



Ballerina

Elegant Integrations with Ballerina Swan Lake - II

Thisaru Guruge

thisaru@wso2.com | Technical Lead | [@ballerinalang](https://twitter.com/ballerinalang) | [WSO2](https://www.wso2.com)

March 2024

Talk Outline

- **Introduction**
 - Recap on Ballerina Swan Lake
 - Introduction to database integration
 - Introduction to GraphQL
- **Database Integrations with Ballerina Persist**
 - Creating the data model
 - Setting up the database connection
 - Perform CRUD operations on the database
- **Expose the database using GraphQL**
 - Design a GraphQL API
 - Implement GraphQL using Ballerina Swan Lake
 - How to test your API



Features of Ballerina - A Quick Recap



Data oriented

Type-safe, declarative processing of JSON, XML, and tabular data with language-integrated queries.

```
type User record { int id; string name; };  
...  
User manu = { id: 92874, name: "manuranga" }
```



Concurrent

Easy and efficient concurrency with sequence diagrams and language-managed threads without the complexity of asynchronous functions.

```
httpClient hello = check new ("http://hello.com");  
MyGreeting greeting = check hello->get("/world");
```

Also see: start, wait and workers



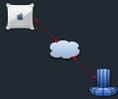
Structurally typed

Uses structural types with support for openness for static typing within a program and for describing service interfaces.

```
type Customer record {|  
    int id;  
    string name;  
    int account;  
|};  
...  
Customer customer = { ... };  
User user = customer;  
addUser(user)
```

Features of Ballerina

Network abstractions



- The `service` and `client` are first-class citizens in Ballerina



- Ballerina has `resource` and `remote` methods to handle network calls

```
service on new http:Listener(9090) {  
}
```

```
http:Client myClient = check new ("http://localhost:9090");  
string greeting = check myClient->/greeting;
```

```
resource function get greeting() returns string {  
    return "Hello, World!";  
}
```

```
remote function greeting() returns string|error {  
    return http:Client->/greeting;  
}
```



Database Integration

Databases

- Databases are an integral part of (almost all) modern day applications
- Integrating a database to your application is cumbersome at times
- Ballerina provides an easy way to integrate a database using connectors
- Ballerina persist functionality helps to go beyond the connectors

```
import ballerina/mysql;
import ballerina/mysql.driver as _;

final mysql:Client dbClient = check new (
    host = "localhost",
    port = 3306,
    user = "user",
    password = "password"
);

public function main() {
    stream<Movie, error?> profiles =
        check dbClient->query(`SELECT * FROM MOVIES`);
}
```



Ballerina Persist

Ballerina Persist

- Generates a client to handle database operations
- Type safe way to handle database operations
- Define the data model using Ballerina records
- CLI tool for generating the code

```
$ bal persist init --module datasource --datastore mysql
```

```
// Data Model
public type Movie record {
    readonly string id;
    string title;
    int year;
    Director director;
};

public type Director record {
    readonly string id;
    string name;
    int birthYear;
    Movie[] movies;
};
```

```
$ bal persist generate
```



Understanding GraphQL

- Query language for APIs
- Developed by Facebook and open-sourced
- Solves over-fetching and under-fetching problems in REST APIs
- Defines the API using a schema
- Strongly-typed type system

```
type Query {  
  movies: [Movie!]!  
  directors: [Director!]!  
}
```

```
type Movie {  
  id: ID  
  title: String  
  year: Int  
  director: Director!  
}
```

```
type Director {  
  id: ID!  
  name: String!  
  birthYear: Int!  
  movies: [Movie!]!  
}
```



Designing GraphQL API (Live coding)

- Using Ballerina visual designer tool for designing GraphQL services
- How to expose a Database using GraphQL
- Lazy-load data from the database
- Export the GraphQL schema using the `bal graphql` tool.
- Using in-built GraphiQL tool to test the GraphQL API



Ballerina Community



Discord : <https://discord.gg/ballerinalang>



SO: <https://stackoverflow.com/questions/tagged/ballerina>



Twitter <https://twitter.com/ballerinalang>



GitHub : <https://github.com/ballerina-platform>



Thank You!

