Fluently NoSQL

Server Side Swift 2019, Copenhagen

Joe Smith

Site Reliability Engineer at Slack

@Yasumoto

- Performance
- Efficiency
- Reliability
- Scalability

Agenda

- Patient Road to Production
- DynamoDB
- `AWSSDKSwift`
- Fluent DynamoDB

Photo by SpaceX on Unsplash

Ecosystem

Photo by Markus Spiske on Unsplash

Introducing New Technology

Photo by Tyler Lastovich on Unsplash

Roadmap

Photo by Ryan Stone on Unsplash

🛑 Emergency Stop

1. What Loadtests are running right now?
2. How do I stop all loadtests NOW?

🍐 All's Well — Loadtesting May Proceed

Message

Explanation Message

Please include a channel name so folks know where to go for more details.

Initiate Emergency Stop!

Running Loadtests

Timestamp	Username	Hostname	Loadtest Tool
10/1/2019, 10:32:55 AM	jmsmith	SFO-M-JMSMITH02	api_blast: EKM Test

Emergency Stop History

Username	Timestamp	Message	Is Incident Ongoing	Version
arka	9/26/2019, 2:10:58 PM	Incident cleared	🍐	24
arka	9/26/2019, 2:10:22 PM	this is paging for Database problems	🍎	23
jrodgers	9/26/2019, 1:08:29 PM	going to continue with #loadtest-private-channel-access-controls	🍐	22

Data Structure – `ServiceLock`

1. **ServiceName** 🌍 always `global`
2. **Version** 📜 A historical log
3. **IsIncidentOngoing** 🔴 the "red button"
4. **Username** 🙋 Who flipped the switch?
5. **Timestamp** ⌚ When?
6. **Message** ➡️ Where's the incident channel?

Photo by Benjamin Davies on Unsplash

Low-risk Production Experience

Photo by Dawn Armfield on Unsplash

Picking a Datastore (at Slack)

- MySQL
- Something Else

Photo by Ian Battaglia on Unsplash

Most of our services... store and retrieve data by primary key and do not require the ***complex querying*** *and management functionality offered by an RDBMS.*

-- Dynamo Paper

Key Concepts

1. Tables
2. Items & Attributes
3. Primary Keys

Table

1. Name
2. Hash Key
3. Range Key
4. Capacity

Items

Photo by Karen Vardazaryan on Unsplash

Attributes (Values)

- Scalars
  - Single items
- Collections
  - Groups of values

Scalar Values

- `Binary`
- `Boolean`
- `Number`
- `Null`
- `String`

Collection Values

1. Binary Set
2. List
3. Map
4. Number Set
5. String Set

Primary Key

1. Simple Primary Key – Hash Key
2. Composite Primary Key
    1. Hash Key
    2. Sort Key

Scalars as Primary Keys

1. String
2. Numbers
3. Binary

Data Structure – `ServiceLock`

1. 🌍 `ServiceName ( String )` – Hash Key
2. 📜 `Version ( Int )` – Sort Key
3. 🔴 `IsIncidentOngoing ( Bool )`
4. 🙋‍♀️ `Username ( String )`
5. ⌚ `Timestamp( String )`
6. ➡️ `Message ( String )`

```
resource "aws_dynamodb_table" "emergency_stop_service_lock" {
  name           = "emergency-stop-service-lock"
  read_capacity  = 10
  write_capacity = 10
  hash_key       = "ServiceName"
  range_key      = "Version"

  attribute {
    name = "ServiceName"
    type = "S"
  }

  attribute {
    name = "Version"
    type = "N"
  }
}
```

Service Name	Version	Is Incident Ongoing	User name	Time stamp	Message
global	1	false	Jim	8/25/2019, 2:04:31 AM	Error!
global	2	true	Joe	8/27/2019, 7:13:40 PM	Fixed

Service Name	Version	Is Incident Ongoing	User name	Time stamp	Message
global	1	false	Jim	8/25/2019, 2:04:31 AM	Error!
global	2	true	Joe	8/27/2019, 7:13:40 PM	Fixed

Photo by Maria Molinero on Unsplash

https://dynamodbguide.com

By Alex DeBrie

Photo by Aron Visuals on Unsplash

AWS SDK Swift

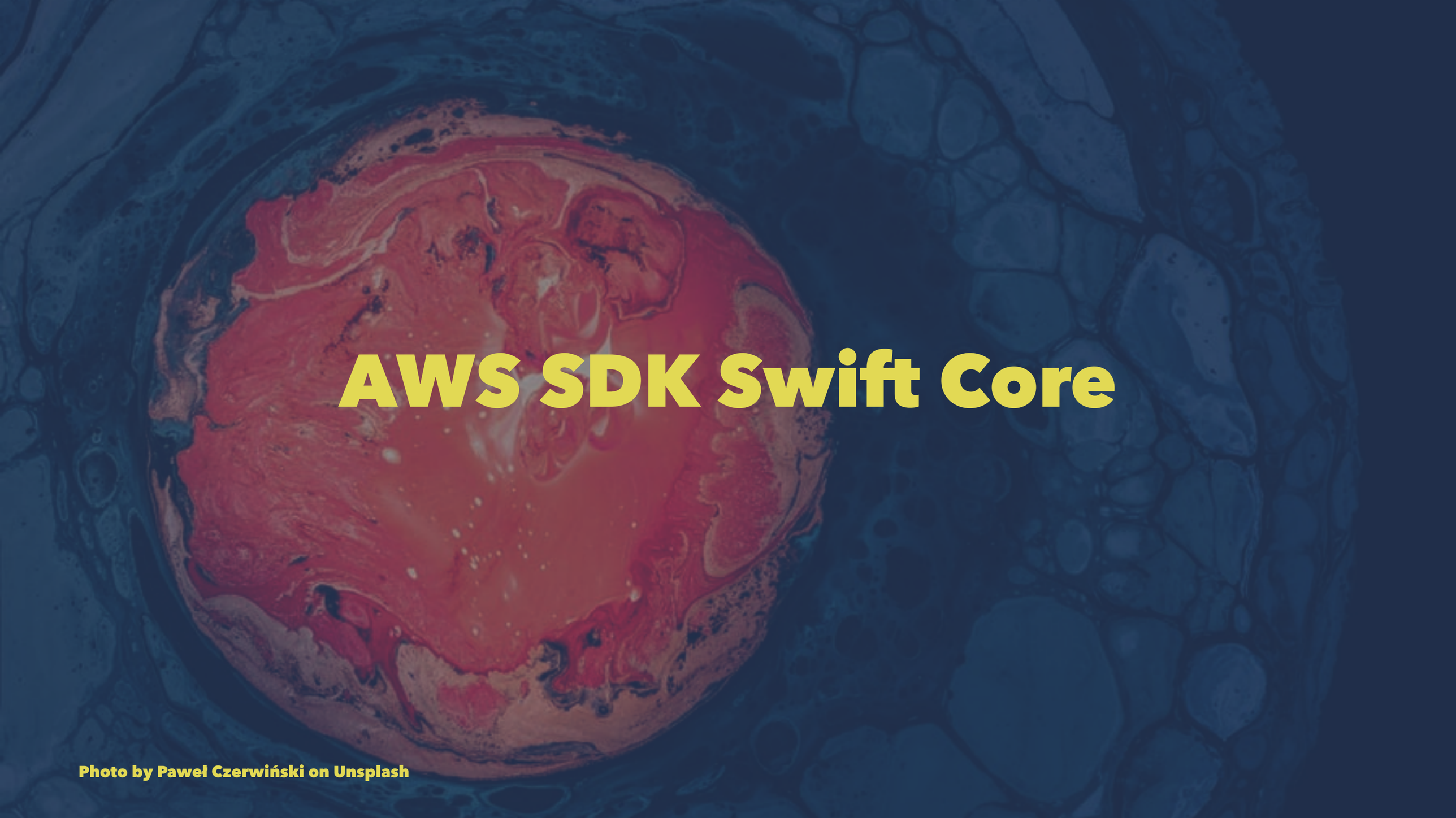
- Created by Yuki (**@noppoMan**)
- Major contributions from **@jonnymacs** and **@adam-fowler**

Photo by Levi Morsy on Unsplash

Two Components

- `aws-sdk-swift-core`
- `aws-sdk-swift`

Photo by Genevieve Perron-Migneron on Unsplash



AWS SDK Swift Core

Photo by Paweł Czerwiński on Unsplash

CredentialProvider

```
/// Protocol defining requirements for object providing AWS credentials
public protocol CredentialProvider {
    var accessKeyId: String { get }
    var secretAccessKey: String { get }
    var sessionToken: String? { get }
    var expiration: Date? { get }
}
```

AWS SDK Swift Core – Hashing

- 💻 `CommonCrypto`
  or
- 🐧 `CAWSSDKOpenSSL`

1. `hmac` – hash-keyed message authentication code
2. `sha256`
3. `md5`

Photo by chris panas on Unsplash

AWS SDK Swift Core – HTTPClient

```
public final class HTTPClient {
    public struct Request {
        var head: HTTPRequestHead
        var body: Data = Data()
    }

    public struct Response {
        let head: HTTPResponseHead
        let body: Data
        public func contentType() -> String?
    }

    public enum HTTPError: Error {
        case malformedHead, malformedBody, malformedURL
    }

    public init(url: URL, eventGroup: EventLoopGroup)

    public func connect(_ request: Request) -> EventLoopFuture<Response>

    public func close(_ callback: @escaping (Error?) -> Void)
}
```

```
public class AWSClient {
    public enum RequestError: Error {
        case invalidURL(String)
    }

    public var endpoint: String

    public static let eventGroup: EventLoopGroup

    public func signURL(url: URL,
                        httpMethod: String,
                        expires: Int = 86400) -> URL
...
```

```
public func send<Output: AWSShape, Input: AWSShape>(
    operation operationName: String,
    path: String,
    httpMethod: String,
    input: Input) -> Future<Output> {
        return signer.manageCredential().thenThrowing { _ in
            let awsRequest = try self.createAWSRequest(
                operation: operationName,
                path: path,
                httpMethod: httpMethod,
                input: input
            )
            return try self.createNioRequest(awsRequest)
        }.then { nioRequest in
            return self.invoke(nioRequest)
        }.thenThrowing { response in
            return try self.validate(operation: operationName, response: response)
        }
}
```

```
public func send<Output: AWSShape, Input: AWSShape>(
    operation operationName: String,
    path: String,
    httpMethod: String,
    input: Input) -> Future<Output> {
        return signer.manageCredential().thenThrowing { _ in
            let awsRequest = try self.createAWSRequest(
                operation: operationName,
                path: path,
                httpMethod: httpMethod,
                input: input
            )
            return try self.createNioRequest(awsRequest)
        }.then { nioRequest in
            return self.invoke(nioRequest)
        }.thenThrowing { response in
            return try self.validate(operation: operationName, response: response)
        }
}
```

```
public func send<Output: AWSShape, Input: AWSShape>(
    operation operationName: String,
    path: String,
    httpMethod: String,
    input: Input) -> Future<Output> {
        return signer.manageCredential().thenThrowing { _ in
            let awsRequest = try self.createAWSRequest(
                operation: operationName,
                path: path,
                httpMethod: httpMethod,
                input: input
            )
            return try self.createNioRequest(awsRequest)
        }.then { nioRequest in
            return self.invoke(nioRequest)
        }.thenThrowing { response in
            return try self.validate(operation: operationName, response: response)
        }
}
```

AWS SDK Swift Dynamo Client

1. `AttributeValue`
2. `getItem` / `putItem`
3. `query`

AttributeValue

```
/// An attribute of type Binary. For example:
/// "B": "dGhpcyB0ZXh0IGlzIGJhc2U2NC1lbmNvZGVk"
public let b: Data?

/// An attribute of type Boolean. For example:  "BOOL": true
public let bool: Bool?

/// An attribute of type Number. For example:  "N": "123.45"
/// Numbers are sent across the network to DynamoDB as strings,
/// to maximize compatibility across languages and libraries.
/// However, DynamoDB treats them as number type attributes
/// for mathematical operations.
public let n: String?

/// An attribute of type Null. For example:  "NULL": true
public let null: Bool?

/// An attribute of type String. For example:  "S": "Hello"
public let s: String?
```

AttributeValue

```
/// An attribute of type Binary Set. For example:
/// "BS": ["U3Vubnk=", "UmFpbnk=", "U25vd3k="]
public let bs: [Data]?

/// An attribute of type List. For example:
/// "L": [ {"S": "Cookies"} , {"S": "Coffee"}, {"N", "3.14159"}]
public let l: [AttributeValue]?

/// An attribute of type Map. For example:
/// "M": {"Name": {"S": "Joe"}, "Age": {"N": "35"}}
public let m: [String: AttributeValue]?

/// An attribute of type Number Set. For example:
/// "NS": ["42.2", "-19", "7.5", "3.14"]
public let ns: [String]?

/// An attribute of type String Set. For example:
/// "SS": ["Giraffe", "Hippo" ,"Zebra"]
public let ss: [String]?
```

```
public class AttributeValue: AWSShape {
    public init(b: Data? = nil, bool: Bool? = nil,
                bs: [Data]? = nil, l: [AttributeValue]? = nil,
                m: [String: AttributeValue]? = nil,
                n: String? = nil, ns: [String]? = nil,
                null: Bool? = nil, s: String? = nil,
                ss: [String]? = nil) {
        self.b = b
        self.bool = bool
        self.bs = bs
        self.l = l
        self.m = m
        self.n = n
        self.ns = ns
        self.null = null
        self.s = s
        self.ss = ss
    }
```

Fetching a Single Item

```
///  The GetItem operation returns a set of attributes
/// for the item with the given primary key.
public func getItem(_ input: GetItemInput) -> Future<GetItemOutput> {
    return client.send(operation: "GetItem",
                       path: "/",
                       httpMethod: "POST",
                       input: input)
}
```

GetItemInput

```
/// Determines the read consistency model
public let consistentRead: Bool?

/// A map representing the primary key of the item to retrieve.
public let key: [String: AttributeValue]

/// The name of the table containing the requested item.
public let tableName: String
```

Setting Items

```
///  Creates a new item, or replaces an old item with a new item.
public func putItem(_ input: PutItemInput) -> Future<PutItemOutput> {
    return client.send(operation: "PutItem",
                       path: "/",
                       httpMethod: "POST",
                       input: input)
}
```

Setting Items

```
///  Creates a new item, or replaces an old item with a new item.
public func putItem(_ input: PutItemInput) -> Future<PutItemOutput> {
    return client.send(operation: "PutItem",
                       path: "/",
                       httpMethod: "POST",
                       input: input)
}
```

PutItemInput

```
/// A map of attribute name/value pairs.
public let item: [String: AttributeValue]

/// Use ReturnValues if you want to get the item attributes
/// as they appeared before they were updated
public let returnValues: ReturnValue?

/// The name of the table to contain the item.
public let tableName: String
```

Querying for Multiple Items

```
///  The Query operation finds items based on primary key values.
public func query(_ input: QueryInput) -> Future<QueryOutput> {
    return client.send(operation: "Query",
                       path: "/",
                       httpMethod: "POST",
                       input: input)
}
```

```
/// One or more substitution tokens for attribute names
/// in an expression.
public let expressionAttributeNames: [String: String]?

/// One or more values that can be substituted in an expression.
public let expressionAttributeValues: [String: AttributeValue]?

/// The condition that specifies the key values for items to
/// be retrieved by the Query action.
public let keyConditionExpression: String?
```

```
let nowDate = Date()
let earlierDate = now.addingTimeInterval(-60.0 * 60.0)

let now = formatter.string(from: nowDate)
let earlier = formatter.string(from: earlierDate)

let expressionAttributeNames = [
        "#S": "ServiceName",
        "#T": "Timestamp"]
let expressionAttributeValues = [
        ":global": DynamoDB.AttributeValue(s: "global"),
        ":then": DynamoDB.AttributeValue(s: earlier),
        ":now": DynamoDB.AttributeValue(s: now)]

let keyConditionExpression = "#S = :global AND #T BETWEEN :then AND :now"
```

AWS SDK Swift Dynamo Client

1. `AttributeValue`
2. `getItem` / `putItem`
3. `query`

History of Changes

Photo by Daniel H. Tong on Unsplash

Service Name	Version	Is Incident Ongoing	User name	Time stamp	Message
global	1	false	Jim	8/25/2019, 2:04:31 AM	Error!
global	2	true	Joe	8/27/2019, 7:13:40 PM	Fixed

Service Name	Version	Is Incident Ongoing	User name	Time stamp	Message
global	1	false	Jim	8/25/2019, 2:04:31 AM	Error!
global	2	true	Joe	8/27/2019, 7:13:40 PM	Fixed

Service Name	Version	Is Incident Ongoing	User name	Time stamp	Message	Current
global	0	true	Joe	8/27/2019, 7:13:40 PM	Fixed	2
global	1	false	Jim	8/25/2019, 2:04:31 AM	Error!	
global	2	true	Joe	8/27/2019, 7:13:40 PM	Fixed	

Service Name	Version	Is Incident Ongoing	User name	Time stamp	Message	Current
global	0	true	Joe	8/27/2019, 7:13:40 PM	Fixed	2
global	1	false	Jim	8/25/2019, 2:04:31 AM	Error!	
global	2	true	Joe	8/27/2019, 7:13:40 PM	Fixed	

Denormalized Data

Photo by Will O on Unsplash

💧 Fluent 3 Concepts

1. 📦 `Database`
2. 🔌 `DatabaseConnection`
3. 😻 `Provider`

📦 Database

```
public protocol Database {

    associatedtype Connection: DatabaseConnection

    /// Creates a new `DatabaseConnection` that will perform
    /// async work on the supplied `Worker`.
    func newConnection(on worker: Worker) -> Future<Connection>
}
```

```
public final class DynamoDatabase: Database {
    public typealias Connection = DynamoConnection
    private let config: DynamoConfiguration

    public init(config: DynamoConfiguration) {
        self.config = config
    }

    internal func openConnection() -> DynamoDB {
        return DynamoDB(accessKeyId: config.accessKeyId,
                        secretAccessKey: config.secretAccessKey,
                        region: config.region,
                        endpoint: config.endpoint)
    }

    public func newConnection(on worker: Worker) -> EventLoopFuture<DynamoConnection> {
        do {
            let conn = try DynamoConnection(database: self, on: worker)
            return worker.future(conn)
        } catch {
            return worker.future(error: error)
        }
    }
```

Photo by israel palacio on Unsplash

```swift
public protocol DatabaseConnection: DatabaseConnectable, Extendable {
    /// This connection's associated database type.
    associatedtype Database: DatabaseKit.Database
        where Database.Connection == Self

    /// If `true`, this connection has been closed and is no
    /// longer valid.
    /// This is used by `DatabaseConnectionPool` to prune
    /// inactive connections.
    var isClosed: Bool { get }

    /// Closes the `DatabaseConnection`.
    func close()
}
```

```
public final class DynamoConnection: DatabaseConnection {
    public typealias Database = DynamoDatabase

    public var isClosed: Bool {
        return self.handle.isClosed()
    }

    public func close() {
        self.handle.close()
    }

    /// Reference to parent `DynamoDatabase` that created this connection.
    private let database: DynamoDatabase

    internal private(set) var handle: DynamoDB!

    internal init(database: DynamoDatabase, on worker: Worker) throws {
        self.database = database
        self.eventLoop = worker.eventLoop
        self.handle = database.openConnection()
    }
}
```

Creating a Connection

Photo by Phix Nguyen on Unsplash

```swift
public struct DatabaseIdentifier<D: Database>: Equatable,
    Hashable, CustomStringConvertible, ExpressibleByStringLiteral {
    /// The unique id.
    public let uid: String

    /// See `CustomStringConvertible`.
    public var description: String {
        return uid
    }

    /// Create a new `DatabaseIdentifier`.
    public init(_ uid: String) {
        self.uid = uid
    }

    /// See `ExpressibleByStringLiteral`.
    public init(stringLiteral value: String) {
        self.init(value)
    }
}
```

```
extension DatabaseIdentifier {
    /// Default identifier for `DynamoDatabase`.
    public static var dynamo: DatabaseIdentifier<DynamoDatabase> {
        return "dynamo"
    }
}
```

```
/// Capable of creating connections to identified databases.
public protocol DatabaseConnectable: Worker {

    func databaseConnection<Database>(
        to database: DatabaseIdentifier<Database>?) ->
        Future<Database.Connection>

}
```

Photo by Kirill Pershin on Unsplash

```
public protocol DatabaseQueryable {

    associatedtype Query

    associatedtype Output

    /// Asynchronously executes a query passing zero
    /// or more output to the supplied handler.
    func query(_ query: Query,
               _ handler: @escaping (Output) throws -> ()) ->
            Future<Void>
}
```

```swift
public enum DynamoQueryAction {
    case set, get, delete, filter
}

/// 🔎 A DynamoDB operation
public struct DynamoQuery {
    /// 🏹 Note this is a var so `get/set` can be flipped easily if desired
    public var action: DynamoQueryAction

    /// 🍴 The name of the table in DynamoDB to work on
    public let table: String

    /// 💸 Which value(s) to perform the action upon
    public let keys: [DynamoValue]
}
```

```swift
/// 🔑 Values to uniquely identify an item stored in a DynamoDB Table
public struct DynamoValue: Codable, Equatable {

    public enum Attribute: Codable, Equatable {
        case mapping([String: Attribute])
        case null(Bool)
        case stringSet([String])
        case binary(Data)
        case string(String)
        case list([Attribute])
        case bool(Bool)
        case numberSet([String])
        case binarySet([Data])
        /// We send over all numbers to Dynamo as Strings, and cannot use Numeric
        case number(String)
}
```

```
public func query(_ query: Query, _ handler: @escaping (Output) throws -> ()) -> Future<Void> {
        switch query.action {
        case .get:
            let inputItem = DynamoDB.GetItemInput(
                key: requestedKey, tableName: query.table)
            return self.handle.getItem(inputItem).map { output in
                return try handler(Output(attributes: output.item))
            }
        case .set:
            let inputItem = DynamoDB.PutItemInput(item: requestedKey, returnValues: .allOld, tableName: query.table)
            return self.handle.putItem(inputItem).map { output in
                return try handler(Output(attributes: output.attributes))
            }
        case .delete:
            let inputItem = DynamoDB.DeleteItemInput(
                key: requestedKey, returnValues: .allOld, tableName: query.table)
            return self.handle.deleteItem(inputItem).map { output in
                return try handler(DynamoValue(attributes: output.attributes))
            }
        case .filter:
            return self.eventLoop.newFailedFuture(error: DynamoConnectionError.notImplementedYet)
        }
    } catch {
        return self.eventLoop.newFailedFuture(error: error)
    }
}
```

```
case .set:
    let inputItem = DynamoDB.PutItemInput(item: requestedKey,
                                          returnValues: .allOld,
                                          tableName: query.table)

     return self.handle.putItem(inputItem).map { output in
        return try handler(DynamoValue(attributes: output.attributes))
     }
```

😻 Provider

```
public protocol Provider {
    /// Register all services you would like to provide the `Container` here.
    ///
    ///     services.register(RedisCache.self)
    ///
    func register(_ services: inout Services) throws

    /// Called before the container has fully initialized.
    func willBoot(_ container: Container) throws -> Future<Void>

    /// Called after the container has fully initialized and after `willBoot(_:)`.
    func didBoot(_ container: Container) throws -> Future<Void>
}
```

```swift
/// 💧 The Provider expected to be registered to easily allow
/// usage of DynamoDB from within a Vapor application
public struct FluentDynamoDBProvider: Provider {
    public func register(_ services: inout Services) throws {
        try services.register(FluentProvider())

        services.register(DynamoConfiguration.self)
        services.register(DynamoDatabase.self)
        var databases = DatabasesConfig()
        databases.add(database: DynamoDatabase.self, as: .dynamo)
        services.register(databases)
    }

    public func didBoot(_ container: Container) throws -> EventLoopFuture<Void> {
        return .done(on: container)
    }
}
```

```
try services.register(FluentDynamoDBProvider())

var databases = DatabasesConfig()

let credentialsPath = Environment.get("CREDENTIALS_FILENAME") ?? "/etc/emergency-stop.json"
let creds = awsCredentials(path: credentialsPath)
let endpoint = Environment.get("ENVIRONMENT") == "local" ? "http://localhost:8000" : nil

let dynamoConfiguration =  DynamoConfiguration(accessKeyId: creds.accessKey,
                                               secretAccessKey: creds.secretKey,
                                               endpoint: endpoint)


let dynamo = DynamoDatabase(config: dynamoConfiguration)
databases.add(database: dynamo, as: .dynamo)
services.register(databases)
```

```swift
/// Write a ServiceLock to DynamoDB
///
/// - Returns:
///     An `EventLoopFuture` used to indicate success or failure
public func write(on worker: Request) -> EventLoopFuture<[DynamoValue]> {

    let key = self.dynamoFormat()

    let query = DynamoQuery(action: .set,
                            table: "limit-break-emergency-stop",
                            keys: [key])

    return worker.databaseConnection(to: .dynamo)
        .flatMap { connection in
            connection.query(query)
        }
}
```

Photo by Seth Reese on Unsplash

Next

1. ⏫ Publish `AWSSDKSwift` 4.0.0
    1. 🎉 Uses `NIO` 2.0
2. 🌄 Create `DynamoKit` with NIO 2.0
3. 💧 Convert `FluentDynamoDB` to Fluent 4
4. 🤦‍♂️ Upgrade `DynamoModel`
5. ✨ Better integration with `swift-log` and `swift-metrics`

Thank you!

- 🔴 Emergency Stop
- 📦 DynamoDB
- 🌴 `AWSSDKSwift`
- 💧 Fluent DynamoDB

https://github.com/Yasumoto/fluent-dynamodb