JSON → CAMT Converter (Apache Camel + Spring Boot)

This repository contains a **runnable** example project that accepts a JSON payload and a template ID, applies a DB-driven mapping, and **returns a CAMT-style ISO20022 XML** (simplified `camt.053` structure). It uses **Apache Camel** for the REST endpoint and routing, Spring Boot, and PostgreSQL (docker-compose included) for template storage.

> **Note:** This is a pragmatic, working implementation for typical banking payloads (account + transactions). ISO20022 is vast and has many versions; the builder here produces a minimal, valid CAMT-like XML that you can extend to match a full ISO schema if required.

What you get

- Spring Boot + Apache Camel REST endpoint `POST /transform/{templateId}` (accepts JSON, responds with CAMT XML)
- Templates stored in Postgres (`transformation_templates` table). A sample template is pre-seeded.
- Mapping logic supports JSONPath -> target keys (including array mapping like `$.transactions[*].amount` → `transactions[].amount`).
- A small CAMT XML builder that produces `Document` → `BkToCstmrStmt` → `Stmt` → `Acct` + `Bal` + `Ntry` (one Ntry per transaction).
- Docker Compose file to run Postgres.

Prerequisites

- Java 17+
- Maven 3.6+
- Docker & docker-compose (for Postgres)

Quick start

1. Build the JAR:

```
mvn -U -DskipTests package
```

1. Start Postgres (docker):

```
docker-compose up -d db
# wait a few seconds for DB to initialize
```

1. Run the app (or build + run via docker-compose if you add the app service):

```
java -jar target/json-to-camt-0.0.1-SNAPSHOT.jar
```

1. Test with sample input (provided in `sample-input.json`):

```
curl -X POST "http://localhost:8080/transform/1" \
  -H "Content-Type: application/json" \
  --data @src/main/resources/sample-input.json
```

You should get a CAMT XML response.

Project file tree (important files)

```
json-to-camt-camel-project/
├─ pom.xml
├─ Dockerfile
├─ docker-compose.yml
├─ README.md
├─ src/
│  ├─ main/
│  │  ├─ java/com/example/camt/
│  │  │  ├─ Application.java
│  │  │  ├─ route/TransformRoute.java
│  │  │  ├─ model/TransformationTemplate.java
│  │  │  ├─ repository/TemplateRepository.java
│  │  │  ├─ service/TemplateService.java
│  │  │  ├─ service/TransformationService.java
│  │  │  └─ util/CamtXmlBuilder.java
│  │  └─ resources/
│  │     ├─ application.properties
│  │     ├─ data.sql
│  │     └─ sample-input.json
```

Key code files (copy-ready)

`pom.xml`

```
<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>
    <groupId>com.example</groupId>
    <artifactId>json-to-camt</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.0</version>
        <relativePath/>
    </parent>

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

    <dependencies>
        <!-- Spring Boot + JPA -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
        </dependency>

        <!-- Apache Camel (Spring Boot) -->
        <dependency>
            <groupId>org.apache.camel.springboot</groupId>
            <artifactId>camel-spring-boot-starter</artifactId>
        </dependency>
        <dependency>
```

```
            <groupId>org.apache.camel.springboot</groupId>
            <artifactId>camel-servlet-starter</artifactId>
        </dependency>

        <!-- JSONPath for mapping extraction -->
        <dependency>
            <groupId>com.jayway.jsonpath</groupId>
            <artifactId>json-path</artifactId>
            <version>2.8.0</version>
        </dependency>

        <!-- Jackson -->
        <dependency>
          <groupId>com.fasterxml.jackson.core</groupId>
          <artifactId>jackson-databind</artifactId>
        </dependency>

        <!-- Tests (optional) -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>${java.version}</source>
                    <target>${java.version}</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

`src/main/java/com/example/camt/Application.java`

```
package com.example.camt;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

`src/main/java/com/example/camt/model/TransformationTemplate.java`

```
package com.example.camt.model;

import jakarta.persistence.*;

@Entity
@Table(name = "transformation_templates")
public class TransformationTemplate {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String templateName;

    @Column(columnDefinition = "text")
    private String description;

    @Lob
    @Column(columnDefinition = "text")
    private String mappingLogic; // JSON string with mapping rules

    private String inputFormat;
    private String outputFormat;

    // getters & setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getTemplateName() { return templateName; }
    public void setTemplateName(String templateName) { this.templateName =
```

```
templateName; }
    public String getDescription() { return description; }
    public void setDescription(String description) { this.description =
description; }
    public String getMappingLogic() { return mappingLogic; }
    public void setMappingLogic(String mappingLogic) { this.mappingLogic =
mappingLogic; }
    public String getInputFormat() { return inputFormat; }
    public void setInputFormat(String inputFormat) { this.inputFormat =
inputFormat; }
    public String getOutputFormat() { return outputFormat; }
    public void setOutputFormat(String outputFormat) { this.outputFormat =
outputFormat; }
}
```

---

src/main/java/com/example/camt/repository/TemplateRepository.java

```
package com.example.camt.repository;

import com.example.camt.model.TransformationTemplate;
import org.springframework.data.jpa.repository.JpaRepository;

public interface TemplateRepository extends
JpaRepository<TransformationTemplate, Long> {
}
```

---

src/main/java/com/example/camt/service/TemplateService.java

```
package com.example.camt.service;

import com.example.camt.model.TransformationTemplate;
import com.example.camt.repository.TemplateRepository;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class TemplateService {
    private final TemplateRepository repo;
    public TemplateService(TemplateRepository repo) { this.repo = repo; }

    public Optional<TransformationTemplate> getTemplate(Long id) {
```

```
        return repo.findById(id);
    }
}
```

---

src/main/java/com/example/camt/route/TransformRoute.java

```
package com.example.camt.route;

import com.example.camt.model.TransformationTemplate;
import com.example.camt.service.TemplateService;
import com.example.camt.service.TransformationService;
import org.apache.camel.Exchange;
import org.apache.camel.builder.RouteBuilder;
import org.apache.camel.model.rest.RestBindingMode;
import org.springframework.stereotype.Component;

@Component
public class TransformRoute extends RouteBuilder {

    private final TemplateService templateService;
    private final TransformationService transformationService;

    public TransformRoute(TemplateService templateService,
TransformationService transformationService) {
        this.templateService = templateService;
        this.transformationService = transformationService;
    }

    @Override
    public void configure() throws Exception {

restConfiguration().component("servlet").contextPath("/").bindingMode(RestBindingMode.off);

        onException(Exception.class)
            .handled(true)
            .setHeader(Exchange.HTTP_RESPONSE_CODE, constant(400))
            .setBody().simple("{\"error\":\"${exception.message}\"}");

        rest("/transform")
            .post("/{templateId}")
            .consumes("application/json")
            .produces("application/xml")
            .route()
            .process(exchange -> {
                String tmpl = exchange.getIn().getHeader("templateId",
```

```
String.class);
                Long templateId = Long.valueOf(tmpl);

                TransformationTemplate template =
templateService.getTemplate(templateId)
                    .orElseThrow(() -> new IllegalArgumentException("Invalid
template id: " + templateId));

                String json = exchange.getIn().getBody(String.class);
                String camtXml =
transformationService.transformJsonToCamt(json, template);

                exchange.getMessage().setBody(camtXml);
                exchange.getMessage().setHeader(Exchange.CONTENT_TYPE,
"application/xml");
            });
    }
}
```

`src/main/java/com/example/camt/service/TransformationService.java`

```
package com.example.camt.service;

import com.example.camt.model.TransformationTemplate;
import com.example.camt.util.CamtXmlBuilder;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.jayway.jsonpath.JsonPath;
import org.springframework.stereotype.Service;

import java.util.*;

@Service
public class TransformationService {
    priva
```