COMMUNITY
THE ASF CONFERENCE
CODE

Zero Code Instrumentation For All-Release Apache Tomcat Observability

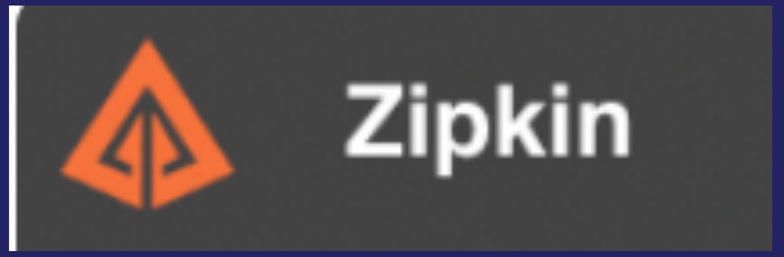
Ziming Liu
Engineer of Alibaba Cloud

COMMUNITY
THE ASF CONFERENCE
CODE

CONTENTS

Background

- Observbility is becoming increasingly important for software systems
- Data collecting is extremely important for observability, there are two main approaches for java middleware like Apache Tomcat:
  1. manual instrumentation
  2. zero code instrumentation



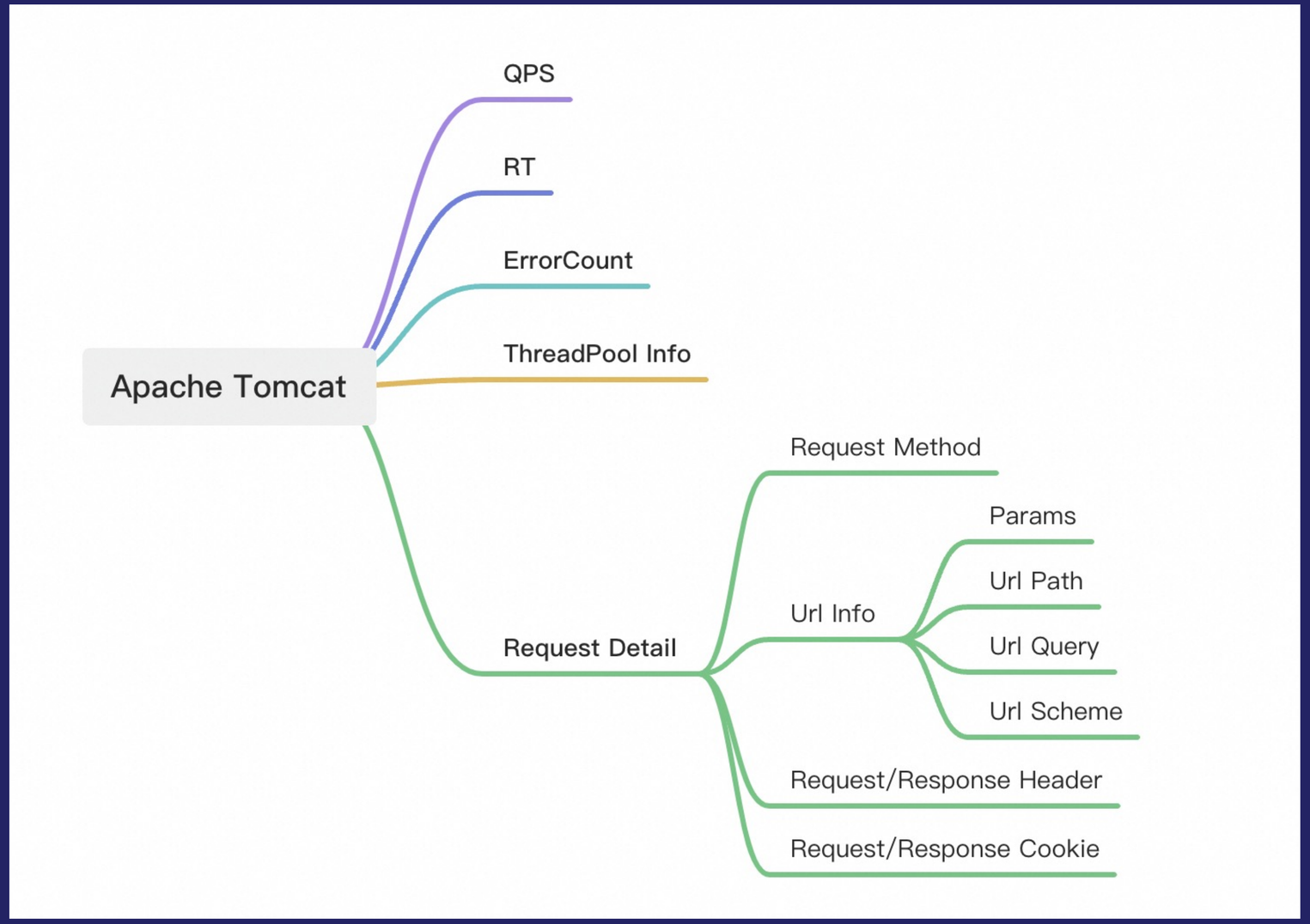
Background

- For manual instrumentation, users usually use sdk to modify the code of the middleware
  - Hard for inexperienced engineers
  - Hard to merge upstream code
- For zero code instrumentation, users usually use a javaagent to modify the code of the middleware
  - Easy for users to use
  - Hard for javaagent developer

Background

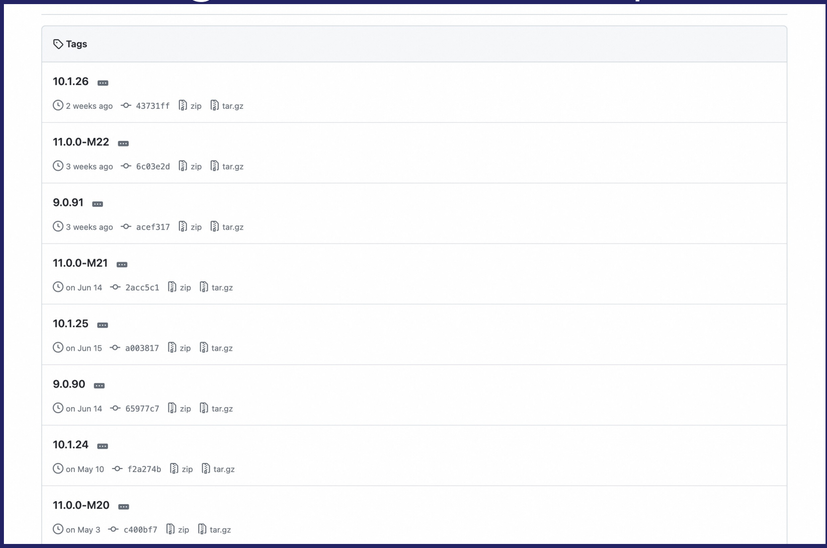
- Previously, we had independently maintained Apache Tomcat branches, where the branch maintainer would do the instrumentation in the key function call to collect observation data
- Now we provide Apache Tomcat observation services for our customers on the Alibaba Cloud, so we are gradually switching to javaagent based zero code instrumentation to reduce the cost of use for the users

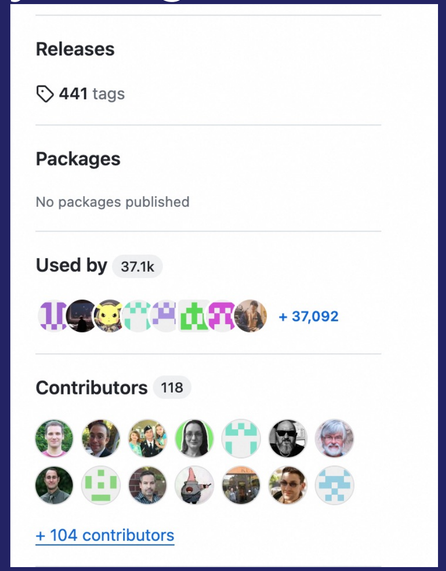
Background



Background

- Although javaagent is non-intrusive to the user's code, the corresponding instrumentation code needs to be adapted accordingly when middleware's code changes
- Apache Tomcat, as an active open-source project with a long history, has so many releases, and Apache Tomcat releases quite frequently, which brings a lot of challenges to the development of javaagent





COMMUNITY
THE ASF CONFERENCE
CODE

Guarantee Of Instrument Comprehensiveness

- There are usually 2 steps while the javaagent doing the zero-code-instrusion instrumentation:
  - Write a matcher to match the class and method that needed to be enhanced
  - Write the code which needed to be inserted around the matched method
- When the class need to be instrumented is loaded, relevant code in javaagent is woven in before and after the matching method

Guarantee Of Instrument Comprehensiveness

- How to ensure the matcher is correct?
- How to ensure the code inserted into Apache Tomcat is correct?

```java
public TomcatServerHandlerInstrumentation(
    String handlerAdviceClassName, String attachResponseAdviceClassName) {
  this.handlerAdviceClassName = handlerAdviceClassName;
  this.attachResponseAdviceClassName = attachResponseAdviceClassName;
}

@Override
public ElementMatcher<TypeDescription> typeMatcher() {
  return named("org.apache.catalina.connector.CoyoteAdapter");
}

@Override
public void transform(TypeTransformer transformer) {
  transformer.applyAdviceToMethod(
      isMethod()
          .and(isPublic())
          .and(named("service"))
          .and(takesArgument(0, named("org.apache.coyote.Request")))
          .and(takesArgument(1, named("org.apache.coyote.Response"))),
      handlerAdviceClassName);

  transformer.applyAdviceToMethod(
      isMethod()
          .and(named("postParseRequest"))
          .and(takesArgument(0, named("org.apache.coyote.Request")))
          .and(takesArgument(2, named("org.apache.coyote.Response")))
          .and(returns(boolean.class)),
      attachResponseAdviceClassName);
  }
}
```

```java
@Override
public HttpServletRequest getServletRequest(Request request) {
  Object note = request.getNote(1);

  if (note instanceof HttpServletRequest) {
    return (HttpServletRequest) note;
  } else {
    return null;
  }
}

@Override
public HttpServletResponse getServletResponse(Response response) {
  Object note = response.getNote(1);

  if (note instanceof HttpServletResponse) {
    return (HttpServletResponse) note;
  } else {
    return null;
  }
}
```

Guarantee Of Instrument Comprehensiveness

- Static Check:
  - Download all-version tomcat jars, and put them into a seperated URLClassloader
  - Use ASM Visitor to collect all the matchers and references in the inserted code
    - Collect all matchers and references in advice recursively
    - Generate matchers and references getter method
  - Match all the matchers with the seperated URLClassloader
  - Match all the references with the seperated URLClassloader

Guarantee Of Instrument Comprehensiveness

- Static Check:

```
transformer.applyAdviceToMethod(
    isMethod()
        .and(isPublic())
        .and(named("service"))
        .and(takesArgument(0, named("org.apache.coyote.Request")))
        .and(takesArgument(1, named("org.apache.coyote.Response"))),
    handlerAdviceClassName);
```

```
public void service(Request req, Response res) throws Exception {
    org.apache.catalina.connector.Request request = (org.apache.catalina.connector.Request) req.getNote( pos
    org.apache.catalina.connector.Response response = (org.apache.catalina.connector.Response) res.getNote(
    if (request == null) {
        request = this.connector.createRequest();
        request.setCoyoteRequest(req);
        response = this.connector.createResponse();
        response.setCoyoteResponse(res);
        request.setResponse(response);
        response.setRequest(request);
        req.setNote( pos: 1, request);
        res.setNote( pos: 1, response);
        req.getParameters().setQueryStringEncoding(this.connector.getURIEncoding());
    }

    if (this.connector.getXpoweredBy()) {
        response.addHeader( name: "X-Powered-By", POWERED_BY);
    }

    boolean comet = false;
    boolean async = false;
```

```
addFlag(MinimumVisibilityFlag.PUBLIC).  no usages  new *

addFlag(ManifestationFlag.NON_INTERFACE).  no usages  new *

addMethod(new Source[]{  no usages  new *
  new Source("io.opentelemetry.javaagent.instrumentation.tomcat.common.TomcatHelper", 37)
},new Flag[]{OwnershipFlag.NON_STATIC,MinimumVisibilityFlag.PROTECTED_OR_HIGHER},"setAttribute",Type.

getType("V"), new Type[]{Type.  no usages  new *

getType("Ljava/lang/String;"),Type.  no usages  new *

getType("Ljava/lang/Object;")}).  no usages  new *

addMethod(new Source[]{  no usages  new *
  new Source(
      "io.opentelemetry.javaagent.instrumentation.tomcat.v8_0_15.Tomcat8HttpAttributesGetter", 28)
},new Flag[]{OwnershipFlag.NON_STATIC,MinimumVisibilityFlag.PROTECTED_OR_HIGHER},"method",Type.

getType("Lorg/apache/tomcat/util/buf/MessageBytes;"), new Type[0]).  no usages  new *

addMethod(new Source[]{  no usages  new *
  new Source(
      "io.opentelemetry.javaagent.instrumentation.tomcat.v8_0_15.Tomcat8HttpAttributesGetter", 34)
},new Flag[]{OwnershipFlag.NON_STATIC,MinimumVisibilityFlag.PROTECTED_OR_HIGHER},"scheme",Type.

getType("Lorg/apache/tomcat/util/buf/MessageBytes;"), new Type[0]).  no usages  new *
```

```
import org.objectweb.asm.Type;var1.put("org.apache.coyote.Request",  new *   542  81
    ClassRef.builder("org.apache.coyote.Request").

addSource("io.opentelemetry.javaagent.instrumentation.tomcat.v8_0_15.Tomcat8AttachResponseAdvice",22).  no us

addSource("io.opentelemetry.javaagent.instrumentation.tomcat.common.TomcatHelper",37).  no usages  new *

addSource("io.opentelemetry.javaagent.instrumentation.tomcat.common.TomcatHelper",52).  no usages  new *

addSource("io.opentelemetry.javaagent.instrumentation.tomcat.common.TomcatHelper",58).  no usages  new *

addSource("io.opentelemetry.javaagent.instrumentation.tomcat.common.TomcatHelper",63).  no usages  new *
```

COMMUNITY
THE ASF CONFERENCE
CODE

Guarantee Of Instrument Comprehensiveness

- Runtime Check:
- At runtime we do a similar check to the static check, and if we detect a mismatch between the code to be inserted and the runtime version of Apache Tomcat, we will not instrument the code, which will avoid `NoSuchMethodException` and other exception.

Guarantee Of Instrument Comprehensiveness

- Latest Depth Check
- Latest Depth Check will pull the latest Apache Tomcat package and then run tests on it. If the APIs of Apache Tomcat has changed(For example: javax.servlet => jajakarta.servlet), we should redesign the instrumentation to fit the latest release

Comprehensive Observation Of Apache Tomcat

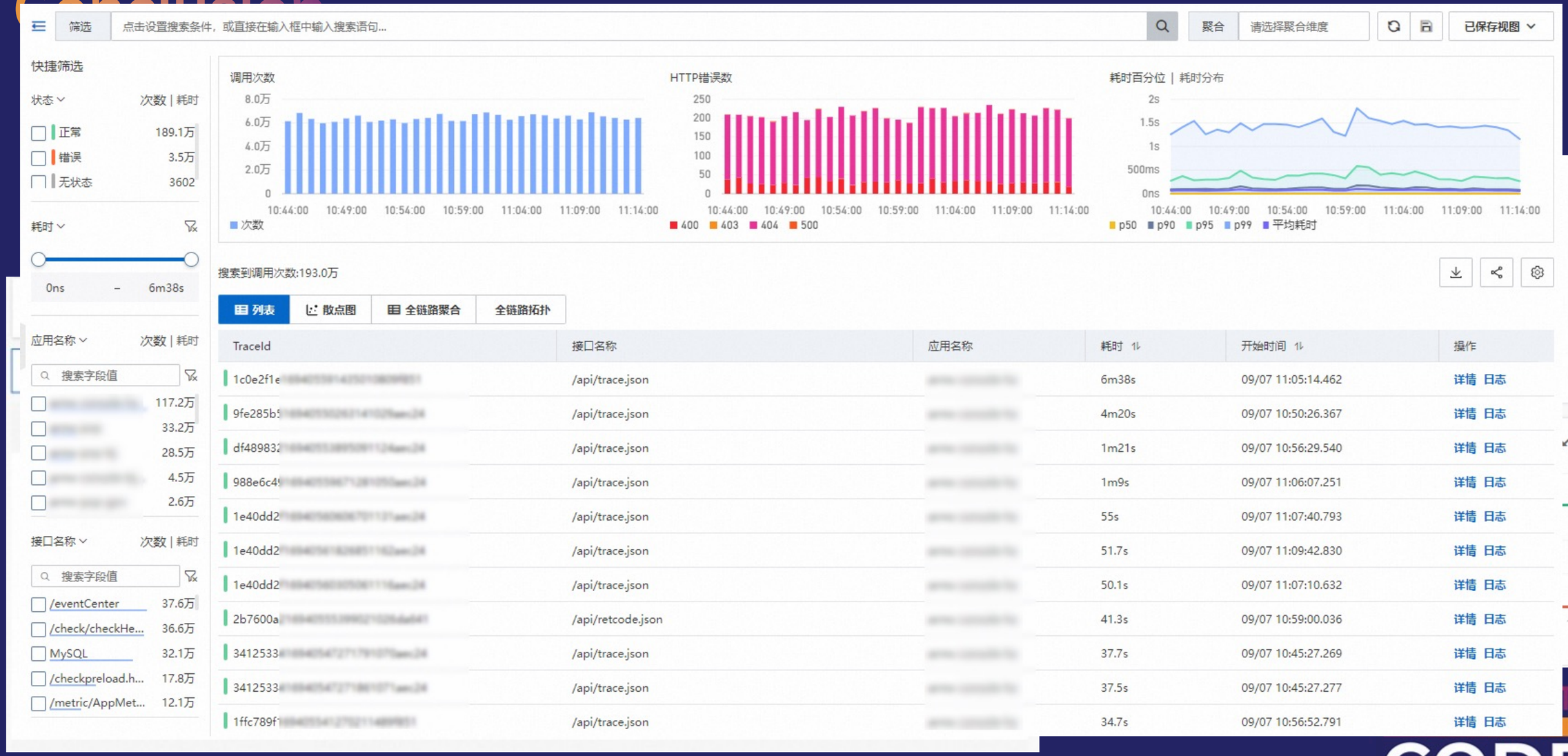
Tomcat Version	Instrument Method	Meaning
7-8.0.15	org.apache.cataline.connector.CoyoteAdapter.service org.apache.cataline.connector.CoyoteAdapter.postParseRequest	**Collect Span / Metrics**
	org.apache.tomcat.util.threads.ThreadPoolExecutor (praent is java.util.concurrent.ThreadPoolExecutor)	**Collect ThreadPool Information**
8.0.15-9.0	org.apache.cataline.connector.CoyoteAdapter.service org.apache.cataline.connector.CoyoteAdapter.postParseRequest (with org.apache.tomcat.util.http.ServerCookies)	**Collect Span / Metrics**
	org.apache.tomcat.util.threads.ThreadPoolExecutor (praent is java.util.concurrent.ThreadPoolExecutor)	**Collect ThreadPool Information**

Comprehensive Observation Of Apache Tomcat

Tomcat Version	Instrument Method	Meaning
9.0-10.0	org.apache.cataline.connector.CoyoteAdapter.service org.apache.cataline.connector.CoyoteAdapter.postParseRequest	**Collect Span / Metrics**
	org.apache.tomcat.util.threads.ThreadPoolExecutor (praent is not java.util.concurrent.ThreadPoolExecutor)	**Collect ThreadPool Information**
10.0-now	org.apache.cataline.connector.CoyoteAdapter.service org.apache.cataline.connector.CoyoteAdapter.postParseRequest (With jarkata.servlet.ReadListener)	**Collect Span / Metrics**
	org.apache.tomcat.util.threads.ThreadPoolExecutor (praent is not java.util.concurrent.ThreadPoolExecutor)	**Collect ThreadPool Information**

COMMUNITY
THE ASF CONFERENCE
CODE

Conclusion



CODE

COMMUNITY
THE ASF CONFERENCE
CODE

Thanks

ZimingLiu
liuziming.lzm@alibaba-inc.com