山东大学 计算机科学与技术 学院

现代软件开发技术 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：2020001300 | 姓名： | | 班级： |
| 实验题目：Spring Boot端点监控 | | | |
| 实验学时：2学时 | | 实验日期： 2023-12-19 | |
| 实验目的：  1.会使用Spring Boot端点监控  2.会编写自己的端点  3.分析电商平台实例 | | | |
| 硬件环境： | | | |
| 软件环境：  操作系统：windows 10  开发平台与工具：IDEA开发平台、jdk-11.0.1、apache-tomcat-9.0.37 | | | |
| 实验内容与问题回答：  1.练习Spring Boot端点监控。  打开课本项目ch10\_1  （1）分析pom.xml文件，  Q:支持应用端点监控的依赖是哪一条？    （2）打开src/main/resources下的application.properties配置文件。  Q:解释其中各项端点设置的含义？  1.spring.jackson.serialization.indent-output=true:  这个配置项用于让Jackson在序列化JSON时进行缩进，以提高JSON输出的可读性。  2.management.endpoint.shutdown.enabled=true:  启用/shutdown端点，该端点用于优雅地关闭应用程序。通过发送POST请求到该端点，可以触发应用程序的关闭。  3.management.endpoints.web.exposure.include=\*:  包含所有的Web端点，使它们对外暴露。这包括默认的端点，如/health、/info等。  4.#management.endpoints.web.exposure.exclude=env,beans:  这是一个被注释掉的设置，如果取消注释，则会排除指定的端点（在此例中是env和beans）不对外暴露。  5.info.app.name=spring-boot-hello:  为应用程序设置一个自定义的信息属性，通常在/info端点中可见。在这里，应用程序的名称被设置为"spring-boot-hello"。  6.info.app.version=v1.0.0:  设置应用程序的版本信息，同样会在/info端点中可见。在这里，版本被设置为"v1.0.0"。  7.management.endpoint.health.show-details=always:  将详细的健康信息显示给所有用户。这意味着/health端点将提供更详细的健康信息，而不仅仅是简单的健康状态。这对于故障排除和监控非常有用。  （3）测试运行项目  运行src/main/java下com.ch.ch10\_1包中Ch101Application.java(Run as Java Application)，  （a）用浏览器访问“http://localhost:8080/actuator”，  Q:显示的是什么格式的信息？  JSON格式的信息：    （b）分别使用下面的href,仔细观察显示的各种配置、指标等监控信息：  应用配置端点的测试：  http://localhost:8080/actuator/conditions  Q:信息分几类？  在conditions端点生成的自动化配置报告中将自动化配置内容的信息分为三部分：  positiveMatches中返回的是条件匹配成功的自动化配置；  negativeMatches中返回的是条件匹配不成功的自动化配置；  unconditionalClasses无条件配置类。  但是在本项目中由于配置文件中开启了所有的自动化配置：  所以只有positiveMatches这一类信息。  度量指标端点的测试：  Q:以下端点分别显示的什么信息？  以下端点分别显示的信息是：  ·metrics 用来返回当前应用的各类重要度量指标 。  ·metrics/jvm.memory.used用来显示 jvm 已经使用的内存信息 。  ·health 用来获取应用的各类健康指标信息 。  ·threaddump 用来获取程序运行中的线程信息。  ·httptrace 用来返回基本的 HTTP 跟踪信息 。  ·scheduledtasks 获取应用程序中调度任务的统计信息。  http://localhost:8080/actuator/metrics    http://localhost:8080/actuator/metrics/jvm.memory.used    http://localhost:8080/actuator/health    操作控制端点的测试：  远程关闭应用：  (1)在配置文件中加入 management.endpoint.shutdown.enabled =true  (2)使用rest-client-master（可从课程网站/softwares/下载），用post方式访问 http://localhost:8080/actuator/shutdown  Q:观察项目是否已关闭？  通过post方式访问发现项目已经关闭。  2.练习自定义端点  打开课本项目ch10\_2  （1）打开项目属性文件配置数据源  （2）分析自定义端点类 DataSourceEndpoint.java  这是一个自定义的Actuator端点类，通过实现Endpoint接口和使用@ReadOperation和@WriteOperation注解，它提供了关于Hikari连接池的监控和操作功能。  @Endpoint(id = "data-source"):  通过@Endpoint注解，这个类被注册为一个Actuator端点。id参数指定了端点的唯一标识，这里设置为"data-source"。  @Component:  使用@Component注解将这个类标记为Spring的组件，以便它可以被Spring容器扫描并实例化。  public DataSourceEndpoint(HikariDataSource ds):  构造函数接受一个HikariDataSource实例，这个实例用于获取连接池的配置信息和运行状态。  @ReadOperation:  使用@ReadOperation注解的info()方法提供了一个只读操作，用于返回关于Hikari连接池的信息。具体来说，它获取连接池的配置信息（如最大连接数）和运行状态（如活动连接数、空闲连接数、等待连接的线程数）。  @WriteOperation:  使用@WriteOperation注解的setMax(int max)方法提供了一个可写操作，用于设置连接池的最大连接数。通过调用setMaximumPoolSize(max)来实现。  （3）测试端点  http://localhost:8080/actuator/data-source    用rest-client-master以post方式发送：  http://localhost:8080/actuator/data-source?max=20  再次访问测试端点 http://localhost:8080/actuator/data-source  Q:显示什么信息？    再次访问显示的信息如上图所示，data-source 中的 max 属性最大连接数被修改为 20。  3.练习自定义HealthIndicator  （1）分析项目ch10\_2中的MyHealthIndicator.java  Q: check()方法返回1，代表什么状态？  check() 方法返回 1，代表健康检查失败或应用程序处于异常状态。  在 MyHealthIndicator 类的 health() 方法中，如果 check() 方法返回的错误代码不等于 0，那么 Health.down() 方法将被调用，表示应用程序的健康状态为 "DOWN"（不健康），并且可以使用 withDetail("message", "error:" + errorCode) 添加额外的异常信息。  因此，返回 1 表示在模拟的健康检查中发生了错误，应用程序的健康状态将被设置为 "DOWN"。  （2）测试端点  http://localhost:8080/actuator/health/my    4.分析eBusiness系统  （1）外部数据库准备  在MySql中创建名为shop的数据库（可利用navicat工具）,用SQL命令文件shop.sql在shop数据库中创建所需的数据表。    （2）打开课本项目eBusiness，  参照“第11章 电子商务平台的设计与实现.PPT”和SPOC网站相关视频，了解系统的系统设计、数据库设计、系统管理模块、组件设计等。  系统管理模块：  主要包括添加相关依赖、HTML页面及静态资源管理、应用的目录结构、配置文件等。  （3）分析项目的配置  pom.xml中的依赖    （4）分析项目的实体、数据访问、控制流程  项目实体：  该项目的项目实体类主要有以下entity：    数据访问和控制流程：  该项目数据访问和控制流程主要由以下部分组成：  在这个项目中，数据访问和控制流程主要涉及到Spring Boot、Thymeleaf、MySQL数据库和MyBatis。以下是主要的数据访问和控制流程：  Spring Boot启动  项目使用Spring Boot作为基础框架，通过Spring Boot的spring-boot-starter-web启动器启动。这使得应用程序能够处理Web请求。  控制层（Controller）:  使用@Controller注解的类处理用户请求。这些类中的方法（使用@RequestMapping等注解）定义了用户请求的处理逻辑。  Thymeleaf模板引擎:  使用spring-boot-starter-thymeleaf启动器，Thymeleaf被用作模板引擎。Controller层的方法返回视图名称，Thymeleaf负责渲染HTML模板。  数据访问层（Data Access Layer）:  使用MyBatis框架处理数据访问。mybatis-spring-boot-starter启动器集成了Spring Boot和MyBatis，简化了配置。通过MyBatis的@Mapper注解或XML文件，定义了数据访问接口和SQL语句。  MySQL数据库:  使用MySQL数据库存储和检索数据。通过mysql-connector-java依赖项，应用程序能够与MySQL数据库建立连接。  Service层:  可能存在Service层，其中包含业务逻辑。Service层可以调用数据访问层的方法，并对数据进行处理，然后返回给Controller层。  实体类:  可能存在用于表示数据模型的Java实体类，这些类与数据库表相对应。MyBatis映射文件或注解用于定义实体类与数据库表之间的映射关系。  前端交互:  用户通过浏览器发起HTTP请求，请求被Spring Boot的控制器处理。Thymeleaf模板引擎负责生成HTML响应，将数据嵌入到HTML页面中。  Q:该项目使用哪种数据访问方式？  根据项目的依赖配置，该项目使用了 MyBatis 作为主要的数据访问方式，采用了Controller层、Service层和数据访问层的架构。  此外，项目中还存在用于定义数据访问接口和 SQL 语句的 MyBatis Mapper 类或 XML 文件。这些文件用于将 Java 对象映射到数据库表，并定义了与数据库的交互方式。  （5）分析项目的视图页面、访问控制等  从系统分析得知，管理员成功登录后，才能管理商品、商品类型、用户、订单等功 能模块。因此，本系统需要对这些功能模块的操作进行管理员登录权限控制。在 com.ch.ebusiness.controller.admin 包中创建 AdminBaseController 控制器类，该类 中有一个@ModelAttribute 注解的方法 isLogin。isLogin 方法的功能是判断管理员是否 已成功登录。需要进行管理员登录权限控制的控制器类继承 AdminBaseController 类即 可，因为带有@ModelAttribute 注解的方法首先被控制器执行。 从系统分析得知，用户成功登录后，才能购买商品、收藏商品、查看购物车、我的 订 单 以 及 个 人 信 息 。 与 管 理 员 登 录 权 限 验 证 同 理 ， 在 com.ch.ebusiness.controller.before 包中创建 BeforeBaseController 控制器类，该 类中有一个@ModelAttribute 注解的方法 isLogin。isLogin 方法的功能是判断前台用户 是 否 已 成 功 登 录 。 需 要 进 行 前 台 用 户 登 录 权 限 控 制 的 控 制 器 类 继 承 BeforeBaseController 类即可。  系统对未登录异常、数据库操作异常以及程序未知异常进行了统一异常处理。  （6）测试运行  管理员访问页面：http://localhost:8080/eBusiness/admin/toLogin    普通用户访问页面：http://localhost:8080/eBusiness    Q:系统为什么加“用户已存在”判断功能？  防止一个人注册多个用户信息。 | | | |
| 思考题解答：  无 | | | |