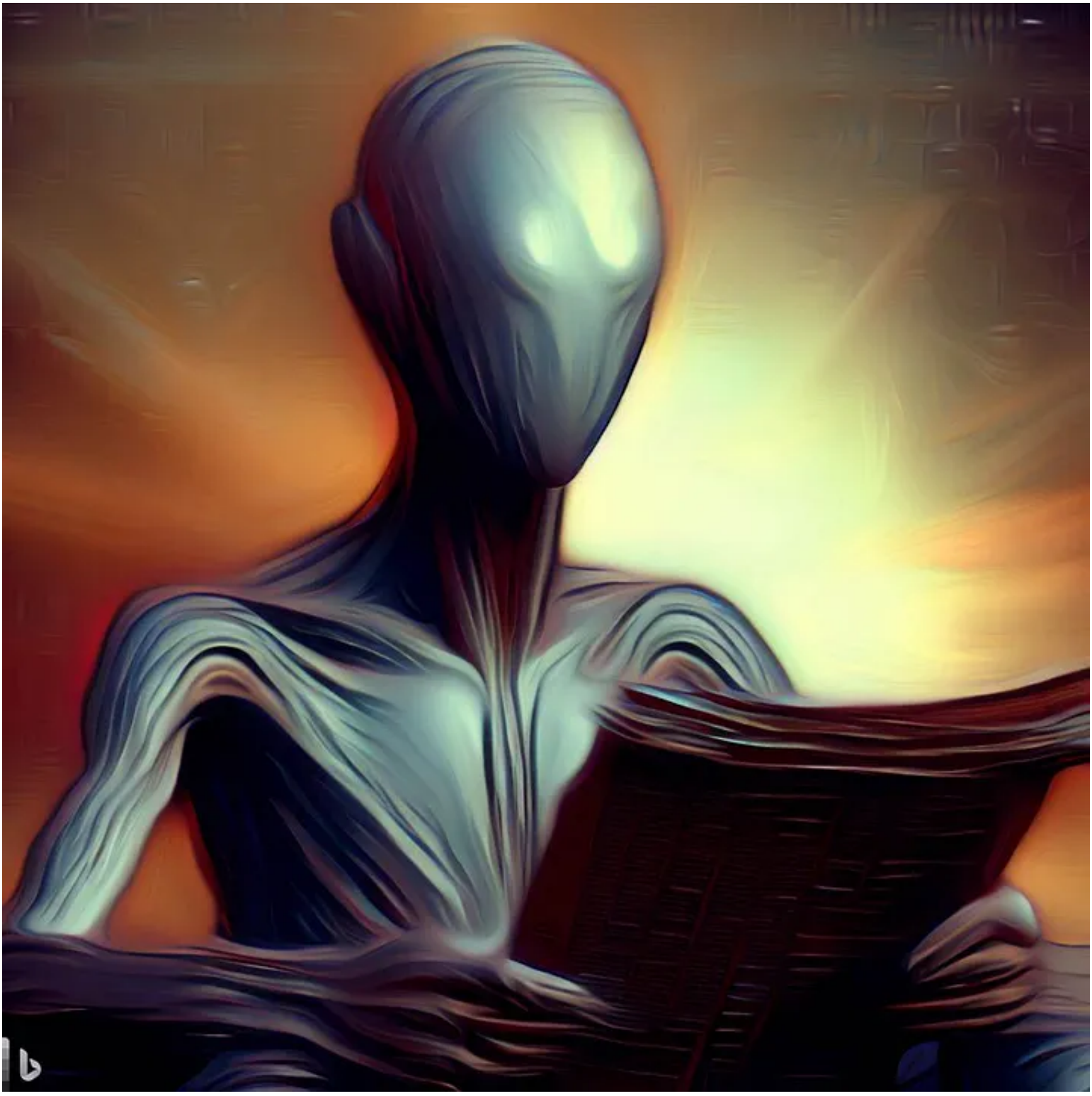
Developing a Hybrid FHIR Provider Using HAPI-FHIR: Access Any Datastore to Retrieve FHIR Data

FHIRFLY · Follow
3 min read · Jul 24, 2023

The HL7 Fast Healthcare Interoperability Resources (FHIR) standard continues to revolutionize the healthcare industry by facilitating data exchange between different healthcare systems. It opens new ways for developers to build more efficient, reliable, and interoperable healthcare applications. To aid in this process, libraries such as HAPI-FHIR provide ready-to-use tools to quickly and easily implement FHIR standards in your applications.

In this article, we'll explore the process of creating a Hybrid FHIR Provider using the HAPI-FHIR library. This hybrid provider will be capable of connecting and retrieving data from virtually any datastore, including SQL, MongoDB, RabbitMQ, Kafka, FHIR Servers, among others.



Introducing HAPI-FHIR

HAPI-FHIR is a complete implementation of the HL7 FHIR standard for healthcare interoperability in Java. Developed by Smile CDR, it is built to offer robust, feature-rich FHIR solutions. HAPI-FHIR aims to make FHIR easy to work with and bring FHIR capabilities to existing applications.

What is a Hybrid FHIR Provider?

A Hybrid FHIR Provider is essentially a data agnostic provider that can connect to any data source to fetch FHIR resources. This flexibility allows the provider to integrate with heterogeneous data architectures, hence 'Hybrid'. It fetches data from a variety of sources and returns it in a FHIR-compliant format, ensuring interoperability and efficiency.

Steps to Develop a Hybrid FHIR Provider

1. Set Up Your Environment

Start by setting up a Java development environment and adding HAPI-FHIR's dependencies to your project. For example, using Maven, you can add the HAPI-FHIR library to your `pom.xml` file.

2. Create a Base FHIR Provider

```
public class HybridPatientProvider extends BaseResourceProvider {
    @Override
    public Class<Patient> getResourceType() {
        return Patient.class;
    }
}
```

3. Define FHIR Operations

Define the FHIR operations your provider will support (e.g., `@Read`, `@Search`, etc.). Each operation should correspond to a method in your provider class:

```
@Read()
public Patient getResourceById(@IdParam IdType theId) {
    // implementation here
}
```

```
@Search()
public List<Patient> searchPatients(@OptionalParam(name = Patient.SP_NAME)
StringParam theName) {
```

```
    // implementation here
}
```

4. Connect to Datastores

At this stage, you can define how your provider connects to various datastores. This typically involves creating connection objects to different databases and using them to fetch data:

```
public class HybridPatientProvider extends BaseResourceProvider {
    // Define connection objects
    private SQLConnection sqlConnection;
    private MongoConnection mongoConnection;
    // ...
```

```
    public HybridPatientProvider() {
        // Initialize connections
        this.sqlConnection = new SQLConnection();
        this.mongoConnection = new MongoConnection();
        // ...
    }
}
```

To fetch data from different datastores, you can switch between connections based on your needs:

```
@Read()
public Patient getResourceById(@IdParam IdType theId) {
    if (someCondition) {
        // Fetch data using SQL connection
        return sqlConnection.fetchData(theId);
    } else {
        // Fetch data using MongoDB connection
        return mongoConnection.fetchData(theId);
    }
}
```

5. Map Data to FHIR Resources

Once you retrieve data from a source, map it to FHIR resources. Depending on the nature of the data, this can be straightforward or complex, involving the creation of custom mapping functions.

6. Deploy and Test Your Provider

Finally, you can deploy your hybrid provider as part of a HAPI FHIR server instance. Ensure to test the implemented FHIR operations with different data sources.

Conclusion

Developing a Hybrid FHIR Provider using the HAPI-FHIR library is a robust solution for organizations dealing with diverse data architectures. This approach enables seamless data interoperability, keeping the complexities of the data source abstracted from the end-user. With such a provider, you can harness the full potential of FHIR, ensuring efficient, reliable, and standards-compliant data exchange in healthcare systems.

Fhir | Fhir Integration | Hapi | Java

Follow

Written by FHIRFLY

74 Followers

SECURE. PRIVATE. AVAILABLE. CONFIDENTIAL. INTEGRAL. INTEROPERABLE. OUT OF THE DARKNESS COMES LIGHT.

More from FHIRFLY