

JVM Bytecode manipulation

When reflection is simply not enough

Piotr Lewandowski

goyello

Not an expert

not even a newbie...

Manipulation

" is the skillful handling, controlling or using of someone

instrumentation Bytecode manipulation

- Modify method body
- Replace statements
- Generate classes

It's **jRebel**
everywhere



SERIOUS SCIENTIFIC ANSWERS

to Absurd Hypothetical Questions



what if?

we make Java more like JavaScript

Hot Reload

Java Debugger API since 2002

Works good when

- 1. Modify method body
- 2. No new classes
- 3. No annotation changes

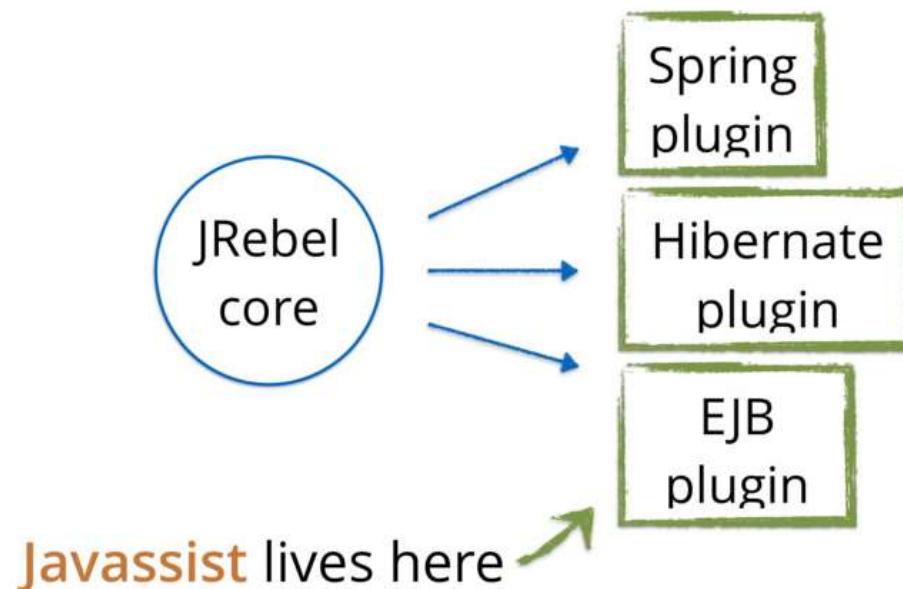


Not good for

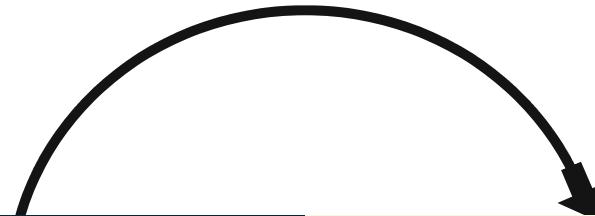
- 1. Method definition change
- 2. New fields declarations
- 3. Spring beans changes



JRebel



Javassist



Result

```
CtClass framework  
= cp.get("com.zt.Framework");
```

```
framework.addInterface(  
    cp.get("com.zt.jrebel.Listener"));
```

```
framework.addMethod(  
    CtNewMethod.make(  
        "public void onEvent() {" +  
        "    configure();" +  
        "}",  
        framework  
    ) );
```

```
class Framework {  
    public void configure() {  
        // Configure beans, set-up endpoints  
    }  
}
```

```
class Framework implements Listener {  
    public void configure() {  
        // ...  
    }  
}
```

```
class Framework implements Listener {  
    public void configure() {  
        // ...  
    }  
  
    public void onEvent() {  
        configure();  
    }  
}
```

It's useful and necessary

Monitoring and profiling

Generating classes from metadata

Caching

Obfuscation

Mocking



```
java.lang.NullPointerException: null
→ at net.piotr1.NewsService.get(NewsService.java:50)
   at net.piotr1.NewsService$$FastClassBySpringCGLIB$$a842897a.invoke(<generated>
   at org.springframework.cglib.proxy.MethodProxy.invoke(MethodProxy.java:204)
   at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invokeJoinpoint
   at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethod
   at org.springframework.transaction.interceptor.TransactionInterceptor$1.proceedWithInvoca
   at org.springframework.transaction.interceptor.TransactionAspectSupport.invokeWithinTransa
   at org.springframework.transaction.interceptor.TransactionInterceptor.invoke(TransactionI
   at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethod
   at org.springframework.aop.framework.CglibAopProxy$DynamicAdvisedInterceptor.intercept(Cg
   at net.piotr1.NewsService$$EnhancerBySpringCGLIB$$b6cab172.get(<generated>)
→ at net.piotr1.NewsController.get(NewsController.java:46)
```

Proxy Class

Reflection

```
/* Handle how proxy behaves */

InvocationHandler handler = (proxy, method, methodArgs) -> {
    if (method.getName().equals("get")) {
        return 42;
    } else {
        throw new UnsupportedOperationException(
            "Unsupported method: " + method.getName());
    }
}

/* Create proxy for Map */

Map mapProxy = (Map) Proxy.newProxyInstance(
    DynamicProxyTest.class.getClassLoader(),
    new Class[] { Map.class },
    handler
);

/* Result */

proxyInstance.get("hello"); // 42
proxyInstance.put("hello", "world"); // exception
```

Example

Proxy Class

Reflection

```
@Entity
class YourEntity {
    // getters ...
}

/* Create proxy */

YourEntity mapProxy = (YourEntity) Proxy.newProxyInstance(
    DynamicProxyTest.class.getClassLoader(),
    new Class[] { YourEntity.class },
    handler
);

/* Result */

java.lang.IllegalArgumentException:
    DynamicProxyTest$1YourEntity is not an interface
```

Proxy Class

Javassist

```
InvocationHandler handler = (proxy, method, methodArgs) -> {
    if (method.getName().equals("get")) {
        return 42;
    } else {
        throw new UnsupportedOperationException(
            "Unsupported method: " + method.getName()
        );
    }
}

public class JavassistLazyInitializer extends BasicLazyInitializer
    implements MethodHandler {

    final JavassistLazyInitializer instance = new JavassistLazyInitializer(...);

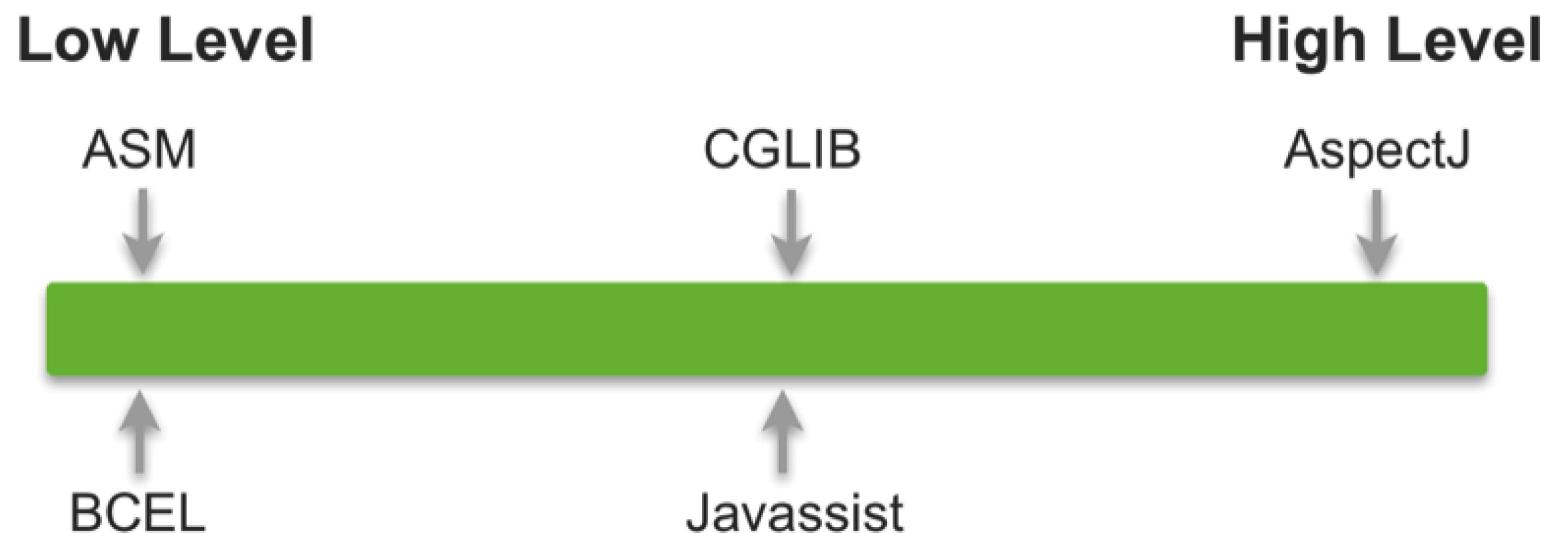
    ProxyFactory factory = new ProxyFactory();
    // factory set-up class metadata

    Class cl = factory.createClass();
    final HibernateProxy proxy = ( HibernateProxy ) cl.newInstance();
    ( (Proxy) proxy ).setHandler( instance );

    return proxy;
}
```



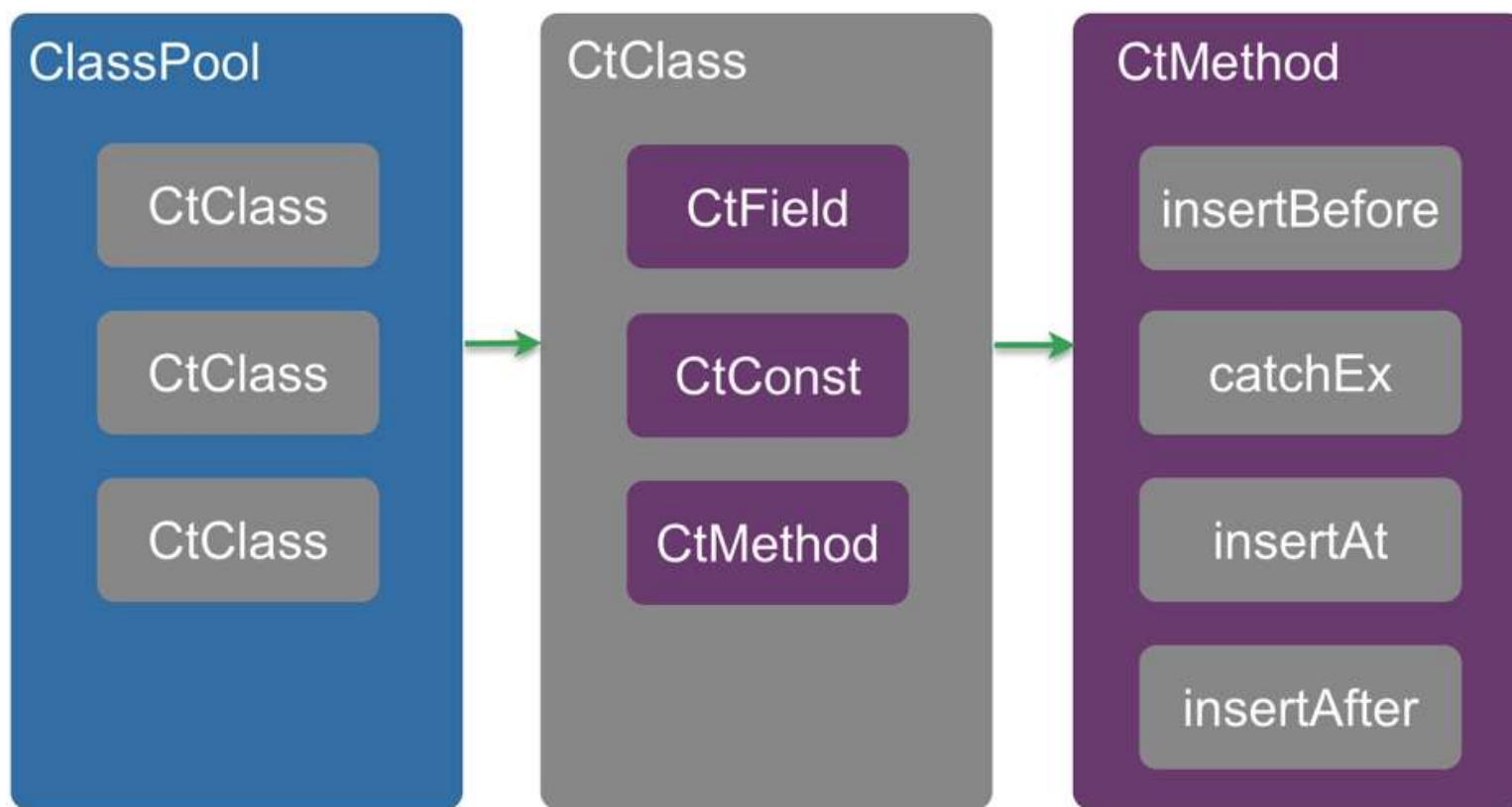
Libraries



Javassist

- Write code in strings
- What You See Is What You Get
- API similar to reflection

Javassist



Demo

Further reading

- Java agents
- [Instrumentation - idea and introduction](#)
- [Javassist - official tutorial](#)
- [Wideo - Having fun with Javassist](#)