DOI 10.34132/mspc2025.01.06.29

В'ячеслав Старченко

ВИКОРИСТАННЯ JAVA SERVER PAGES ДЛЯ ВІДДАЛЕНОГО МОНІТОРИНГУ АПАРАТНОЇ СКЛАДОВОЇ ВЕБСЕРВЕРУ

Розглянуто підхід до реалізації системи віддаленого моніторингу апаратної складової вебсерверу на базі Java Server Pages (JSP). Запропоноване рішення використовує інструмент Java ProcessBuilder для виконання системних команд (sensors, apcaccess, df) на сервері під керуванням ОС Debian з метою отримання даних про температурний режим, параметри системи живлення та стан дискової підсистеми. Отримані результати передаються на клієнтську сторону у форматі JSON, де за допомогою технології Ajax відбувається їх динамічне відображення без перезавантаження сторінки. Описана архітектура та технічна реалізація системи, включаючи приклади коду JSP-сторінки та JavaScript-модуля для обробки Ajax-запитів. Проаналізовано ефективність цього рішення в контексті Internet of Things (IoT), відзначено простоту реалізації, зручність використання, масштабованість та гнучкість підходу для віддаленого моніторингу апаратних параметрів вебсервера.

Ключові слова: *JSP, Apache Tomcat, ProcessBuilder, JSON, веброзробка, системні команди.*

Вступ

У сучасних системах Internet of Things (IoT) все більшого значення набуває віддалений моніторинг апаратних параметрів серверів та мережевих пристроїв. Одним із підходів до вирішення цієї задачі є використання Java Server Pages (JSP) у комбінації з технологіями, які дозволяють виконувати системні перевірки в режимі реального часу [1]. У цьому контексті розглядається можливість створення за допомогою класу ProcessBuilder окремого Java-процесу, що виконує системні команди для діагностики апаратної складової сервера, який працює під керуванням ОС Debian [2]. Результати зазначених перевірок повертаються у форматі JSON [3], після чого на боці клієнта за допомогою технології Ajax відбувається їх динамічне відображення [4].

Архітектура системи

Запропоноване рішення базується на використанні програмного інструменту Java ProcessBuilder, який дозволяє створювати та керувати системними процесами безпосередньо з коду JSP-сторінки. Це дозволяє виконувати стандартні системні команди OS Debian для отримання важливих показників апаратного забезпечення [5].

Системні команди, що використовуються для моніторингу поточного стану сервера наведено у табл. 1.

Таблиця 1.

Системні команди OS Debian, що використовуються для моніторингу поточного стану сервера

Команда	Опис
sensors	Перевірка температурного режиму
apcaccess	Моніторинг параметрів системи живлення
df	Аналіз використання дискового простору

За допомогою класу ProcessBuilder створюється окремий процес, який почергово виконує кожну з команд, наведених у табл. 1. Це дає змогу ізолювати процес виконання системних перевірок від основного процесу обробки запитів клієнтів в середовищі Tomcat, тим самим забезпечуючи стабільність роботи вебсервера.

Контролюються наступні ключові апаратні параметри вебсервера:

1) *температурний режим* – за допомогою команди `sensors`, яка надає повну інформацію про температуру процесора, материнської плати та інших критичних компонентів комп'ютера;

2) *параметри системи живлення* – за допомогою команди `apcaccess`, яка надає дані про напругу джерела живлення та поточне споживання енергії, стан зарядки акумулятора та інші параметри системи неперервного живлення сервера;

3) *стан дискової підсистеми* – за допомогою команди `df` для моніторингу вільного місця та стану розділів дискової підсистеми сервера.

Після збору даних вони перетворюються у формат JSON. Цей формат є легким і широко поширеним для обміну даними між вебсервером та клієнтом. За допомогою технології Ajax на клієнтській стороні

забезпечується асинхронне оновлення сторінки, що створює користувачу відчуття миттєвості та безперервного моніторингу.

Така архітектура дозволяє створити зручний інтерфейс для віддаленого моніторингу, де дані оновлюються в режимі реального часу без необхідності перезавантаження сторінки. Крім того, використання технології Ajax сприяє оптимізації використання ресурсів, оскільки між сервером і клієнтом передаються лише необхідні дані.

Технічна реалізація

На рис. 1 наведено приклад ключового компоненту JSP-сторінки, що відповідає за збір даних.

```
<%@ page language="java" contentType="application/json; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.io.*,java.util.*,org.json.*"  %>
<%
   // Створюємо JSON об’єкт для відповіді
   JSONObject  responseData = new JSONObject();

   try {
      // Отримання даних про температуру
      ProcessBuilder sensorsProcess = new ProcessBuilder("sensors");
      Process sensorExec = sensorsProcess.start();
      String tempData = readProcessOutput(sensorExec.getInputStream());
      responseData.put("temperature", parseTemperatureData(tempData));

      // Отримання даних про живлення
      ProcessBuilder powerProcess = new ProcessBuilder("apcaccess");
      Process powerExec  = powerProcess.start();
      String powerData = readProcessOutput(powerExec.getInputStream());
      responseData.put("power", parsePowerData(powerData));

      // Отримання інформації про диски
      ProcessBuilder dfProcess = new ProcessBuilder("df", "-h");
      Process dfExec  = dfProcess.start();
      String diskData = readProcessOutput(dfExec.getInputStream());
      responseData.put("disk", parseDiskData(diskData));

      responseData.put("status", "success");
   } catch (Exception e) {
      responseData.put("status", "error");
      responseData.put("message", e.getMessage());
   }

   // Виведення JSON відповіді
   out.print(responseData.toString());
%>
```

Рис. 1. Приклад коду JSP-сторінки, що відповідає за збір даних

На боці клієнта дані відображаються за допомогою Ajax-запитів і представляються в зручній для користувача формі. Приклад JavaScript-коду з отримання та розбору Ajax-запитів наведено на рис. 2.

```
$(document).ready(function() {
   function updateSystemStatus() {
      $.ajax({
         url: ‘systemMonitor.jsp’,
         type: ‘GET’,
         dataType: ‘json’,
         success: function(data) {
            if(data.status === "success") {
               // Оновлення інформації на веб-сторінці
               updateTemperatureDisplay(data.temperature);
               updatePowerDisplay(data.power);
               updateDiskDisplay(data.disk);
            }
```

```
        },
        error: function(xhr, status, error) {
          console.error("Помилка отримання даних: " + error);
        },
        complete: function() {
          // Оновлення кожні 5 секунд
          setTimeout(updateSystemStatus, 5000);
        }
      });
  }

  // Запуск моніторингу
  updateSystemStatus();
});
```

Рис. 2. Приклад коду з отримання та розбору Ajax-запитів на боці клієнта

Аналіз ефективності рішення

Запропонований підхід має низку суттєвих переваг для використання в контексті IoT:

1) простота реалізації:

– *стандартні API Java:* Використання ProcessBuilder не вимагає додаткових зовнішніх бібліотек. Це дозволяє розробникам, які вже знайомі з Java, оперативно інтегрувати цей функціонал у свої проєкти;

– *JSP та Tomcat:* JSP є невід'ємною частиною стандартного стеку технологій Java для веброзробки, що робить його зручним вибором для розробки як серверних застосунків, так і IoT-платформ;

– *JSON та Ajax:* Простота роботи з JSON для серіалізації/десеріалізації даних та асинхронна природа Ajax роблять інтерфейс користувача більш інтуїтивним та швидким;

– *мінімальний програмний код:* Використовується компактний JSP-скрипт, що не потребує складних налаштувань на боці сервера;

– *використання стандартних системних утиліт:* Не потрібно розробляти власні інструменти системної діагностики;

2) зручність використання:

– *універсальність:* Система працює на будь-якому сервері з підтримкою JSP (Tomcat, Jetty тощо);

– *інтеграція з наявними системами:* Оскільки метод використовує стандартні системні команди та API, інтегрувати його у вже наявні проєкти на базі Tomcat дуже легко;

– *модульність рішення:* Кожна частина моніторингу (температура, параметри живлення, стан дискової підсистеми) працює окремо. Це дає змогу розширювати систему за потребою або адаптувати її під специфічні вимоги IoT-платформ;

– *гнучкість у розгортанні:* Застосунок на базі JSP можна легко розгорнути в стандартному середовищі Tomcat, що є перевагою для IoT-сценаріїв, де важлива швидкість впровадження та масштабованість;

– *вебінтерфейс:* Забезпечує доступ до діагностичної інформації з будь-якого пристрою через браузер;

– *JSON-формат даних:* Є промисловим стандартом та легко інтегрується з іншими системами та фреймворками;

– *реальний час:* Регулярне оновлення даних за допомогою Ajax-запитів без перезавантаження сторінки;

3) ефективність у контексті IoT:

– *реальний час:* Постійний моніторинг апаратних параметрів є критично важливим для IoT-систем, де віддалене керування та контроль обладнання можуть суттєво підвищити стійкість та надійність функціонування великої інфраструктури;

– *гнучкість та легкість адаптації:* Системні команди, описані у доповіді, можуть бути легко адаптовані під специфічні вимоги різних апаратних платформ. Це дозволяє використовувати запропоноване рішення практично у будь-якому апаратному середовищі, що працює під керуванням ОС Debian;

– *масштабованість:* Рішення є модульним, що дозволяє додавати інші параметри моніторингу, наприклад, аналіз мережевого трафіку або стану оперативної пам'яті. Це робить його універсальним для IoT-інфраструктури. Існує також можливість моніторингу великої кількості серверів за допомогою єдиного інтерфейсу;

– *низьке навантаження на систему сервера:* Мінімальне використання ресурсів системи для збору та передачі даних;

– *безпека:* Можливість налаштування різних рівнів доступу через стандартні механізми захисту Tomcat.

Висновки

Запропонований підхід, що полягає у використанні Java Server Pages у комбінації з інструментом Java ProcessBuilder для виконання системних команд та обробкою отриманих результатів з представленням їх у форматі JSON з подальшим відображенням за допомогою Ajax, є простим, зручним та ефективним рішенням для моніторингу апаратної складової серверів. Він дозволяє швидко інтегрувати моніторинг в інформаційну систему на базі Apache Tomcat, забезпечуючи при цьому високу швидкість оновлення даних

та гнучке розширення функціоналу у майбутньому. З огляду на зростаючу потребу у віддаленому моніторингу в IoT-системах, такий підхід може стати надійною основою для створення розумних, високоефективних рішень, які здатні опрацьовувати великі обсяги даних в режимі реального часу.

References

1. JavaServer Pages API Documentation. Retrieved from: https://docs.oracle.com/cd/E17802_01/products/products/jsp/2.1/docs/jsp-2_1-pfd2/index.html (Last accessed: 10.05.2025).
2. ProcessBuilder API. Retrieved from: https://docs.oracle.com/javase/8/docs/api/java/lang/ProcessBuilder.html (Last accessed: 10.05.2025).
3. Introducing JSON. Retrieved from: https://www.json.org/json-en.html (Last accessed: 10.05.2025).
4. AJAX Introduction. Retrieved from: https://www.w3schools.com/Js/js_ajax_intro.asp (Last accessed: 10.05.2025).
5. Debian Reference Retrieved from: https://www.debian.org/doc/manuals/debian-reference/debian-reference.en.pdf (Last accessed: 10.05.2025).

DOI 10.34132/mspc2025.01.06.29

Viacheslav Starchenko

USING JAVA SERVER PAGES FOR REMOTE MONITORING OF WEB SERVER HARDWARE

An approach to implementing a system for remote monitoring of the hardware component of a web server based on Java Server Pages (JSP) is considered. The proposed solution uses the Java ProcessBuilder tool to execute system commands (sensors, apcaccess, df) on a server running the Debian OS in order to obtain data on the temperature regime, power system parameters, and the state of the disk subsystem. The results obtained are transmitted to the client side in JSON format, where they are dynamically displayed without reloading the page using Ajax technology. The architecture and technical implementation of the system are described, including examples of JSP page code and a JavaScript module for processing Ajax requests. The effectiveness of this solution in the context of the Internet of Things (IoT) is analyzed, and the simplicity of implementation, ease of use, scalability, and flexibility of the approach for remote monitoring of web server hardware parameters are noted.

Keywords: *JSP, Apache Tomcat, ProcessBuilder, JSON, web development, system commands.*

Відомості про автора / Information about the Author

В'ячеслав Старченко, старший викладач кафедри комп'ютерної інженерії Чорноморського національного університету імені Петра Могили, м. Миколаїв, Україна. E-mail: viacheslav.starchenko@chmnu.edu.ua, orcid: https://orcid.org/0000-0002-7039-2332.

Viacheslav Starchenko, Senior Lecturer in Computer Engineering Department at the Petro Mohyla Black Sea National University, Mykolaiv, Ukraine. E-mail: viacheslav.starchenko@chmnu.edu.ua, orcid: https://orcid.org/0000-0002-7039-2332.

Стаття отримана редакцією 20.05.2025.