



Case Study Title: Employee Info API using Spring Boot AutoConfiguration



Objective:

To build a simple Spring Boot application that exposes an API endpoint to retrieve basic employee information using **Spring Boot AutoConfiguration**. The endpoint will be tested via a browser and Postman using only @GetMapping.



Background:

Spring Boot simplifies application setup with its **AutoConfiguration** feature. Instead of manually defining bean configurations, Spring Boot intelligently guesses what you need and configures it behind the scenes.

This case study helps you understand:

- What AutoConfiguration does.
- How to leverage it using minimal configuration.
- How to expose a basic REST endpoint with @GetMapping.



Components Involved:

1. **Spring Boot Starter Web** – Automatically brings in all dependencies for building REST APIs.
2. **AutoConfiguration** – Behind the scenes, it configures the DispatcherServlet, Tomcat server, and other beans automatically.
3. **REST Controller** – A simple Java class using @RestController and @GetMapping.
4. **Browser/Postman** – For testing the GET API.



Scenario:

You are a developer working in the HR software team. Your task is to expose employee information (like name, ID, and department) through a simple HTTP GET API without manually configuring any server, servlet, or web.xml file.



Steps in the Case Study:

1. Create the Spring Boot Project

- Use Spring Initializr (<https://start.spring.io/>)
- Project metadata:
 - Group: com.company ◦ Artifact: employee-api
 - Dependencies:
 - Dependencies:
 - Spring Web

2. Directory Structure AutoCreated by Spring Boot

Spring Boot automatically generates the following:

src/

```
main/  java/  
      com.company.employeeapi/  
          EmployeeApiApplication.java    controller/  
          EmployeeController.java    resources/  
          application.properties
```

3. Understanding AutoConfiguration

- No need to configure DispatcherServlet, JSON converter, or server port.
- When you add spring-boot-starter-web, it:
 - Configures embedded Tomcat server.

- Registers Jackson for JSON conversion.
- Sets up DispatcherServlet for handling REST requests.
- Starts server on port 8080.

4. Creating a Simple GET Endpoint

- The @RestController and @GetMapping("/employee") annotations automatically expose a REST endpoint due to AutoConfiguration.

5. Running the Application

- Just run the main class EmployeeApiApplication.java.
- Spring Boot auto-starts the embedded server and makes the endpoint live.

6. Testing the API

Open browser or Postman.

Hit: <http://localhost:8080/employee>

Expected JSON output:

```
{
  "id": 101,
  "name": "John Doe", "department": "Engineering"
}
```

EmployeeApiApplication.java:

```
package com.company.employeeapi;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class EmployeeApiApplication {
    public static void main(String[] args) {
        SpringApplication.run(EmployeeApiApplication.class, args);
    }
}
```

EmployeeController.java:

```
package com.company.employeeapi.controller;
import com.company.employeeapi.model.Employee;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class EmployeeController {
    @GetMapping("/employee")
    public Employee getEmployeeInfo() {
        return new Employee(101, "John Doe", "Engineering");
    }
}
```

Employee.java:

```
package com.company.employeeapi.model;
public class Employee {
    private int id;
    private String name;
    private String department;
```

```

public Employee() {
}
public Employee(int id, String name, String department) {
    this.id = id;
    this.name = name;
    this.department = department;
}
public int getId() { return id; }
public void setId(int id) { this.id = id; }
public String getName() { return name; }
public void setName(String name) { this.name = name; }
public String getDepartment() { return department; }
public void setDepartment(String department) { this.department = department; }
}

```

Application.properties:

```

spring.application.name=employee-api
server.port=8080

```

pom.xml:

```

<?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
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.5.4</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.company</groupId>
    <artifactId>employee-api</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>employee-api</name>
    <description>Company project for Spring Boot</description>
    <url/>
    <licenses>
        <license/>
    </licenses>
    <developers>
        <developer/>
    </developers>
    <scm>
        <connection/>
        <developerConnection/>
    </scm>

```

```

<tag/>
<url/>
</scm>
<properties>
    <java.version>21</java.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

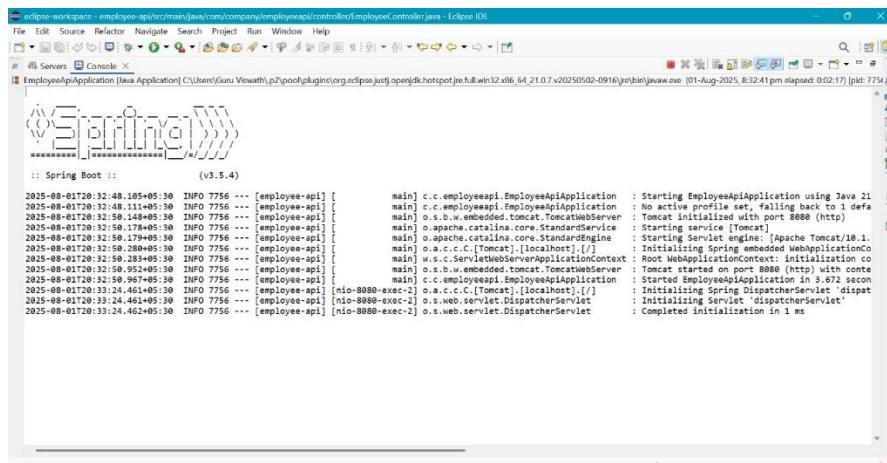
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>

```

</project>

Outputs:



```

eclipse-workspace - employee-api/src/main/java/com/company/employeeapi/controller/EmployeeController.java - Eclipse IDE
File Edit Source Refactor Search Project Run Window Help
Servers Console X
EmployeeApiApplication [Java Application] C:\Users\Guru Viswath\p2\pool\plugins\org.eclipse.jdt.core\openjdk-hotspot.jdt.win32.x64_21.0.7.v20250502-0916\jre\bin\javaw.exe [01-Aug-2025, 8:32:41 pm elapsed: 0:02:17] [pid: 775]
:: Spring Boot :: (v3.5.4)
2025-08-01T20:32:48.185+05:30 INFO 7756 --- [employee-api] [main] c.c.employeeapi.EmployeeApiController : Starting EmployeeApiApplication using Java 21
2025-08-01T20:32:48.185+05:30 INFO 7756 --- [employee-api] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : No active profile set, falling back to 1 defa
2025-08-01T20:32:50.148+05:30 INFO 7756 --- [employee-api] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port: 8080 (http)
2025-08-01T20:32:50.178+05:30 INFO 7756 --- [employee-api] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-08-01T20:32:50.180+05:30 INFO 7756 --- [employee-api] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.
2025-08-01T20:32:50.280+05:30 INFO 7756 --- [employee-api] [main] o.s.c.C.[Tomcat].[localhost].[/] : Starting Spring embedded WebApplicationCo
2025-08-01T20:32:50.283+05:30 INFO 7756 --- [employee-api] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialized on co
2025-08-01T20:32:50.952+05:30 INFO 7756 --- [employee-api] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port: 8080 (http) with conte
2025-08-01T20:32:50.953+05:30 INFO 7756 --- [employee-api] [nio-8080-exec-2] o.a.c.c.[Tomcat].[localhost].[/] : Started EmployeeApiApplication in 1.67 secon
2025-08-01T20:33:24.461+05:30 INFO 7756 --- [employee-api] [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2025-08-01T20:33:24.461+05:30 INFO 7756 --- [employee-api] [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
2025-08-01T20:33:24.462+05:30 INFO 7756 --- [employee-api] [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet :

```

The screenshot shows the Postman interface with a history of requests. A specific GET request to `http://localhost:8080/employee` is selected. The response is a JSON object:

```

1  {
2   "id": 101,
3   "name": "John Doe",
4   "department": "Engineering"
5 }

```



2. Spring Boot – Actuators

Case Study: Monitoring an Inventory System

Problem Statement:

You deploy an Inventory Management app and want to **monitor** its health, memory usage, bean loading, and environment settings without building these endpoints manually.



Key Concept:

Spring Boot Actuator exposes production-ready features like health checks, metrics, beans, and custom endpoints.



Scenario:

You add the `spring-boot-starter-actuator` dependency, and enable the `/actuator` endpoint in `application.properties`.

With zero code changes, you get:

- `/actuator/health` → Health of the service.
- `/actuator/beans` → Beans created in the container.
- `/actuator/metrics` → JVM and HTTP metrics.
- `/actuator/env` → Current environment values.

InventoryMonitoringApplication.java:

```

package com.company.inventorymonitoring;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class InventoryMonitoringApplication {
    public static void main(String[] args) {
        SpringApplication.run(InventoryMonitoringApplication.class, args);
    }
}

```

application.properties:

```

spring.application.name=inventory-monitoring
management.endpoints.web.exposure.include=*
server.port=8082

```

InventoryMonitoringApplicationTests.java:

```
package com.company.inventory_monitoring;
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
@SpringBootTest
class InventoryMonitoringApplicationTests {
    @Test
    void contextLoads() {
    }
}
```

Pom.xml:

```
<?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
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.5.4</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.company</groupId>
    <artifactId>inventory-monitoring</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>inventory-monitoring</name>
    <description>Inventory monitoring project for Spring Boot</description>
    <url/>
    <licenses>
        <license/>
    </licenses>
    <developers>
        <developer/>
    </developers>
    <scm>
        <connection/>
        <developerConnection/>
        <tag/>
        <url/>
    </scm>
    <properties>
        <java.version>21</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
```

```

</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>

```

</project>

Outputs:

```

:: Spring Boot :: (v3.5.4)
2025-08-01T20:44:33.868+05:30 INFO 23424 --- [inventory-monitoring] main c.c.i.InventoryMonitoringApplication : Starting InventoryMonitoringApplication
2025-08-01T20:44:33.873+05:30 INFO 23424 --- [inventory-monitoring] main o.s.w.e.e.tomcat.TomcatWebServer : No active profile set, falling back to Tomcat's internal 'tomcat' profile
2025-08-01T20:44:33.873+05:30 INFO 23424 --- [inventory-monitoring] main o.apache.catalina.core.StandardService : Tomcat initialized [Tomcat]
2025-08-01T20:44:35.749+05:30 INFO 23424 --- [inventory-monitoring] main o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat /]
2025-08-01T20:44:35.805+05:30 INFO 23424 --- [inventory-monitoring] main o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebAplica
2025-08-01T20:44:35.806+05:30 INFO 23424 --- [inventory-monitoring] main w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initializat
2025-08-01T20:44:35.827+05:30 INFO 23424 --- [inventory-monitoring] main o.s.w.e.e.EmbeddedLinksResolver : Exposing 13 endpoints beneath base path
2025-08-01T20:44:36.831+05:30 INFO 23424 --- [inventory-monitoring] main o.s.w.e.e.EmbeddedTomcatWebServer : Exposing 13 endpoints beneath base path
2025-08-01T20:44:36.831+05:30 INFO 23424 --- [inventory-monitoring] main c.c.i.InventoryMonitoringApplication : Started InventoryMonitoringApplication
2025-08-01T20:45:19.756+05:30 INFO 23424 --- [inventory-monitoring] [nio-8082-exec-3] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet
2025-08-01T20:45:19.757+05:30 INFO 23424 --- [inventory-monitoring] [nio-8082-exec-3] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2025-08-01T20:45:19.761+05:30 INFO 23424 --- [inventory-monitoring] [nio-8082-exec-3] o.s.web.servlet.DispatcherServlet : Completed initialization in 3 ms

```

You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.

History

New Import GET http://localhost:8082/actuator/health + ***

Today

GET http://localhost:8082/actuator/health
GET http://localhost:8082/actuator/health
GET http://localhost:8080/employee
GET http://localhost:8080/manage
GET http://localhost:8085/manage
GET http://localhost:8085/products
GET http://localhost:9090/products
GET http://localhost:8082/products
GET http://localhost:8082/products
GET http://localhost:8080/products

http://localhost:8082/actuator/health

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies Bulk Edit

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1
2 "status": "UP"
3

```

Create collections in Postman
Use collections to save your requests and share them with others.

Create a Collection

You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.

History

New Import

GET http://localhost:8082/actuator/beans

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "contexts": [
3     "inventory-monitoring": {
4       "beans": [
5         "endpointsInCachingOperationInvokerAdvisor": {
6           "aliases": [],
7           "scope": "singleton",
8           "type": "org.springframework.boot.actuate.endpoint.invoke.advisor.CachingOperationInvocationAdvisor",
9           "resource": "class path resource [org/springframework/boot/actuate/autoconfigure/endpoint/EndpointAutoConfiguration.class]",
10          "dependencies": [
11            "org.springframework.boot.actuate.autoconfigure.endpoint.EndpointAutoConfiguration",
12            "environment"
13          ]
14        }
15      }
16    }
17  ]
18 }
```

Create collections in Postman

Use collections to save your requests and share them with others.

Create a Collection

Console Not connected to a Postman account

You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.

History

New Import

GET http://localhost:8082/actuator/metrics

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "names": [
3     "application.ready.time",
4     "application.started.time",
5     "disk.free",
6     "disk.total",
7     "executor.active",
8     "executor.completed",
9     "executor.pool.core",
10    "executor.pool.max",
11    "executor.pool.size",
12    "executor.queue.remaining",
13    "executor.queue",
14    "http.server.requests",
15    "http.server.requests.active",
16    "jvm.bytes.count",
17    "jvm.bytes.read",
18    "jvm.bytes.write"
19  ]
20 }
```

Create collections in Postman

Use collections to save your requests and share them with others.

Create a Collection

Console Not connected to a Postman account

You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.

History

New Import

GET http://localhost:8082/actuator/env

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "activeProfiles": [],
3   "defaultProfiles": [
4     "default"
5   ],
6   "propertySources": [
7     {
8       "name": "server.ports",
9       "properties": [
10         "local.server.port": {
11           "value": "*****"
12         }
13       ]
14     },
15     {
16       "name": "servletContextInitParams",
17     }
18   ]
19 }
```

Create collections in Postman

Use collections to save your requests and share them with others.

Create a Collection

Console Not connected to a Postman account