryName); }
}
```



ИНСТРУМЕНТЫ ГЕНЕРАЦИИ КОДА

square / javapoet

Watch 291 | Star 4,773 | Fork 602

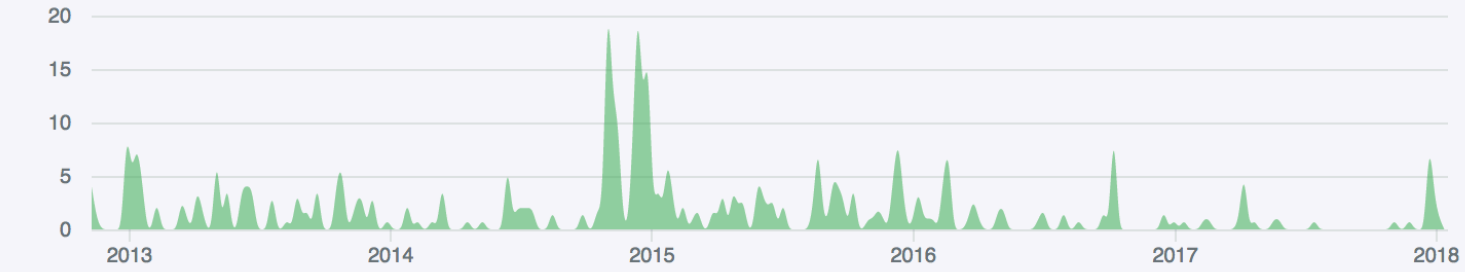
Code | Issues 43 | Pull requests 1 | Insights

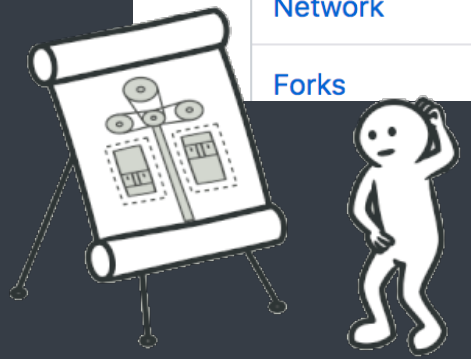
- Pulse
- **Contributors**
- Community
- Commits
- Code frequency
- Dependency graph
- Network
- Forks

Dec 9, 2012 – Feb 16, 2018

Contributions: **Commits**

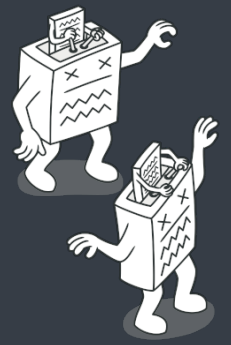
Contributions to master, excluding merge commits





```
compile 'com.squareup:javapoet:1.9.0'
```

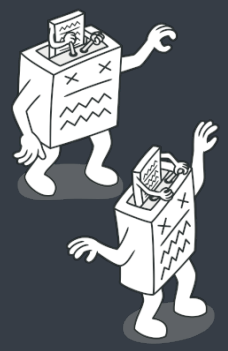
JAVAPOET



```
ClassName marker = ClassName.get( packageName: "ru.sbertech.android.codegen.myapplication",  simpleName: "Country");
ClassName stringClass = ClassName.get(String.class);

ClassName mapClass = ClassName.get(Map.class);
TypeName mapOfStringMarker = ParameterizedTypeName.get(mapClass, stringClass, marker);

ClassName hashMapClass = ClassName.get(HashMap.class);
TypeName hashMapOfStringMarker = ParameterizedTypeName.get(hashMapClass, stringClass, marker);
```

```
String putEntityString = "";
for (Element element : roundEnvironment.getElementsAnnotatedWith(CountryCodegenFactory.class)) {
    putEntityString = putEntityString + "put(\"" + element.getAnnotation(CountryCodegenFactory.class).value()
            + "\", new " + element.toString() + "()); \n        ";
}
```

JAVAPOET

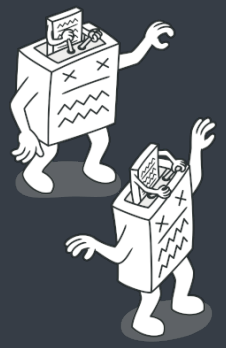


```
FieldSpec factoryMap = FieldSpec.builder(mapOfStringMarker, name: "__factoryMap")
        .addModifiers(Modifier.PRIVATE, Modifier.FINAL)
        .initializer("$T.unmodifiableMap(new $T()  {\n    {\n        $L \n    } \n})",
                Collections.class,
                hashMapOfStringMarker, putEntityString)
        .build();

MethodSpec create = MethodSpec.methodBuilder( name: "getCountry")
        .addModifiers(Modifier.PUBLIC, Modifier.STATIC)
        .returns(TypeName.get(Country.class))
        .addParameter(String.class,  name: "name")
        .addStatement("return __factoryMap.get(name)")
        .build();


TypeSpec autogenerateClass = TypeSpec.classBuilder( name: "Factory$$$Autogenerate")
        .addSuperinterface(CountryFactory.class)
        .addModifiers(Modifier.PUBLIC)
        .addField(factoryMap)
        .addMethod(create)
        .build();
```

```
JavaFile javaFile = JavaFile.builder( packageName: "ru.sbertech.android.codegen.myapplication.loggen",
        autogenerateClass)
        .build();
```

JAVAPOET

```
public class Factory$$$Autogenerate {
  private static final Map<String, Country> __factoryMap = Collections.unmodifiableMap(new Has
      {
          put("Russia", new ru.sbertech.android.codegen.myapplication.country.Russia());
          put("USA", new ru.sbertech.android.codegen.myapplication.country.USA());
          put("Germany", new ru.sbertech.android.codegen.myapplication.country.Germany());
      }
  });

  public static Country create(String value) { return __factoryMap.get(value); }
}
```



```
public class ProviderFactory {
    public static Country create(String countryName) {
        return Factory$$$Autogenerate.create(countryName);
    }
}
```

РЕФЛЕКСИЯ

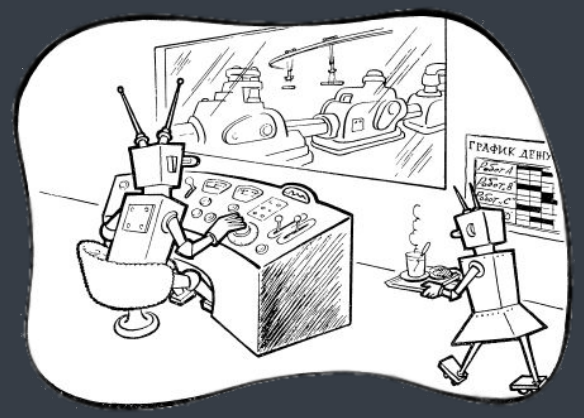

```
public FactoryReflection(Class<?>... clazz) { Collections.addAll(mObjects, clazz); }

public static Country create(String countryName) {
    for (Class<?> clazz : mObjects) {
        if (clazz.getAnnotation(CountryReflectionFactor.class) != null &&
                clazz.getAnnotation(CountryReflectionFactor.class).value().equals(countryName)) {
            try {
                Object o = clazz.newInstance();
                if (o instanceof Country) {
                    return (Country) o;
                }
            } catch (InstantiationException e) {
                e.printStackTrace();
```

ИТОГИ

```
FactoryHandmade.create("Germany");

Factory$$$Autogenerate.create("Russia");

new FactoryReflection(USA.class, Russia.class, Germany.class);
FactoryReflection.create("USA");

ProviderFactory.create("Russia");
```

Nexus 6p

№	*handmade*	*generated*	*reflection*
	nanoseconds		
1	143050	144650	447750
2	132460	138230	250820
3	132790	141450	276480
	nanoseconds		
1	161010	140800	449040
2	135040	132790	244410
3	150110	156840	274240



HTTPS://GITHUB.COM/AKURTUKOV/CODEGEN