2단계: 디렉토리 구조

전체 트리

```
rodi-x-svc/
│
├── bin/
│   └── rodi-x-svc.js          ← 서비스 진입점 (부트스트랩)
│
├── index.js                   ← 로봇 API 초기화 및 시뮬레이션 모드 설정
│
├── config/
│   ├── framework.json         ← MQTT, 로깅, 이벤트 토픽 설정
│   └── runtime.json           ← Python/Java 런타임 경로
│
├── modules/                   ← 서비스 자체 코드
│   ├── applicationContext/    ← [핵심] 플러그인 관리 컨테이너
│   ├── interface/             ← [핵심] 외부 이벤트 ↔ 내부 커맨드 어댑터
│   ├── internalService/       ← [핵심] 비즈니스 서비스 (페이지, 데몬)
│   ├── exceptionHandler/      ← 플러그인 에러 핸들링
│   ├── logger/                ← 로깅 (Winston + IPC)
│   ├── checker/               ← 시스템 파일/라이선스 검증
│   ├── utils/                 ← 유틸리티 함수
│   └── test/                  ← 테스트용 가상 호출자
│
├── externals/                 ← Git 서브모듈 (다른 서비스와 공유)
│   ├── core-framework/        ← DI 컨테이너, 이벤트, MQTT 매니저
│   ├── persisted-domain/      ← 데이터 영속성 레이어
│   ├── robot-api/             ← 로봇 하드웨어 API (ClinkAPI)
│   ├── ui-api/                ← UI 인터랙션 API (다이얼로그, 알림)
│   ├── script-interpreter/    ← 스크립트 실행 엔진
│   └── data-migration/        ← 데이터 마이그레이션
│
├── docs/                      ← 문서
├── utils/                     ← INtime 라이선스 파일
├── event_list                 ← 이벤트 목록 정의
├── package.json
└── X_RELEASE.md               ← 릴리즈 노트
```

modules/ 상세 구조

applicationContext/ (플러그인 관리 컨테이너)

이 프로젝트에서 가장 중요한 모듈입니다. 플러그인의 로딩부터 실행까지 전체를 관장합니다.

```
applicationContext/
├── index.js                   ← 메인: 컨테이너 초기화, 어셈블리 로딩
│
├── assembly/                  ← 플러그인 발견 및 로딩
│   ├── assemblyDirectoryInfo.js   ← 디렉토리에서 .asar/.js 어셈블리 탐색
```

```
│   ├── extractor/
│   │   └── asarAssemblyExtractor.js ← ASAR 아카이브 추출
│   └── loader/
│       └── jsAssemblyLoader.js    ← JS 어셈블리 로딩 → AssemblyCatalog
│
├── inject/                    ← 의존성 주입 및 도메인 모델
│   ├── index.js               ← Injector 메인 (APIObject 등 생성)
│   ├── contribution/
│   │   └── daemon/            ← 데몬 서비스 Contribution 래퍼
│   ├── domain/
│   │   ├── api/               ← 14개 도메인 모델
│   │   │   ├── robotModel/    ← 로봇 상태, 좌표, 조인트
│   │   │   ├── ioModel/       ← I/O 포트 관리
│   │   │   ├── eventModel/    ← 이벤트 모델
│   │   │   ├── programModel/  ← 프로그램 노드 (if, loop, move 등)
│   │   │   ├── variableModel/ ← 변수 관리
│   │   │   ├── functionModel/ ← 함수 관리
│   │   │   ├── commandModel/  ← 명령 모델
│   │   │   ├── coordinateModel/ ← 좌표 모델
│   │   │   ├── controllerModel/ ← 컨트롤러 모델
│   │   │   ├── tcpModel/      ← TCP 모델
│   │   │   ├── systemSettings/ ← 시스템 설정
│   │   │   ├── utilsModel/    ← 유틸리티 모델
│   │   │   ├── rodiModel/     ← RODI 모델
│   │   │   └── identifierBuilder/ ← 식별자 생성
│   │   ├── data/              ← 스크립트 변수/함수/프로그램노드 맵
│   │   ├── script/            ← 스크립트 생성기
│   │   └── ui/                ← UI 페이지 정의 (xPages.json)
│   └── ui/
│       └── xPages.json        ← X 컴포넌트 정의
│
├── presentation/              ← UI 프레젠테이션
│   └── core.js                ← Factory-Register-Selector 패턴
│
├── service/                   ← 서비스 등록/검증
│   ├── context.js             ← ServiceContext (플러그인이 사용)
│   ├── validator.js           ← 서비스 타입 검증
│   └── selector.js            ← 서비스 타입별 선택
│
└── data/                      ← 데이터 저장
    ├── repository.js          ← Repository 인터페이스
    └── store.js               ← 파일 기반 KV 저장소
```

interface/ (이벤트 ↔ 커맨드 어댑터)

외부(MQTT)와 내부(비즈니스 로직)를 연결하는 브릿지 역할입니다.

```
interface/
├── sdkCommand.js              ← 커맨드 핸들러 정의 (22개 커맨드)
└── sdkService.js              ← MQTT 구독 → sdkCommand 라우팅
```

데이터 흐름: MQTT 이벤트 → sdkService (구독) → sdkCommand[이벤트명] (핸들러 실행) → 결과 콜백

internalService/ (비즈니스 서비스)

플러그인의 각 서비스 타입에 대한 실행 및 관리를 담당합니다.

```
internalService/
├── page/                          ← 페이지 관련 서비스
│   ├── pageDispatcher.js          ← 페이지 열기/닫기/이벤트 라우팅
│   ├── extensionPageService.js    ← Extension 서비스 실행/관리
│   ├── programNodePageService.js  ← ProgramNode 서비스 실행/관리
│   └── widgetNodePageService.js   ← Widget 서비스 실행/관리
│
├── daemon/
│   └── index.js                   ← Daemon 서비스 실행/관리
│
└── interop/                       ← 서비스 간 통신
    ├── serviceInfoProvider.js     ← 서비스 메타데이터 제공
    └── marshal/                   ← 직렬화/역직렬화
        ├── extensionMarshal.js
        ├── programMarshal.js
        ├── daemonMarshal.js
        ├── widgetMarshal.js
        └── scriptMarshal.js
```

externals/ 상세

externals/는 Git 서브모듈로, 다른 rodi 서비스들과 공유하는 코드입니다.

| 서브모듈 | 핵심 파일 | 역할 |

|---|---|---|

| core-framework | index.js, modules/ | DI 컨테이너, ModuleProvider, EventManager(MQTT), TopologyManager |

| persisted-domain | persistedDomain.js | 설정, 변수, 좌표, 로봇모델 등의 영속화 |

| robot-api | index.js, clink_api.js | ClinkAPI 래퍼, 로봇 DLL 호출, 시뮬레이션 DLL 전환 |

| ui-api | index.js, userInteraction/ | 메시지 다이얼로그, 브라우저, 알림 UI |

| script-interpreter | interpreter.js, executor.js | 플러그인 스크립트 분석/실행/검증 |

| data-migration | data/, program/, system/ | 데이터 버전 마이그레이션 |

> 주의: externals 코드를 수정하면 다른 서비스에도 영향을 줍니다. 변경 전 반드시 영향 범위를 확인하세요.

주요 파일 빠른 참조

"이 기능을 수정하려면 어디를 봐야 하는가?"

| 수정 대상 | 파일 위치 |

|---|---|

| MQTT 이벤트 토픽 추가/수정 | config/framework.json |

| 새 커맨드 핸들러 추가 | modules/interface/sdkCommand.js |

| 커맨드 이벤트 구독 추가 | modules/interface/sdkService.js |

| 플러그인 로딩 로직 | modules/applicationContext/assembly/ |

| 페이지 열기/닫기 로직 | modules/internalService/page/pageDispatcher.js |

| 데몬 서비스 관리 | modules/internalService/daemon/index.js |

| 도메인 모델 (로봇, IO 등) | modules/applicationContext/inject/domain/api/ |

| 에러 핸들링 | modules/exceptionHandler/XPluginExceptionHandler.js |

| DI 컨테이너/모듈 등록 | bin/rodi-x-svc.js (moduleProvider 등록부) |

| 시뮬레이션 모드 전환 | index.js |

| 서비스 디스커버리(토폴로지) | externals/core-framework/modules/topologyManager.js |

다음 단계

파일 배치를 파악했으면, 3단계: 아키텍처에서 이 파일들이 어떤 설계 원칙으로 연결되어 있는지 살펴봅니다.