WSO2

Ballerina

Elegant Integrations with Ballerina Swan Lake

Oct 2023

Talk Outline

History of WSO2

Our Journey with Programming Languages

Ballerina Language

- Data-Oriented Design
- Concurrency
- Structural Types

Demo

History of WSO2

Short clip from WSO2 founder and CIO, Sanjiva Weerawarana.

Our Journey with Programming Languages

WSO2 enables thousands of enterprises, including hundreds of the world's largest corporations, top universities, and governments, to drive their digital transformation journeys—executing more than 18 trillion transactions and managing more than 500 million identities annually.

Apache Synapse

XML based DSL for specifying service mediation. Community project, heavily contributed by WSO2.

Siddhi

Stream Processor with a Streaming SQL

Jaggery Runtime

JS runtime written in Java.

Integration Studio

Graphical editor for Apache Synapse

Timeline Of Ballerina

- Started out as Synapse replacement language back in late 2016. Inspired by sequence diagrams and graphical editing.
- Initial implementation as AST interpreted language (2017)
- Internal vm (BVM) with internal ByteCode (late 2017)
- Backend/frontend separation via BIR. JVM bytecode as the backend (late 2018).
- Swan Lake version GA release in , with major improvements and extensive set of standard libraries and connectors (early 2022).
- Continues update to Swan Lake version. Currently update 8.

Features of Ballerina

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.

```
http:Client 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 - Data oriented

Rich set of built in data types	`boolean, int, float, string, array, tuple, map, record`
	`decimal, xml (byte, char, json)`

Data litrals for json (more generally records and lists) and xml	`json person = { id: 92874, name: "manu" };`
	``xml book = xml`<book>The Lost World</book>`;``

Data independent form behavior	`type User record { int id; string name; };`
	`type Point3D [int, int, int];`

Table type for efficiently look up data	`employeesById.put({ id: 92874, name: "manu" });`
	`io:println(employeesById[92874]);`

Immutable values	`int[] & readonly arr = [1, 2, 3];`

Values are comparable	`io:println([1, "hello"] == [1, "hello"]);`
	`io:println([1, "hello"] < [2, "hello"]);`

Features of Ballerina - Concurrent

Concurrency in Ballerina is enabled by strands, which are lightweight threads.

Concurrent by default - Below code will block the strand without blocking the thread, no special keyword (like await) needed.

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

works and start keyword can be used to start strands explicitly

```
future<int> resultA = start calculateA();
future<int> resultB = start calculateB();
int|error result = wait resultA | resultB;
```

Features of Ballerina - Structurally typed

Ballerina has a unique type system based on set theory.

```
type Colour "RED"|"GREEN"|"BLUE";
...
Colour? c = "RED";
```

Think of type as a set and subtype as a subset.

```
type User record { string name; };
type Employee record { int id; string name; };
...
Employee emp = ...;
User user = emp;
```

Features of Ballerina - Structurally typed

New types can be constructed using set operation (union and intersect) on existing types.

```
int[] & readonly arr = [1, 2, 3];
```

Since types can work as schemas when working defining network interfaces.

```
service / on ln {

    resource function post calc(CalcReq args) returns CalcResp {

    }
}
```

Mathematically sound - based on the works of Giuseppe Castagna

Demo

```
import ballerina/http;

type ITunesAlbumData record {
   string collectionViewUrl;
   string collectionName;
};

type ITunesResult record {
   ITunesAlbumData[] results;
};

type Album record {|
   string url;
   string name;
|};

service http:Service /pickagift on new http:Listener(9090) {

   resource function get albums(string artist) returns Album[]|error {
       http:Client iTunes = check new ("https://itunes.apple.com/");
       ITunesResult search = check iTunes->get(searchUrl(artist));
       return from ITunesAlbumData a in search.results
              select {name: a.collectionName, url : a.collectionViewUrl};
   }
}

function searchUrl(string artist) returns string {
   return "search?term=" + artist + "&entity=album&attribute=allArtistTerm";
}
```

Thank you

Please give us a try : https://ballerina.io/downloads/

Follow us on : Discord community and https://twitter.com/ballerinalang

Star us on GitHub : https://github.com/ballerina-platform/ballerina-lang