



Caching Spring context in tests: How to speed up testing process



**Ilia
Chernov**

Wrike

 @il_chernov

 @mugenor



JPoint



Wrike

Wrike – A Leader in Work Management

Our collaborative workflow management platform helps teams gain visibility, simplify planning, enable collaboration, and streamline their workflow.



**Founded
in 2006**



**9 Offices
Globally**



**20,000+
Customers
Globally**



**1100+
Employees**



**5 years in the
Fast 500**

Speaker

- Backend developer in Wrike
- Experiment with RL as a student
- Fan of video gaming
- Like learning new IT stuff



Ilia Chernov

Content

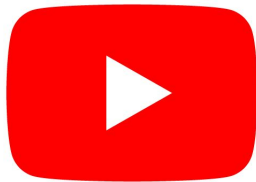
- [Idea](#)
- [Demo of IDEA plugin](#)
- [Custom HotSwapAgent Plugin](#)

Please, wait

⌵ ⚙ Instantiating tests...

Problem

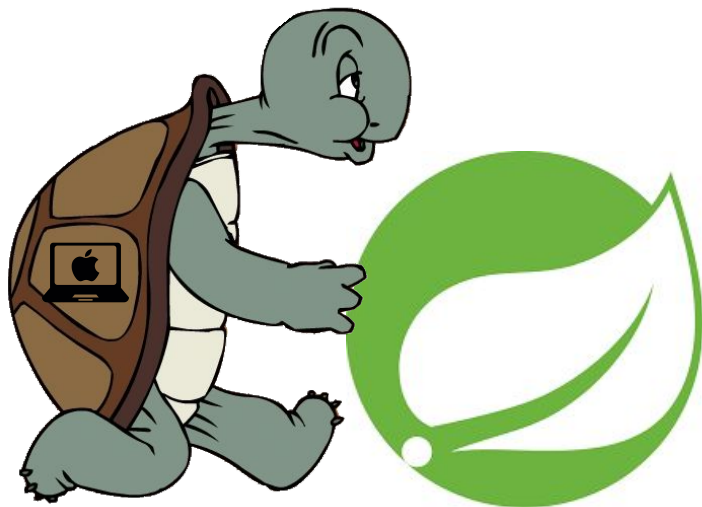
A lot of time is taken to execute test with Spring context.



Problem root

▼ ✨ Instantiating tests...

- Spring context initialization — up to several minutes
- Tests itself — milliseconds or seconds



Idea

Reuse Spring context

Don't restart JVM between every test run, so the Spring context is not lost



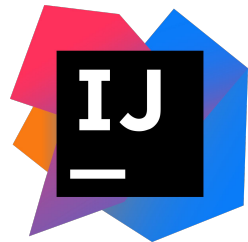
Use HotSwap for code updates

Utilise the power of DCEVM and HotSwapAgent

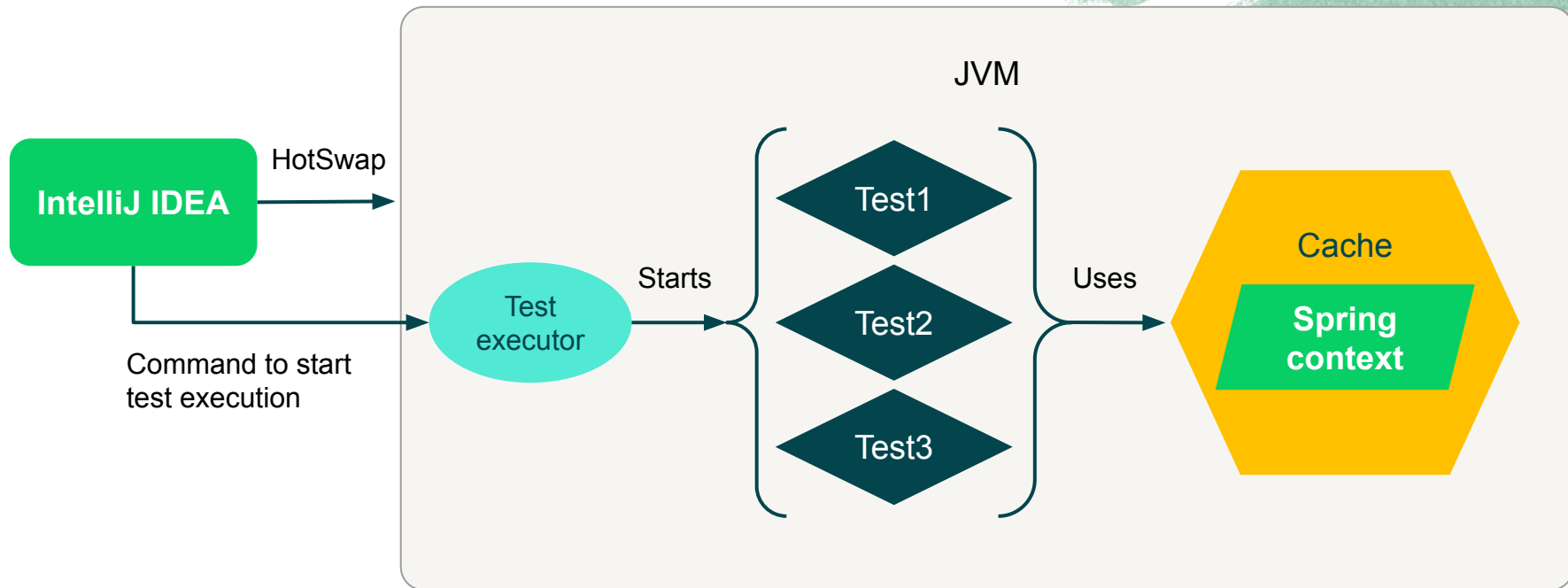


Make it easy to use

Create IntelliJ Idea plugin



Idea visualised



Idea

Reuse Spring context

Don't restart JVM between every test run, so the Spring context is not lost



Use HotSwap for code updates

Utilise the power of DCEVM and HotSwapAgent



Make it easy to use

Create IntelliJ Idea plugin



How will be the context cached?

```
/** Default static cache of Spring application contexts. */  
static final ContextCache defaultContextCache = new DefaultContextCache();
```

Spring does not close context after test execution.
Unless `@DirtiesContext` is used.

Idea

Reuse Spring context

Don't restart JVM between every test run, so the Spring context is not lost



Use HotSwap for code updates

Utilise the power of DCEVM and HotSwapAgent



Make it easy to use

Create IntelliJ Idea plugin



Why DCEVM (Dynamic Code Evolution VM)? [Source](#)

Change type	Vanilla HotSwap
Modify method body	+
Add and remove method	
Modify method definition	
Add and remove field	
Add and remove class	
Add and remove inner class	
Add and remove static field	
Add and remove annotation	
Add and remove enum value	
Hierarchy manipulation	

Why DCEVM (Dynamic Code Evolution VM)? [Source](#)

Change type	Vanilla HotSwap	DCEVM
Modify method body	+	+
Add and remove method		+
Modify method definition		+
Add and remove field		+
Add and remove class		+
Add and remove inner class		+
Add and remove static field		+
Add and remove annotation		+
Add and remove enum value		
Hierarchy manipulation		

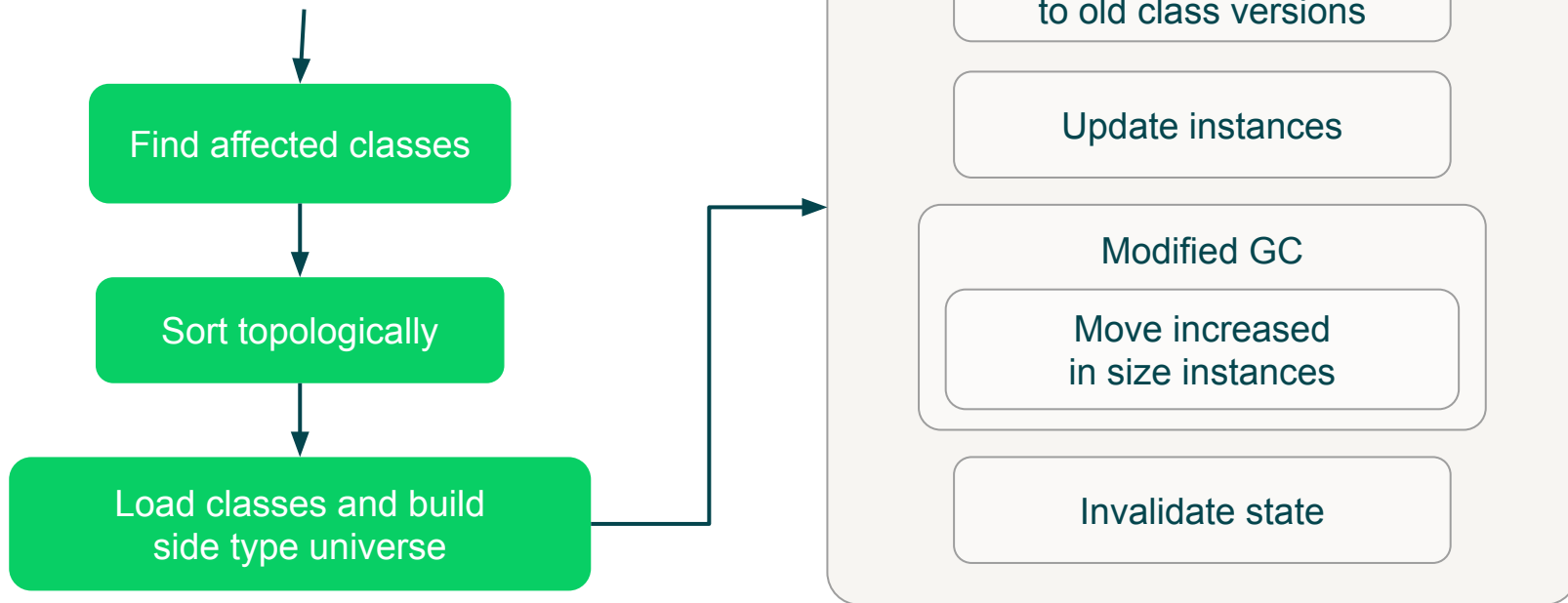
Tested with JVM
17*

What is DCEVM?

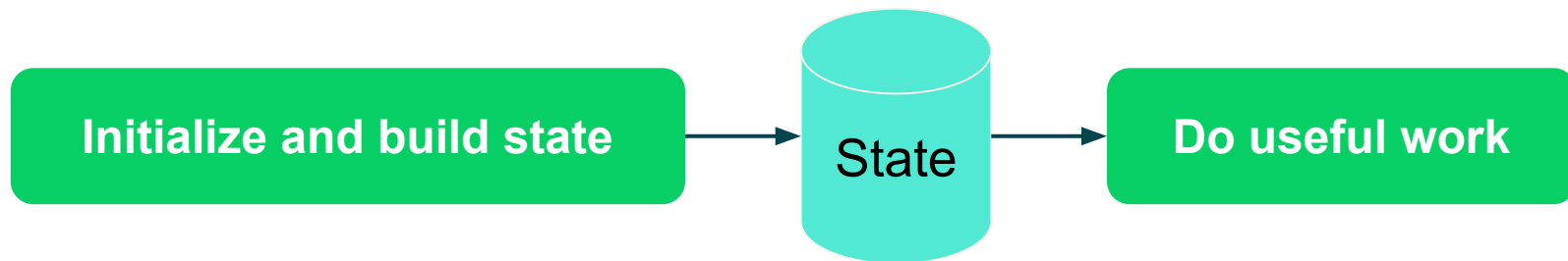
- HotSpot modification
- [PhD work](#)
- Since Java 17: Distributed with [JetBrainsRuntime](#)
- For other versions: [official instructions](#)

How does DCEVM work?

Class redefinition command



Generic framework workflow schema



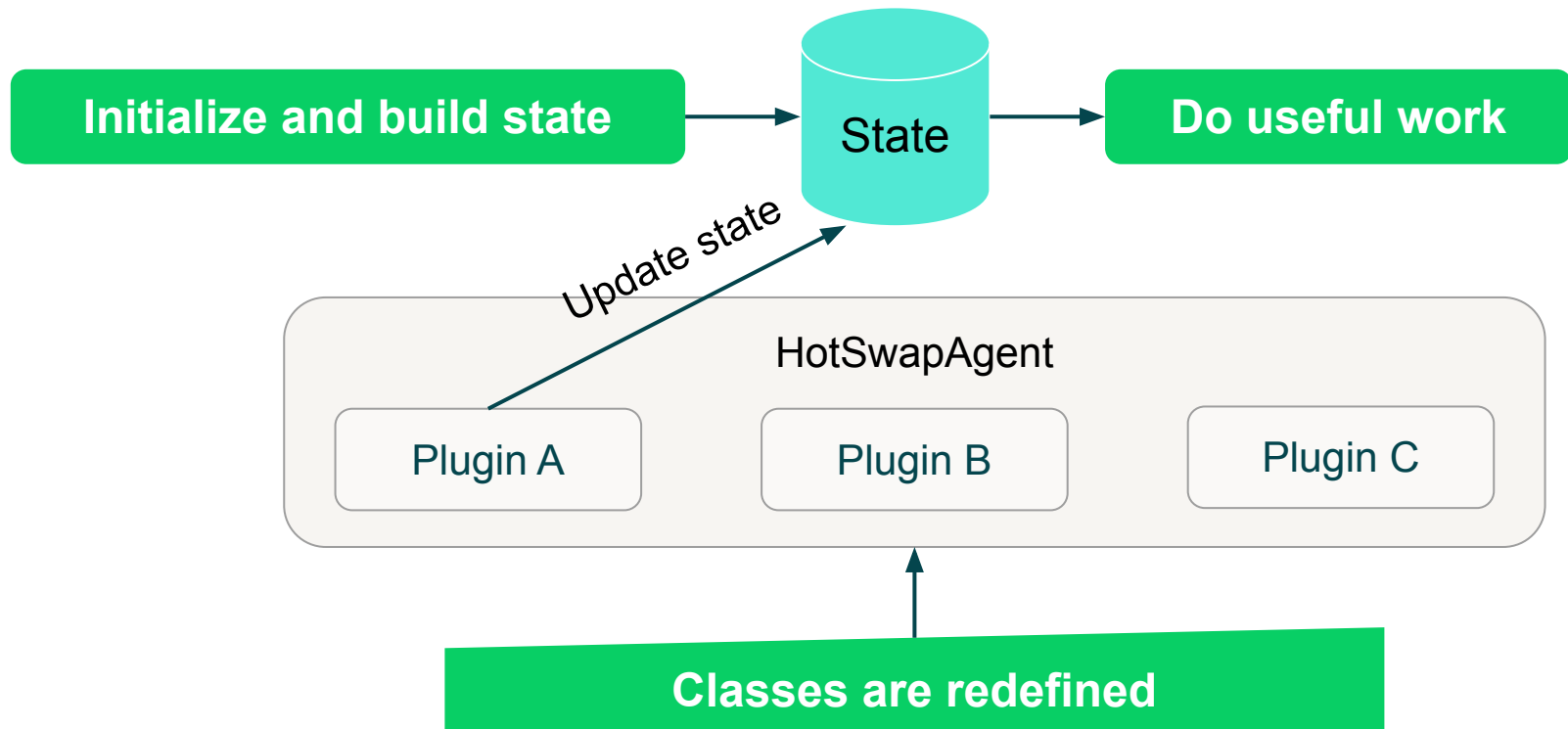
How to update framework state?

Classes are redefined

HotSwapAgent

- Java agent
- Updates application state when class or resource is updated
- Plugin system
- Built in plugins for many frameworks and instruments
- You can write plugins by yourself

HotSwapAgent



What about JRebel?

- Alternative to DCEVM+HotSwapAgent
- Costs money and not opensource
- Better support for frameworks and hotswap features

Change type	Vanilla HotSwap	DCEVM	JRebel
Modify method body	+	+	+
Add and remove method		+	+
Modify method definition		+	+
Add and remove field		+	+
Add and remove class		+	+
Add and remove inner class		+	+
Add and remove static field		+	+
Add and remove annotation		+	+
Add and remove enum value			+
Hierarchy manipulation			+

Idea

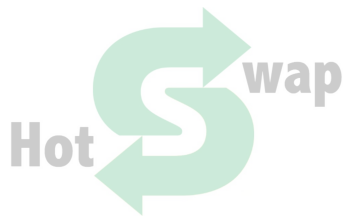
Reuse Spring context

Don't restart JVM between every test run, so the Spring context is not lost



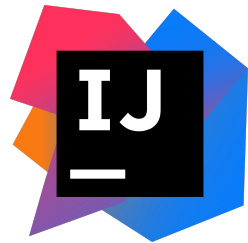
Use HotSwap for code updates

Utilise the power of DCEVM and HotSwapAgent



Make it easy to use

Create IntelliJ Idea plugin



Content

- [Idea](#)
- [Demo of IDEA plugin](#)
- [Custom HotSwapAgent Plugin](#)

Content

- [Idea](#)
- [Demo of IDEA plugin](#)
- [Custom HotSwapAgent Plugin](#)

Custom HotSwapAgent plugin for MyBatis



What is MyBatis?

- Persistence framework
- Native SQL
- Object mappings
- Other features

MyBatis. XML Configuration

```
public interface MyBatisMapper {  
    int selectInt();  
}
```

```
<mapper namespace="org.example.MyBatisMapper">  
    <select id="selectInt">  
        select 1  
    </select>  
</mapper>
```

MyBatis. Annotation Configuration

```
public interface MyBatisMapper {  
    @Select("select 1")  
    int selectInt();  
}
```



MyBatis. HotSwap problem

```
@Select("select 1")  
int selectInt();
```



```
@Select("select 2")  
int selectInt();
```

HotSwapAgent MyBatis plugin

XML



Annotation



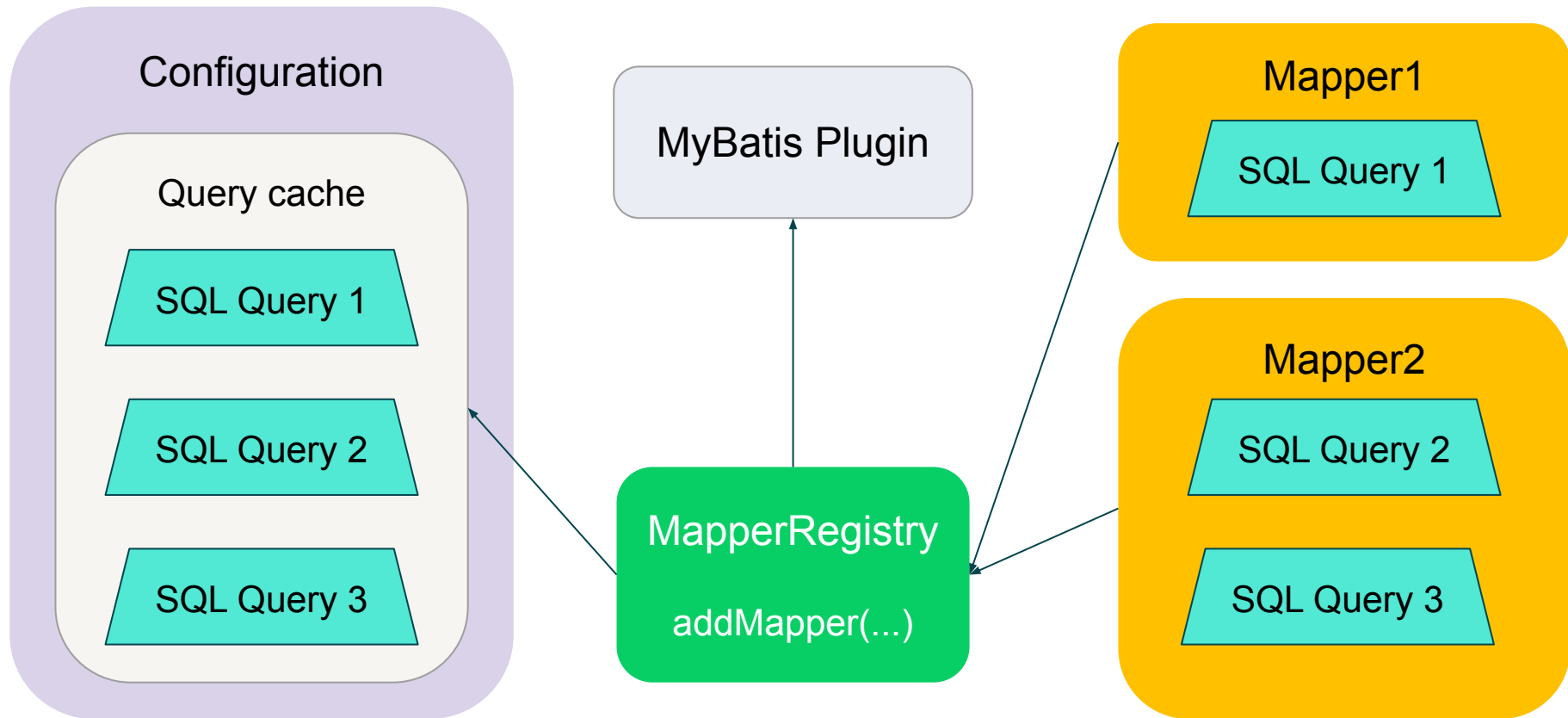
Plugin registration

```
@Plugin(  
    name = "ConfMyBatisPlugin",  
    testedVersions = {"3.5.9"}  
)  
  
public class ConfMyBatisPlugin {  
    -  
}
```

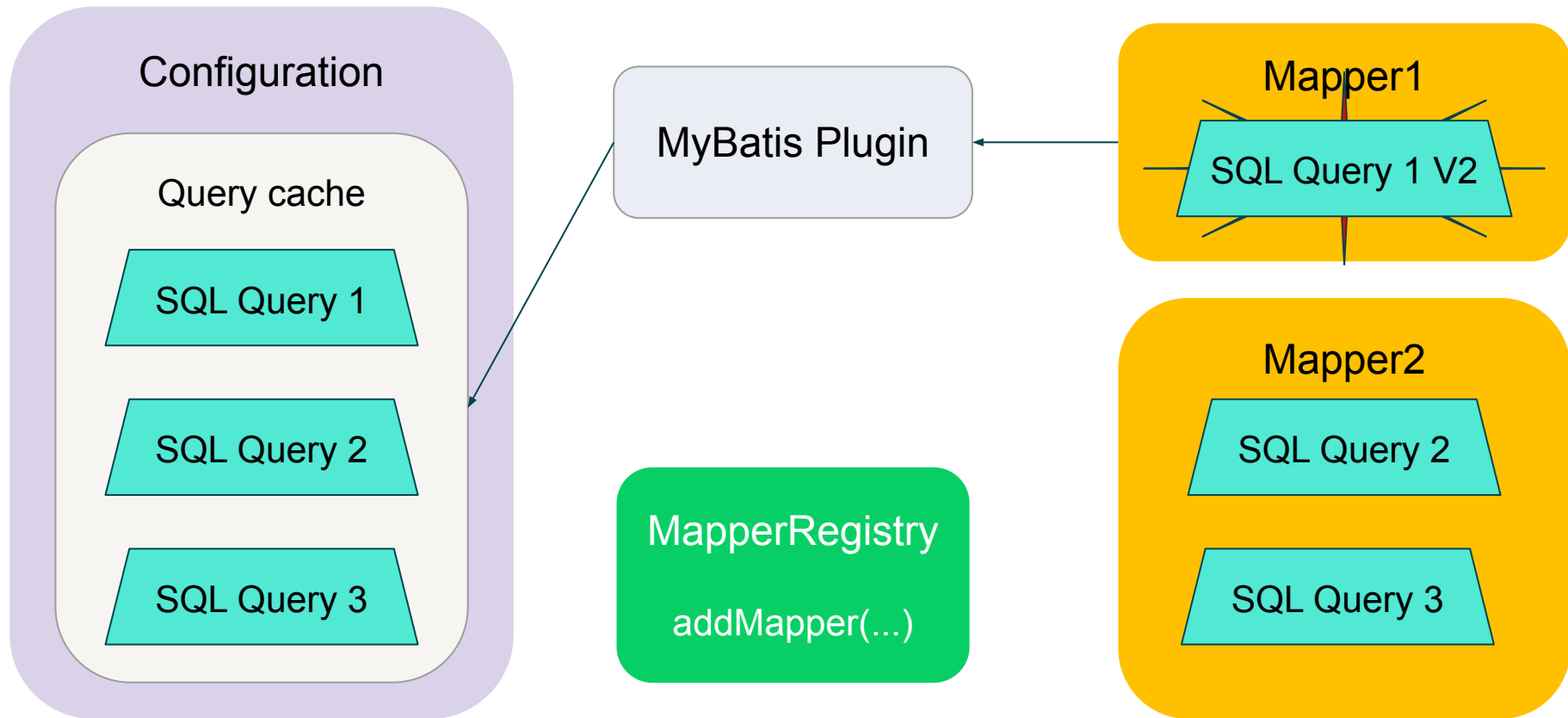
Hotswap properties:

```
pluginPackages=org.example.hotswap.plugin
```

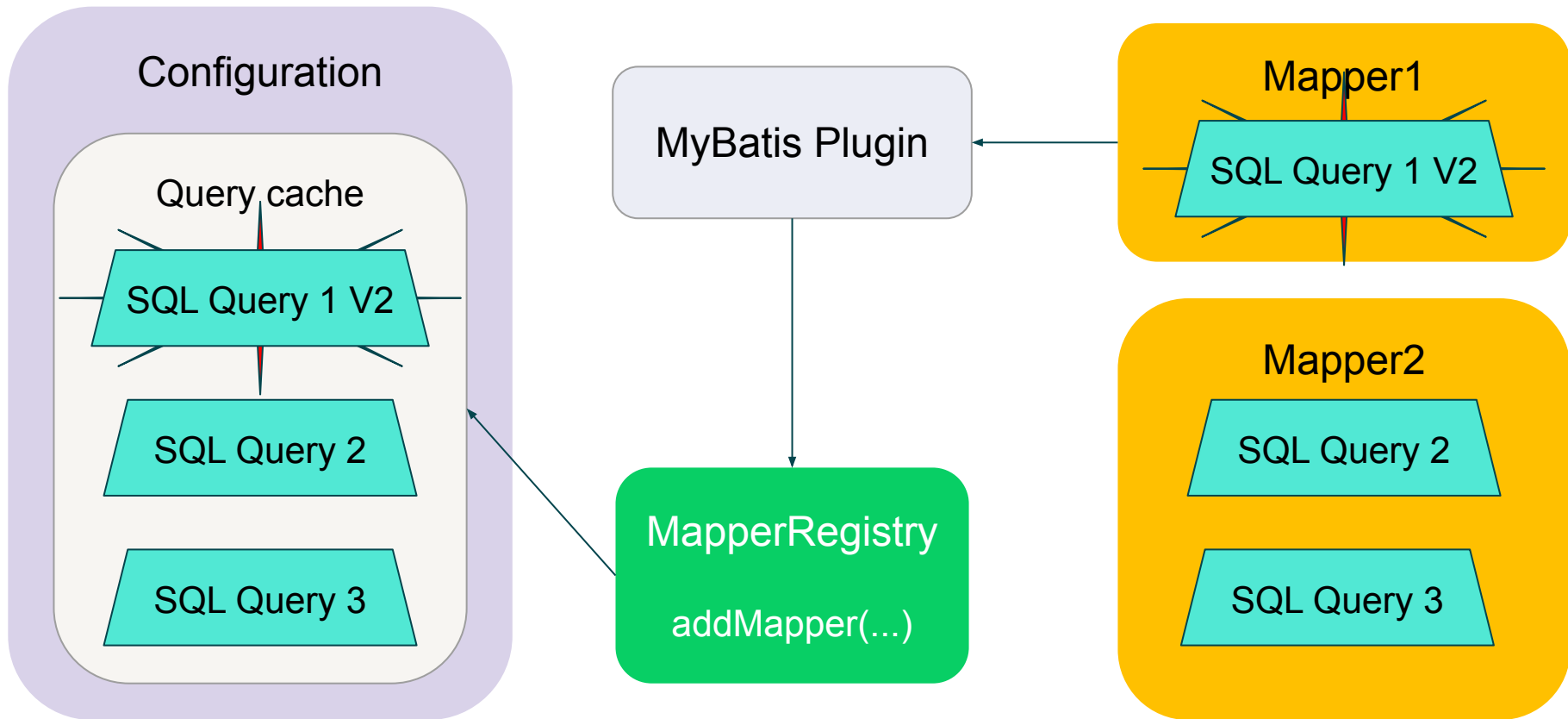
MyBatis configuration classes overview



MyBatis configuration classes overview



MyBatis configuration classes overview



MapperRegistry.addMapper

```
try {  
    knownMappers.put(k: type, v: new MapperProxyFactory<(type));  
    MapperAnnotationBuilder parser = new MapperAnnotationBuilder(config, type);  
    parser.parse();  
    loadCompleted = true;  
} finally {  
    if (!loadCompleted) {  
        knownMappers.remove(o: type);  
    }  
}
```

MapperRegistry.addMapper

```
try {  
    knownMappers.put(k: type, v: new MapperProxyFactory<(type));  
    MapperAnnotationBuilder parser = new MapperAnnotationBuilder(config, type);  
    parser.parse();  
    loadCompleted = true;  
    Call plugin method here  
} finally {  
    if (!loadCompleted) {  
        knownMappers.remove(o: type);  
    }  
}
```


MapperRegistry.addMapper

```
try {
    knownMappers.put(k: type, v: new MapperProxyFactory<(type));
    MapperAnnotationBuilder parser = new MapperAnnotationBuilder(config, type);
    parser.parse();
    loadCompleted = true;
    //region Call plugin method here

    HotSwapAgent.getPlugin(pluginClass: ConfMyBatisPlugin.class)
        .registerMapperClass(mapperClass: type, config);
    //endregion
} finally {
    if (!loadCompleted) {
        knownMappers.remove(o: type);
    }
}
```

Javaassist

- JVM bytecode manipulation library
- Source level and bytecode level APIs

Javaassist. Example

```
→ ClassPool classPool = ClassPool.getDefault();  
→ CtClass barCls = classPool.makeClass( classname: "foo.Bar");  
→ CtMethod printMethod = CtNewMethod.make(  
    src: "public void print() { System.out.println(\"Welcome to the Bar\"); }",  
    declaring: barCls  
);  
→ barCls.addMethod( m: printMethod);  
   useItSomehow(barCls.toBytecode());
```

MapperRegistry patch

```
@OnClassLoadEvent(classNameRegexp = "org.apache.ibatis.binding.MapperRegistry", events = {LoadEvent.DEFINE})  
public static void patchMapperRegistry(final CtClass mapperRegistryClass, final ClassPool classPool)
```

```
    throws NotFoundException, CannotCompileException {
```

```
    String registerPluginAndMapperClass = "{" +
```

```
        PluginManagerInvoker.buildInitializePlugin(pluginClass: ConfMyBatisPlugin.class) +
```

```
        PluginManagerInvoker.buildCallPluginMethod(pluginClass: ConfMyBatisPlugin.class, method: "registerMapperClass",
```

```
            ...paramValueAndType: "type", "java.lang.Class", "this.config", "java.lang.Object") +
```

```
    "};
```

```
CtMethod addMapperMethod = mapperRegistryClass.getDeclaredMethod(name: "addMapper",
```

```
    params: new CtClass[]{classPool.get("java.lang.Class")});
```

```
// insert initialization block right after successful add of mapper. See class source code.
```

```
addMapperMethod.insertAt(lineNum: 74, src: registerPluginAndMapperClass);
```

```
}
```

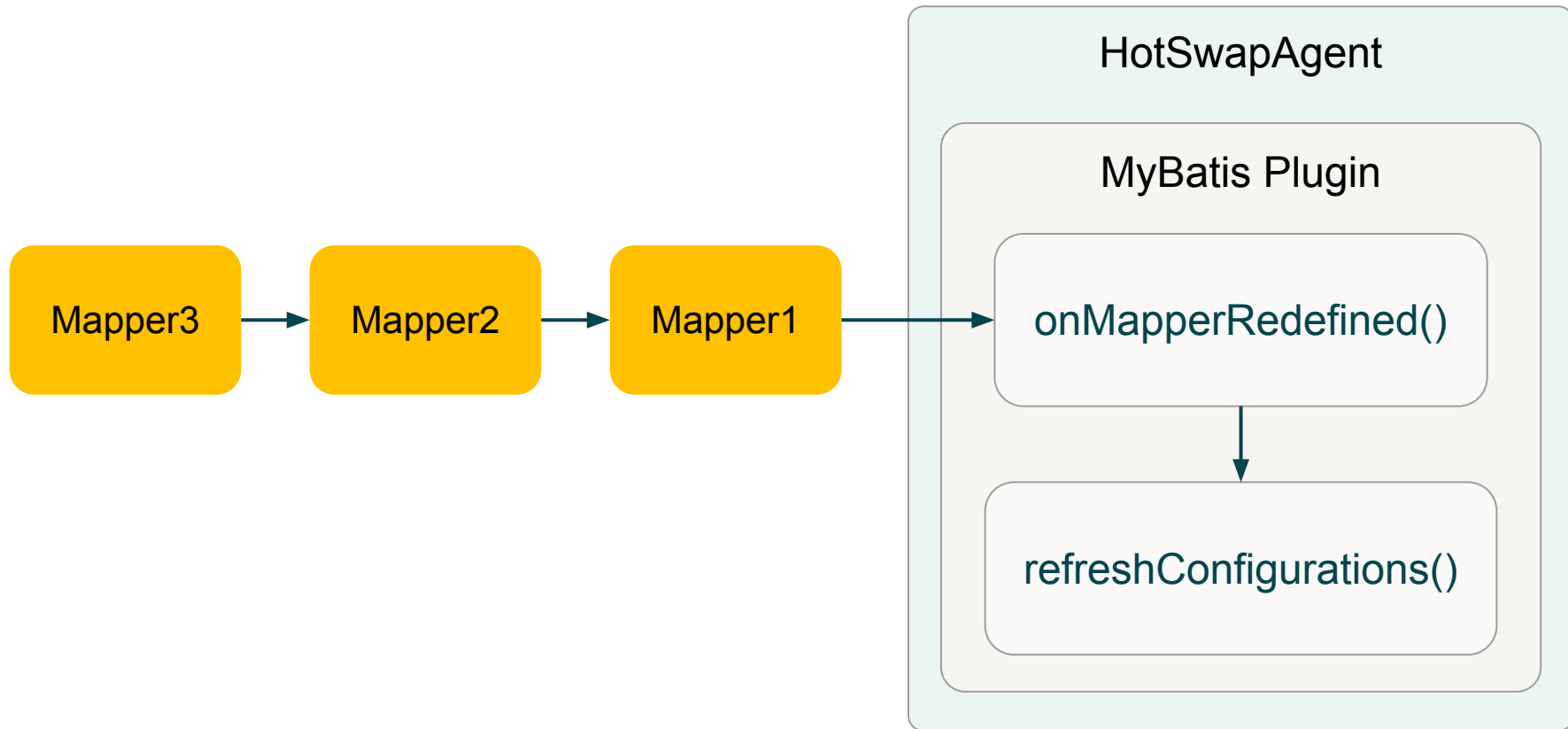
```
Map<Object, Set<Class<?>>> configurationToMapperClasses
```

```
Map<Class<?>, Set<Object>> mapperClassToConfigurations
```

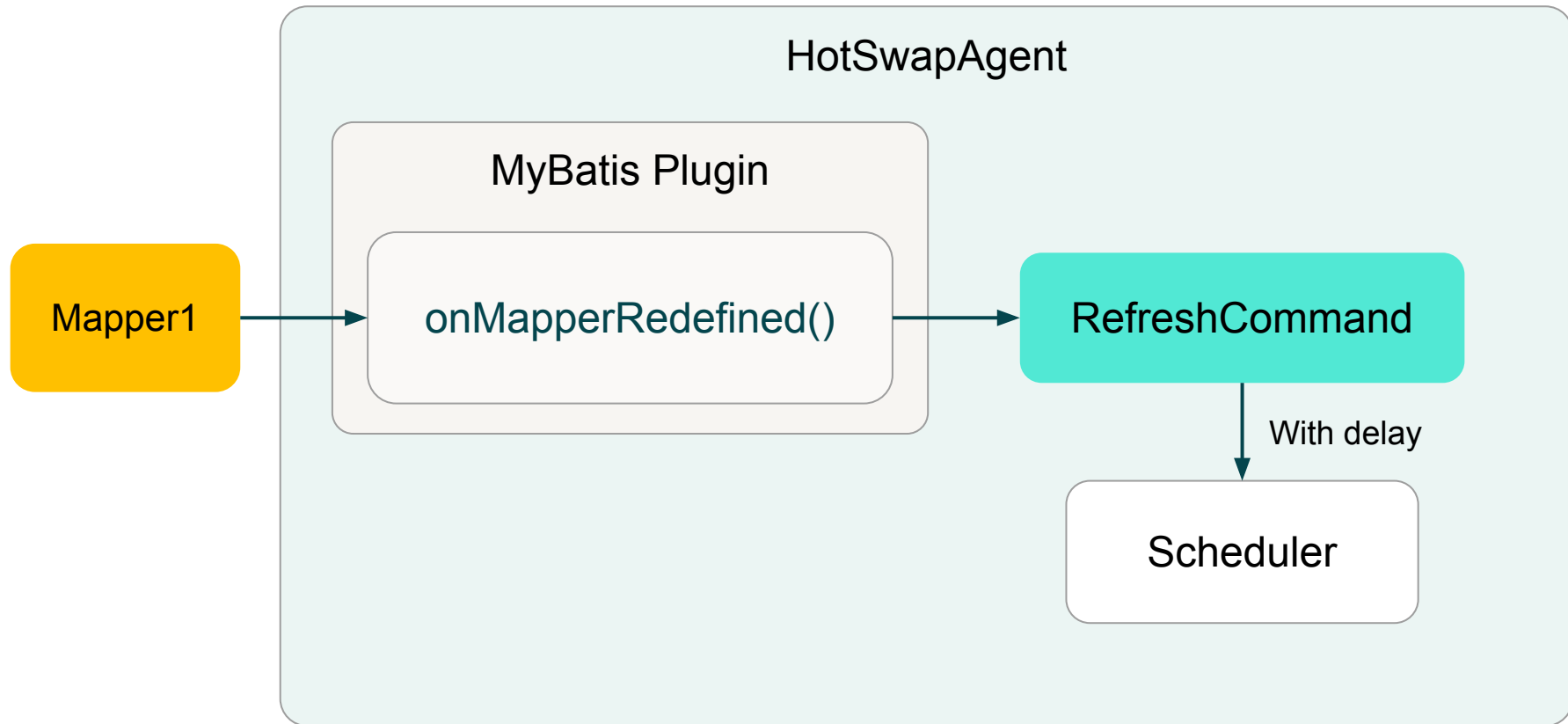
On mapper redefined

```
@OnClassLoadEvent(classNameRegexp = ".*", events = {LoadEvent.REDEFINE})  
public void onMapperInterfaceRedefined(final Class<?> mapperClass) {  
    Set<Object> configurationsToRefresh = mapperClassToConfigurations.get(mapperClass);  
    if (!CollectionUtils.isEmpty(collection: configurationsToRefresh)) {  
        refreshMyBatisConfigurations(configurationsToRefresh);  
    }  
}
```

Refresh configurations



Refresh configurations



Refresh configurations

```
Map<Object, Set<Class<? >>> configurationToRefreshToMappersMap = configurationsToRefresh.stream()
    .collect( collector: toMap( keyMapper: identity(), valueMapper: configurationToMapperClasses::get));

scheduler.scheduleCommand(
    command: new MyBatisRefreshConfigurationsCommand(configurationToRefreshToMappersMap),
    timeout: 1_000
);
```



Custom MyBatis plugin. Refresh command

```
for (Map.Entry<Object, Set<Class<?>>> configurationWithMappers : configurationToRefreshToMappersMap.entrySet()) {  
    Object configuration = configurationWithMappers.getKey();  
    Set<Class<?>> mapperClasses = configurationWithMappers.getValue();  
    resetConfiguration(configuration);  
    for (Class<?> mapperClass: mapperClasses) {  
        ReflectionHelper.invoke(  
            target: configuration,  
            clazz: configuration.getClass(),  
            methodName: "addMapper",  
            parameterTypes: new Class[]{Class.class},  
            ...args: mapperClass  
        );  
    }  
}
```

The background features several broad, diagonal brushstrokes in a teal or seafoam green color, set against a dark navy blue background. The strokes are thick and have a visible, textured edge, giving the impression of being painted with a large brush. They sweep across the frame from the top left towards the bottom right.

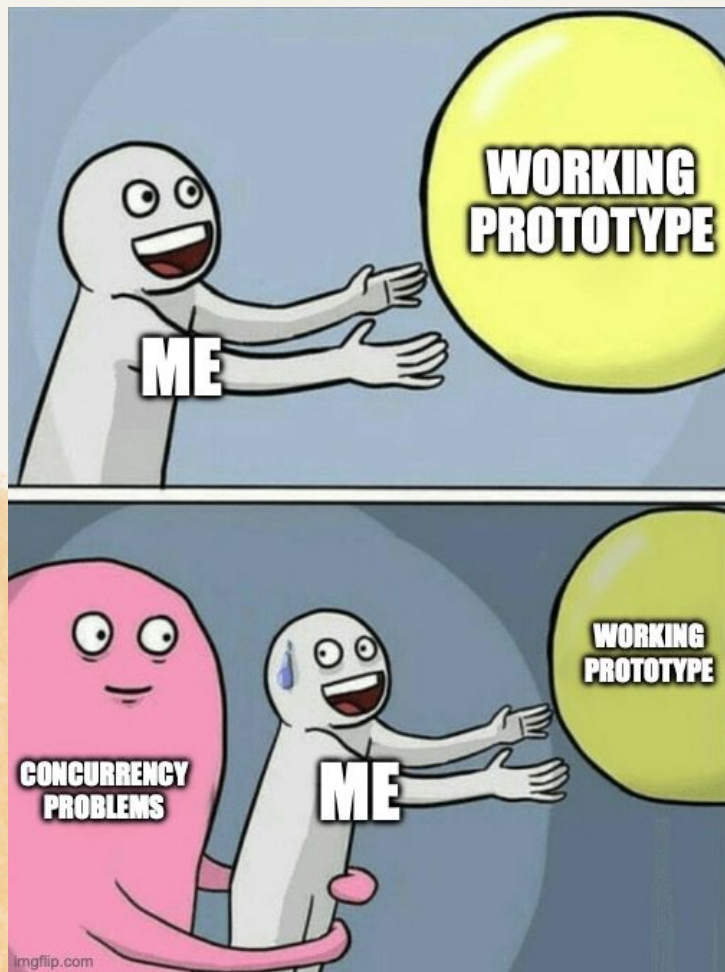
That's it? Let's check

What is wrong?

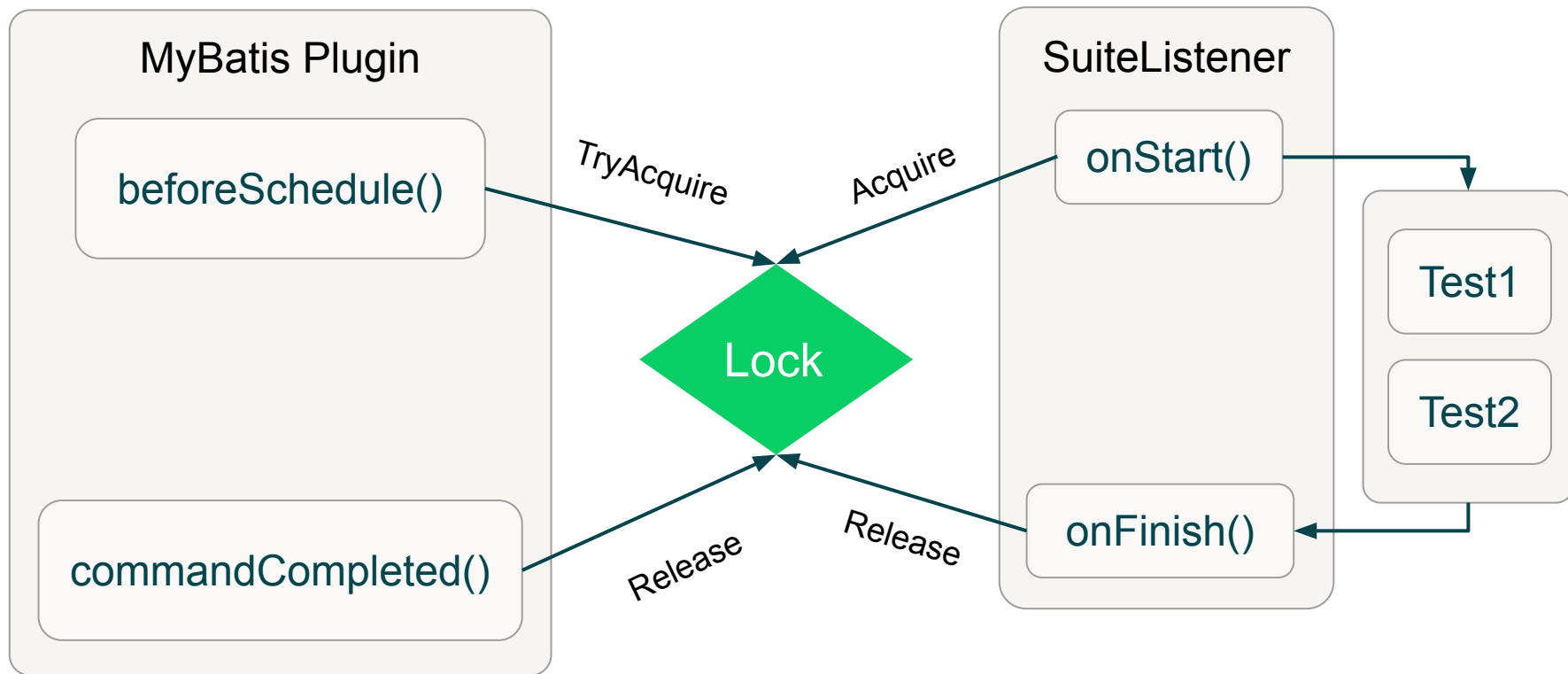
- 1) Configuration caches are not reset
- 2) Refresh configuration command is not executed
-  3) Refresh configuration command is not executed yet
- 4) Annotation value with query is not redefined

Command delay

```
Map<Object, Set<Class<? >>> configurationToRefreshToMappersMap = configurationsToRefresh.stream()
    .collect( collector: toMap( keyMapper: identity(), valueMapper: configurationToMapperClasses::get));
scheduler.scheduleCommand(
    command: new MyBatisRefreshConfigurationsCommand(configurationToRefreshToMappersMap),
    timeout: 1_000
);
```



Fix idea



SuiteListener patch

```
public class MySuiteListener implements ISuiteListener {  
    @Override  
    public void onStart(ISuite suite) {  
        System.out.println("Suite started");  
    }  
  
    @Override  
    public void onFinish(ISuite suite) {  
        System.out.println("Suite finished");  
    }  
}
```

SuiteListener patch

```
public class MySuiteListener implements ISuiteListener {  
    @Override  
    public void onStart(ISuite suite) {  
        HotSwapAgent.getPlugin(pluginClass: ConfMyBatisPlugin.class)  
            .testStartBarrierAcquire();  
    }  
  
    @Override  
    public void onFinish(ISuite suite) {  
        HotSwapAgent.getPlugin(pluginClass: ConfMyBatisPlugin.class)  
            .testStartBarrierRelease();  
    }  
}
```


Command scheduling update

```
Map<Object, Set<Class<?>>> configurationToRefreshToMappersMap = configurationsToRefresh.stream()
    .collect( collector: toMap( keyMapper: identity(), valueMapper: configurationToMapperClasses::get));
scheduler.scheduleCommand(
    command: new MyBatisRefreshConfigurationsCommand(configurationToRefreshToMappersMap),
    timeout: 1_000
);
```

Command scheduling update

```
testStartBarrier.tryAcquire();
```

```
Map<Object, Set<Class<?>>> configurationToRefreshToMappersMap = configurationsToRefresh.stream()
    .collect( collector: toMap( keyMapper: identity(), valueMapper: configurationToMapperClasses::get));
scheduler.scheduleCommand(
    command: new MyBatisRefreshConfigurationsCommand(
        configurationToRefreshToMappersMap,
        afterRun: testStartBarrier::release
    ),
    timeout: 1_000
);
```



Does it work now?

Conclusions

- Hotswap is useful in testing
- DCEVM+HotSwapAgent is a powerful combination
- You can adapt hotswap behaviour with HotSwapAgent!
- IDEA prototype plugin is working well

Join Wrike's TechClub community to grow and share ideas

Join our community of knowledge-sharers by attending our events, watching videos, reading articles, and more.



Q&A

Useful links:

- [JDK-17 with DCEVM install instructions](#)
- [HotSwap agent CMD options](#)
- [HotSwap agent properties file](#)
- [DCEVM PhD paper](#)

Wrike TechClub

www.wrike.com/techclub





Thank you!



ed.