Spring Cloud Function – 2025

Spring Cloud Function is a framework that simplifies building serverless functions within the Spring ecosystem. In essence, Spring Cloud Function enables developers to write business logic as functions and deploy them in a serverless manner, leveraging the benefits of the Spring ecosystem.

* Functional Programming: **Writing business logic as functions**, Supports different programming styles (reactive, imperative, or hybrid).
* Serverless Deployment: Provides a **unified programming model for different serverless providers**. Supports deploying functions as standalone applications or within a PaaS.
* Adapters for Exposure/Decoupled Runtime: **Functions can be exposed as HTTP endpoints** or can run as web endpoints, stream processors, or tasks without being tied to a specific runtime.

Spring Cloud Function embraces 3 core functional interfaces.

* Supplier<O>
* Function<I, O>
* Consumer<I>

Spring Cloud Function has the following features:

* Choice of programming styles - reactive, imperative or hybrid.
* Function composition and adaptation (e.g., composing imperative functions with reactive).
* Transparent type conversion of inputs and outputs.
* Adapters for AWS Lambda, Azure, Google Cloud Functions, and possibly other "serverless" service providers.

| **Method** | **Path** | **Request** | **Response** | **Status** |
| --- | --- | --- | --- | --- |
| GET | /{supplier} | - | Items from the named supplier | 200 OK |
| POST | /{consumer} | JSON object or text | Mirrors input and pushes request body into consumer | 202 Accepted |
| PUT | /{consumer} | JSON object or text | Mirrors input and pushes request body into consumer | 202 Accepted |
| DELETE | /{consumer} | JSON object or text | - | 204 NO CONTENT |
| POST | /{function} | JSON object or text | The result of applying the named function | 200 OK |
| PUT | /{function} | JSON object or text | The result of applying the named function | 200 OK |
| GET | /{function}/{item} | - | Convert the item into an object and return the result of applying the function | 200 OK |

**Complete Example on Spring Cloud Function**

Relevant pom.xml

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.4.2</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<java.version>17</java.version>

<spring-cloud.version>2024.0.0</spring-cloud.version>

</properties>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-function-web</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-function-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-function-context</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-configuration-processor</artifactId>

<optional>true</optional>

</dependency>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

**application.properties**

**# Multiple packages can be provided with comma(,)**

spring.cloud.function.scan.packages=com.ddlab.rnd.basic.fn,com.ddlab.rnd.core.fn;com.ddlab.rnd.more.fn

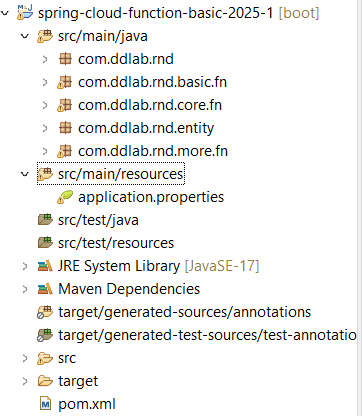
**# Below is the context path**

spring.cloud.function.web.path=/fns

**# Not required definitions should be mentioned here, this should not be called**

spring.cloud.function.ineligible-definitions=employeeSalaryInfo

**Project Structure**



**Main class**

@SpringBootApplication

**public** **class** BasicFunctionApp {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(BasicFunctionApp.**class**, args);

}

}

Basic CRUD Operations are given below.

**Create Employee (Package: com.ddlab.rnd.basic.fn)**

/\*\*

\* **POST http://localhost:8080/fns/empCreation**

\* Below is the message body

\* {

"id": 123,

"name": "John Abraham"

}

Response: 202 Accepted

\*/

**public** **class** EmpCreation **implements** **Consumer**<Employee> {

@Override

**public** **void** **accept**(Employee e) {

System.***out***.println("Employee info has been saved in the system:"+e);

}

}

**Read Employee Information**

/\*\*

\* **GET http://localhost:8080/fns/employeeInfo/23**

\*

\* Response: 200

\* Response Body:

\* {

"id": 23,

"name": "John"

}

\*/

**public** **class** EmployeeInfo **implements** Function<Integer, Employee> {

@Override

**public** Employee apply(Integer t) {

**return** **new** Employee(t.intValue(), "John");

}

}

**Update Employee Information**

/\*\*

\* **PUT OR POST http://localhost:8080/fns/empUpdation**

\* Below is the message body

\* {

"id": 123,

"name": "Vidya Balan"

}

Response: 200

Response Body: Info updated successfully ...

\*/

**public** **class** EmpUpdation **implements** Function<Employee, String> {

@Override

**public** String apply(Employee e) {

System.***out***.println("Incoming emp info: "+e);

System.***out***.println("Employee info has been updated in the system");

**return** "Info updated successfully ...";

}

}

**Delete Employee Information**

/\*\*

\* **'DELETE' can only be mapped to Consumer**

\* There can not be Request Body for HTTP DELETE

\* **DELETE http://localhost:8080/fns/empRemovalById/233**

\*/

**public** **class** EmpRemovalById **implements** Consumer<Integer> {

@Override

**public** **void** accept(Integer e) {

System.***out***.println("Incoming Emp Id: "+e);

System.***out***.println("Employee info deleted successfully ...");

}

}

The above can also be written in the below format.

@Component  
public class CoreFunctions {  
  
 */\*\*  
 \* POST or PUT http://localhost:8080/fns/consumeEmp  
 \* Below is the message body  
 \* {  
 "id": 123,  
 "name": "John Abraham"  
 }  
 Response: 202 Accepted  
 \*/* @Bean  
 public Consumer<Employee> **consumeEmp**() {  
 return emp -> System.*out*.println("Employee: " + emp);  
 }  
  
 */\*\*  
 \* GET http://localhost:8080/fns/getEmpInfo/23  
 \* <p>  
 \* Response: 200  
 \* Response Body:  
 \* {  
 \* "id": 23,  
 \* "name": "John"  
 \* }  
 \*/* @Bean  
 public Function<Integer, Employee> **getEmpInfo**() {  
 return id -> new Employee(id, "John");  
 }  
  
 */\*\*  
 \* GET http://localhost:8080/fns/getAppVersion  
 \* <p>  
 \* Response: 200  
 \* Response Body: Version: 1.0.0  
 \*/* @Bean  
 public Supplier<String> **getAppVersion**() {  
 return () -> "Version: 1.0.0";  
 }  
  
 */\*\*  
 \* DELETE http://localhost:8080/fns/removeEmpById/23  
 \* <p>  
 \* Response: 200  
 \* Response Body: Version: 1.0.0  
 \*/* @Bean  
 public Consumer<String> **removeEmpById**() {  
 return val -> System.*out*.println("Emp Id: " + val+" has been removed");  
 }

*/\*\*  
 \* PUT or POST http://localhost:8080/fns/updateEmp  
 \* <p>  
 \* Below is the message body  
 \* {  
 \* "id": 123,  
 \* "name": "Vidya Balan"  
 \* }  
 \* Response: 200  
 \* Response Body: Info updated successfully ...  
 \*  
 \* @return  
 \*/* @Bean  
 public Function<Employee, String> **updateEmp**() {  
 return emp -> {  
 System.*out*.println("Incoming emp info: " + emp);  
 System.*out*.println("Employee info has been updated in the system");  
 return "Info updated successfully ...";  
 };  
 }  
}

**Exclusion of functions**

You have to use the following property in application.properties.

**spring.cloud.function.ineligible-definitions=employeeSalaryInfo**

Code is given below.

/\*\*

\* **GET http://localhost:8080/fns/employeeSalaryInfo/John**

Response: 202 Accepted

"status": 404,

"error": "Not Found",

\*/

**public** **class** EmployeeSalaryInfo **implements** Function<String, EmpSalary> {

@Override

**public** EmpSalary **apply**(String name) {

System.***out***.println("Employee Name: "+name);

**return** **new** EmpSalary(123, name, 1234);

}

}

**How to use Supplier**

*\*\*  
 \* GET http://localhost:8080/fns/getAppVersion  
 \* <p>  
 \* Response: 200  
 \* Response Body: Version: 1.0.0  
 \*/*@Bean  
public Supplier<String> getAppVersion() {  
 return () -> "Version: 1.0.0";  
}

**Input/Output Message Enrichment**

@Component

**public** **class** Enrichment {

*/\*\*  
 \* GET http://localhost:8080/fns/getEnrichEmpInfoById/23  
 \* Response: 200  
 \* Response Body: Given below  
 \* {  
 \* "id": 34,  
 \* "name": "John Abraham"  
 \* }  
 \* Response Header:  
 \* bar: 456  
 \* foo: 123  
 \*/*@Bean  
public Function<Message<Integer>, Message<Employee>> **getEnrichEmpInfoById**() {  
 return empId -> {  
 Employee emp = new Employee(empId.getPayload(), "John Abraham");  
 return **MessageBuilder.*withPayload*(emp)  
 .setHeader("foo", 123)  
 .setHeader("bar", 456)  
 .build();**  
 };  
}

*/\*\*  
 \* GET* ***http://localhost:8080/fns/getEnrichEmpInfoById2/23*** *\* Response: 200  
 \*  
 \* @return  
 \*/*@Bean  
public Function<Integer, Message<Employee>> g**etEnrichEmpInfoById2**() {  
 return empId -> {  
 Employee emp = new Employee(empId, "John Abraham");  
 return **MessageBuilder.*withPayload*(emp)  
 .setHeader("foo", 123)  
 .setHeader("bar", 456).  
 build();** };  
}

/\*\*

\* PUT http://localhost:8080/fns/getSome

\* Response: 200

\* Response Body: Hello Mr Sam

\* Response Header:

\* bar: 456

\* foo: 123

\*/

@Bean

**public** Supplier<Message<String>> getSome() {

**return** () -> MessageBuilder.*withPayload*("Hello Mr Sam")

.setHeader("foo", 123)

.setHeader("bar", 456)

.build();

// return () -> MessageBuilder.withPayload("Hello Mr").build();

}

/\*\*

\* PUT http://localhost:8080/fns/consumeSome

\* Request Body:

\* {

"id": 123,

"name": "Vidya Balan"

}

\* Response: 202

\* Response Header:

\* bar: 456

\* foo: 123

\*/

@Bean

**public** Consumer<Message<Employee>> consumeSome() {

**return** msg -> {

System.***out***.println("Some Randome Message: " + msg);

System.***out***.println("Message Payload---->" + msg.getPayload());

};

}

}

**Conversion Pattern**

**POST http://localhost:8080/fns/empCreation**

public class EmpCreation implements Consumer<Employee> {  
 @Override  
 public void accept(Employee e) {  
 System.*out*.println("Employee info has been saved in the system:"+e);  
 }  
}

Can be written as below

**@Bean** 🡺 **POST or PUT http://localhost:8080/fns/consumeEmp**  
public Consumer<Employee> consumeEmp() {  
 return emp -> System.*out*.println("Employee: " + emp);  
}

**GET http://localhost:8080/fns/getEmpInfo/23**

public class EmployeeInfo implements Function<Integer, Employee> {  
 @Override  
 public Employee apply(Integer t) {  
 return new Employee(t.intValue(), "John");  
 }  
}

Can be written as below

**@Bean** 🡺 **GET http://localhost:8080/fns/getEmpInfo/23**  
public Function<Integer, Employee> getEmpInfo() {  
 return id -> new Employee(id, "John");  
}

**DELETE http://localhost:8080/fns/empRemovalById/233**

public class EmpRemovalById implements Consumer<Integer> {  
 @Override  
 public void accept(Integer e) {  
 System.*out*.println("Incoming Emp Id: "+e);  
 System.*out*.println("Employee info deleted successfully ...");  
 }  
}

Can be written as below

**@Bean** 🡺 **DELETE http://localhost:8080/fns/removeEmpById/23**public Consumer<String> removeEmpById() {  
 return val -> System.*out*.println("Emp Id: " + val+" has been removed");  
}

**PUT OR POST http://localhost:8080/fns/empUpdation**

public class EmpUpdation implements Function<Employee, String> {  
  
 @Override  
 public String apply(Employee e) {  
 System.*out*.println("Incoming emp infor: "+e);  
 System.*out*.println("Employee info has been updated in the system");  
 return "Info updated successfully ...";  
 }  
}

Can be written as below

**@Bean 🡺 PUT or POST http://localhost:8080/fns/updateEmp**  
public Function<Employee, String> updateEmp() {  
 return emp -> {  
 System.*out*.println("Incoming emp info: " + emp);  
 System.*out*.println("Employee info has been updated in the system");  
 return "Info updated successfully ...";  
 };  
}

**DELETE http://localhost:8080/fns/empRemovalById/233**

public class EmpRemovalById implements Consumer<Integer> {  
 @Override  
 public void accept(Integer e) {  
 System.*out*.println("Incoming Emp Id: "+e);  
 System.*out*.println("Employee info deleted successfully ...");  
 }  
}

Can be written as below

**@Bean 🡺 DELETE http://localhost:8080/fns/removeEmpById/23**  
public Consumer<String> removeEmpById() {  
 return val -> System.*out*.println("Emp Id: " + val+" has been removed");  
}

--spring.cloud.function.definition=foo;bar This will only export function foo and function bar regardless how many functions are available in catalog (e.g., localhost:8080/foo).

--spring.cloud.function.definition=foo|bar;baz This will only export function composition foo|bar and function baz regardless how many functions are available in catalog (e.g., localhost:8080/foo,bar).

CRUD REST with Spring Cloud Function By now it should be clear that functions are exported as REST endpoints and can be invoked using various HTTP methods. In other words a single function could be triggered via GET, POST, PUT etc.

However, it is not always desirable and certainly does not fit the CRUD concept. And while SCF does not support and has no intention of supporting all the features of Spring web stack, the framework does provide support for CRUD mappings where a single function could be mapped to a particular HTTP method(s). It is done via spring.cloud.function.http. property.

For example,

spring.cloud.function.http.GET=uppercase;reverse;foo|bar spring.cloud.function.http.POST=reverse spring.cloud.function.http.DELETE=deleteById