**Henry Schein One SET Test**

Given that a large part of what we do is API based, test automation is a key part of our SDLC. This application primarily focusses on RESTful APIs working with a Person object. Our GET, POST, and DELETE allow you to test your automation skills against an h2 DB.

**Henry Schein One is focused on securing all API’s, and maintaining quality data in our databases. The goal of this test is to verify if our application is ready to be deployed to production.**

**Getting Started:**

Before testing 3 endpoints against our application it will require the following prerequisites:

* Java – This is to run the jar file and test automation code against application
* Jar File – Runs the application code locally

**Setup:**

**Java**

--Note-- Skip Java setup if you already have a JDK installed

* Open browser to <https://adoptopenjdk.net/>
* Download OpenJDK 11 (LTS) with HotSpot JVM
  + Follow installation instructions if needed <https://adoptopenjdk.net/installation.html>

**Jar File**

* Download hstest-0.0.1-SNAPSHOT.jar from <https://github.com/audiomojo/testAutomation/blob/master/target/hstest-0.0.1-SNAPSHOT.jar>
* Open a terminal or CMD window
* Change directory to location jar file is located

--Example jar located in Downloads directory



* Run command “java -jar hstest-0.0.1-SNAPSHOT.jar --server.port=8083”

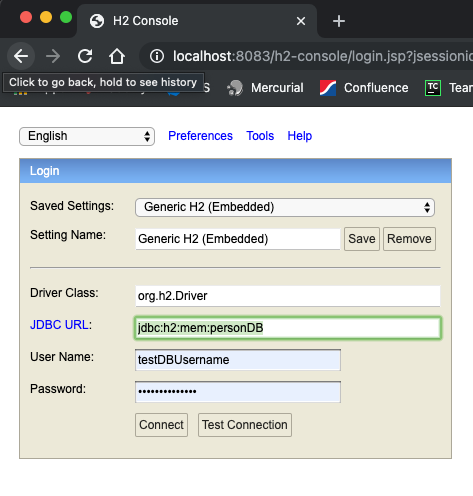


Database:

**H2 Database**

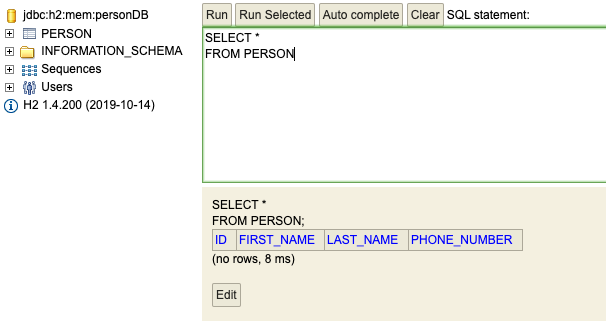
How to access DB from Brower:

* + Open Browser to <http://localhost:8083/h2-console>
  + Change JDBC URL to “jdbc:h2:mem:personDB”
  + Change User Name to “testDBUsername”
  + Change Password to “testDBPassword”



Run SQL commands from the Person Table to verify data is correctly inserted, and deleted

**Example:**



**How to access H2 DB from code in Java:**

* Below code sample is using the java classes provided in java.sql package.
* Candidate can use any library to setup database connection for testing, given that proper instruction/jar-setup should be provided to run tests if necessary.

import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
  
public class SQLDatabaseConnection *{* // Connect to your database.  
 // Replace server name, username, and password with your credentials  
 public static void main*(*String*[]* args*) {* String connectionUrl =  
 "jdbc:h2:tcp://localhost:8091/mem:personDB;"  
 + "user=testDBUsername;"  
 + "password=testDBPassword;";  
  
 ResultSet resultSet = null;  
  
 try *(*Connection connection = DriverManager.*getConnection(*connectionUrl*)*;  
 Statement statement = connection.createStatement*()*;*) {* // Create and execute a SELECT SQL statement.  
 String selectSql = "SELECT TOP 10 \* from Person";  
 resultSet = statement.executeQuery*(*selectSql*)*;  
  
 // Print results from select statement  
 while *(*resultSet.next*()) {* System.*out*.println*(*resultSet.getString*(*2*)* + " " + resultSet.getString*(*3*))*;  
 *}  
 }* catch *(*SQLException e*) {* e.printStackTrace*()*;  
 *}  
 }  
}*

**How to access H2 DB from code in Python(3.9):**

* **Download the h2 jar from**: “https://github.com/audiomojo/testAutomation/blob/master/target/h2-1.4.200.jar”
* **JDBC** **URL**: “jdbc:h2:tcp://localhost:8091/mem:personDB”
* **UserName**: “testDBUsername”
* **Password**: “testDBPassword

> import jaydebeapi

>>> conn = jaydebeapi.connect("org.h2.Driver",

... "jdbc:h2:tcp//localhost:8091/mem:personDB",

... ["testDBUsername", "testDBPassword"],

... "<AbsolutePathToH2Jar>",)

>>> curs = conn.cursor()

>>> curs.execute("select \* from PERSON")

>>> curs.fetchall()

>>> curs.close()

>>> conn.close()

More information can be found at <https://pypi.org/project/JayDeBeApi/#usage>

**Endpoints**

All endpoints have Basic Authentication. There are two types of authorization:

Read Access:

Username – testUsername

Password – testPassword

Write. Access:

Username – admin

Password -- testPassword

**POST** -- /v1/post-person **(Needs WRITE Access)**

Allows a person to be inserted into the H2 DB

* Body needed
  + Fields
    - firstName: String
    - lastName: String
    - phoneNumber: String

**GET** -- /v1/get-person/{personId} **(Needs either READ or WRITE Access)**

Allows one person to be selected based on ID. We are only searching for a person based off ID at this time

* Path Variable required

**DELETE** -- /v1/delete-person/{personId} **(Needs WRITE Access)**

Allows one person to be deleted based on ID. We are only deleting a person based off ID at this time

* Path Variable required

**Useful Tools**

Here are a few tools we believe could help you be more successful

**Source code**

Source Code can be found at <https://github.com/audiomojo/testAutomation>

**Swagger**

Swagger is an Interface Description Language for describing RESTful APIs expressed using JSON

How to Access Swagger

* + Open Browser to <http://localhost:8083/swagger-ui.html>
  + Swagger Docs <http://localhost:8083/v3/api-docs>

**Postman**

**Postman** is a great tool when trying to dissect RESTful APIs

**Chrome**

**PyCharm**

Great IDE for writing automation in python