SPRING SUMMIT 2017

开放共享 原生共融

北京 | 2017年8月26日

# Spring Boot at AliExpress

许晓斌

阿里巴巴高级技术专家

AliExpress
Smarter Shopping, Better Living!

# About Me

- Juven has 10 years' experience on software development.
- He's currently leading a team in AliExpress, focusing on improve technical productivity and stability using methods like Microservices and DevOps.
- He authored a book about Apache Maven.


AliExpress™
Smarter Shopping, Better Living!

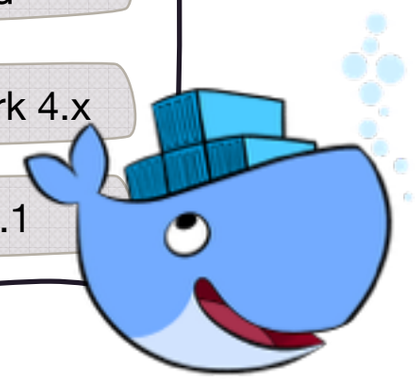
- A subsidiary of Alibaba Group.
- The largest cross border trading platform in the world.
- In 2016 **11.11** shopping festival, from **230** countries, over **6 millions** consumers placed **35.78 millions** of orders.

## 2013

- svn
- maven 2
- java 6
- JBoss 4
- war
- SpringFramework 2.x
- Servlet API 2.x

## *NOW*

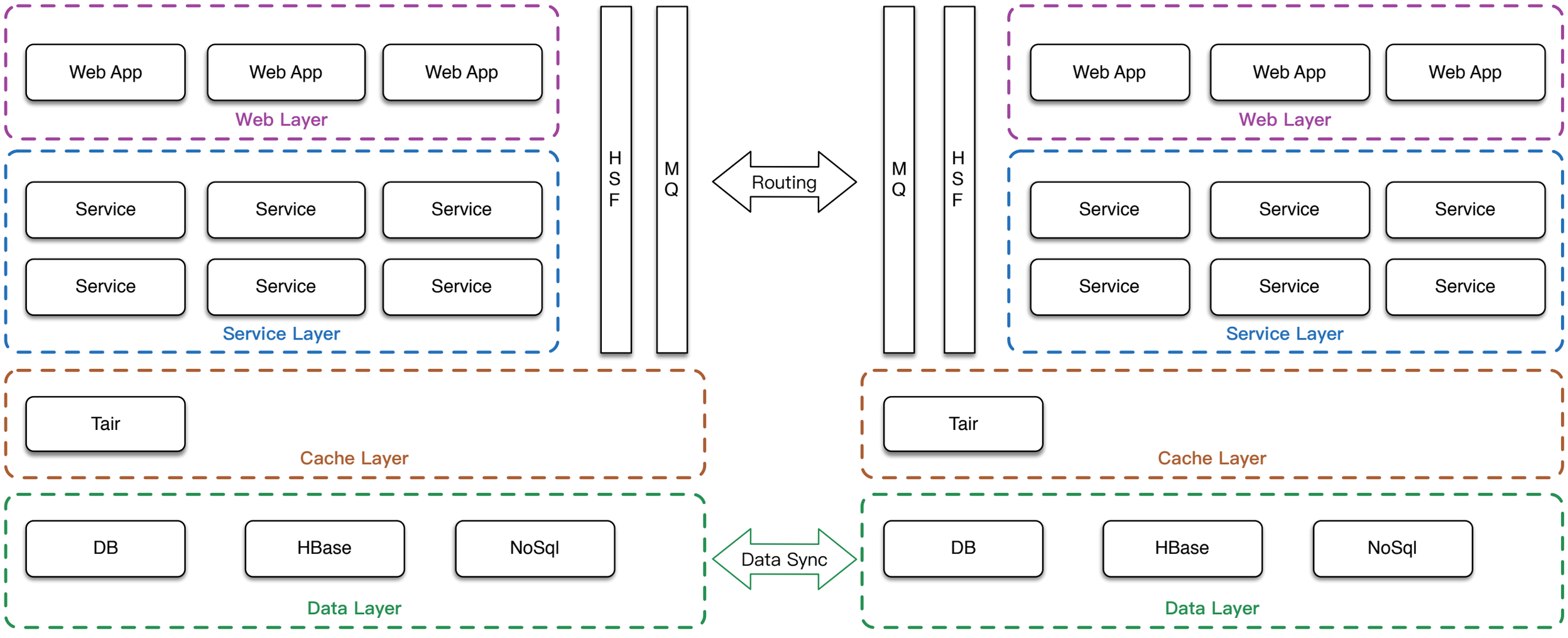
- Git & Gitlab
- Maven 3
- Java 8
- Embedded Tomcat 8
- jar
- Spring Boot &
  Spring Cloud
- SpringFramework 4.x
- Servlet API 3.1



# The Big Picture

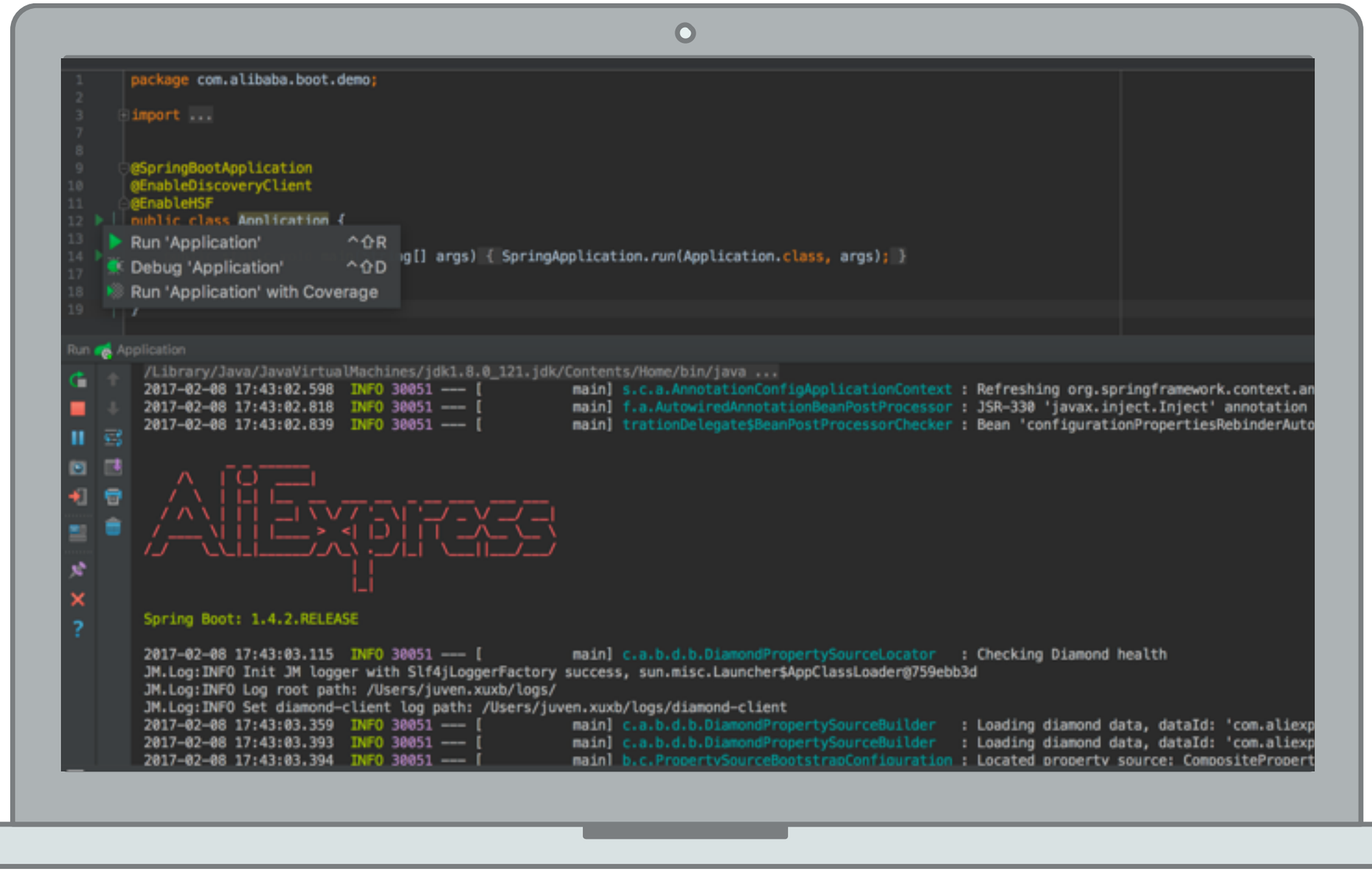
- AliExpress has hundreds of applications most of which are Java applications.
- Mainly communicated in sync way using HSF (similar to gRPC)
- Some communicated in async way using MetaQ (similar to Kafka)
- Multiple data centers.
- All apps use the same release system and monitoring system.

# The Big Picture



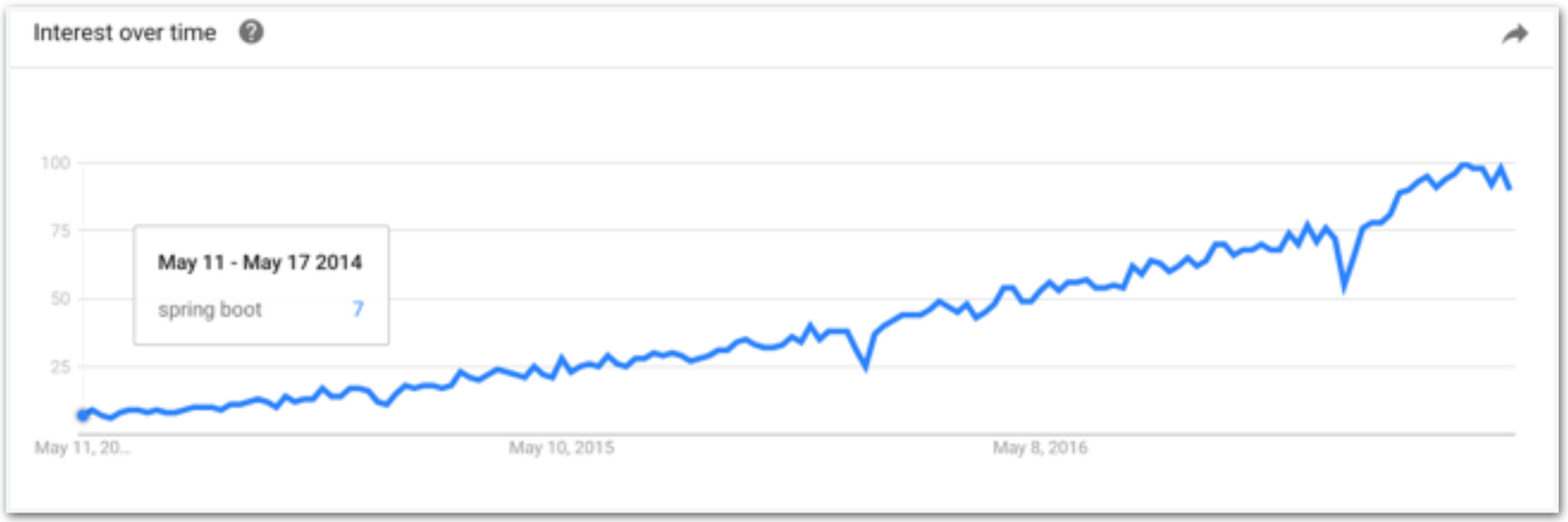
AliExpress™
Smarter Shopping, Better Living!

# Why SpringBoot?



# Why SpringBoot?

- It improves our speed of development.
- It has mature community.
- It’s a cloud native framework. (see The Twelve-Factor App)



AliExpress™
Smarter Shopping, Better Living!

# Simple but Powerful Spring Boot techniques

# BOM

```
<properties>
    <alibaba-spring-boot.version>1.4.2.4</alibaba-spring-boot.version>
    <spring-boot.version>1.4.5.RELEASE</spring-boot.version>
    <spring-cloud.version>Camden.SR5</spring-cloud.version>
</properties>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.alibaba.boot</groupId>
            <artifactId>alibaba-spring-boot-dependencies</artifactId>
            <version>${alibaba-spring-boot.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-dependencies</artifactId>
            <version>${spring-boot.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>${spring-cloud.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
</dependencyManagement>
```

- Much easier to use Spring Boot
- The order matters

AliExpress
Smarter Shopping, Better Living!

# Health Indicator


```
{
  "description": "Spring Cloud Eureka Discovery Client",
  "status": "UP",
  "discoveryComposite": { … }, // 4 items
  "diamond": {
      "status": "UP",
      "dataId: 'com.aliexpress:application.properties'
      "dataIds": [ … ] // 2 items
  },
  "eagleEye": {
      "status": "UP",
      "logfile":
      "timestamp"
  },
  "hsf": {
      "status": "UP"
  },
  "region": {
      "status": "UP",
      "hsf extension": "loaded",
      "ArtRouteReadService": "ok"
  },
  "tair": {
      "status": "UP",
      "exchangerate": { … } // 2 items
  },
  "diskSpace": { … }, // 4 items
  "refreshScope": {
      "status": "UP"
  },
  "hystrix": { … } // 1 item
}
```


- Its critical for register/unregister to load balancer

AliExpress
Smarter Shopping, Better Living!

# Health Indicator

```
public class DiamondHealthIndicator extends AbstractHealthIndicator {
  //...

  protected void doHealthCheck(Health.Builder builder) throws Exception {
    dataIds.forEach(dataId -> {
          try {
              String config = Diamond.getConfig(dataId, diamondProperties.getGroup(),
diamondProperties.getTimeOut());
              if (StringUtils.isEmpty(config)) {
                  builder.down().withDetail(String.format("dataId: '%s', group: '%s'", dataId,
diamondProperties.getGroup()), "config is empty");
              }
          } catch (Exception e) {
              builder.down().withDetail(String.format("dataId: '%s', group: '%s'", dataId,
diamondProperties.getGroup()), e.getMessage());
          }
      }
    );
    builder.up().withDetail("dataIds", dataIds);
  }
}
```

AliExpress
Smarter Shopping, Better Living!

# Failure Analyzer

```
***************************
APPLICATION FAILED TO START
***************************

Description:

The Tomcat connector configured to listen on port 8196 failed to start. The port may

Action:

Verify the connector's configuration, identify and stop any process that's listening
```

- Fail fast and fail clearly

# Failure Analyzer

```
public class DiamondConnectionFailureAnalyzer extends
AbstractFailureAnalyzer<DiamondConnectionFailureException> {

    @Override
    protected FailureAnalysis analyze(Throwable rootFailure, DiamondConnectionFailureException cause) {
        return new FailureAnalysis("Application failed to connect to Diamond, unable to access http://" +
cause.getDomain() + ":" + cause.getPort() + "/diamond-server/diamond",
                "Make sure you are in Alibaba Intranet, your VPN/Proxy Server(代理服务器) and /etc/hosts are
configured correctly.", cause);
    }
}
```

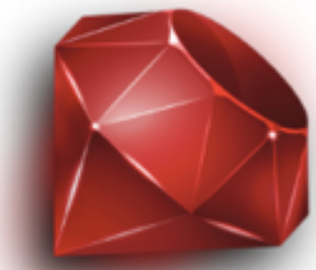
# SpringApplicationEvent

```
public class AliEnvironmentApplicationListener implements ApplicationListener<ApplicationStartedEvent> {

    private boolean initialized = false;

    private AliEnvironmentDetector aliEnvironmentDetector = new AliEnvironmentDetector();

    private DiamondPropertySourceBuilder diamondPropertySourceBuilder = new
DiamondPropertySourceBuilder();

    private static final String ALI_SPRING_BOOT_SUPER_PROPERTIES_ID = "alibaba-spring-
boot:application.properties";

    @Override
    public void onApplicationEvent(ApplicationStartedEvent event) {
        if (initialized) {
            return;
        }

        diamondPropertySourceBuilder.build(ALI_SPRING_BOOT_SUPER_PROPERTIES_ID, "DEFAULT_GROUP",
false).ifPresent(
                propertySource -> {
                    aliEnvironmentDetector.detect(propertySource, System.getenv()).forEach(env -> {
                                event.getSpringApplication().setAdditionalProfiles(env.getName());
                            }
                    );
                    initialized = true;
                }
        );
    }
}
```

# PropertySourceLocator

**spring-cloud-starter-config**

```
@Value("${archimedes.masterEnabled}")
private boolean masterEnabled;

@Value("${archimedes.messagingProducerDestination}")
private String messagingProducerDestination;
```



**diamond-spring-boot-starter**

```
spring.application.name=archimedes-master
spring.application.group=com.aliexpress.archimedes
```

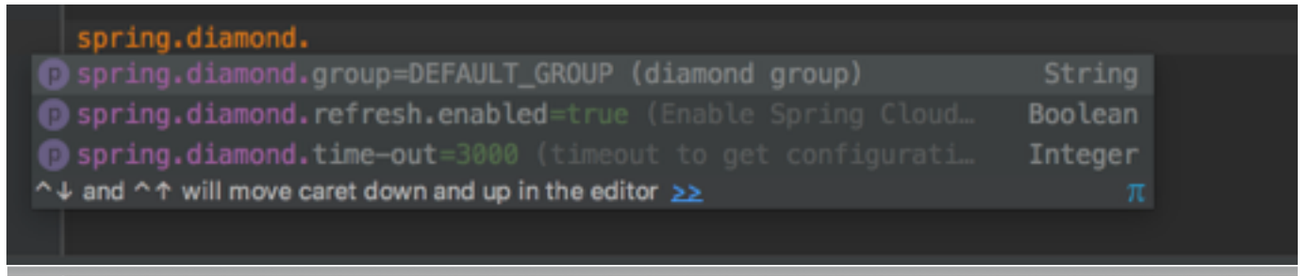
AliExpress
Smarter Shopping, Better Living!

# PropertySourceLocator

```
public class DiamondPropertySourceLocator implements PropertySourceLocator {

    private static final String DIAMOND_PROPERTY_SOURCE_NAME = "diamond";

    private DiamondPropertySourceBuilder diamondPropertySourceBuilder = new DiamondPropertySourceBuilder();

    @Override
    public PropertySource<?> locate(Environment environment) {
        String applicationName = environment.getProperty("spring.application.name");
        String applicationGroup = environment.getProperty("spring.application.group");

        if (StringUtils.isEmpty(applicationName)) {
            throw new IllegalStateException("'spring.application.name' must be configured.");
        }
        if (StringUtils.isEmpty(applicationGroup)) {
            throw new IllegalStateException("'spring.application.group' must be configured.");
        }

        CompositePropertySource compositePropertySource = new
CompositePropertySource(DIAMOND_PROPERTY_SOURCE_NAME);

        loadGroupConfigurationRecursively(compositePropertySource, applicationGroup);

        loadApplicationConfiguration(compositePropertySource, environment, applicationGroup, applicationName);

        return compositePropertySource;
    }
}
```

# Configuration - Global

```java
@ConfigurationProperties(prefix = "spring.diamond")
public class DiamondProperties {
    /**
     * diamond group
     */
    private String group = "DEFAULT_GROUP";
    /**
     * timeout to get configuration
     */
    private int timeOut = 3000;

    public String getGroup() {
        return group;
    }

    public void setGroup(String group) {
        this.group = group;
    }

    public int getTimeOut() {
        return timeOut;
    }

    public void setTimeOut(int timeOut) {
        this.timeOut = timeOut;
    }
}
```



# Configuration - Bean Specific

```java
@HSFProvider(serviceInterface = ServiceForSameServiceDifferentVersion.class, serviceVersion = "1.0.1")
class ServiceForSameServiceDifferentVersionImplB implements ServiceForSameServiceDifferentVersion {
    @Override
    public String sayHello(String name) { return "1.0.1"; }
}
```

```java
List<HSFApiProviderBean> hsfApiProviderBeans = applicationContext.getBeansWithAnnotation(HSFProvider.class)
    .entrySet().stream()
    .map(entry -> buildHsfProviderDefinition(entry.getKey(), entry.getValue()))
    .filter(HsfProviderDefinition::isEnabled)
    .map(this::buildHsfProviderBean)
    .collect(toList());
```

# logback-spring.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <include resource="com/aliexpress/boot/logging/logback/base.xml"/>
    <springProfile name="!dev">
        <root level="INFO">
            <appender-ref ref="ALIMONITOR"/>
            <appender-ref ref="APPLICATION"/>
        </root>
    </springProfile>
    <springProfile name="dev">
        <root level="INFO">
            <appender-ref ref="CONSOLE"/>
            <appender-ref ref="APPLICATION"/>
        </root>
    </springProfile>
</configuration>
```

# logback-spring.xml

```
<included>
    <include resource="org/springframework/boot/logging/logback/defaults.xml"/>

    <springProperty name="logging.file" scope="context" source="logging.file"/>
    <springProperty name="logging.path" scope="context" source="logging.path"/>
    <springProperty name="logging.pattern.file" scope="context" source="logging.pattern.file"/>
    <springProperty name="logging.pattern.console" scope="context" source="logging.pattern.console"/>
    <springProperty name="spring.application.name" scope="context" source="spring.application.name"/>

    <property name="LOG_PATH" value="${logging.path:-${user.home}/${spring.application.name}/logs}"/>
    <property name="LOG_FILE" value="${logging.file:-${LOG_PATH}/application.log}"/>

    <appender name="ALIMONITOR" class="com.alibaba.alimonitor.jmonitor.plugin.logback.JMonitorLogbackAppender"/>

    <appender name="APPLICATION" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${LOG_FILE}</file>
        <encoder>…</encoder>
        <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">…</rollingPolicy>
    </appender>

    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">..</appender>

</included>
```

# endpoints

archimedes.alibaba-inc.com/ endpoints/


```
{
  "endpoints": [
    {
      "name": "alimonitor"
    },
    {
      "name": "diamond"
    },
    {
      "name": "docker"
    },
    {
      "name": "maven"
    },
    {
      "name": "hsf"
    }
  ]
}
```


archimedes.alibaba-inc.com/ /maven/

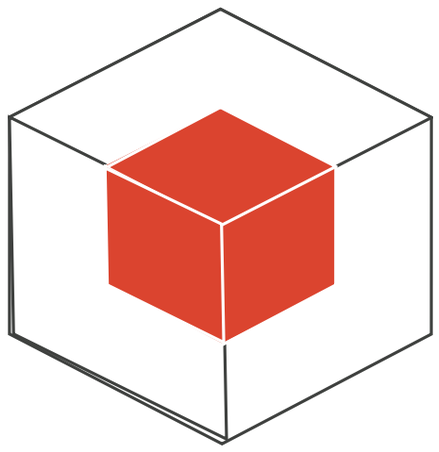

```
{
  "endpoint": {
    "docs": "http://gitlab.alibaba-inc.com/spring-boot/maven-spring-boot-starter",
    "name": "maven",
    "scm": "http://gitlab.alibaba-inc.com/spring-boot/maven-spring-boot-starter",
    "version": "1.1.1",
    "authors": [
      "juven.xuxb"
    ]
  },
  "runtime": {
    "groupId": "com.alibaba.intl.base.fileserver",
    "dependencyCount": 356,
    "artifactId": "fileserver2-rpc",
    "snapshotDependencies": [ … ], // 18 items
    "version": "1.0-SNAPSHOT",
    "snapshotDependencyCount": 18,
    "dependencies": [
      "ch.qos.logback:logback-classic:1.1.7",
      "ch.qos.logback:logback-core:1.1.7",
      "com.ali.com.google.guava:guava.hsf:18.0",
      "com.alibaba.alimonitor:alimonitor-jmonitor:1.1.8",
      "com.alibaba.aliyunid:aliyunid.client.java:1.0.2",
      "com.alibaba.boot:alimetrics-spring-boot-starter:1.0.0",
      "com.alibaba.boot:alimonitor-spring-boot-starter:1.0.4",
      "com.alibaba.boot:diamond-spring-boot-starter:1.2.1",
      "com.alibaba.boot:docker-spring-boot-starter:1.0.1",
      "com.alibaba.boot:maven-spring-boot-starter:1.1.1",
      "com.alibaba.configserver.google.code.gson:ali-gson:2.2",
      "com.alibaba.dts:dts-client:1.6.13",
```

# endpoints

```java
public abstract class AbstractAliExpressEndpoint extends AbstractEndpoint<Map<String, Object>> {

    public AbstractAliExpressEndpoint(String id) {
        super(id, false, true);
    }

    /**
     * Get the name of the endpoint
     *
     * @return endpoint name
     */
    @NotNull
    public abstract String getName();

    /**
     * Get the version of the Endpoint
     *
     * @return current version
     */
    @NotNull
    public abstract String getVersion();

    /**
     * Get the authors of the Endpoint
     *
     * @return author list
     */
    @NotNull
    public abstract List<String> getAuthors();
```

AliExpress
Smarter Shopping, Better Living!

# endpoints



Git Info
Docker Image Info
Spring Boot Version
Java Dependencies

Startup Time
Package Size

Release Rate
Most Recent Release

Load
GC Info
Disk Usage

*Collect informations using endpoints,*
*and analyze them.*

AliExpress
Smarter Shopping, Better Living!

# Recap

- The Big Picture of AliExpress technical stack.
- Why we use Spring Boot.
- Simple but powerful Spring Boot techniques.
    - BOM
    - Health Indicator
    - Failure Analyzer
    - SpringApplicationEvent
    - PropertySourceLocator
    - Configuration - Global
    - Configuration - Bean Specific
    - logback-spring.xml
    - endpoints

AliExpress™
Smarter Shopping, Better Living!

# Q&A

juven.xuxb@alibaba-inc.com
@juvenxu

Pivotal | InfoQ

# Thanks!