UI Tests

„Integration and UI testing with my local swift web server“

Christian Menschel
CocoaHeads Aachen 27.02.2020

Testing UI & content

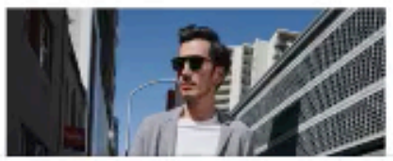
… might be easy

Möchten Sie Ihre letzten Einkäufe einsehen?

Jetzt anmelden >

Suche

Trends und Kategorien



Home Shop Favoriten Warenkorb Konto

XCUITests for a framework?

We have XCTests!

(Unit Tests should be enough)

Frameworks

Will always be Integrated


gif-finder.com

Instana
APM for iOS

Measures

- HTTP Requests
- View changes
- Errors
- more soon (frame rate,…)

MYGIGS

7d
Jan 21 - Jan 28

Live

Analyze Sessions

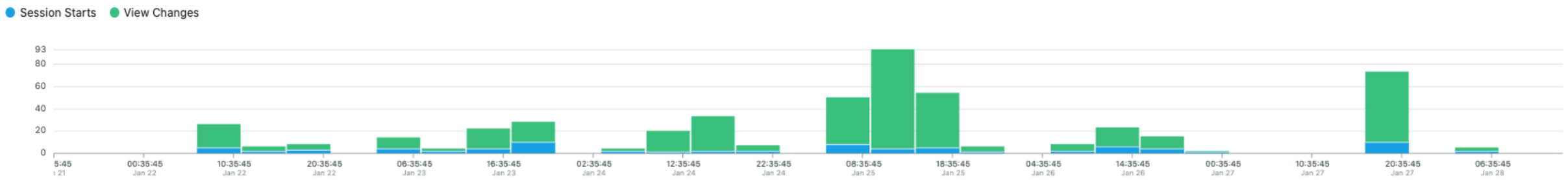
Filters | Platform | OS | Bundle | Version | Country | Subdivision | Meta

Summary | HTTP Requests | Geography | Views | Configuration

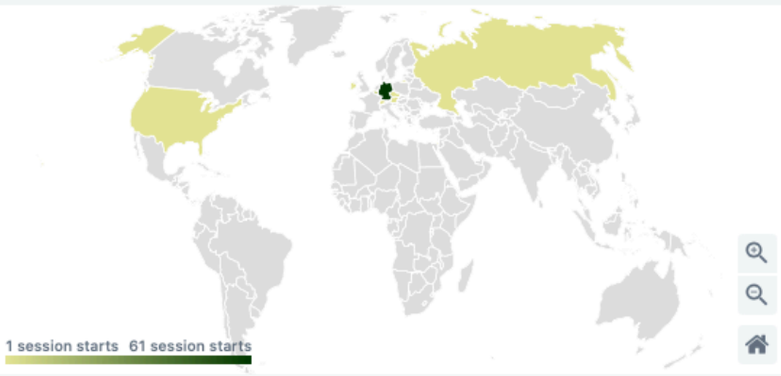
Session Starts
82

View Changes
426

Activity



Geography



Top HTTP Request Origins

Calls | Errors

https://lastfm.freetls.fastly.net	4,335
https://lastfm-img2.akamaized.net	4,293
https://e.crashlytics.com	1,374
https://api.mygigs.tapwork.de	919
https://www.eventim.de	891

View all origins

Top Views

VenuesNearBy	0
Background	0
EventsWatchList	0
EventDetails	0
VenuesWatchList	0

View all views

MYGIGS

7d Jan 21 - Jan 28 | Live

Analyze Sessions

Filters | Platform | OS | Bundle | Version | Country | Subdivision | Meta

Summary | HTTP Requests | Geography | Views | Configuration

< Back to list of HTTP request origins

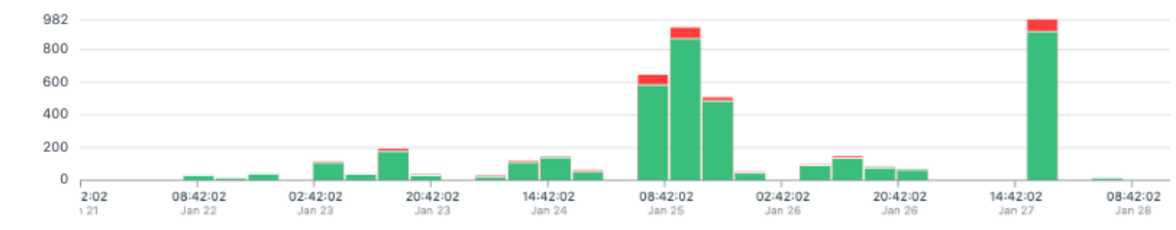
Analyze HTTP Request Origin

Origin

https://lastfm.freetls.fastly.net

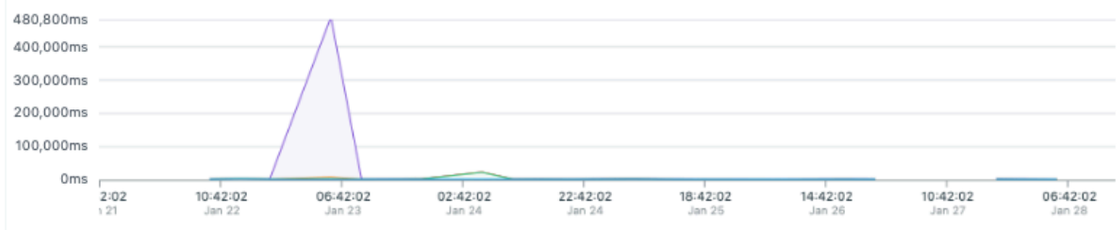
Calls

Success Failure



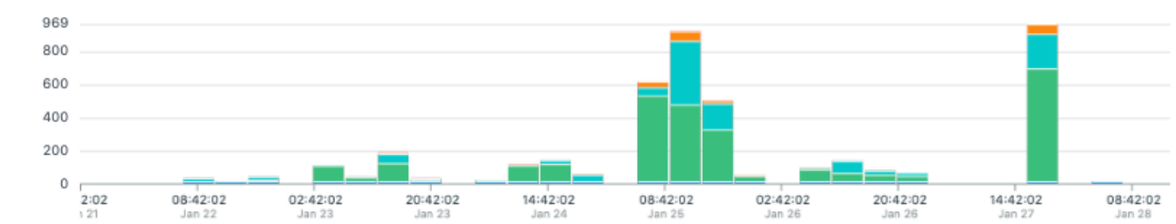
Latency

50th 90th 95th 99th Max Mean



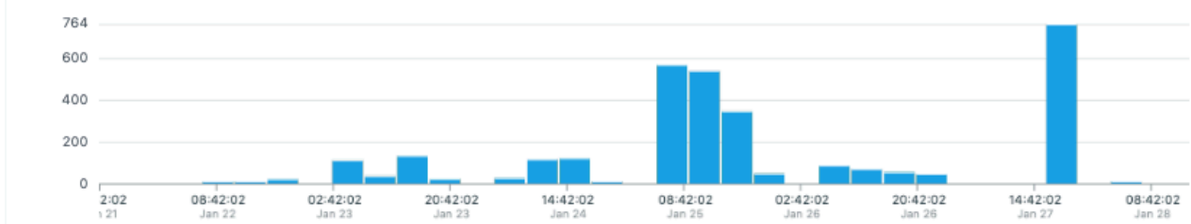
HTTP Status Code Breakdown

1XX 2XX 3XX 4XX 5XX



HTTP Method Breakdown

GET POST PUT DELETE



Views

Calls | Errors

EventsNearBy	3,605
VenueDetails	383
EventDetails	283
VenuesNearBy	27
ArtistDetails	15

View all views

Paths

Calls | Latency | Errors

/i/u/300x300/2a96cbd8b46e442fc41c2b86b821562f.jpg	153
/i/u/300x300/f2c701689a60438a89d38c13f8c1cc4d.jpg	61
/i/u/300x300/1f2016c001c8441f929859143c06c22d.jpg	48
/i/u/300x300/676e4932fc33464da6e074cf2e353004.jpg	36
/i/u/300x300/4629f1a46207443da0d7467f6703f1fd.jpg	33

View all paths

Error Types

HTTP 404	213
Cancelled	53
Timeout	6

View all error types

Use XCUITests for Integration Testing

But how to mock the backend in Swift?

There is no such Dependency Injection for XCUITests

Network.framework

Let’s use a Swift webserver

Mock Web server in Swift

It’s cool because…

- Runs in the test process

- Access to received data on the Web server

- Enables stubbing of HTTP Response (i.e. 404 or 403)

- Launch and configure individually for each test

Let’s build a test

```
override func setUp() {
    super.setUp()
}
```

```
override func setUp() {
    super.setUp()

    webserver = Webserver(port: port)
    webserver.start()
```

```
}
```

```
override func setUp() {
    super.setUp()

    webserver = Webserver(port: port)
    webserver.start()

    app = XCUIApplication()
    app.launchArguments = ["-reportingURL", "http://127.0.0.1:9999",
                           "-key", theKey,
                           "-IgnoreZIPReporting", "true"]
    app.launch()
}
```

```
func test_Launch_and_enter_url() {

    // When
    load("https://api.mygigs.tapwork.de")

}
```

```
func test_Launch_and_enter_url() {

    // When
    load("https://api.mygigs.tapwork.de")


    // Then
    verify(app.textViews.staticTexts["{\"message\":\"api.mygigs.tapwork.de\"}"])
    delay(3.0)
    webserver.verifyBeaconReceived(key: "t", value: "httpRequest")
    webserver.verifyBeaconReceived(key: "hu", value: "https://api.mygigs.tapwork.de")
    webserver.verifyBeaconReceived(key: "k", value: instanaKey)



}
```

```
func test_Launch_and_enter_url() {

    // When
    load("https://api.mygigs.tapwork.de")


    // Then
    verify(app.textViews.staticTexts["{\"message\":\"api.mygigs.tapwork.de\"}"])
    delay(3.0)
    webserver.verifyBeaconReceived(key: "t", value: "httpRequest")
    webserver.verifyBeaconReceived(key: "hu", value: "https://api.mygigs.tapwork.de")
    webserver.verifyBeaconReceived(key: "k", value: instanaKey)

    let types = webserver.values(for: "t")
    XCTAssertEqual(types.count, 3)
    XCTAssertTrue(types.contains("sessionStart"))
    XCTAssertTrue(types.contains("viewChange"))
    XCTAssertTrue(types.contains("httpRequest"))
}
```

Let’s build a Swift Web server

```
public class Webserver {

    private let queue = DispatchQueue.global()
    private let listener: NWListener
```

```
public class Webserver {

    private let queue = DispatchQueue.global()
    private let listener: NWListener

    private var connectionsByID: [Int: Connection] = [:]
    var connections: [Connection] { connectionsByID.map {$0.value} }
```

```
        connection.stop()
```

```
public class Webserver {

    private let queue = DispatchQueue.global()
    private let listener: NWListener

    private var connectionsByID: [Int: Connection] = [:]
    var connections: [Connection] { connectionsByID.map {$0.value} }

    init(port: UInt16) {
        let tcpprotocol = NWProtocolTCP.Options()
        tcpprotocol.connectionTimeout = 60
        listener = try! NWListener(using: NWParameters(tls: nil,
                                   tcp: tcpprotocol), on: NWEndpoint.Port(rawValue: port)!)
    }









            connection.stop()
```

```
public class Webserver {

    private let queue = DispatchQueue.global()
    private let listener: NWListener

    private var connectionsByID: [Int: Connection] = [:]
    var connections: [Connection] { connectionsByID.map {$0.value} }

    init(port: UInt16) {
        let tcpprotocol = NWProtocolTCP.Options()
        tcpprotocol.connectionTimeout = 60
        listener = try! NWListener(using: NWParameters(tls: nil,
                                   tcp: tcpprotocol), on: NWEndpoint.Port(rawValue: port)!)
    }

    func start() {
        listener.stateUpdateHandler = stateDidChange(to:)
        listener.newConnectionHandler = didAccept(nwConnection:)
        listener.start(queue: .main)
        print("Signaling server started listening on port \(listener.port!)")
    }
```

```
func start() {
    listener.stateUpdateHandler = stateDidChange(to:)
    listener.newConnectionHandler = didAccept(nwConnection:)
    listener.start(queue: .main)
    print("Signaling server started listening on port \(listener.port!)")
}

public func stop() {
    listener.stateUpdateHandler = nil
    listener.newConnectionHandler = nil
    listener.cancel()
    for connection in connectionsByID.values {
        connection.stop()
        connection.didStopCallback = nil
    }
}
```

```
func stateDidChange(to newState: NWListener.State) {
    switch newState {
    case .setup:
        break
    case .waiting:
        break
    case .ready:
        break
    case .failed(let error):
        print("server did fail, error: \(error)")
        stop()
    case .cancelled:
        stop()
    default:
        break
    }
}
```

```
func stateDidChange(to newState: NWListener.State) {
    switch newState {
    case .setup:
        break
    case .waiting:
        break
    case .ready:
        break
    case .failed(let error):
        print("server did fail, error: \(error)")
        stop()
    case .cancelled:
        stop()
    default:
        break
    }
}

private func didAccept(nwConnection: NWConnection) {
    let connection = Connection(nwConnection: nwConnection)
    connectionsByID[connection.id] = connection
    connection.didStopCallback = { _ in
        self.connectionDidStop(connection)
    }
    connection.start()
    print("server did open connection \(connection.id)")
}
```

```swift
public class Connection {
    private static var nextID: Int = 0
    let nwConnection: NWConnection
    var didStopCallback: ((Error?) -> Void)? = nil
    let id: Int
    var received = [String]()
    let MTU = 65536

    init(nwConnection: NWConnection) {
        self.nwConnection = nwConnection
        self.id = Connection.nextID
        Connection.nextID += 1
    }
```

```swift
public class Connection {
    private static var nextID: Int = 0
    let nwConnection: NWConnection
    var didStopCallback: ((Error?) -> Void)? = nil
    let id: Int
    var received = [String]()
    let MTU = 65536

    init(nwConnection: NWConnection) {
        self.nwConnection = nwConnection
        self.id = Connection.nextID
        Connection.nextID += 1
    }

    func start() {
        print("connection \(id) will start")
        nwConnection.stateUpdateHandler = stateDidChange(to:)
        setupReceive()
        nwConnection.start(queue: .main)
    }
```

```
public class Connection {
    private static var nextID: Int = 0
    let nwConnection: NWConnection
    var didStopCallback: ((Error?) -> Void)? = nil
    let id: Int
    var received = [String]()
    let MTU = 65536

    init(nwConnection: NWConnection) {
        self.nwConnection = nwConnection
        self.id = Connection.nextID
        Connection.nextID += 1
    }

    func start() {
        print("connection \(id) will start")
        nwConnection.stateUpdateHandler = stateDidChange(to:)
        setupReceive()
        nwConnection.start(queue: .main)
    }

    func stop(error: Error? = nil) {
        print("connection \(id) will stop")
        nwConnection.stateUpdateHandler = nil
        nwConnection.cancel()
        if let callback = didStopCallback {
            didStopCallback = nil
            callback(error)
        }
    }
```

```
private func setupReceive() {
    nwConnection.receive(minimumIncompleteLength: 1,
                         maximumLength: MTU) {
                          [weak self] (data, _, isComplete, error) in


        }
    }
}
```

```
private func setupReceive() {
    nwConnection.receive(minimumIncompleteLength: 1,
                         maximumLength: MTU) {
                          [weak self] (data, _, isComplete, error) in
        guard let self = self else { return }
        if let data = data, let received = String(data: data, encoding:.utf8) {
            print("MockWebServer \(self.id) did receive: \(received)")
            self.received.append(received)
        }


            }
        }
}
```

```
private func setupReceive() {
    nwConnection.receive(minimumIncompleteLength: 1,
                         maximumLength: MTU) {
                          [weak self] (data, _, isComplete, error) in
        guard let self = self else { return }
        if let data = data, let received = String(data: data, encoding:.utf8) {
            print("MockWebServer \(self.id) did receive: \(received)")
            self.received.append(received)
        }
        if data?.body != nil {
            self.respond()
        } else if let error = error {
            stop(error: error)
        } else {
            self.setupReceive()
        }
    }
}
```

```
func respond() {
    nwConnection.send(content: Data.response,
                      completion: .contentProcessed( {[weak self] error in
        guard let self = self else { return }
        if let error = error {
            stop(error: error)
            return
        }
        stop(error: nil)
    }))
}
```

```
func respond() {
    nwConnection.send(content: Data.response,
                      completion: .contentProcessed( {[weak self] error in
        guard let self = self else { return }
        if let error = error {
            stop(error: error)
            return
        }
        stop(error: nil)
    }))
}

    private func stateDidChange(to state: NWConnection.State) {
        switch state {
        case .setup: break
        case .waiting(let error): break
        case .preparing: break
        case .ready: break
        case .failed(let error):
            stop(error: error)
        case .cancelled: break
        default: break
        }
    }
}
```

```swift
extension Data {
    static var response: Data {
        let message = "OK"
        let statusCode = 200
        let dateString = DateFormatter.http.string(from: Date())
        return """
        HTTP/1.1 \(statusCode) \(message)\r\n
        Date: \(dateString)\r\n
        Server: Apache/2.4.25 (Debian)\r\n
        Cache-Control: no-cache, private\r\n
        Access-Control-Allow-Origin: *\r\n
        Content-Type: text/html; charset=UTF-8\n
        Connection: close
        """.data(using: .utf8) ?? Data()
    }
}
```

Recap

- Use XCUITests for integration tests

- …even there is no UI involved

- Use Network.framework to build a mock Web server

- Verify transmitted data on the Web server in the test

Christian Menschel
CocoaHeads Aachen 27.02.2020

Swift Mock Webserver

https://github.com/instana/iOSAgent/blob/master/tools/Webserver/Webserver.swift

Example UI Tests

https://github.com/instana/iOSAgent/blob/master/Dev/iOSAgentExampleUITests/

Swift Forum about the Network.framework

https://forums.swift.org/t/socket-api/19971/10

Christian Menschel
CocoaHeads Aachen 27.02.2020