POM

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>com.example</groupId>  <artifactId>website-monitor</artifactId>  <version>1.0.0</version>  <packaging>jar</packaging>  <name>website-monitor</name>  <description>Website Monitoring Spring Boot Application</description>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.6.3</version> <!-- Use the latest stable version -->  <relativePath/> <!-- lookup parent from repository -->  </parent>  <properties>  <java.version>11</java.version>  </properties>  <dependencies>  <!-- Spring Boot Starter dependencies -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter</artifactId>  </dependency>  <!-- Spring Web Starter for web applications -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <!-- Spring Mail Starter for sending emails -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-mail</artifactId>  </dependency>  <!-- Spring Context Starter for managing the application context -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-actuator</artifactId>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

JSON

|  |
| --- |
| {  "websites": [  {  "name": "Example Website 1",  "url": "https://example.com",  "timeout": 5000,  "expectedResponse": "Expected Content for Example Website 1"  },  {  "name": "Example Website 2",  "url": "https://example2.com",  "timeout": 5000,  "expectedResponse": "Expected Content for Example Website 2"  },  {  "name": "Example Website 3",  "url": "https://example3.com",  "timeout": 5000,  "expectedResponse": "Expected Content for Example Website 3"  }  ]  } |

application.properties:

|  |
| --- |
| # Email Configuration  spring.mail.host=your-smtp-host  spring.mail.port=587  spring.mail.username=your-email@example.com  spring.mail.password=your-email-password  spring.mail.properties.mail.smtp.auth=true  spring.mail.properties.mail.smtp.starttls.enable=true  # Website JSON File Path  website.json-file=/path/to/websites.json  # Scheduling Configuration (Check websites every 15 minutes)  website.check-interval=900000  # Logging Configuration  logging.level.org.springframework=INFO  logging.level.com.example.websitemonitor=DEBUG |

**MainApplication.java:**

|  |
| --- |
| import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.scheduling.annotation.EnableScheduling;  @SpringBootApplication  @EnableScheduling  public class MainApplication {  public static void main(String[] args) {  SpringApplication.run(MainApplication.class, args);  }  } |

### EmailService.java:

|  |
| --- |
| import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.mail.SimpleMailMessage;  import org.springframework.mail.javamail.JavaMailSender;  import org.springframework.stereotype.Service;  @Service  public class EmailService {  @Autowired  private JavaMailSender javaMailSender;  public void sendEmail(String subject, String message) {  // Implement logic to send email  SimpleMailMessage mailMessage = new SimpleMailMessage();  mailMessage.setSubject(subject);  mailMessage.setText(message);  // Set email recipient, sender, etc.  javaMailSender.send(mailMessage);  }  } |

### WebsiteCheckerService.java:

|  |
| --- |
| import com.fasterxml.jackson.databind.JsonNode;  import com.fasterxml.jackson.databind.ObjectMapper;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.beans.factory.annotation.Value;  import org.springframework.http.ResponseEntity;  import org.springframework.scheduling.annotation.Scheduled;  import org.springframework.stereotype.Service;  import org.springframework.web.client.RestTemplate;  import javax.annotation.PostConstruct;  import java.io.File;  import java.io.IOException;  import java.net.HttpURLConnection;  import java.util.ArrayList;  import java.util.List;  @Service  public class WebsiteCheckerService {  private List<WebsiteInfo> websites;  @Autowired  private EmailService emailService;  @Value("${website.json-file}")  private String jsonFilePath;  @PostConstruct  public void init() {  loadWebsitesFromJson();  }  @Scheduled(fixedRateString = "${website.check-interval}")  public void scheduleWebsiteCheck() {  checkWebsites();  }  private void checkWebsites() {  for (WebsiteInfo website : websites) {  checkWebsite(website);  }  }  private void checkWebsite(WebsiteInfo website) {  RestTemplate restTemplate = new RestTemplate();  // Set timeout for the request  restTemplate.setRequestFactory(new SimpleClientHttpRequestFactoryWithTimeout(website.getTimeout()));  try {  ResponseEntity<String> responseEntity = restTemplate.getForEntity(website.getUrl(), String.class);  // Check if the response status code indicates an error  if (responseEntity.getStatusCode().isError()) {  // Send email about the error  emailService.sendEmail("Website Response Issue", "Check website: " + website.getUrl() +  "\nHTTP Status Code: " + responseEntity.getStatusCodeValue() +  "\nResponse Body: " + responseEntity.getBody());  } else {  // Check the response content against the expected content from the JSON file  if (!isExpectedResponse(responseEntity.getBody(), website.getExpectedResponse())) {  // Send email about the unexpected response  emailService.sendEmail("Unexpected Website Response", "Check website: " + website.getUrl() +  "\nUnexpected Response Body: " + responseEntity.getBody());  }  }  } catch (Exception e) {  // Handle timeout or other exceptions  emailService.sendEmail("Website Response Timeout", "Timeout for website: " + website.getUrl());  }  }  private boolean isExpectedResponse(String actualResponse, String expectedResponse) {  // Implement logic to compare actual and expected responses  // Return true if they match, false otherwise  return actualResponse.equals(expectedResponse);  }  private void loadWebsitesFromJson() {  try {  ObjectMapper objectMapper = new ObjectMapper();  File jsonFile = new File(jsonFilePath);  // Read JSON file and parse into List<WebsiteInfo>  JsonNode jsonNode = objectMapper.readTree(jsonFile);  JsonNode websitesNode = jsonNode.get("websites");  websites = new ArrayList<>();  for (JsonNode websiteNode : websitesNode) {  String name = websiteNode.get("name").asText();  String url = websiteNode.get("url").asText();  int timeout = websiteNode.get("timeout").asInt();  String expectedResponse = websiteNode.get("expectedResponse").asText();  websites.add(new WebsiteInfo(name, url, timeout, expectedResponse));  }  } catch (IOException e) {  e.printStackTrace();  }  }  private static class WebsiteInfo {  private String name;  private String url;  private int timeout;  private String expectedResponse;  public WebsiteInfo(String name, String url, int timeout, String expectedResponse) {  this.name = name;  this.url = url;  this.timeout = timeout;  this.expectedResponse = expectedResponse;  }  public String getName() {  return name;  }  public String getUrl() {  return url;  }  public int getTimeout() {  return timeout;  }  public String getExpectedResponse() {  return expectedResponse;  }  }  // Custom RequestFactory to set timeout  private static class SimpleClientHttpRequestFactoryWithTimeout extends SimpleClientHttpRequestFactory {  private final int timeout;  public SimpleClientHttpRequestFactoryWithTimeout(int timeout) {  this.timeout = timeout;  }  @Override  protected void prepareConnection(HttpURLConnection connection, String httpMethod) throws IOException {  super.prepareConnection(connection, httpMethod);  connection.setConnectTimeout(timeout);  connection.setReadTimeout(timeout);  }  }  } |