SCIENCES SORBONNE UNIVERSITÉ

Communications between Watch and Phone

Fabrice.Kordon@lip6.fr

LIP6

F. Kordon - Sorbonne Université - CC2018

As an introduction…

Phone to watch

- Thanks to messages
  - **Of interest to provide elements from the main app**

Watch to Phone

- You may monitor your App from the Watch
  - **Even if it is in background**
- You use messages to exchange information
  - **Serialisation possibly integrated for free**

Involved classes

WCSession & WCSessionDelegate
Framework **WatchConnectivity**

F. Kordon - Sorbonne Université - CC2018

WCSession

Objectives

- Handles a session (both sides)

Principle

- Get a reference to the shared object
  - **Property default**
- Checks
  - **Property isPaired and isReachable**

```
class func isSupported() -> Bool
```

- Configure session
  - **Set the delegate**
- Activate the session

```
func activate()
```

F. Kordon - Sorbonne Université - CC2018

Exchanging messages

Sending (WCSession)

```
func sendMessage(_ message: [String : Any],
                 replyHandler: (([String : Any]) -> Void)?,
                 errorHandler: ((Error) -> Void)? = nil)

func sendMessageData(_ data: Data,
                     replyHandler: ((Data) -> Void)?,
                     errorHandler: ((Error) -> Void)? = nil)
```

Receiving messages (WCSessionDelegate)

```
func session(_ session: WCSession,
             didReceiveMessage message: [String : Any])

func session(_ session: WCSession,
             didReceiveMessage message: [String : Any],
             replyHandler: @escaping ([String : Any]) -> Void)

func session(_ session: WCSession,
             didReceiveMessageData messageData: Data)

func session(_ session: WCSession,
             didReceiveMessageData messageData: Data,
             replyHandler: @escaping (Data) -> Void)
```

Exchanging messages

Sending (WCSession)

```
func sendMessage(_ message: [String : Any],
                 replyHandler: (([String : Any]) -> Void)?,
                 errorHandler: ((Error) -> Void)? = nil)

func sendMessageDa( _ data: Data,
                    ...Handler: ((Da...) -> Void)?,
```

Careful with dictionary

Serialisable types required

Receiving m...te)

```
func session(_ sess...
             didReceiveMessage message: [...

func session(_ session: WCSession,
             didReceiveMessage message: [String : Any],
             replyHandler: @escaping ([String : Any]) -> Void)

func session(_ session: WCSession,
             didReceiveMessageData messageData: Data)

func session(_ session: WCSession,
             didReceiveMessageData messageData: Data,
             replyHandler: @escaping (Data) -> Void)
```

WCSessionDelegate, other methods

Handling error

```
func session(_ session: WCSession,
             activationDidCompleteWith
                  activationState: WCSessionActivationState,
             error: Error?)
```

Handling session

```
func sessionDidBecomeInactive(_ session: WCSession)
func sessionDidDeactivate(_ session: WCSession)
func sessionWatchStateDidChange(_ session: WCSession)
```

Handling reachability

```
func sessionReachabilityDidChange(_ session: WCSession)
```

File transfert

```
func session(_ session: WCSession,
             didReceive file: WCSessionFile)
```

Etc.

As a conclusion…

Useful

- Numerous use by WatchApp companions of regular Apps
  - **Music players**

Do not hesitate!

- But remember you have a distributed asynchronous system
  - **In a nice configuration (2 hosts)**