Reporter: Nguyen Duong Thu Thuy

**Database**

1. **SQLite**
2. **What is SQLite?**

* SQLite database feature is inbuilt provided by the iOS framework
* Structured Query Language (SQL)
* A pure Swift interface
* A type-safe, optional-aware SQL expression builder
* A flexible, chainable, lazy-executing query layer i.e a flexible coding style with queries that can be chained together to form a single query

1. **Realm**
2. **What is a Realm?**

* An instance of Realm Mobile Database container
* Loyal, synchronized, or in-memory

1. **Some notes**

* Realm is not a single application-wide database: an application often uses multiples Realms
* A Realm is not a table: A realm can contain multiple kinds of objects
* A Realm is not a schemaless document store

1. **Realm Model**

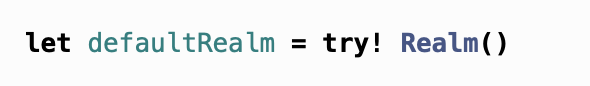
* Configuration

When you open a Realm, you pass the constructor a configuration object that defines how to access it. The configuration object specifies where the Realm database is located:

* A path on the device’s local file system
* A URL to a Realm Object Server, with appropriate access credentials (user/password, auth token)
* An identifier for an in-memory Realm



If you don’t provide a configureation object, you’ll open the default Realm, which is a local Realm specific to that application



* Realm URLS
* 3 types: **public**, **private** and **shared**.
* They’re all accessed the same way. The difference between them is access controls, which users can read and write to them. The URL format may also look a little different:

+ **Public**: can be accessed by all users. Owned by the admin user on the Realm Object Server, read-only to non-admins. ( realms://server/realm-name )

+ **Private**: Only the user that creates it has read and write permissions. ( realms://server/user-id/realm-name )

+ **Shared**: a private realm and owners grant other users read (and possibly write) access. ( realms://server/user-id/realm-name ) user-id is the userid of the owner. Everyone in that group has their own local copies of the Realm, but there’s only one “master” copy synced through the Object Server

* Permissions

Permission are set on each Realm using 3 boolean flags:

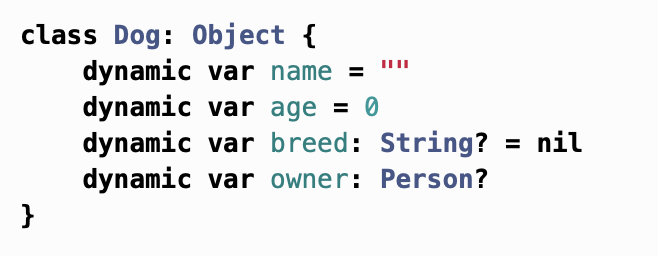
+ mayRead: can read

+ mayWrite: can write

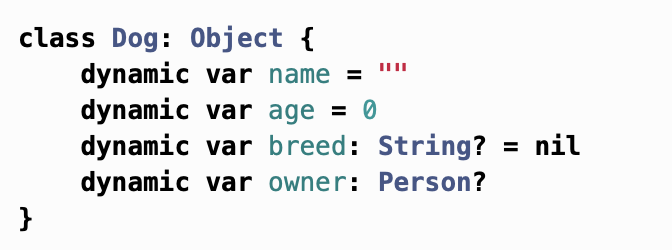
+ mayManage: can change permissions on the Realm for other users

By default, a Realm is private.

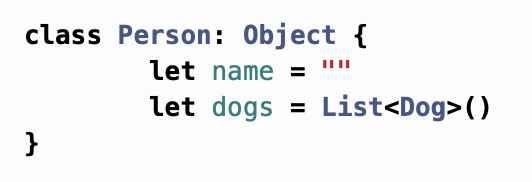
* Models and Schema



* To-one Relations



* To-Many Relationships

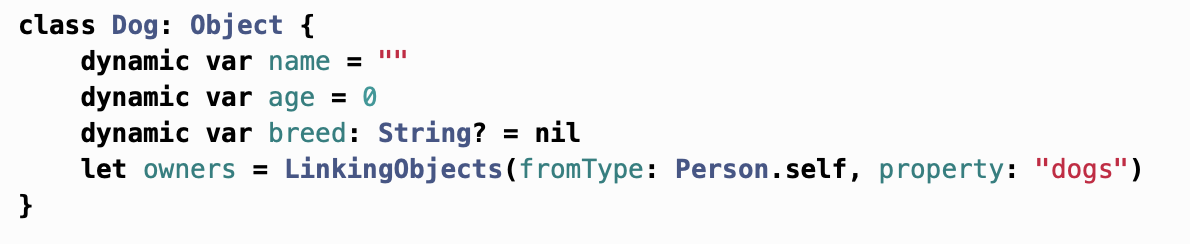


A list in Realm contains one or more Realm objects. To add Fido to Bob’s list of dogs:

Bob.dogs.append(fido)

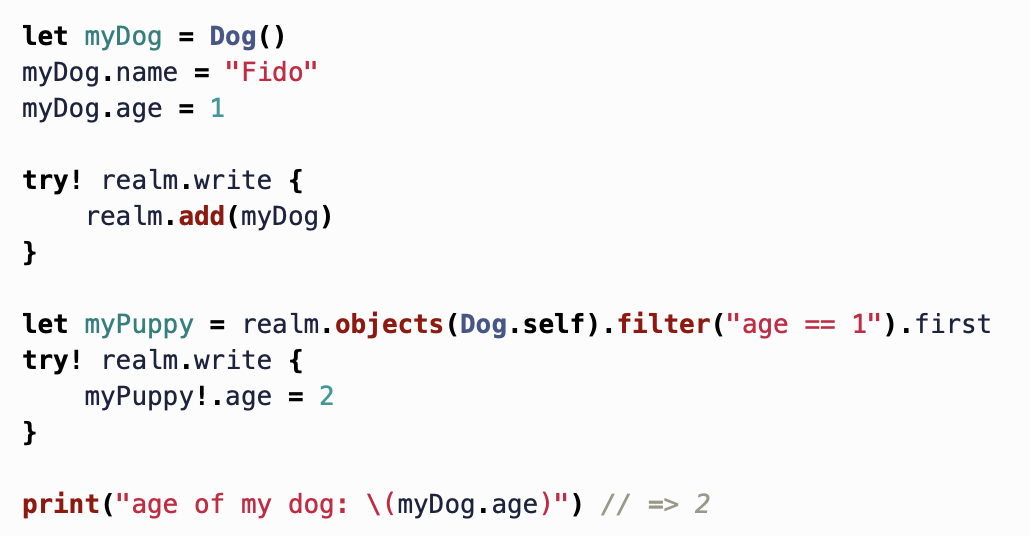
* Inverse Relationship

You’ll note that defining the Person has-many Dogs relationship didn’t automatically create a Dog belongs-to Person relationship; both sides of the relationship need to be set explicitly. It’s important to define both sides of this relationship.



1. **Realm Objects**

Object instances are live, auto-updateing views into the underlying data. You never have to refresh objects.



* Keep Realm fast and efficient, it also allow your code to be simpler and more reactive. If your UI code is dependent on a specific Realm object, you don’t need to worry about refreshing or re-fetching it before triggering a UI redraw.

1. **Some limitations**

* Some limits relate to class size, property name, data and string type.