Implementing an MCP Server in Java

Java Day
Istanbul, Turkey
April 18, 2026

Boni García
https://bonigarcia.dev/

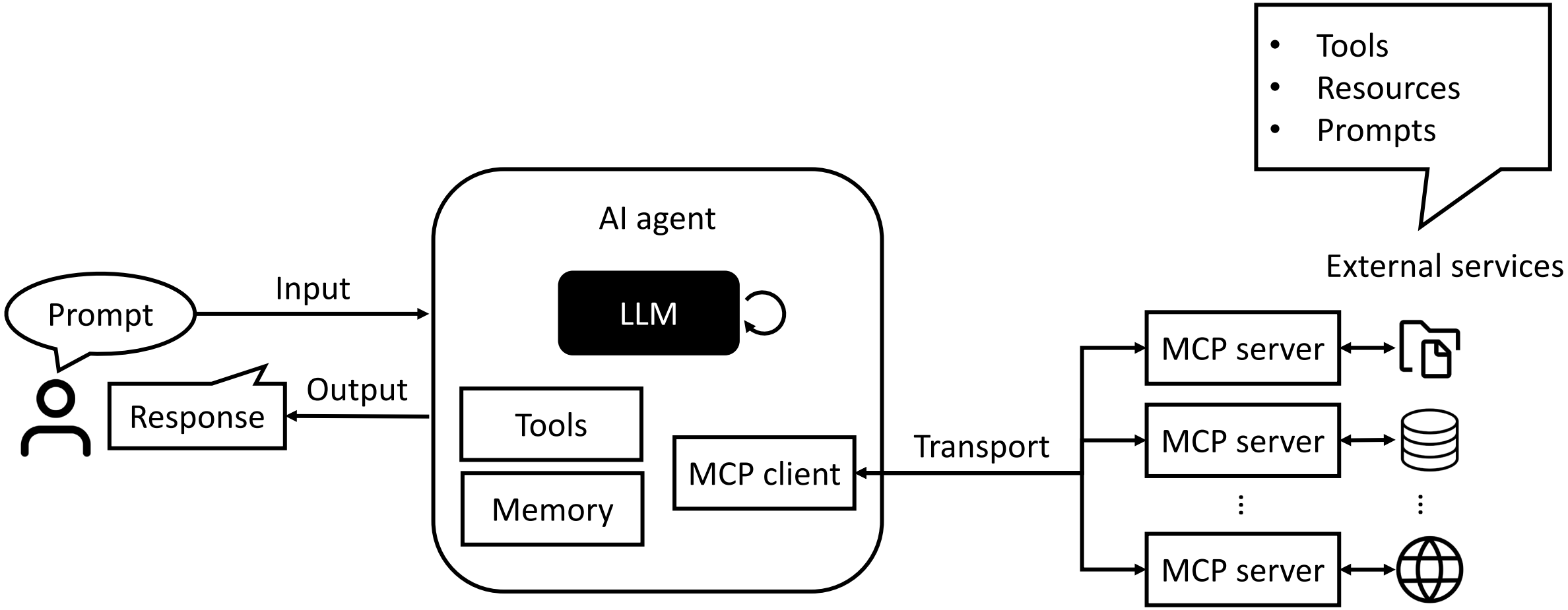
What is MCP?

> ***MCP (Model Context Protocol)*** *is an open standard for connecting AI agents to external services*

- Created by Anthropic in 2024
- Currently, it is part of the Linux Foundation (Agentic AI Foundation, AAIF)

https://modelcontextprotocol.io/

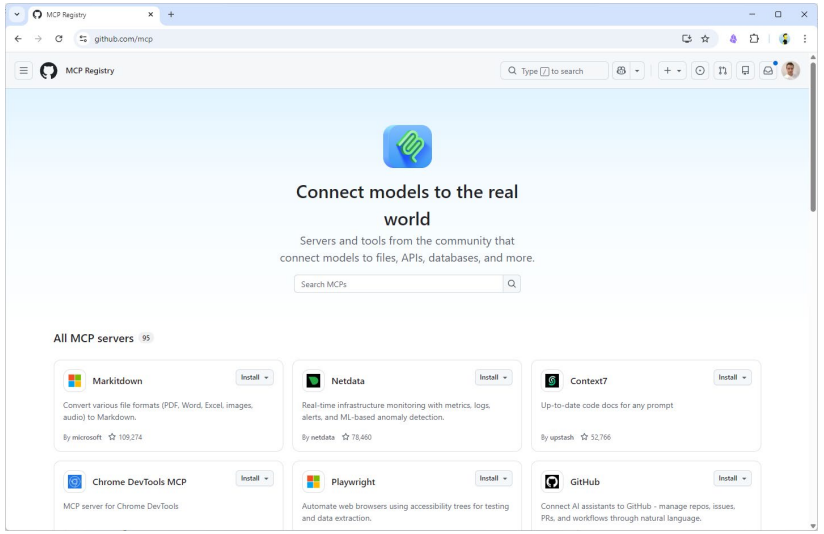
What is MCP?


Tools
Resources
Prompts
External services
AI agent
LLM
Prompt
Input
Response
Output
Tools
Memory
MCP client
Transport
MCP server
MCP server
MCP server

MCP servers

- There are plenty of MCP servers available, e.g.:
  - Filesystem MCP Server:
    https://github.com/modelcontextprotocol/servers/
  - GitHub MCP Server:
    https://github.com/github/github-mcp-server
  - Docker MCP Server:
    https://github.com/docker/hub-mcp/
  - Context7
    https://github.com/upstash/context7
  - …

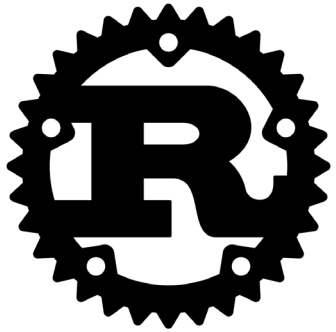
Find more on the official MCP registry

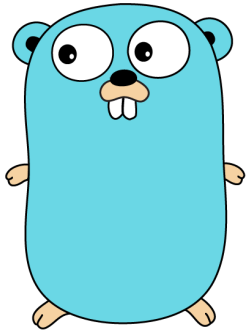


https://github.com/mcp

Implementing an MCP server

- MCP provides different official SDKs for building MCP servers



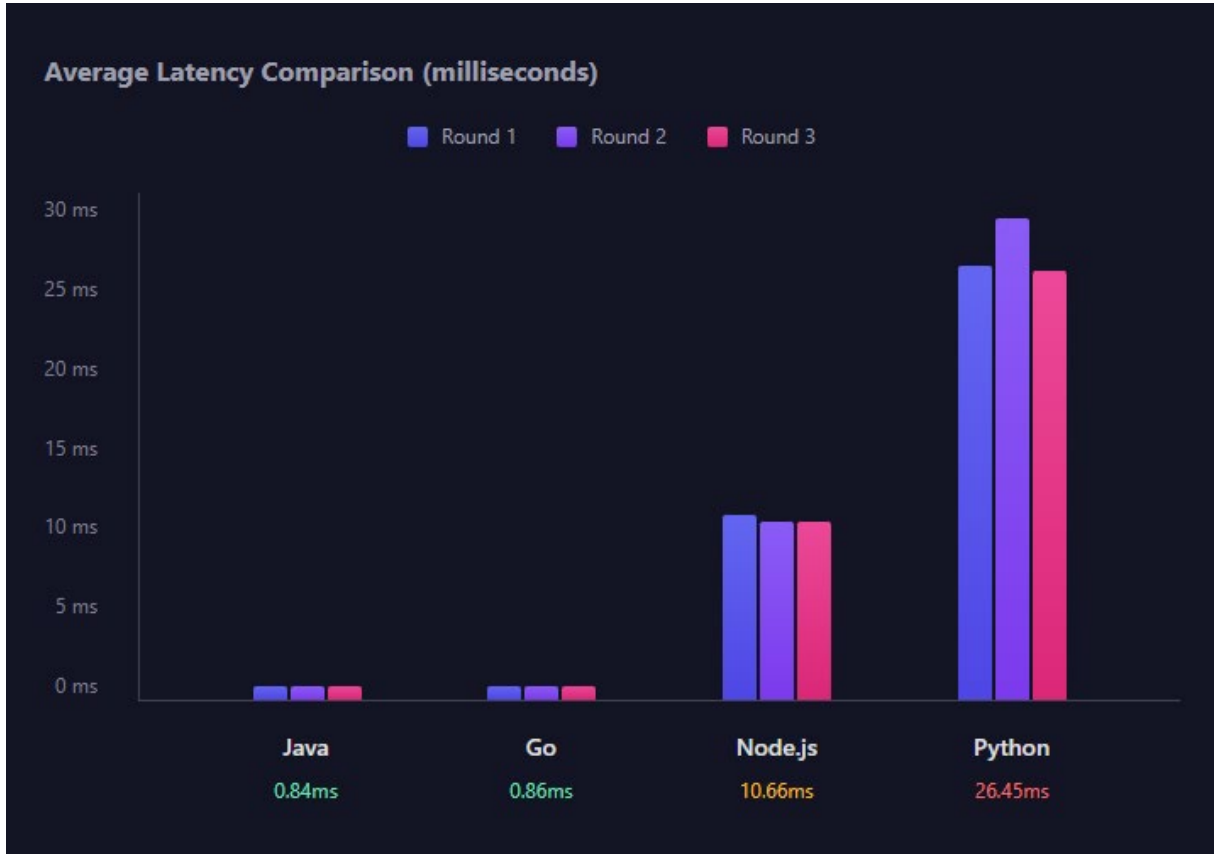





https://modelcontextprotocol.io/docs/sdk

Implementing an MCP server

- Why Java?



https://www.tmdevlab.com/mcp-server-performance-benchmark.html

Implementing an MCP server

https://java.sdk.modelcontextprotocol.io/latest/

https://docs.spring.io/spring-ai/reference/api/mcp/mcp-overview.html

---

https://docs.quarkiverse.io/quarkus-mcp-server/dev/index.html

---

https://micronaut-projects.github.io/micronaut-mcp/latest/guide/

Choosing a Java stack for an MCP server

Dimension	Java SDK	Spring AI	Quarkus	Micronaut
Programming model	Builder API, explicit wiring	Annotation-based, auto-configured	Annotation-based, build-time oriented	Annotation-based, compile-time metadata
Transport support	STDIO, HTTP	STDIO, HTTP	STDIO, HTTP, WebSocket	STDIO, HTTP
Enterprise integration	Minimal by default	Best fit for Spring ecosystem	Strong cloud-native stack	Good, lighter enterprise stack
Main trade-off	More manual wiring	Heavier stack, more framework magic	More opinionated platform	Smaller MCP community/ecosystem
Choose it when…	Maximum protocol control and portability	Safest default for enterprise Java	Runtime efficiency and ops features	You want a lean service with low overhead

*Feature-based opinionated comparison (not a benchmark)

Case study: Selenium

> ***Selenium*** *is a browser automation library*

- Multi-language

- Cross-browser

- Open-source and community-driven since 2004

https://selenium.dev/

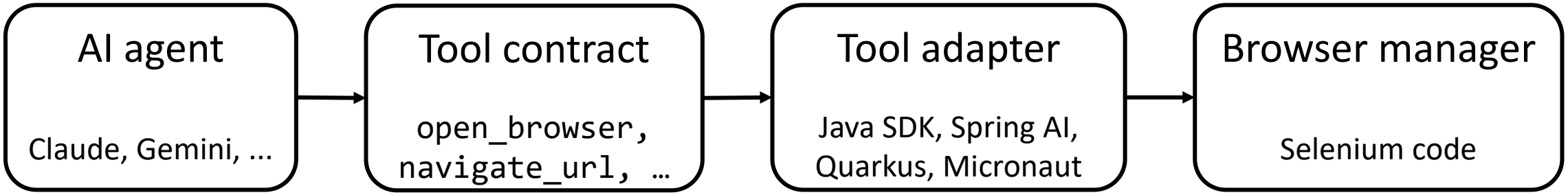
Case study: Selenium

- Why Selenium?

```
WebDriver driver = new ChromeDriver();

driver.get("https://modelcontextprotocol.io/");

driver.findElement(By.tagName("body")).getText();

driver.quit();
```

Tool	Description	Parameters
open_browser	Launches a new browser instance	browser_name (string): The name of the browser to open (e.g., "chrome", "firefox")
navigate_url	Navigate the browser to a specified URL	url (string): The complete URL to navigate to (e.g., https://example.com)
get_browser_text	Read the visible text of the entire page	-
close_browser	Closes the browser	-

Case study: Selenium



In general, we should keep business logic separate from the MCP adapter and the rest becomes replaceable

Case study: Selenium

> ***Context engineering**: the art and science of shaping context-aware AI systems*

https://github.com/bonigarcia/context-engineering

Case study: Selenium – Browser manager

Tool	Description	Parameters
open_browser	Launches a new browser instance	browser_name (string): The name of the browser to open (e.g., "chrome", "firefox")

```java
public class BrowserManager {

    private WebDriver driver;

    public BrowserResult start(String browserName) {
        if (driver != null) {
            return BrowserResult.error("Browser is already open.");
        }
        try {
            BrowserType type = BrowserType.from(browserName);
            switch (type) {
            case CHROME -> driver = new ChromeDriver();
            case FIREFOX -> driver = new FirefoxDriver();
            }
            return BrowserResult.ok("Browser started successfully.");
        } catch (IllegalArgumentException e) {
            return BrowserResult.error("Unsupported browser: " + browserName);
        } catch (Exception e) {
            return BrowserResult.error("Error starting browser: " + e.getMessage());
        }
    }
    // ...
}
```

Case study: Selenium – Browser manager

Tool	Description	Parameters
`navigate_url`	Navigate the browser to a specified URL	`url` (string): The complete URL to navigate to (e.g., https://example.com)

```java
public class BrowserManager {

    // ...

    public BrowserResult navigate(String url) {
        if (driver == null) {
            return BrowserResult.error("Browser not started.");
        }
        try {
            driver.get(url);
            return BrowserResult.ok("Navigation successful.");
        } catch (Exception e) {
            return BrowserResult.error("Error navigating: " + e.getMessage());
        }
    }

    // ...

}
```

Case study: Selenium – Browser manager

Tool	Description	Parameters
`get_browser_text`	Read the visible text of the entire page	-

```java
public class BrowserManager {

    // ...

    public BrowserResult readText() {
        if (driver == null) {
            return BrowserResult.error("Browser not started.");
        }
        try {
            String text = driver.findElement(By.tagName("body")).getText();
            return BrowserResult.ok(text);
        } catch (Exception e) {
            return BrowserResult.error("Error reading page: " + e.getMessage());
        }
    }

    // ...

}
```

Case study: Selenium – Browser manager

Tool	Description	Parameters
`close_browser`	Closes the browser	-

```java
public class BrowserManager {

    // ...

    public BrowserResult close() {
        if (driver == null) {
            return BrowserResult.error("Browser not started.");
        }
        try {
            driver.quit();
            return BrowserResult.ok("Browser closed successfully.");
        } catch (Exception e) {
            return BrowserResult
                    .error("Error closing browser: " + e.getMessage());
        } finally {
            driver = null;
        }
    }

}
```

Case study: Selenium – Java SDK


```gradle
plugins {
    id 'java'
    id 'application'
}

application {
    mainClass = 'io.github.bonigarcia.ce.McpSeleniumServer'
}

tasks.jar {
    duplicatesStrategy = DuplicatesStrategy.EXCLUDE
    manifest {
        attributes 'Main-Class': application.mainClass
    }
    from {
        configurations.runtimeClasspath.collect { it.isDirectory() ? it : zipTree(it) }
    }
}

dependencies {
    implementation platform('io.modelcontextprotocol.sdk:mcp-bom:1.1.1')
    implementation 'io.modelcontextprotocol.sdk:mcp'
    implementation 'org.seleniumhq.selenium:selenium-java:4.43.0'
}
```

Gradle

java-sdk

```xml
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>io.modelcontextprotocol.sdk</groupId>
            <artifactId>mcp-bom</artifactId>
            <version>1.1.1</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
        <groupId>io.modelcontextprotocol.sdk</groupId>
        <artifactId>mcp</artifactId>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.43.0</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>3.6.2</version>
            <!-- executions -->
        </plugin>
    </plugins>
</build>
```

Maven

Case study: Selenium – Java SDK

```java
public class McpSeleniumServer {

    public static void main(String[] args) throws Exception {
        BrowserTools tools = new BrowserTools();
        var transportProvider = new StdioServerTransportProvider(
                McpJsonDefaults.getMapper());

        McpAsyncServer server = McpServer.async(transportProvider)
                .serverInfo("mcp-selenium-server", "1.0.0")
                .capabilities(McpSchema.ServerCapabilities.builder().tools(true)
                        .logging().build())
                .tools(tools.browserStart(), tools.browserNavigate(),
                        tools.readPageText(), tools.browserClose())
                .build();

        CountDownLatch keepAlive = new CountDownLatch(1);

        Runtime.getRuntime().addShutdownHook(new Thread(() -> {
            try {
                server.close();
            } finally {
                keepAlive.countDown();
            }
        }));

        // Block forever until the Java process is terminated
        keepAlive.await();
    }
}
```

java-sdk

Case study: Selenium – Java SDK

```java
public class BrowserTools {

    private final BrowserManager browserManager = new BrowserManager();

    public McpServerFeatures.AsyncToolSpecification browserStart() {
        final String browserNamergument = "browser_name";

        McpSchema.JsonSchema inputSchema = new McpSchema.JsonSchema("object",
                Map.of(browserNamergument, Map.of("type", "string",
                        "description",
                        "The name of the browser to open. Supported values: 'chrome', 'firefox'")),
                List.of(browserNamergument), null, null, null);
        Tool tool = McpSchema.Tool.builder().name("open_browser").description(
                "Launches a new browser instance. Supports Chrome and Firefox browsers")
                .inputSchema(inputSchema).build();

        return McpServerFeatures.AsyncToolSpecification.builder().tool(tool)
                .callHandler((exchange, args) -> {
                    String browserName = (String) args.arguments()
                            .get(browserNamergument);
                    var result = browserManager.start(browserName);
                    return Mono.just(McpSchema.CallToolResult.builder()
                            .addTextContent(result.message())
                            .isError(result.error()).build());
                }).build();
    }

    // ... other tool specifications ...
}
```

java-sdk

Case study: Selenium – Spring AI


```
plugins {
    id 'java'
    id 'org.springframework.boot' version '4.0.5'
    id 'io.spring.dependency-management' version '1.1.7'
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter'
    implementation platform('org.springframework.ai:spring-ai-bom:2.0.0-M4')
    implementation 'org.springframework.ai:spring-ai-starter-mcp-server'
    implementation 'org.springframework:spring-web'
    implementation 'org.seleniumhq.selenium:selenium-java'
}
```

Gradle


```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.ai</groupId>
            <artifactId>spring-ai-bom</artifactId>
            <version>2.0.0-M4</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-starter-mcp-server</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
    </dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Maven

Case study: Selenium – Spring AI


```java
@SpringBootApplication
public class McpSeleniumServer {

    public static void main(String[] args) {
        SpringApplication.run(McpSeleniumServer.class, args);
    }

}
```

application.properties

```
spring.main.web-application-type=none

# Keep stdout clean for MCP stdio
spring.main.banner-mode=off
logging.console.enabled=false

spring.ai.mcp.server.name=selenium-mcp-server
spring.ai.mcp.server.version=1.0.0

logging.file.name=./target/selenium-mcp-server.log
```

Case study: Selenium – Spring AI


```
@Service
public class BrowserService {

    private final BrowserManager browserManager = new BrowserManager();

    @McpTool(name = "open_browser", description = "Launches a new browser instance. Supports Chrome and Firefox browsers")
    public String startBrowser(
            @McpToolParam(description = "The name of the browser to open. Supported values: 'chrome', 'firefox'",
            required = true) String browserName) {
        return browserManager.start(browserName).message();
    }

    @McpTool(name = "navigate_url", description = "Navigate the browser to a specified URL. The browser must be started first")
    public String navigate(
            @McpToolParam(description = "The complete URL to navigate to (e.g., https://example.com)", required = true) String url) {
        return browserManager.navigate(url).message();
    }

    @McpTool(name = "close_browser", description = "Close the browser")
    public String closeBrowser() {
        return browserManager.close().message();
    }

    @McpTool(name = "get_browser_text", description = "Read the visible text of the entire page")
    public String readPageText() {
        return browserManager.readText().message();
    }

}
```

Case study: Selenium – Quarkus

Gradle

```
plugins {
    id 'java'
    id 'io.quarkus' version '3.34.3'
}

dependencies {
    implementation enforcedPlatform("io.quarkus.platform:quarkus-bom:3.34.3")
    implementation "io.quarkiverse.mcp:quarkus-mcp-server-stdio:1.11.1"
    implementation "org.seleniumhq.selenium:selenium-java:4.43.0"
}
```

QUARKUS

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>${quarkus.platform.group-id}</groupId>
            <artifactId>${quarkus.platform.artifact-id}</artifactId>
            <version>${quarkus.platform.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>io.quarkiverse.mcp</groupId>
        <artifactId>quarkus-mcp-server-stdio</artifactId>
        <version>1.11.1</version>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.43.0</version>
    </dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <groupId>io.quarkus.platform</groupId>
            <artifactId>quarkus-maven-plugin</artifactId>
            <version>${quarkus.platform.version}</version>
            <extensions>true</extensions>
            <!-- executions -->
        </plugin>
    </plugins>
</build>
```

Maven

Case study: Selenium – Quarkus


```java
@ApplicationScoped
public class BrowserService {

    private final BrowserManager browserManager = new BrowserManager();

    @Tool(description = "Launches a new browser instance. Supports Chrome and Firefox browsers")
    public String open_browser(
            @ToolArg(description = "The name of the browser to open. Supported values: 'chrome', 'firefox'")
            String browser_name) {
        return browserManager.start(browser_name).message();
    }

    @Tool(description = "Navigate the browser to a specified URL. The browser must be started first")
    public String navigate_url(
            @ToolArg(description = "The complete URL to navigate to (e.g., https://example.com)")
            String url) {
        return browserManager.navigate(url).message();
    }

    @Tool(description = "Close the browser")
    public String close_browser() {
        return browserManager.close().message();
    }

    @Tool(description = "Read the visible text of the entire page")
    public String get_browser_text() {
        return browserManager.readText().message();
    }

}
```

application.properties

```
quarkus.banner.enabled=false
quarkus.log.console.stderr=true
```

Case study: Selenium – Micronaut

```
plugins {
    id 'java'
    id 'application'
    id 'io.micronaut.application' version '4.6.2'
    id 'com.gradleup.shadow' version '9.4.1'
}

application {
    mainClass = 'io.github.bonigarcia.ce.McpSeleniumServer'
}

micronaut {
    version = '4.10.11'
}

dependencies {
    implementation 'io.micronaut.mcp:micronaut-mcp-server-java-sdk'
    implementation 'io.micronaut:micronaut-jackson-databind'
    implementation 'org.seleniumhq.selenium:selenium-java:4.43.0'
    runtimeOnly 'ch.qos.logback:logback-classic'
}
```

```
<parent>
    <groupId>io.micronaut.platform</groupId>
    <artifactId>micronaut-parent</artifactId>
    <version>4.10.11</version>
    <relativePath />
</parent>

<properties>
    <exec.mainClass>io.github.bonigarcia.ce.McpSeleniumServer</exec.mainClass>
</properties>

<dependencies>
    <dependency>
        <groupId>io.micronaut.mcp</groupId>
        <artifactId>micronaut-mcp-server-java-sdk</artifactId>
    </dependency>
    <dependency>
        <groupId>io.micronaut</groupId>
        <artifactId>micronaut-jackson-databind</artifactId>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.43.0</version>
    </dependency>
    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-classic</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>io.micronaut.maven</groupId>
            <artifactId>micronaut-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

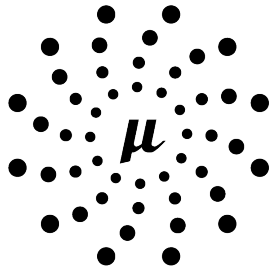
Fork me on GitHub

Case study: Selenium – Micronaut


```java
public class McpSeleniumServer {

    public static void main(String[] args) {
        Micronaut.build(args).mainClass(McpSeleniumServer.class).start();
    }
}
```

application.properties

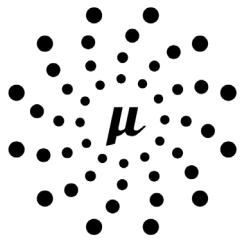
```
micronaut.mcp.server.transport=STDIO
micronaut.mcp.server.info.name=mcp-selenium-server
micronaut.mcp.server.info.version=1.0.0
micronaut.mcp.server.reactive=true
micronaut.banner.enabled=false
micronaut.server.enabled=false
logging.level.io.micronaut=ERROR
logging.level.root=ERROR
```



MICRONAUT®

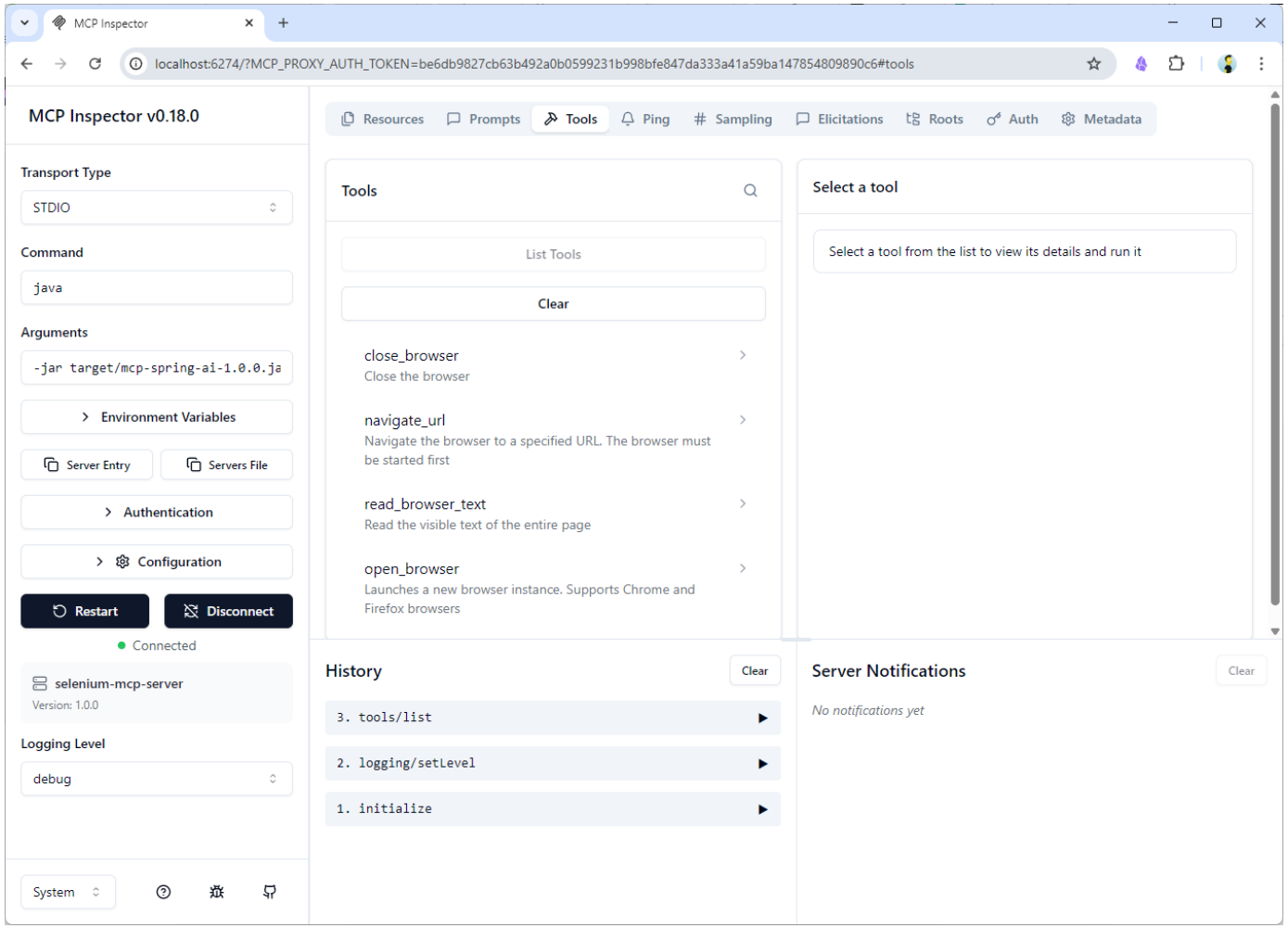
Case study: Selenium – Micronaut


```java
@Singleton
public class BrowserService {

    private final BrowserManager browserManager = new BrowserManager();

    @Tool(description = "Launches a new browser instance. Supports Chrome and Firefox browsers")
    public Mono<String> open_browser(
            @ToolArg(description = "The name of the browser to open. Supported values: 'chrome', 'firefox'") String browser_name) {
        return Mono.just(browserManager.start(browser_name).message());
    }

    @Tool(description = "Navigate the browser to a specified URL. The browser must be started first")
    public Mono<String> navigate_url(
            @ToolArg(description = "The complete URL to navigate to (e.g., https://example.com)") String url) {
        return Mono.just(browserManager.navigate(url).message());
    }

    @Tool(description = "Close the browser")
    public Mono<String> close_browser() {
        return Mono.just(browserManager.close().message());
    }

    @Tool(description = "Read the visible text of the entire page")
    public Mono<String> get_browser_text() {
        return Mono.just(browserManager.readText().message());
    }
}
```



Case study: Selenium – Debugging

```
npx @modelcontextprotocol/inspector java -jar mcp-server.jar
```



Takeaways

- Most direct and portable MCP implementation

- Safest default for mainstream enterprise Java teams

- Strong choice for cloud-native production workloads

- Lightweight option with low runtime overhead

Implementing an MCP Server in Java

Get these slides at:

https://bonigarcia.dev/

Thank you so much!

Boni García
boni.garcia@uc3m.es

Read this story at:

https://medium.com/@boni.gg