SpringBoot2.0 基础案例(16)：配置 Actuator 组件，实现系统监控

本文源码：GitHub·点这里 || GitEE·点这里

一、Actuator 简介

1、监控组件作用

在生产环境中，需要实时或定期监控服务的可用性。Spring Boot 的 actuator（健康监控）功能提供了很多监控所需的接口，可以对应用系统进行配置查看、相关功能统计等。

2、监控分类

Actuator 提供 Rest 接口，展示监控信息。
接口分为三大类：
应用配置类：获取应用程序中加载的应用配置、环境变量、自动化配置报告等与 SpringBoot 应用相关的配置类信息。
度量指标类：获取应用程序运行过程中用于监控的度量指标，比如：内存信息、线程池信息、HTTP 请求统计等。
操作控制类：提供了对应用的关闭等操作类功能。

二、与 SpringBoot2.0 整合

1、核心依赖 Jar 包

```
<!-- 监控依赖 -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

2、Yml 配置文件

```yaml
# 端口
server:
  port: 8016
spring:
  application:
    # 应用名称
    name: node16-boot-actuator
management:
  endpoints:
    web:
      exposure:
        # 打开所有的监控点
        include: "*"
      # 自定义监控路径 monitor
      # 默认值：http://localhost:8016/actuator/*
      # 配置后：http://localhost:8016/monitor/*
      base-path: /monitor
  endpoint:
    health:
      show-details: always
    shutdown:
      # 通过指定接口关闭 SpringBoot
      enabled: true
  # 可以自定义端口
  # server:
  #   port: 8089
```

```
# 描述项目基础信息
info:
  app:
    name: node16-boot-actuator
    port: 8016
    version: 1.0.0
    author: cicada
```

三、监控接口详解

1、Info 接口

Yml 文件中配置的项目基础信息

路径：`http://localhost:8016/monitor/info`

输出：

```
{
    "app": {
        "name": "node16-boot-actuator",
        "port": 8016,
        "version": "1.0.0",
        "author": "cicada"
    }
}
```

2、Health 接口

health 主要用来检查应用的运行状态

路径：`http://localhost:8016/monitor/health`

输出：


```
{
    "status": "UP",
    "details": {
        "diskSpace": {
            "status": "UP",
            "details": {
                "total": 185496236032,
                "free": 140944084992,
                "threshold": 10485760
            }
        }
    }
}
```


3、Beans 接口

展示了 bean 的类型、单例多例、别名、类的全路径、依赖 Jar 等内容。

路径：`http://localhost:8016/monitor/beans`

输出：


```
{
    "contexts": {
        "node16-boot-actuator": {
        "beans": {
            "endpointCachingOperationInvokerAdvisor": {
                "aliases": [],
                "scope": "singleton",
```

```
                "type":
"org.springframework.boot.actuate.endpoint.invoker.cache.CachingOperationIn
vokerAdvisor",

                "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/EndpointAutoConfig
uration.class]",

                "dependencies": ["environment"]

            }

        }

    }

}
```


4、Conditions 接口

查看配置在什么条件下有效，或者自动配置为什么无效。

路径：`http://localhost:8016/monitor/conditions`

输出：


```
{

    "contexts": {

        "node16-boot-actuator": {

            "positiveMatches": {

                "AuditAutoConfiguration#auditListener": [{

                    "condition": "OnBeanCondition",

                    "message": "@ConditionalOnMissingBean"

                }],

    }

}
```


5、HeapDump 接口

自动生成 Jvm 的堆转储文件 HeapDump，可以使用监控工具 VisualVM 打开此文件查看内存快照。

路径：http://localhost:8016/monitor/heapdump

6、Mappings 接口

描述 URI 路径和控制器的映射关系

路径：http://localhost:8016/monitor/mappings

输出：


```
{
    "contexts": {
        "node16-boot-actuator": {
            "mappings": {
                "dispatcherServlets": {
                    "dispatcherServlet": [ {
                        "handler": "Actuator web endpoint 'auditevents'",
                        "predicate": "{GET /monitor/auditevents ||
application/json]}",
                        "details": {
                            "handlerMethod": {
                                "className":
"org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpoi
ntHandlerMapping.Operat
                                "name": "handle",
                                "descriptor":
"(Ljavax/servlet/http/HttpServletRequest;Ljava/util/Map;)Ljava/lang/Object;
"
                            },
                            "requestMappingConditions": {
                                "consumes": [],
                                "headers": [],
                                "methods": ["GET"],
                                "params": [],
```

```
                                    "patterns": ["/monitor/auditevents"],
                                    "produces": [{
                                        "mediaType": "application/vnd.spring-
boot.actuator.v2+json",
                                        "negated": false
                                    }, {
                                        "mediaType": "application/json",
                                        "negated": false
                                    }]
                                }
                            }
                        }
                }
    }
}
```


7、ThreadDump 接口

展示线程名、线程 ID、是否等待锁、线程的状态、线程锁等相关信息。

路径：`http://localhost:8016/monitor/threaddump`

输出：


```
{
    "threads": [{
        "threadName": "DestroyJavaVM",
        "threadId": 34,
        "blockedTime": -1,
        "blockedCount": 0,
        "waitedTime": -1,
        "waitedCount": 0,
```

```
        "lockName": null,
        "lockOwnerId": -1,
        "lockOwnerName": null,
        "inNative": false,
        "suspended": false,
        "threadState": "RUNNABLE",
        "stackTrace": [],
        "lockedMonitors": [],
        "lockedSynchronizers": [],
        "lockInfo": null
    }
    ]
}
```


8、ShutDown 接口

优雅关闭 Spring Boot 应用，默认只支持 POST 请求。

路径：`http://localhost:8016/monitor/shutdown`

四、源代码地址

GitHub·地址

https://github.com/cicadasmile/spring-boot-base

GitEE·地址

https://gitee.com/cicadasmile/spring-boot-base