Why Gilhari® for JSON integration?
An introduction to the microservice framework

What is Gilhari?

Gilhari™ is a cutting-edge microservice framework designed to provide seamless persistence for JSON objects in relational databases. This framework, encapsulated in a Docker image, offers a highly configurable solution tailored to specific app models, ensuring smooth data exchange and CRUD operations through a RESTful interface.
Gilhari, named after the Hindi word for squirrel, transfers JSON data between an application and a database. Gilhari's ability to easily work with even an existing database schema makes it ideal for leveraging legacy data.
The functional focus of a Gilhari microservice is to exchange JSON data with a relational database. Architecturally, it would typically fit behind other application components (e.g., another microservice and/or an API gateway) that take care of business logic, authentication, and authorizations. Gilhari, running in a Docker container, can scale effectively in Kubernetes environments.

Why Gilhari?

1. *Rapid Development*: Streamline the creation of microservices for data integration.
2. *Simplified Data Pipelines*: Easily export JSON data into JDBC-compliant databases.
3. *Effortless Data Retrieval*: Retrieve JSON data for downstream consumption.
4. *Mobile Client Support*: Facilitate JSON data exchange with remote server databases.
5. *Leverage Existing Data*: Utilize existing schema and data without relying on non-standard JSON data types.
6. *Cloud Integration*: Enhance cloud services development.

What sets Gilhari apart?

- *Encapsulated Capability*: Exchange JSON objects with relational databases efficiently.
- *Object Model Independence*: Compatible with any object model.
- *Database-Agnostic*: Works with any JDBC-compliant database.
- *Supports Complex Relationships*: Handle one-to-one, one-to-many, and many-to-many relationships.
- *Metadata-Driven*: No code development needed for CRUD operations.
- *Flexible RESTful Interface*: Provides synchronous and asynchronous REST APIs.
- *Scalability*: Ideal for container orchestration systems like Kubernetes.
- *Docker Integration*: Easy to configure, deploy, and scale.

So how do we use Gilhari?

Configuring and using Gilhari involves five straightforward steps:
1. *Define Domain Model Classes*: Create and compile empty Java classes corresponding to your JSON object types.
2. *Create ORM Specification*: Define a declarative Object-Relational Mapping (ORM) specification mapping JSON object attributes.
3. *Build a Dockerfile*: Create a Dockerfile to combine the base Gilhari image with app-specific artifacts like domain model classes, ORM specification, and JDBC driver.
4. *Build Docker Image*: Use the Dockerfile to build an app-specific Docker image for Gilhari.
5. *Deploy and Run*: Deploy the Docker image in a container to provide RESTful APIs for JSON persistence.

An outline of step by step Example: Employee object

Step 1: Define the JSON_Employee Java class to encapsulate Employee data, extending JDX_JSONObject for compatibility with Gilhari's JSON handling.

Step 2: Create an ORM specification (myproject.jdx) to map Employee attributes (id, name, exempt, compensation) to corresponding database fields (id, name, exempt, salary).

Step 3: Prepare a Dockerfile to configure Gilhari within a Docker container, setting up necessary directories (bin and config) and specifying the gilhari_service.config file.

Step 4: Build the Docker image (my_app_gilhari:1.0) using the Dockerfile, bundling your application-specific configurations and Gilhari setup.

Step 5: Deploy and run the Docker container (my_app_gilhari:1.0), exposing Gilhari's RESTful APIs on port 80, which can now handle CRUD operations for Employee JSON objects interfacing with your relational database.

By following these steps, you leverage Gilhari's capabilities to efficiently persist JSON data (such as Employee objects) into a relational database, simplifying the development and deployment of data-intensive applications.

Conclusion

Gilhari™ simplifies the process of storing JSON objects in relational databases, making development more straightforward and enhancing data integration. By using Gilhari, you can streamline your workflow and ensure efficient handling of JSON data within your relational database systems. It's designed to make JSON persistence easier and more efficient for developers.

Further reading:

https://github.com/ashhraaffff/software-tree-gilhari.git
https://www.softwaretree.com/v1/products/gilhari/gilhari-restfulAPI.php

https://www.softwaretree.com/v1/products/gilhari/gilhari-usage-tips.php

https://www.softwaretree.com/v1/products/gilhari/download-gilhari.php