# HotCode2原理与应用

朱勇（千臂）

共享业务事件部–中间件–应用容器

2014.09

共享业务事业部 Shared Services Platform

## 主要内容

- HotCode功能介绍
- HotCode2核心原理
- HotCode2生态系统
- HotCode2使用与规划
- Q&A

# Part 1

## HotCode功能介绍

# HotCode2 Feature List

| 特性 | 修改方法体 | 添加删除方法 | 添加删除构造函数 | 添加删除域 | 添加删除注解 | 修改静态域 | 添加删除枚举值 | 修改接口 | 将一个类的超类换成另一个 | 给一个类添加删除接口 |
|---|---|---|---|---|---|---|---|---|---|---|
| ***HotCode2*** | √ | √ | √ | √ | √ | √ | √ | √ | ✗ | ✗ |
| ***Hotswap*** | √ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| 框架支持 | Spring 2.5.6/3.2.4<br>新增spring bean配置<br>Class 修改后自动autowire | | | Webx3 3.0.x/3.2.x<br>新增module<br>修改form.xml/resources.xml/uris.xml | | | IBatis 2.3.4<br>修改resultMap<br>修改sqlMap | | | **SDK**<br>开发工具包 |
| 容器/*IDE*支持 | | | jetty:// | | JBoss | | eclipse | | IntelliJIDEA<br>***comming soon*** | |

# Java Web常用开发框架

下面的Java开发框架，你都使用过吗?

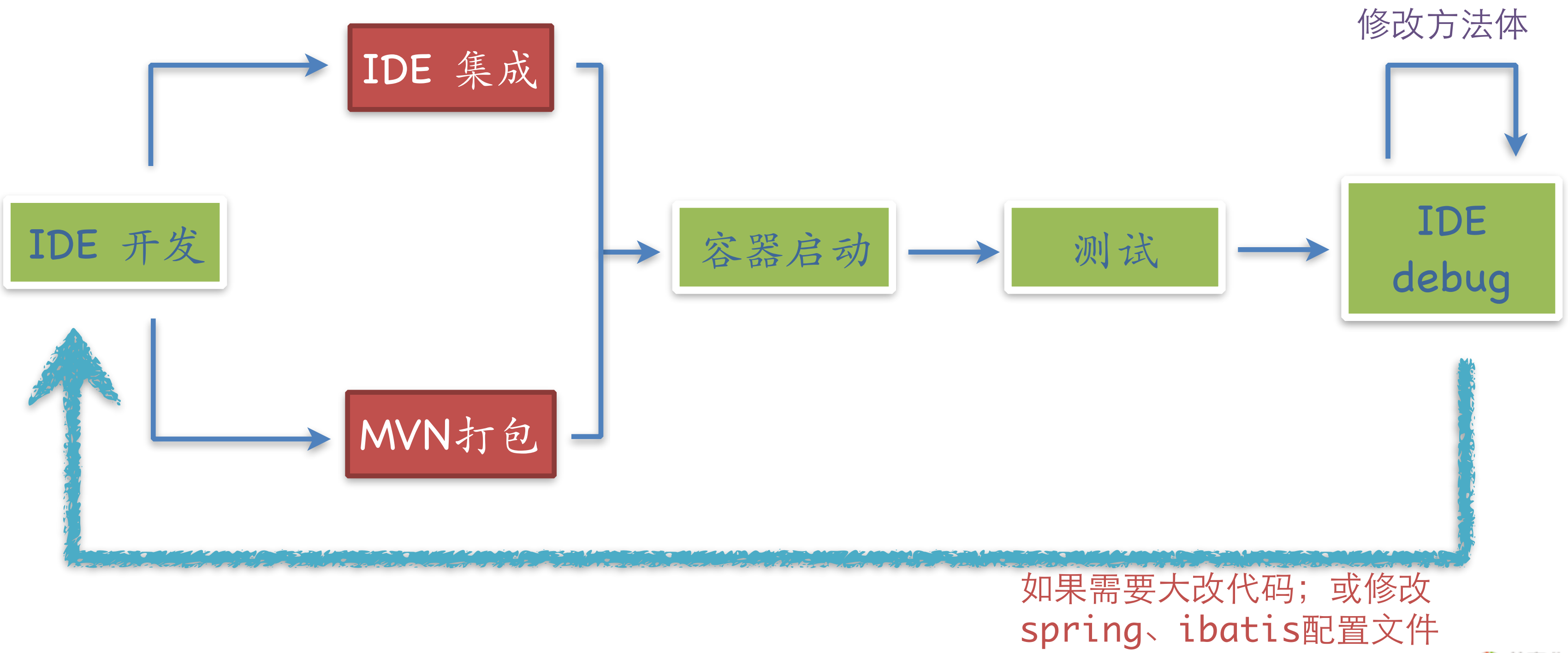
WebX 3 Powered By

iBATIS

spring

MyBatis

# 你的开发、部署流程是怎样的?



## 如何优化开发、部署流程?

HotCode is comming

Talk is cheap, show me your code!

## petstore示例

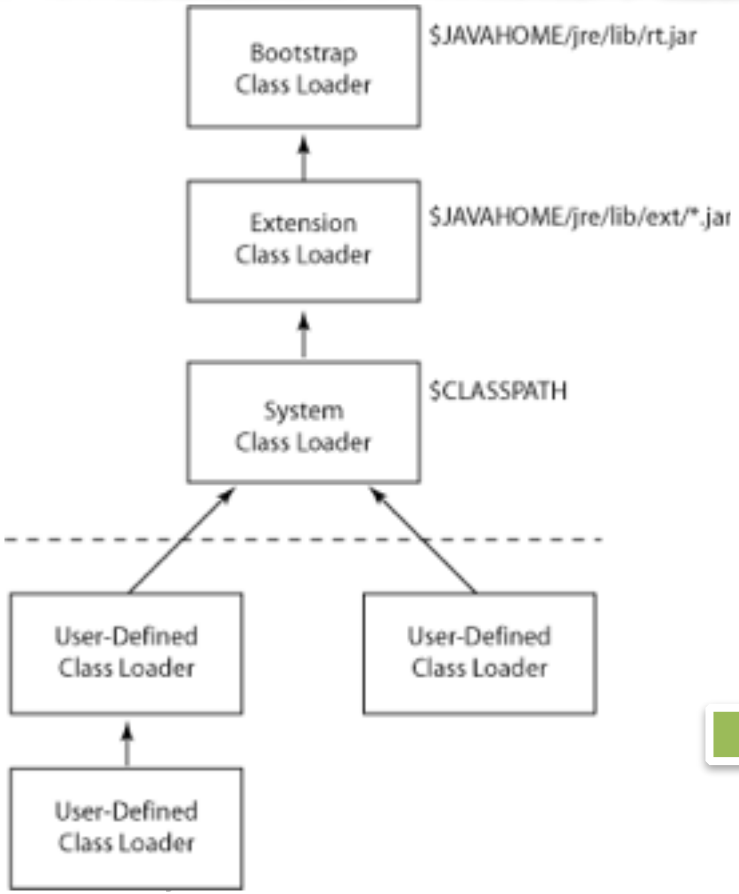
| | | | |
|---|---|---|---|
| *screen* | *bean*注入 | 参数注入 | 新增/修改方法 |
| *form.xml* | 修改验证器属性 | 新增验证器属性 | 新增属性 |
| *uris.xml* | 新增*broker* | 修改*broker* | |
| *sqlmap.xml* | 修改*sql*语句 | 增加*sql*语句 | |
| *spring.xml* | 新增*bean* | | |

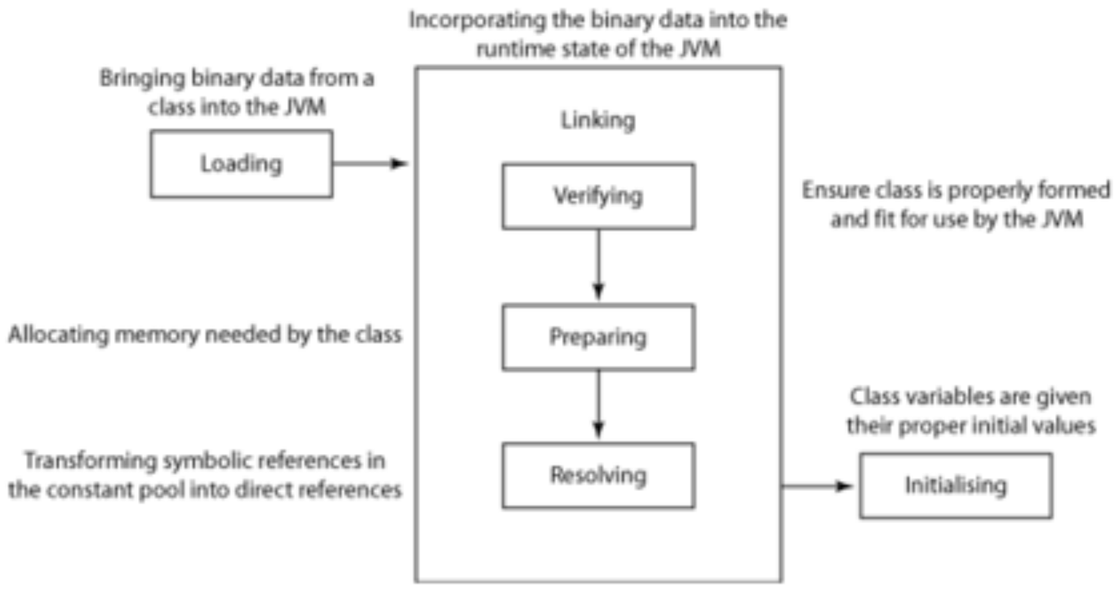
# Part 2

HotCode2原理与使用

## HotCode2核心原理

# JVM类加载体系

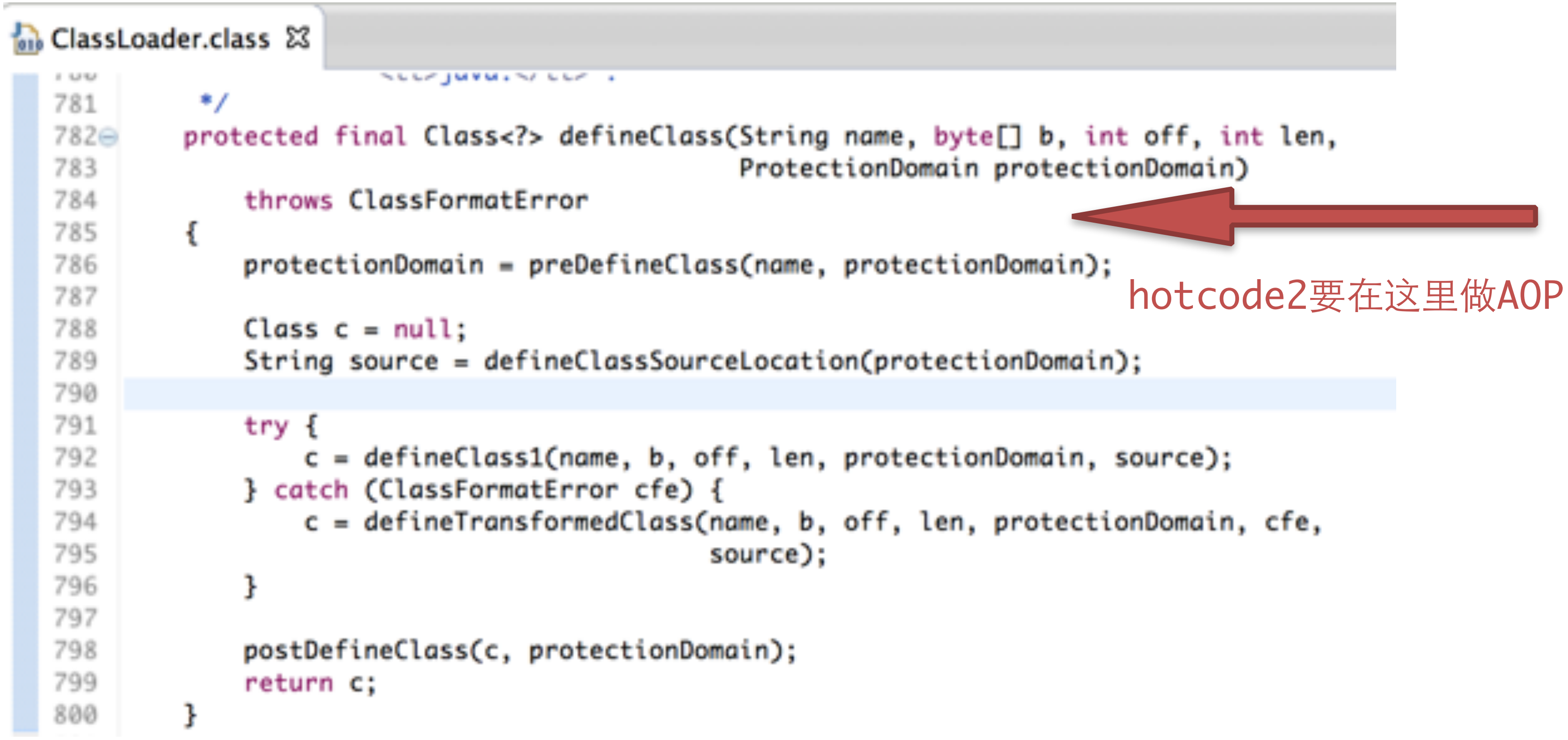

Bootstrap Class Loader
$JAVAHOME/jre/lib/rt.jar
Extension Class Loader
$JAVAHOME/jre/lib/ext/*.jar
System Class Loader
$CLASSPATH
User-Defined Class Loader
User-Defined Class Loader
User-Defined Class Loader


Incorporating the binary data into the runtime state of the JVM
Bringing binary data from a class into the JVM
Loading
Linking
Verifying
Ensure class is properly formed and fit for use by the JVM
Allocating memory needed by the class
Preparing
Class variables are given their proper initial values
Transforming symbolic references in the constant pool into direct references
Resolving
Initialising

hotcode2切入点：

java/lang/ClassLoader

# java/lang/ClassLoader



ClassLoader.class

```
    */
    protected final Class<?> defineClass(String name, byte[] b, int off, int len,
                                         ProtectionDomain protectionDomain)
        throws ClassFormatError
    {
        protectionDomain = preDefineClass(name, protectionDomain);

        Class c = null;
        String source = defineClassSourceLocation(protectionDomain);

        try {
            c = defineClass1(name, b, off, len, protectionDomain, source);
        } catch (ClassFormatError cfe) {
            c = defineTransformedClass(name, b, off, len, protectionDomain, cfe,
                                       source);
        }

        postDefineClass(c, protectionDomain);
        return c;
    }
```

hotcode2要在这里做AOP

## Java Instrumentation

- Java SE 6引入
- 最大作用，是类定义动态改变和操作
- 独立于应用程序的代理程序（Agent）
- AOP on JVM

## HotCode2 Agent

```
public static void premain(String agentArgs, Instrumentation inst) {
    ClassRedefiner.setInstrumentation(inst);
    ConfigurationFactory.getInstance();
    HotCodeEnv.asyncPrintHotCodeInfo();
    redefineJdkClasses();
}
```

# redefineJdkClasses

```
private static void redefineJdkClasses() {
    for (Entry<Class<?>, Class<? extends ClassVisitor>> entry :
            JdkClassProcessorFactory.JDK_CLASS_PROCESSOR_HOLDER.entrySet()) {
        InputStream is = null;
        try {
            ClassWriter cw = new ClassWriter(ClassWriter.COMPUTE_MAXS + ClassWriter.COMPUTE_FRAMES);
            ClassVisitor cv = cw;
            Constructor<? extends ClassVisitor> c = entry.getValue().getConstructor(ClassVisitor.class);
            cv = c.newInstance(cv);
            is = ClassLoader.getSystemResourceAsStream(Type.getInternalName(entry.getKey()) + ".class");
            ClassReader cr = new ClassReader(is);
            cr.accept(cv, ClassReader.EXPAND_FRAMES);
            byte[] transformedByte = cw.toByteArray();
            ClassRedefiner.redefine(entry.getKey(), transformedByte);
            ……
        } catch (Exception e) {
            ……
        } finally {
            ……
        }
    }
}
```

## java.lang.ClassLoader->ClassLoaderAdapter

```
public class ClassLoaderAdapter extends ClassVisitor {

    public ClassLoaderAdapter(ClassVisitor cv) {
        super(Opcodes.ASM4, cv);
    }

    @Override
    public MethodVisitor visitMethod(int access, String name, String desc, String
            signature, String[] exceptions) {
        MethodVisitor mv = super.visitMethod(access, name, desc, signature,
                exceptions);

        // Our HotCode hack code

        return mv;
    }
}
```

# HotCode hack code

```
if (name.equals("defineClass")
    && desc.equals(Type.getMethodDescriptor(Type.getType(Class.class), Type.getType(String.class),
            Type.getType(byte[].class), Type.INT_TYPE, Type.INT_TYPE,
            Type.getType(ProtectionDomain.class)))) {
    return new MethodVisitor(Opcodes.ASM4, mv) {
        public void visitCode() {
            super.visitCode();
            mv.visitVarInsn(Opcodes.ALOAD, 0);
            mv.visitMethodInsn(Opcodes.INVOKESTATIC, Type.getInternalName(CRMManager.class),
                    "registerClassLoader", "(Ljava/lang/ClassLoader;)V");
            mv.visitVarInsn(Opcodes.ALOAD, 1);
            mv.visitVarInsn(Opcodes.ALOAD, 0);
            mv.visitVarInsn(Opcodes.ALOAD, 2);
            mv.visitMethodInsn(Opcodes.INVOKESTATIC, Type.getInternalName(ClassTransformer.class),
                    "transformNewLoadClass", "(Ljava/lang/String;Ljava/lang/ClassLoader;[B)[B");
            mv.visitVarInsn(Opcodes.ASTORE, 2);
            mv.visitVarInsn(Opcodes.ALOAD, 2);
            mv.visitInsn(Opcodes.ARRAYLENGTH);
            mv.visitVarInsn(Opcodes.ISTORE, 4);
        }
    };
}
```

hotcode2一切字节码变换的入口

## hotcode dump & 字节码查看工具

/tmp/hotcode2 --> hotcode transform过的类被加载时的class文件dump目录
/tmp/hotcode2/reload --> hotcode管理的类在修改后reload的dump目录

javap -v -p : Jdk自带

Bytecode Outline : Eclipse plugin

JD-GUI : 桌面工具

## Field

```java
public class A {

    private byte bt;
    private boolean b;
    private int i;
    private long l;
    private float f;
    private double d;
    private char c;
    ……
}
```

hotcode2变换后

```java
public class A {
        private byte bt;
        private boolean b;
        private int i;
        private long l;
        private float f;
        private double d;
        private char c;
        ……
        private FieldHolder
                __hotcode_instance_fields__;
}
```

# 新增Field

```
public class A {

    private byte bt;
    private boolean b;
    private int i;
    private long l;
    private float f;
    private double d;
    private char c;
    ……
    private Object o;
}
```

reload后

```
public class A {
    private byte bt;
    private boolean b;
    private int i;
    private long l;
    private float f;
    private double d;
    private char c;
    ……
    private FieldHolder
            __hotcode_instance_fields__;
}
```

Object o在哪里?

FieldHolder本质上是一个Map

## Constructor

```
public class A {

    public A() {

    }

    ……
}
```

hotcode2变换后

```
public class A {

    public A() {

    }
    ……

    public A(HotCodeGenConstructorMarker marker,
            int index, Object[] args) {
        ……
    }
}
```

## 特殊的Constructor

```
public Base(com.taobao.hotcode2.adapter.marker.HotCodeGenConstructorMarker, int, java.lang.Object[]);
flags: ACC_PUBLIC
Code:
  stack=4, locals=4, args_size=4
     0: getstatic     #260                // Field __hotcode_class_reloader_field__:ClassReloader
     3: invokevirtual #265                // Method ClassReloader.checkAndReload:()Z
     6: iconst_1
     7: if_icmpne     18
    10: aload_0
    11: aload_1
    12: iload_2
    13: aload_3
    14: invokespecial #353                // Method "<init>":(LHotCodeGenConstructorMarker;I[LObject;)V
    17: return
    18: new           #333                // class HotCodeException
    21: dup
    22: ldc_w         #355                // String un reachable <init> code.
    25: invokespecial #339                // Method HotCodeException."<init>":(LString;)V
    28: athrow
  StackMapTable: number_of_entries = 1
       frame_type = 18 /* same */
```

## 新增Constructor

```java
public class A {

    private int i;

    public A() {

    }

    public A(String s) {
        if (s != null) {
            Assert.assertEquals("AAA", s);
            i = 1;
        } else {
            i = 11;
        }
    }
```

reload后

javap看结果?

## Method

```
public static Object __hotcode_static_method_router__com$taobao$Base(
        int, Object[]);

public Object __hotcode_private_instance_method_router__com$taobao$Base(
        int, Object[]);

public Object __hotcode_instance_method_router__(
        int, Object[]);

public Object __hotcode_package_instance_method_router__com$taobao(
        int, Object[]);
```

# HotCode辅助类

- XxxShadowClass
  - 解决Field/Constructor/Method的反射
  - 解决Class/Field/Constructor/Method的Annotation
  - 仅仅具有代码的框架
- XxxAssistClass
  - 解决Interface的reinit
  - 协助方法的调用

## 其他需要处理的细节

- Reflect (Field/Method/Constructor)
- Enum
- Annation (Class/Method/Field/Param)
- Jdk Proxy (Cglib……)
- AccessCheck
- Visibility

# 一些有代表性的问题和Bug

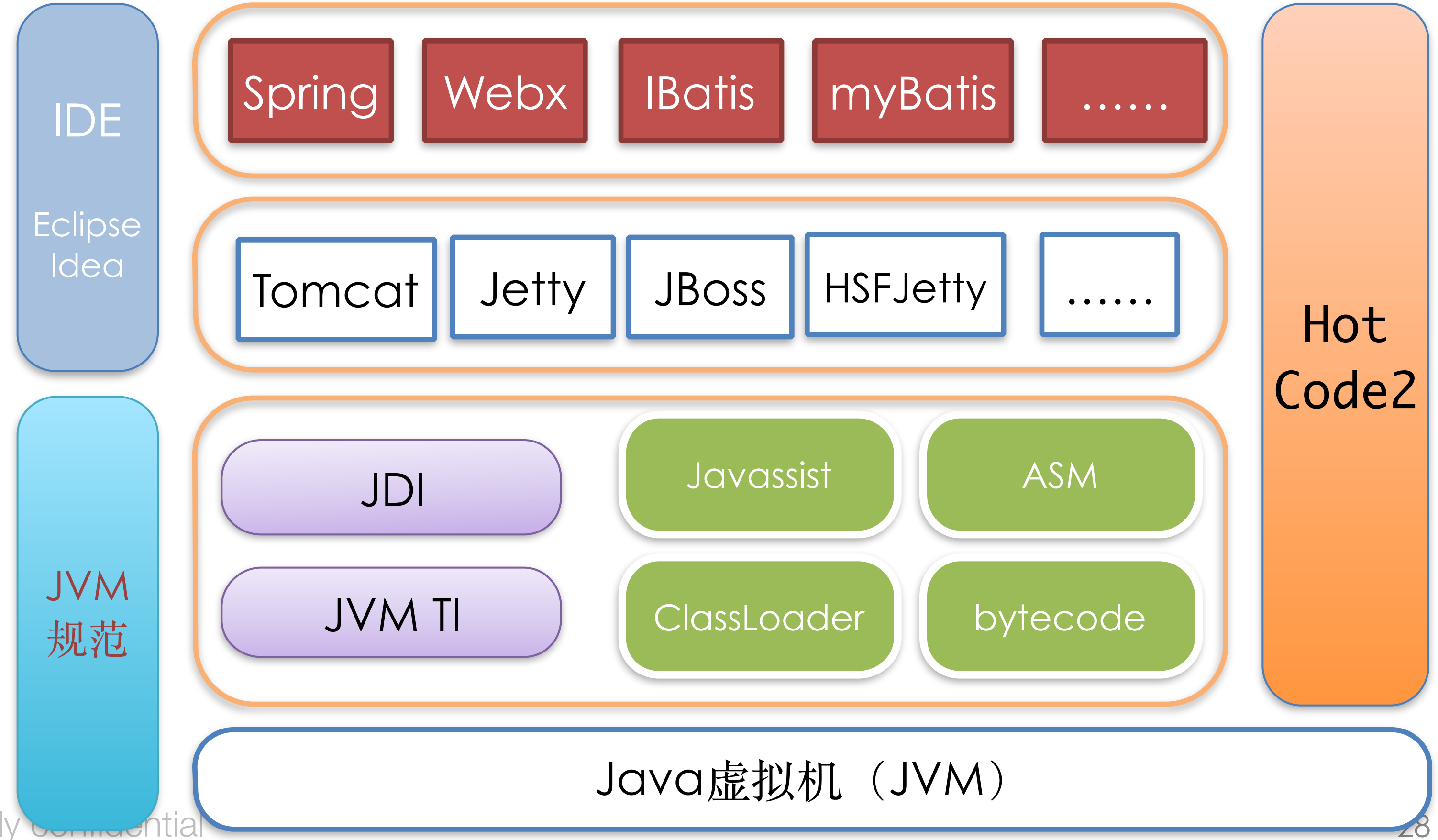
- 类相互引用导致的StackOverFlow问题
- static final带默认值的Field处理
- Interface reinit问题
- Constructor局部变量偏移问题
- Assist类访问受限问题
- byte code error导致GC hang住问题
- ……

# Part 3

HotCode2原理与使用

HotCode2生态系统

# HotCode2生态系统



## HotCode2生态建设

- 应用容器适配
- 框架集成、插件开发
- IDE集成开发环境
- 提高生产力，人员、能力、成长

# Part 4

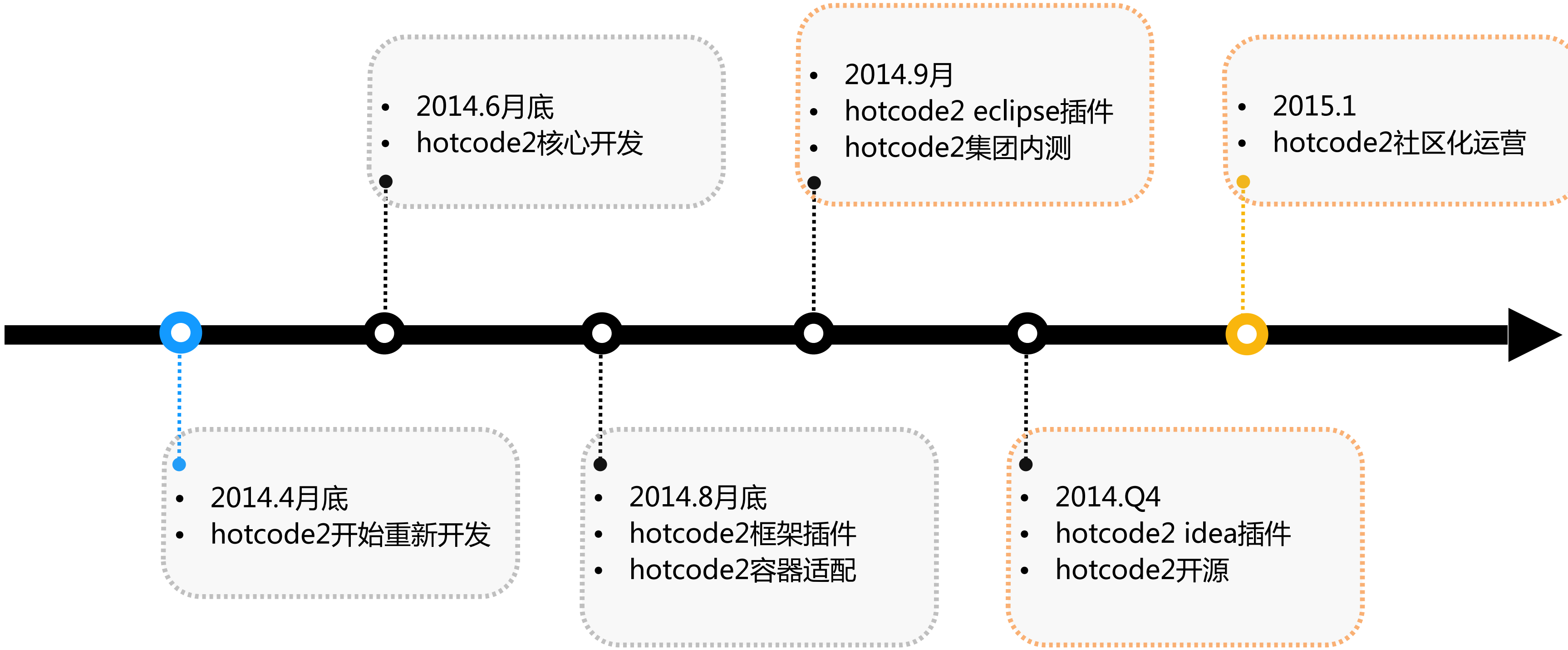
HotCode2原理与使用

HotCode2使用与规划

## 如何使用hotcode2

- 命令行直接添加agent
- 使用eclipse插件集成hsf.jetty run.jetty

# Road map



- 2014.4月底
- hotcode2开始重新开发

- 2014.6月底
- hotcode2核心开发

- 2014.8月底
- hotcode2框架插件
- hotcode2容器适配

- 2014.9月
- hotcode2 eclipse插件
- hotcode2集团内测

- 2014.Q4
- hotcode2 idea插件
- hotcode2开源

- 2015.1
- hotcode2社区化运营

## Q & A

http://hotcode.alibaba-inc.com

寻求帮助：千臂

HotCode2答疑支持群：639513230（hotcode2）