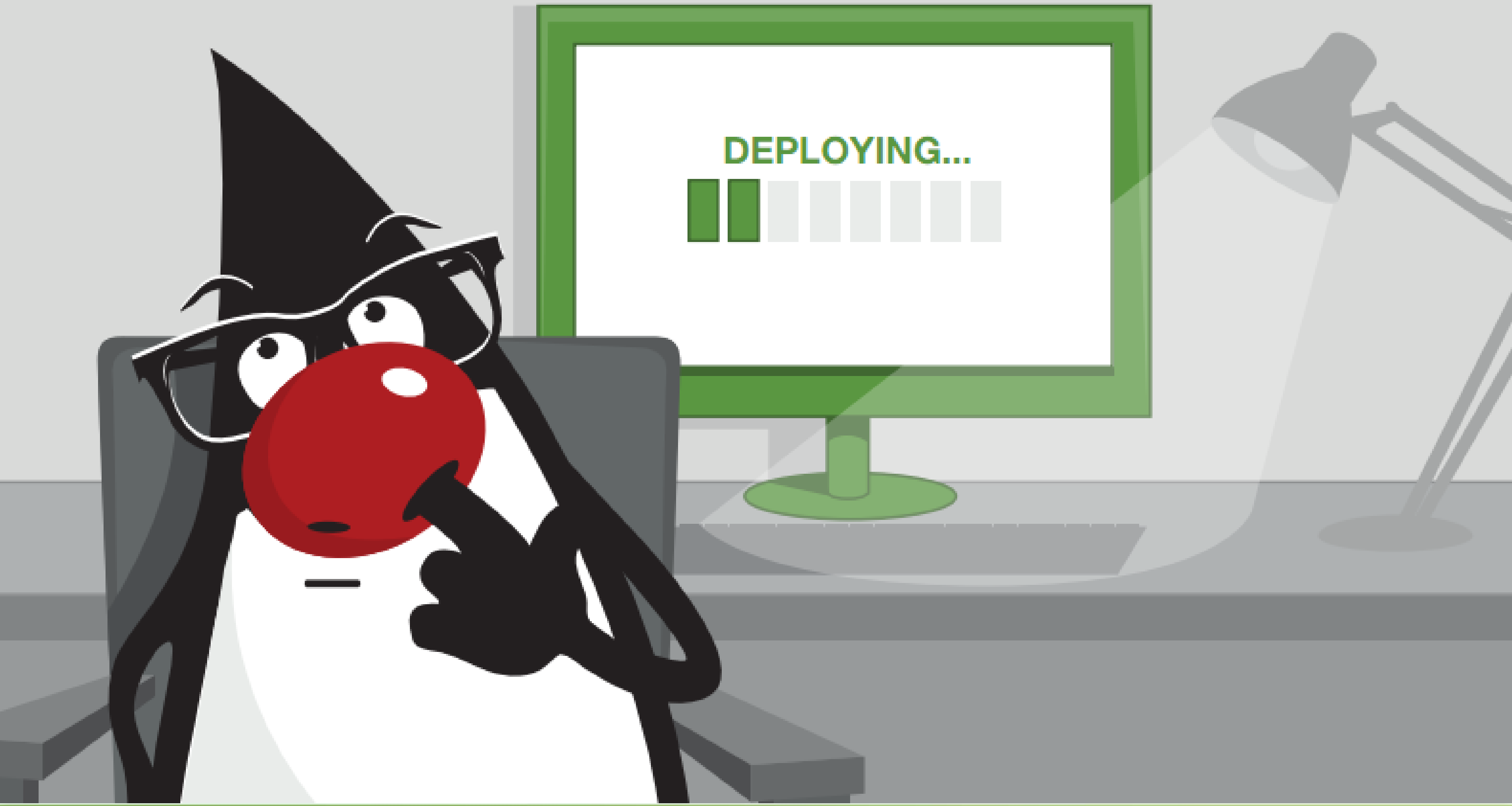
Redeploying in development?



Binary patching of Java classes for fun *and profit*

Jfokus 2011, Stockholm

whoami

Anton Arhipov
ZeroTurnaround
JRebel

http://arhipov.blogspot.com
@antonarhipov
@javarebel

What’s Binary Patching?

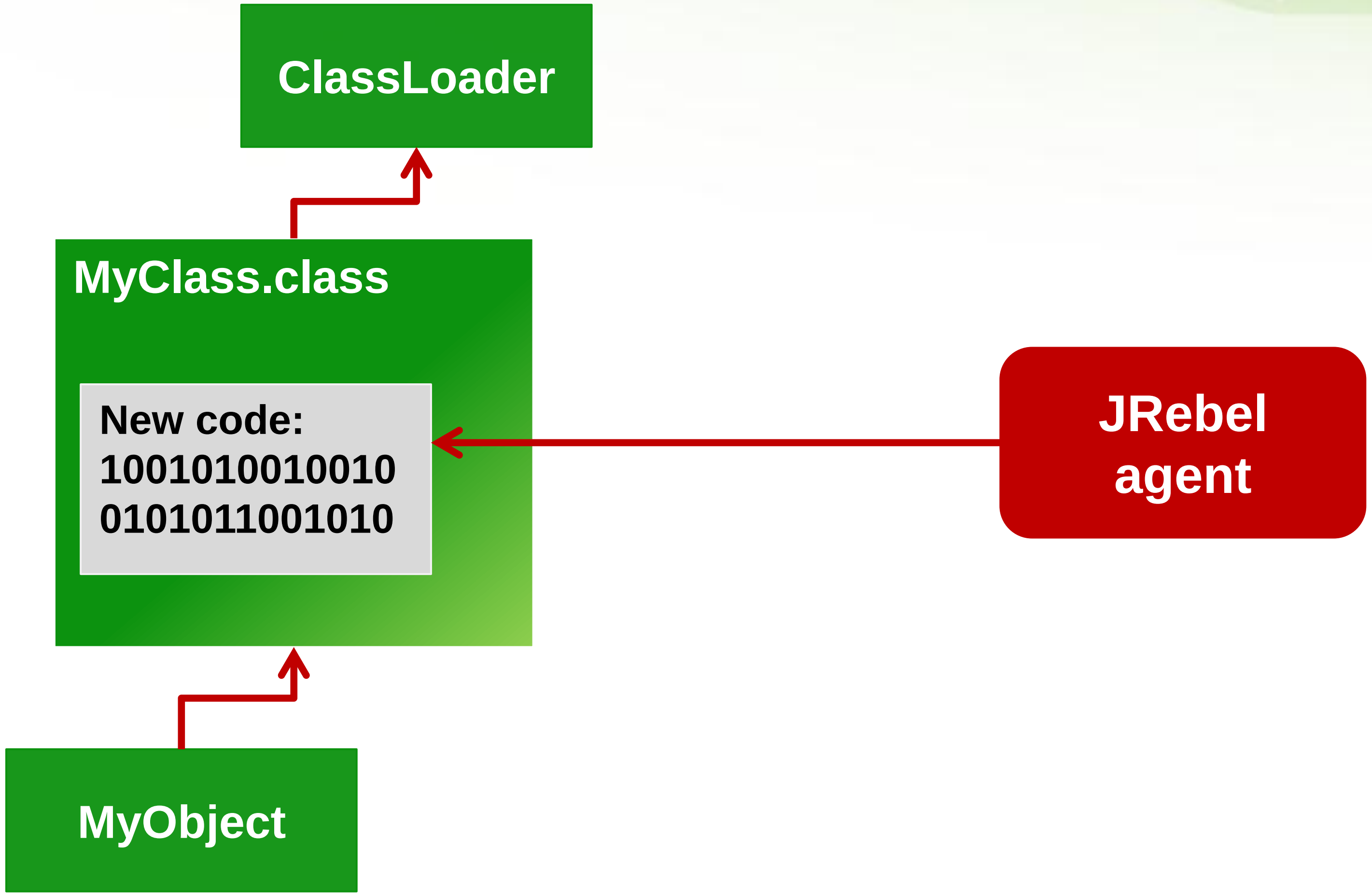
Ninja.class

```
10101010101
11000101010
10101010001
00010001110
11011101011
```

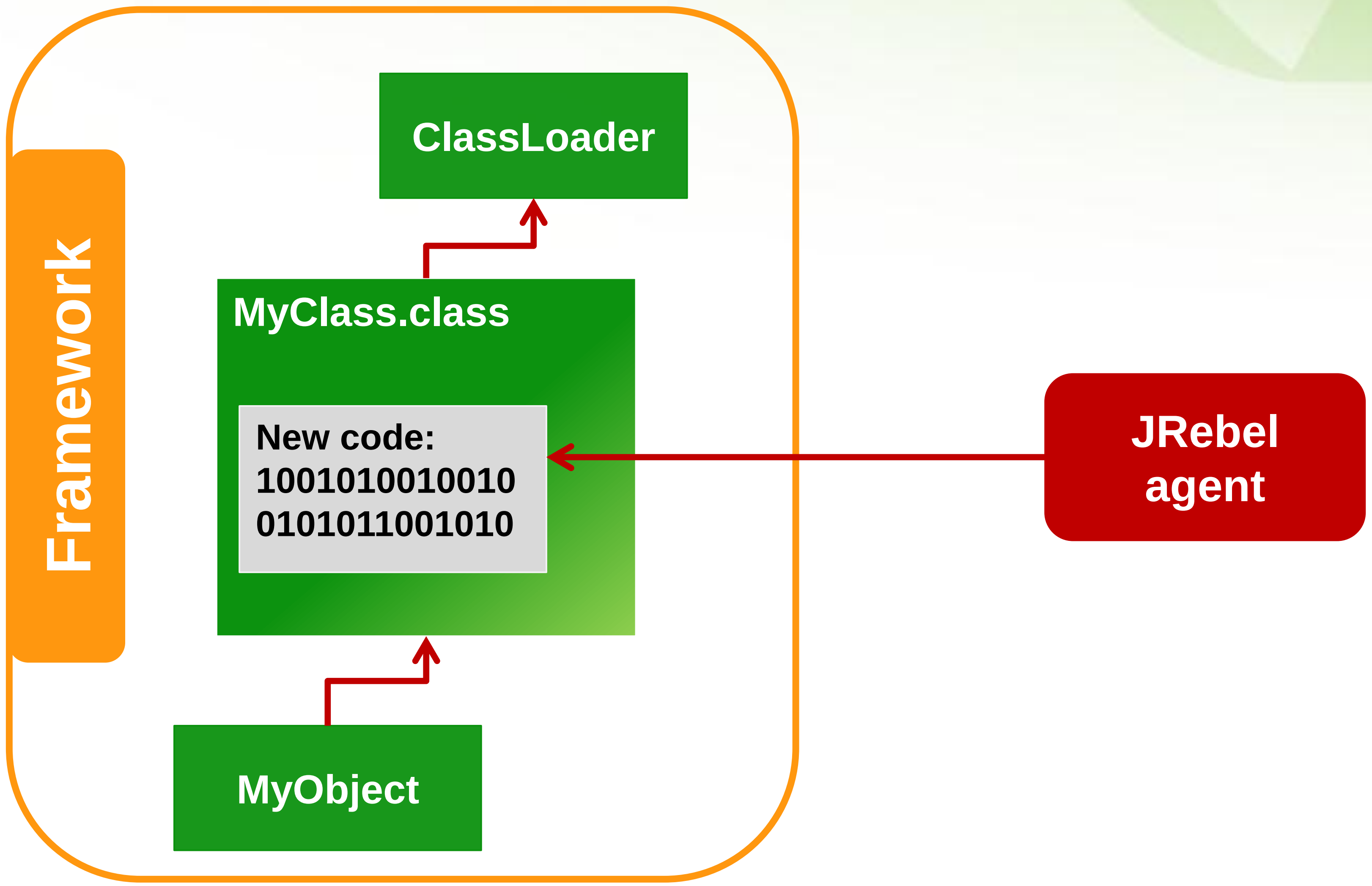
Ninja.class’

```
10101010101
11100001010
10101010001
00010001110
11011101110
```

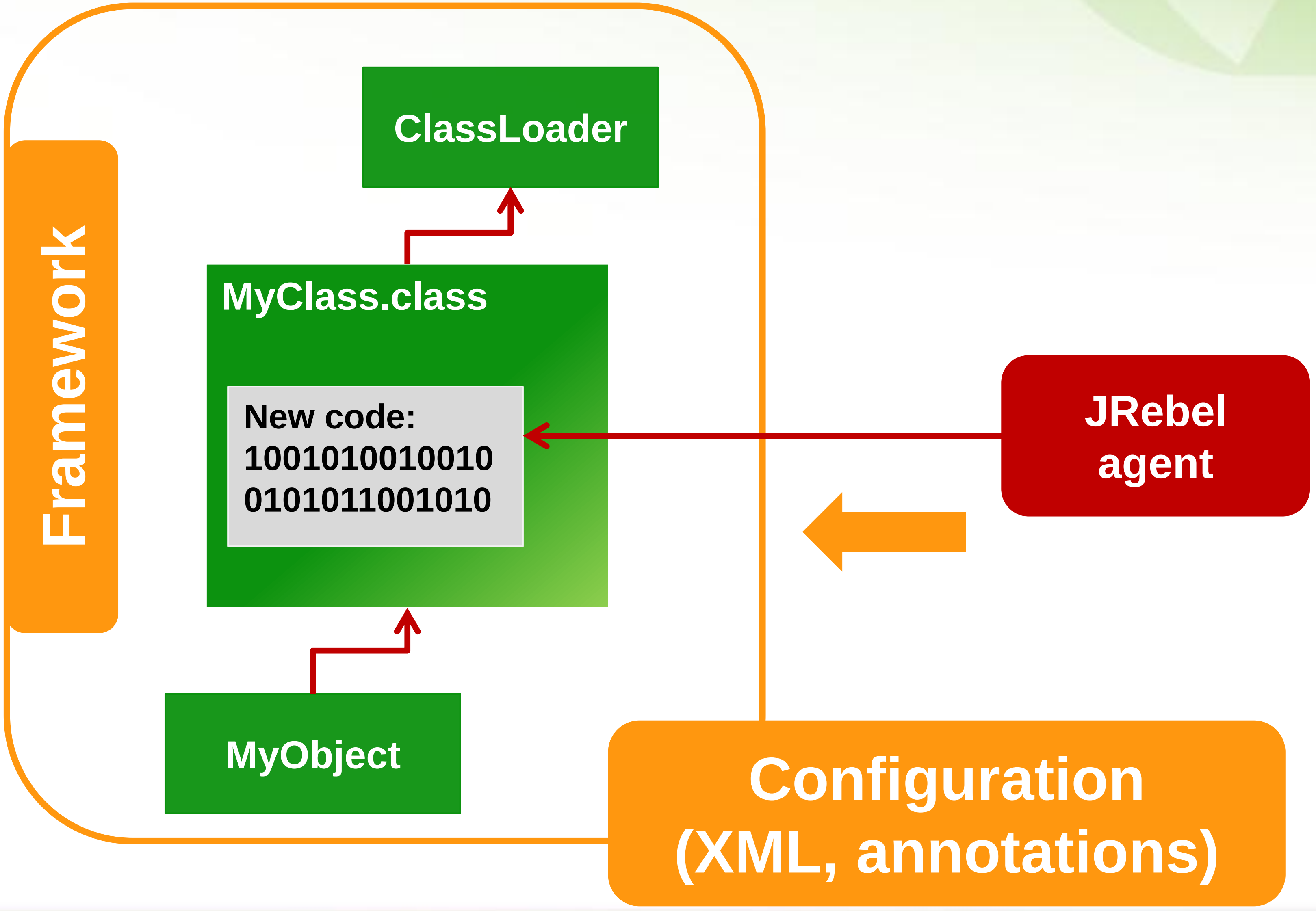
Why Binary Patching?



Why Binary Patching?



Why Binary Patching?



How?

- Using **–javaagent** to hook into class loading process
- Using bytecode manipulation libraries (e.g. **Javassist**)

java.lang.instrument

```java
import java.lang.instrument.ClassFileTransformer;
import java.lang.instrument.Instrumentation;

public class Agent {
  public static void premain(String args, Instrumentation inst)
        throws Exception {
      inst.addTransformer(new ClassFileTransformer { … });
   }
}
```

java.lang.instrument

```
import java.lang.instrument.ClassFileTransformer;
import java.lang.instrument.Instrumentation;

public class Agent {
  public static void premain(String args, Instrumentation inst)
        throws Exception {
      inst.addTransformer(new ClassFileTransformer { … });
   }
}
```

```
META-INF/MANIFEST.MF
Agent-Class: Agent
```

java.lang.instrument

```
import java.lang.instrument.ClassFileTransformer;
import java.lang.instrument.Instrumentation;

public class Agent {
  public static void premain(String args, Instrumentation inst)
       throws Exception {
     inst.addTransformer(new ClassFileTransformer { … });
  }
}
```

```
META-INF/MANIFEST.MF
Agent-Class: Agent
```

```
java –javaagent:agent.jar …
```

ZEROTURNAROUND

java.lang.instrument + Javassist

```
new ClassFileTransformer() {
  public byte[] transform(ClassLoader loader, String className,
                          Class<?>classBeingRedefined,
                          ProtectionDomain protectionDomain, byte[] classfileBuffer){









   }
}
```

java.lang.instrument + Javassist

```
new ClassFileTransformer() {
  public byte[] transform(ClassLoader loader, String className,
                                   Class<?>classBeingRedefined,
                                   ProtectionDomain protectionDomain, byte[] classfileBuffer){
    ClassPool cp = ClassPool.getDefault();
    CtClass ct = pool.makeClass(new
                           ByteArrayInputStream(classfileBuffer));



  }
}
```

java.lang.instrument + Javassist

```
new ClassFileTransformer() {
  public byte[] transform(ClassLoader loader, String className,
                          Class<?>classBeingRedefined,
                          ProtectionDomain protectionDomain, byte[] classfileBuffer){
    ClassPool cp = ClassPool.getDefault();
    CtClass ct = pool.makeClass(new
                          ByteArrayInputStream(classfileBuffer));

    transformClass(ct, cp);



  }
}
```

java.lang.instrument + Javassist

```
new ClassFileTransformer() {
  public byte[] transform(ClassLoader loader, String className,
                          Class<?>classBeingRedefined,
                          ProtectionDomain protectionDomain, byte[] classfileBuffer){
    ClassPool cp = ClassPool.getDefault();
    CtClass ct = pool.makeClass(new
                          ByteArrayInputStream(classfileBuffer));

    transformClass(ct, cp);

    return ct.toBytecode();
  }
}
```

JRebel SDK

Javassist + JRebel

```
cp.importPackage("org.zeroturnaround.javarebel");
```

Javassist + JRebel

```
cp.importPackage("org.zeroturnaround.javarebel");

ct.addInterface(
        cp.get(ClassEventListener.class.getName()));
```

Javassist + JRebel

```
cp.importPackage("org.zeroturnaround.javarebel");

ct.addInterface(
          cp.get(ClassEventListener.class.getName()));

ct.addMethod(CtNewMethod.make(
"public void onClassEvent(int eventType, Class clazz) {"+
    "cache.evict();" +
 "}", ct));
```

Javassist + JRebel

```
CtClass ct = …
CtConstructor[] cs = ct.getConstructors();
```

Javassist + JRebel

```
CtClass ct = ...
CtConstructor[] cs = ct.getConstructors();

for (CtConstructor c : cs) {
  if (c.callsSuper()) {
    c.insertAfter("ReloaderFactory.getInstance()
           .addClassReloadListener($0);");
  }
}
```

Javassist + JRebel

```
CtClass ct = …

ct.getDeclaredMethod("service")
.insertBefore(
    "ReloaderFactory.getInstance()
       .checkAndReload(Application.class);");
```

Integration Highlights

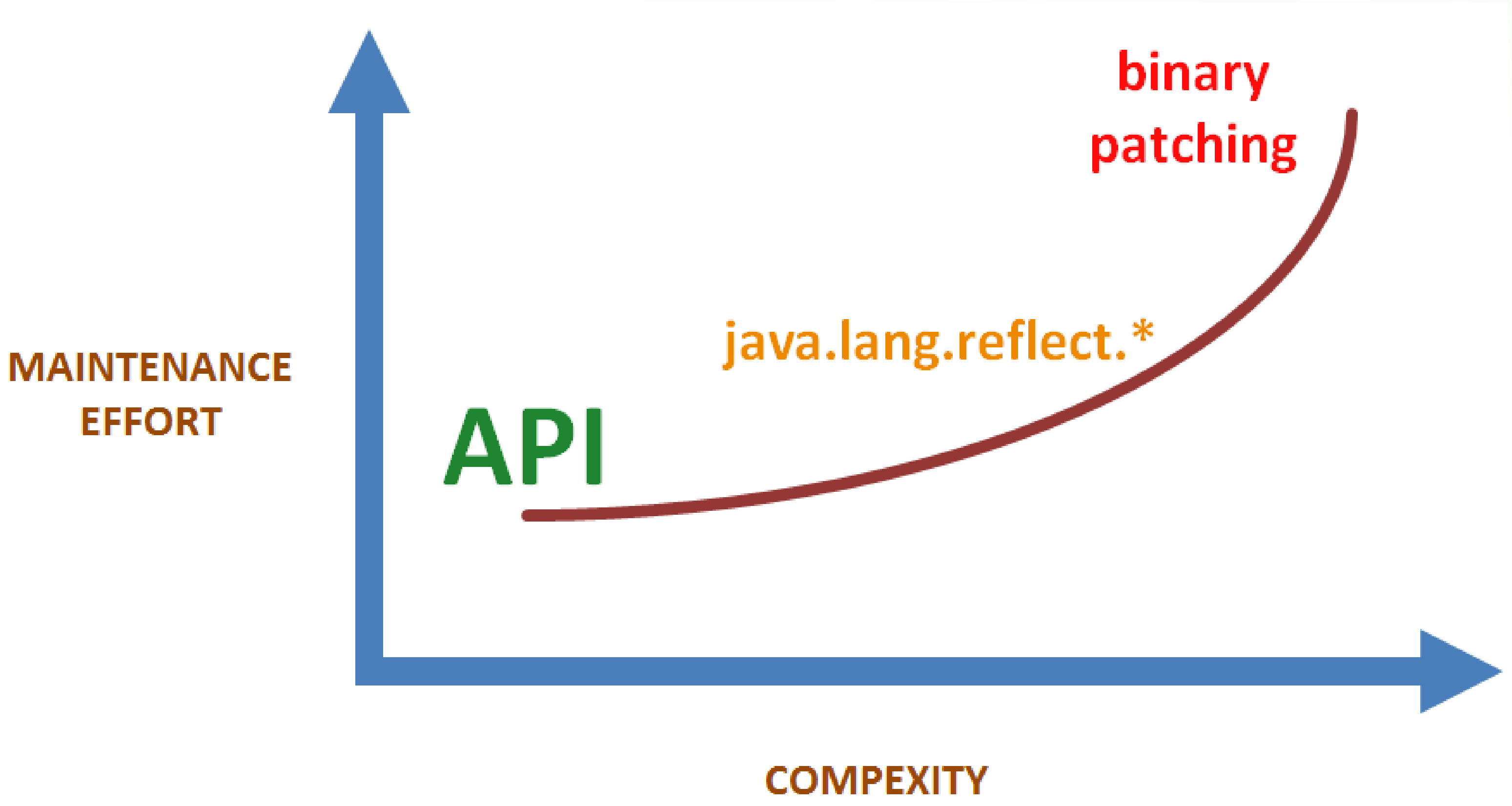
- Implement **ClassEventListener**
- Register **listener** instance to **JRebel**

```
ReloaderFactory#addClassReloadListener(…);
```

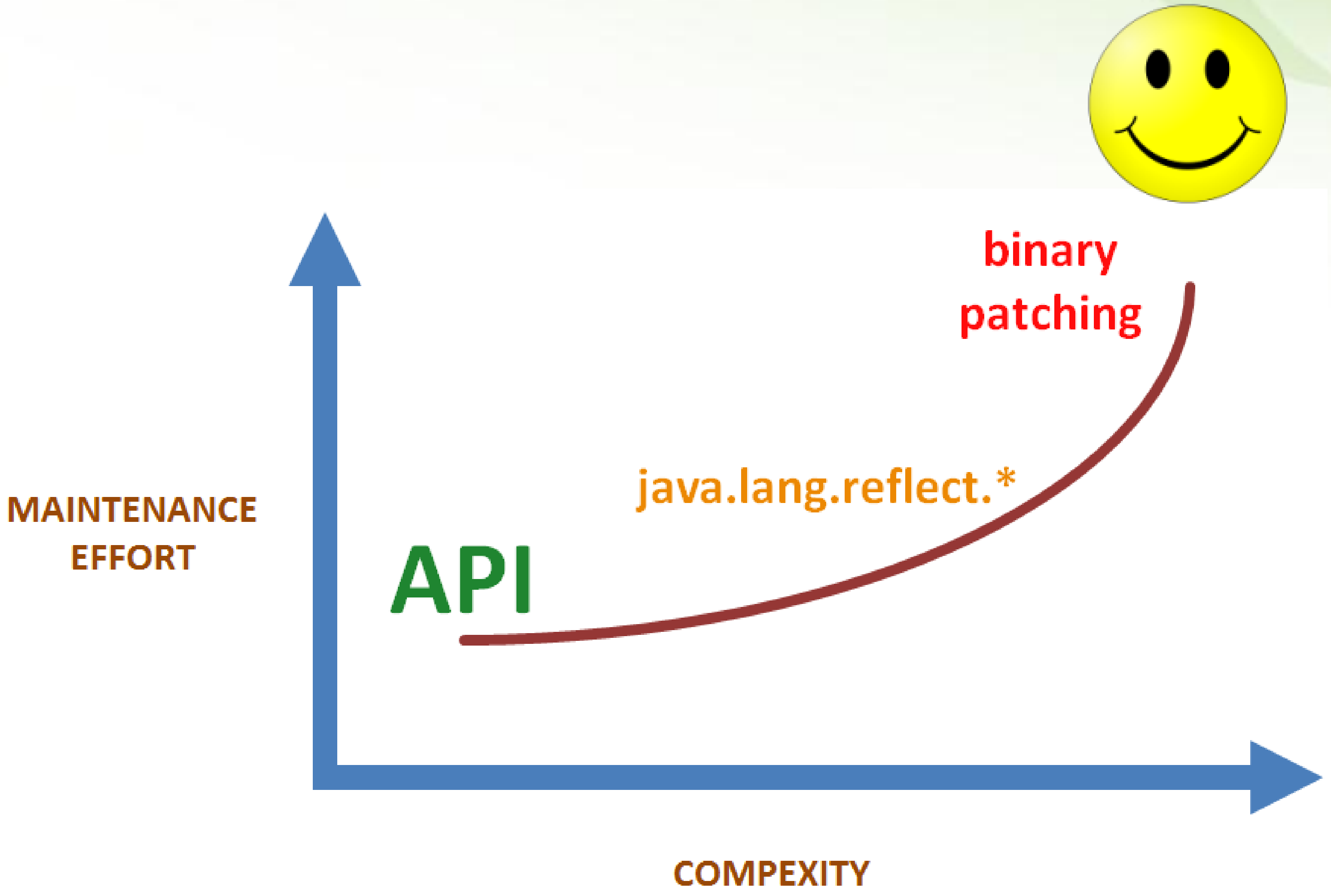
- Trigger the re-load event

```
ReloaderFactory#checkAndReload(…);
```

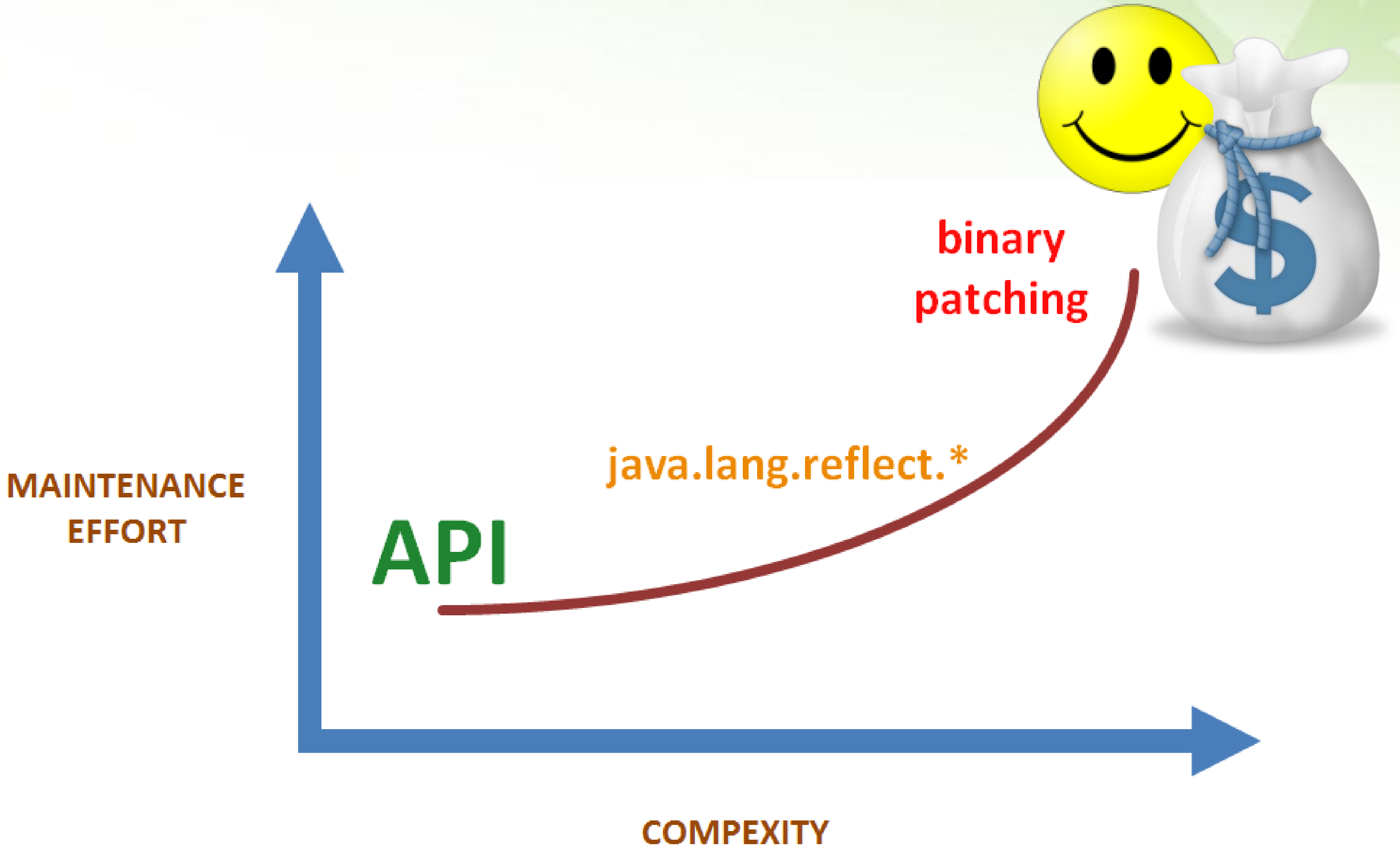
Pros/Cons


binary
patching
java.lang.reflect.*
MAINTENANCE
EFFORT
API
COMPEXITY

Pros/Cons


binary
patching
java.lang.reflect.*
MAINTENANCE
EFFORT
API
COMPEXITY

Pros/Cons


binary
patching
java.lang.reflect.*
API
MAINTENANCE
EFFORT
COMPEXITY

Thx!

Want JRebel free?

Come to our booth!