COCOAHEADS PARIS

OCTOBER 2018

CODABLE XML

FRANK LEFEBVRE

PROGRAM

- Swift Codable Internals
- XML Encoder Implementation

SWIFT CODABLE

USAGE (ENCODING)

```
struct Person: Codable {
  var firstName: String
  var lastName: String
}

let person = Person(firstName: "Arthur", lastName: "Dent")

let encoder = JSONEncoder()
let jsonData = try encoder.encode(person)
let jsonString = String(data: jsonData, encoding: .utf8)

// result: {"firstName":"Arthur","lastName":"Dent"}
```

USAGE (DECODING)

```
struct Person: Codable {
  var firstName: String
  var lastName: String
}

let jsonData = """
  {"firstName":"Arthur","lastName":"Dent"}
  """.data(using: .utf8)

let decoder = JSONDecoder()
let person = try decoder.decode(Person.self, from: jsonData)

// result: Person(firstName: "Arthur", lastName: "Dent")
```

ENCODABLE & DECODABLE

- `typealias Codable = Encodable & Decodable`
- From now on: Encodable

SYNTHESIZED CODE

```
struct Person: Encodable {
  var firstName: String
  var lastName: String

  private enum CodingKeys: String, CodingKey {
    case lastName = "name"
  }


  func encode(to encoder: Encoder) throws {
    var container = encoder.container(keyedBy: CodingKeys.self)
    try container.encode(self.firstName, forKey: .firstName)
    try container.encode(self.lastName, forKey: .lastName)
  }
}
```

ENCODER IMPLEMENTATION

- `let encoder = CustomEncoder()`
- `let encoded = try encoder.encode(value)`
- CustomEncoder does not conform to Encoder
  - By convention: `func encode(_ value: Encodable) throws -> ?`
  - May return any type
  - Creates instance of Encoder-conforming type upon request
  - Invokes `value.encode(to encoder:)`

CONTAINERS

- An Encoder implementation must provide 3 types of containers:
  - singleValueContainer
  - keyedContainer
  - unkeyedContainer
- Storage
- Stack

RECURSION

```
class _CustomEncoder: Encoder {
  // ...
  struct CustomUnkeyedEncodingContainer: UnkeyedEncodingContainer {
    // ...
    mutating func encode<T>(_ value: T) throws where T : Encodable {
      var childEncoder = _CustomEncoder()
      try value.encode(to: childEncoder)
      self.storage.append(childEncoder.rootElement)
    }
  }
}
```

XML ENCODER

XML CHALLENGES

- Namespaces
- Attributes
- XML Elements = neither dictionaries nor arrays
- Tags within strings
- Whitespace collapsing
- Escaped characters, XML entities

IMPLEMENTATION DECISIONS

- Platforms: macOS, Linux
- Swift Package Manager
- Foundation XMLDocument classes
  - https://github.com/apple/swift-corelibs-foundation/blob/master/Docs/Status.md
- Output: XMLDocument
- Strong Typing
- Unit Tests

NAIVE IMPLEMENTATION

- No Namespaces
- No Attributes
- Keyed Container: [XMLElement]
- Unkeyed Container: [XMLElement] (name = “element”)
- Heavy Recursion
  - _XMLEncoder instances
  - CodingPath not implemented

NAMESPACES

- XMLNamespaceProvider
  - Mapping between qualified names and local names
- Specific treatment for Linux
  - Extension on XMLElement
  - Namespaces simulated using attributes
- XMLQualifiedKey protocol

NAMESPACES

```
struct Person: Encodable {
  var firstName: String
  var lastName: String

  private enum CodingKeys: String, CodingKey, XMLQualifiedKey {
    case firstName
    case lastName

    var namespace: String? {
      switch(self) {
      case .lastName:
        return "http://some.url.example.com/ns"
      default:
        return nil
      }
    }
  }
}
```

NAMESPACES

```
let person = Person(firstName: "Zaphod", lastName:
"Beeblebrox")
```

```
<Person xmlns:ns1="http://some.url.example.com/ns">
   <firstName>Zaphod</firstName>
   <ns1:lastName>Beeblebrox</ns1:lastName>
</Person>
```

ATTRIBUTES

- Storage
  - elements: [XMLNode]
  - attributes: [XMLNode]
- CodableXMLString<T>: ExpressibleByStringLiteral
- typealias CodableXMLInlineText = CodableXMLString<InlineTextType>
- typealias CodableXMLAttribute = CodableXMLString<AttributeType>

ATTRIBUTES

```
struct Quote: Encodable {
  var id: CodableXMLAttribute
  var firstName: String
  var lastName: String
  var text: CodableXMLInlineText
}

let quote = Quote(id: "42", firstName: "Ford", lastName: "Prefect",
text: "Six pints of bitter. And quickly please, the world's about to
end.")

<quote id="42"><firstName>Ford</firstName><lastName>Prefect</
lastName>Six pints of bitter. And quickly please, the world's about
to end.</quote>
```

CODE

- https://github.com/franklefebvre/XMLCoder
- BSD License
- Version 0.1
- Encoder only
- Swift 4.2

NEXT STEPS

- Finish implementation (superEncoder etc)
- Implement Decoder
- Write documentation
- Optimize (recursion, CodingPath)
- Follow Foundation evolution on Linux (XMLNode)
- (Later) Auto-generate structs from DTD or XMLSchema

FURTHER DOCUMENTATION

- Mike Ash: Friday Q&A, 2017-07-14, 2017-07-28
- Chris Eidhof, Ole Begemann, Ben Cohen: Advanced Swift
- Chris Eidhof, Ole Begemann: Swift Talk, episodes 115-116
- Mattt Zmuda: Flight School Guide to Swift Codable
- Kaitlin Mahar: Server Side Swift Conference 2018
  - https://kaitlinmahar.com/files/encoder_decoder_slides.pdf
- Nicolas Zinovieff: Coder & Codable
  - https://blog.krugazor.eu/2018/09/07/dictionary-coder-codable

OTHER ENCODER/DECODER IMPLEMENTATIONS

- XML (minimal): https://github.com/ShawnMoore/XMLParsing
- XML (Decoder, Recurly): https://github.com/objcio/S01E115-building-a-custom-xml-decoder
- BSON: https://github.com/mongodb/mongo-swift-driver
- In-memory: https://github.com/krugazor/DictionaryCoding
- Binary: https://github.com/mikeash/BinaryCoder

QUESTIONS

@franklefebvre