# Members micro-service description

## App structure

Application structure is generated by “Spring initializr” as maven project for spring-boot.
Release is built by maven  (mvnw package)

## Database

entity: MEMBERS table

long ID
String firstName
String lastName
Sting postalCode
Date birthDate

DBMS to use:
H2 database (configured by adding maven dependency)

## Layers

| Layer | Description | Package |
|---|---|---|
| Model | entity objects | org.tc.demo.model |
| Repository | JPA repository to access entity objects | org.tc.demo.repo |
| Dto | data transfer object to (de)serialize data | org.tc.demo.dto |
| Service | Service for high level data access | org.tc.demo.services |
| Controller | REST controller | org.tc.demo.controllers |
| Configuration | Database configuration | org.tc.demo.config |
| Utilities | Utilities and exception classes | org.tc.demo.utils |

## Implementation

There are specific configurations inheriting abstract database configurations.
The “test” profile activates testing database configuration and the “server” profile activates production configuration. In this simple example it is not actually needed but in real application it would allow to configure differently for production and tests: use different property files, or to only scan subset of of packages for beans and configurations.

The errors are reported by throwing an exception declaring http error status via annotation.

## Testing strategy

- The implementation followed TDD principle.
- The implementation was done by layer (first services then controllers)
- The methods of tested classes were declared with body throwing common lang NotImplementedException
- The unit tests were written for each tested method, method by method. The implementation code was added to fix the negative test results until test describing desired functionality was complete and positive.
- The controllers are tested using Mockito mocking the services layer, to be real unit tests and not integration tests.
- Controllers test both XML and JSON service output (“accept” header) and input format.
- Controller tests expect the http error status code according to thrown exception
- A single integration test requests list of members and expects http ok to see if the server is up and configured correctly for json consumer client.

## Server

Default is Tomcat configured by the Spring configuration. Jetty is possible to configure by adding a maven dependency

Formats: JSON and XML. JSON is supported by Spring out-of-box, XML added by adding jackson-dataformat-xml maven dependency.