ufw.application

a lego application toy

Goals

- Modularity
- Configuration reflecting application structure
- Consistent lifecycle management
- Singleton-free (consistent dependencies)
- Modules rewiring without code change

Configuration - YAML

```
---

application:
  entities:
# ====== general purpose entities ======
   - name: LOGGER
     config: |
        [Core]
        DisableLogging=false
        LogSeverity=error

        [Sinks.Console]
        Destination=Console
        Format="%TimeStamp(format=\"%H:%M:%S.%f\")% | %Severity(format=\"%6s\")% | %ThreadPID% | %Entity% - %Tag%%Message%"
        Asynchronous=true
        AutoFlush=true

# ====== loaders ======
    - name: FIX_SESSION
      config:
        config_file: quickfix.properties


# ====== managed FIX sessions ======
    - name: ECN
      loader_ref: FIX_SESSION
      config:
        handler_ref: FIX_ROUTER
        begin_string: FIX.4.4
        sender_comp_id: CID_ROUTER
        target_comp_id: CID_ECN
        forward_exception_text: true
```

Terminology

- *entity* - an identifiable building block
- *application* - container of *entities*
- *loader* - an *entity* capable of loading other *entities*
- *lifecycle_participant* - an *entity* with application managed lifecycle

Lifecycle Phases

- init() - *lifecycle_participants* may/should discover and cache strongly typed references to each other and fail fast if anything is missing or is of a wrong type
- start() - *lifecycle_participants* may/should establish required connections, spawn threads, etc.
- up() - *lifecycle_participants* may start messaging others
- stop() - opposite of start()
- fini() - opposite of init()

Loaders

- *loaders* are *entities*
- *loaders* can load other *loaders*
- *default_loader* - a special “seed” loader used by application to load *entities* by name (including other *loaders)*
- whether or not an *entity is* loaded with loader is specified in the config (flexibility!)
- *entities* can be added to application programmatically without *loaders (e.g. default_loader)*

Concurrency

- Initialisation is single threaded (main thread)
- A single thread “application context” (main thread :)) is available
- All other concurrency is incremental to the ufw.application