프론트 구조와 프로그래밍 (2021.4.7)

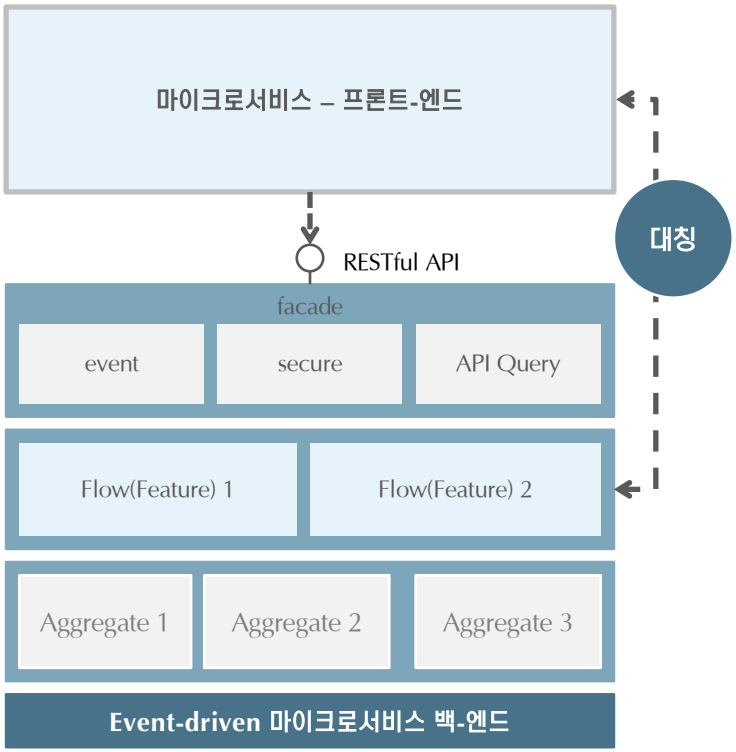
목차

posco ict

1. 프론트엔드 애플리케이션 이해(1/2)

posco ict

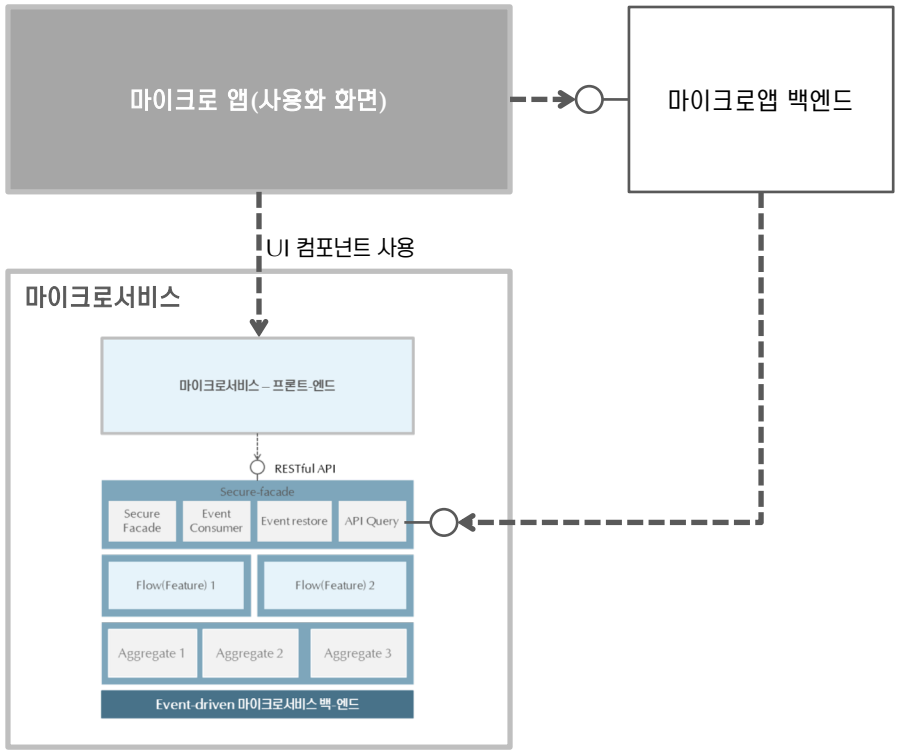
✓ 마이크로서비스는 백엔드와 프론트엔드가 API로 통신하며 한 페어를 이루어 작동합니다.
✓ 마이크로서비스 프론트엔드는 해당 서비스가 제공하는 업무를 수행하는 UI 컴포넌트로 이루어져 있습니다.
✓ 백엔드 설계와 유사하게 프론트엔드도 레이어를 분리하여 각 레이어의 역할과 책임을 명확히 하였습니다.
✓ 마이크로서비스는 최종 사용자 UI인 마이크로앱들에 컴포넌트로 포함되어 재사용될 수 있습니다.



1. 프론트엔드 애플리케이션 이해(2/2)

posco ict

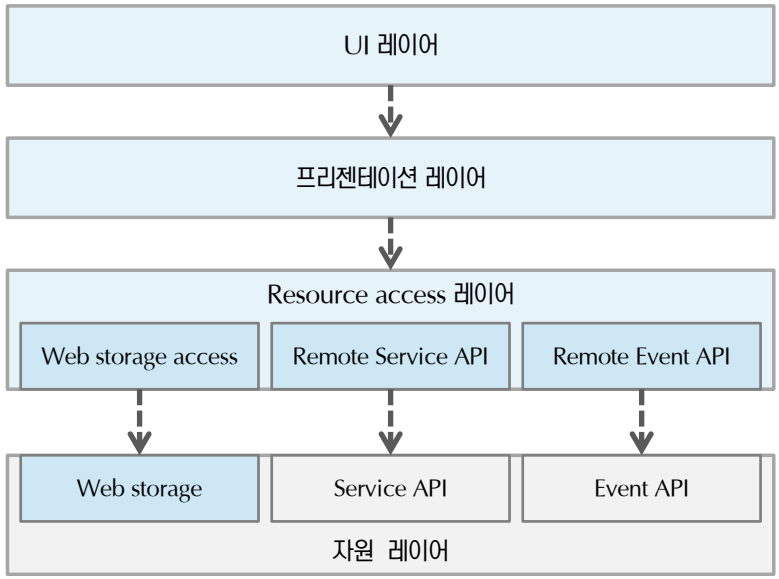
- ✓ 마이크로앱은 마이크로서비스(의 프론트와 백엔드)를 사용하여 구성한 최종 사용자 UI입니다.
- ✓ 마이크로앱은 주로 마이크로서비스의 프론트엔드를 조합하여 사용하지만, 자체 UI 컴포넌트도 포함할 수 있습니다.
- ✓ 마이크로서비스들 간의 복잡한 협업이 필요한 경우, 마이크로앱 백엔드를 중간에 두어 프론트엔드의 부담을 줄이는 협업 로직을 작성할 수 있습니다.



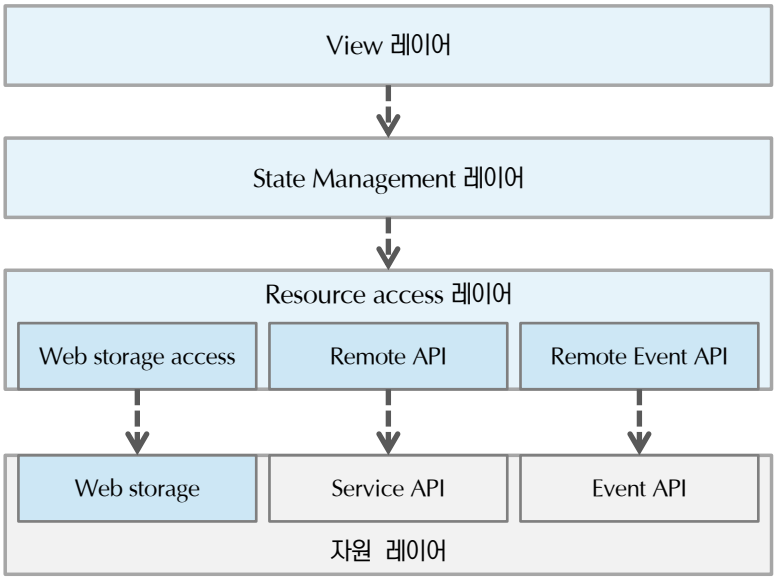
2. 레이어 설계(1/8) – 전형적인 레이어

posco ict

- ✓ 마이크로서비스 프론트-엔드는 하나의 Application으로 보고 설계합니다.
- ✓ 하지만, 프론트-엔드 특성으로 때문에 도메인 로직보다는 프리젠테이션 로직을 가지므로, 프리젠테이션 레이어를 둡니다.
- ✓ HTML5 표준에 로컬 웹 스토리지를 정의함으로써, 자체 저장소를 가질 수 있습니다.
- ✓ 서비스 API 외에 이벤트 API를 두면, 클릭 스트림 분석 등에 효율적으로 활용할 수 있습니다.



[전통적인 클라이언트 APP based on Layered Style]

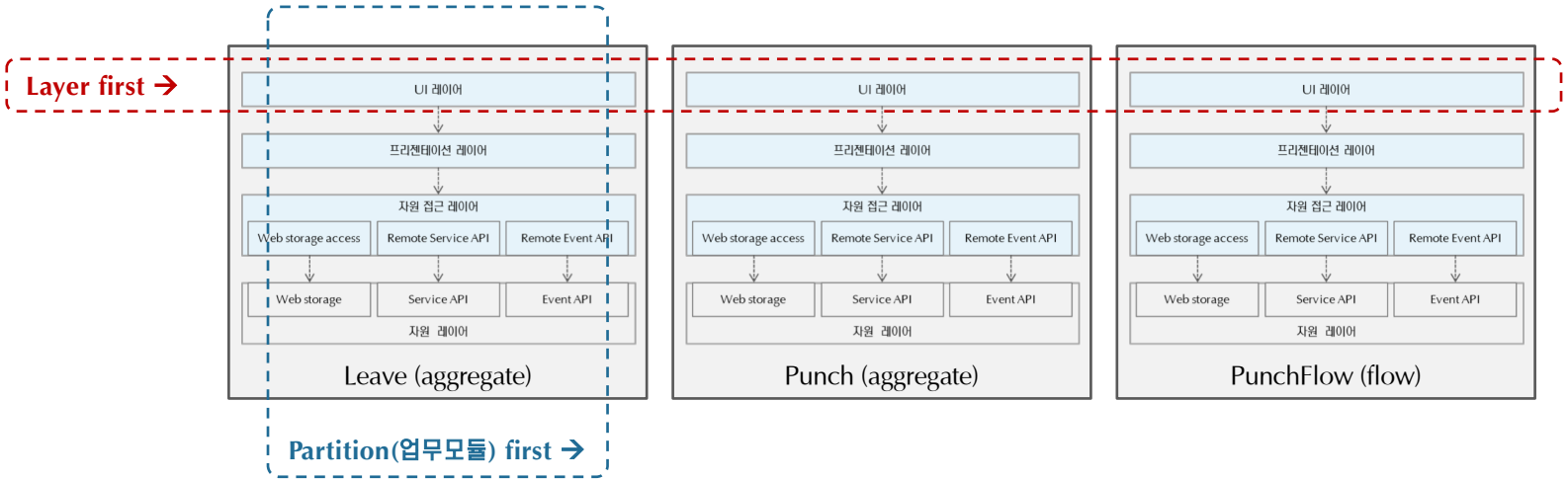


[Single Page APP based on Layered Style]

2. 레이어 설계(2/8) – 접근 방법

posco ict

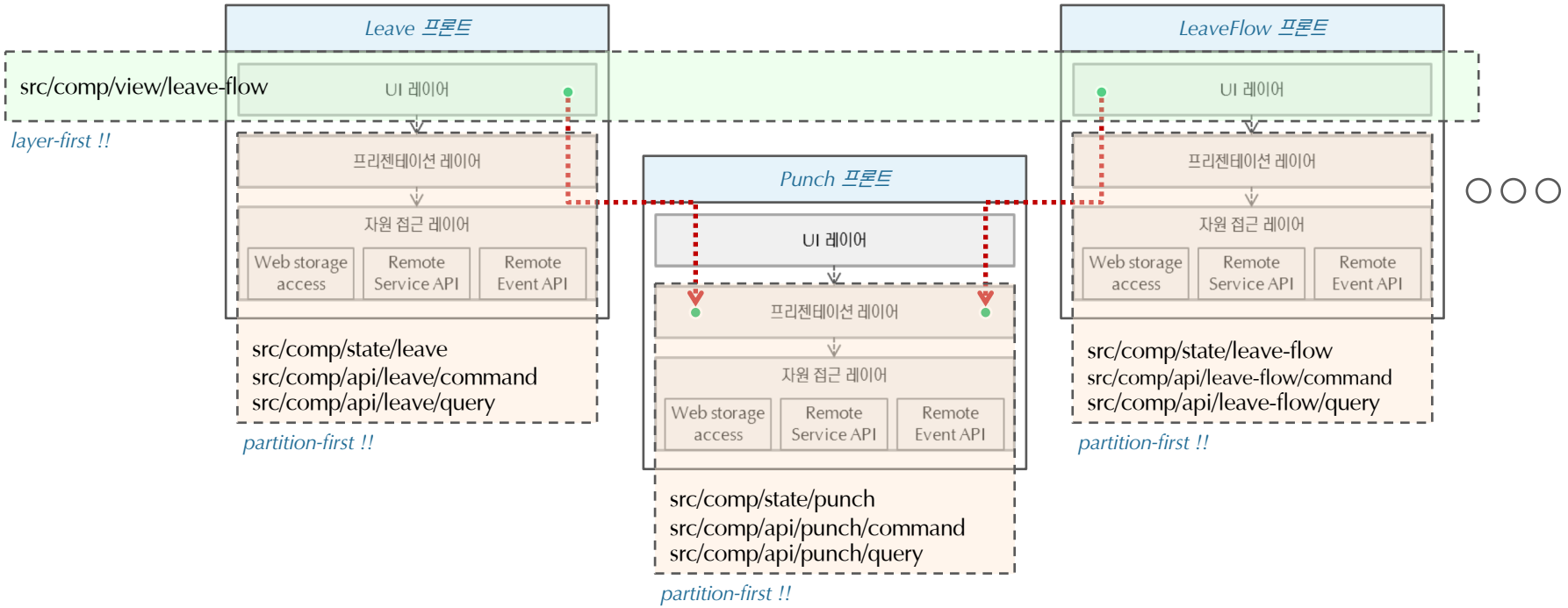
✓ 레이어 별로 패키지를 구성하는 것이 일반적이지만, 패키지가 너무 많아질 수 있습니다.
✓ 따라서, 레이어와 파티션(업무 모듈)을 조합하여 최적의 패키지 구조를 설계하여야 합니다.
✓ 프리젠테이션과 자원 접근을 묶어서 하나의 패키지로 처리하는 것이 효율적입니다.
✓ UI 컴포넌트는 여러 모듈의 프리젠테이션 로직을 사용하여 구성하는 것이 일반적이므로 하나로 묶어 줍니다.



2. 레이어 설계(3/8) – 분리 기준

posco ict

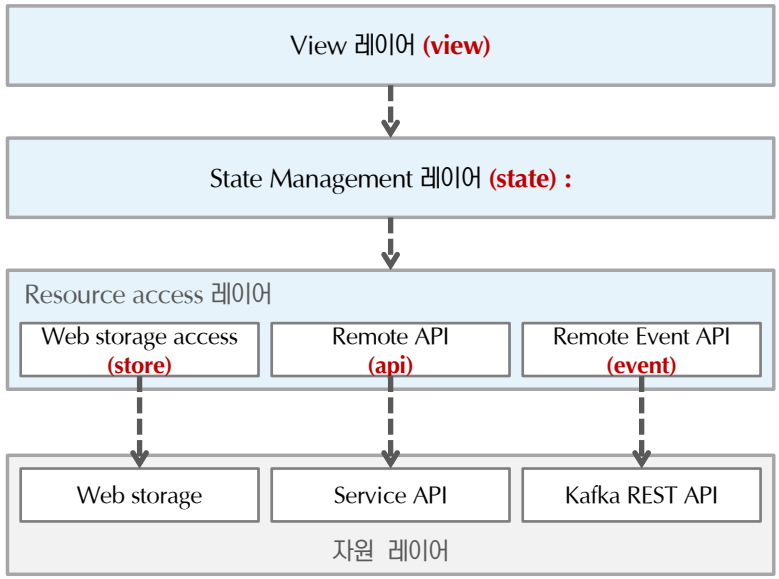
- ✓ UI 레이어에서 View 컴포넌트를 설계할 때, 다른 모듈의 프리젠테이션을 볼 수 있고, 보아야 합니다.
- ✓ 이것은 백엔드에서 Flow 모듈이 아래 계층에 있는 여러 Aggregate 모듈을 보는 것과 같은 맥락입니다.
- ✓ UI 레이어는 flow(feature) 단위로 패키지를 구성합니다. flow는 언제든 독립 마이크로서비스로 분가할 수 있습니다.
- ✓ 프리젠테이션 이하 레이어는 모듈 단위 응집성이 강하므로, 모듈 단위로 패키징을 합니다.



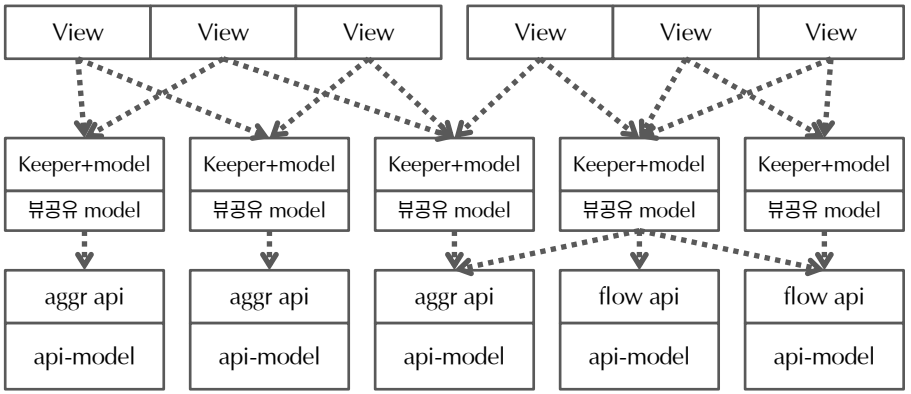
2. 레이어 설계(4/8) – Layer 1

posco ict

- ✓ 뷰 레이어는 뷰 컴포넌트를 기준 단위로 구성하되, 피처(flow)를 기준으로 묶음을 만듭니다.
- ✓ State 관리자, 또는 State 컨테이너는 프론트 전에서 중재자 역할을 하며 State를 관리합니다. → Keeper 라고 부름
- ✓ 자원 접근 레이어에서 대부분의 활동은 원격 api 호출입니다. 간혹 웹 스토리지나 이벤트 플랫폼을 사용하는 경우도 있습니다.
- ✓ 레이어 별로 파티션을 위한 규칙이 서로 다르므로 규칙을 정확하게 이해하고 패키지를 구성하여야 합니다.



[Single Page APP based on Layered Style]

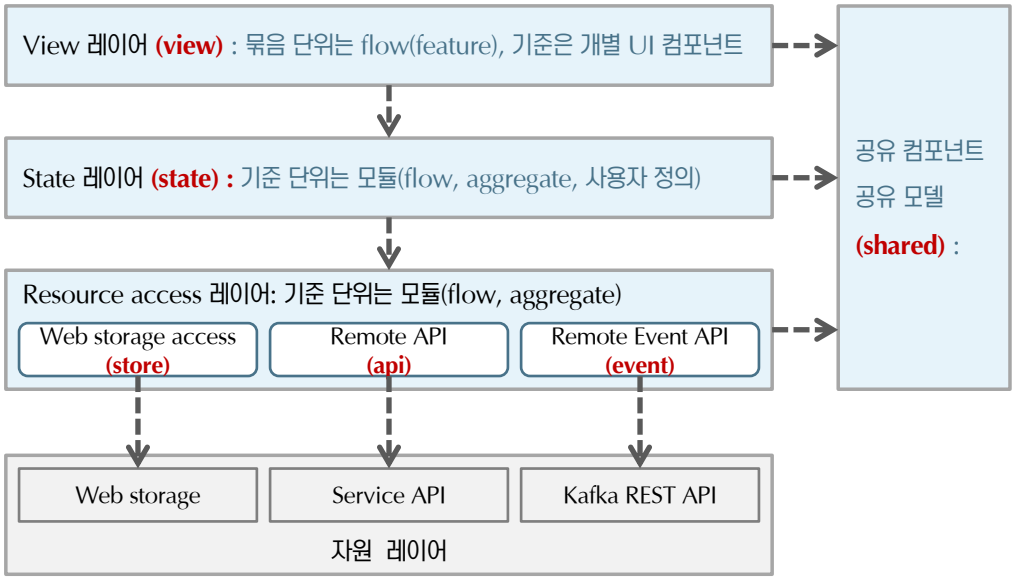


- Keeper : State container 또는 State manager를 의미합니다. State keeper.
- Api-model: api모델은 백엔드 도메인 모델로, 접미어(Base)를 붙입니다.

2. 레이어 설계(5/8) – Layer 2

posco ict

✓ 각 레이어 별로 묶음 단위는 서로 다릅니다. 앞 장에서 파티션 기반 분리에서 파티션의 단위가 다르기 때문입니다.
✓ view 레이어의 묶음은 flow(feature)이고 그 아래 각 UI 컴포넌트 단위 파티션을 둡니다.
✓ state는 view와 api를 연결하는 브로커/컨테이너/컨트롤러 역할이며, 기준은 모듈(flow, aggregate, 사용자 정의)입니다.
✓ api는 백엔드의 REST API 발행의 기준이 되는 모듈과 대칭으로 구성합니다.

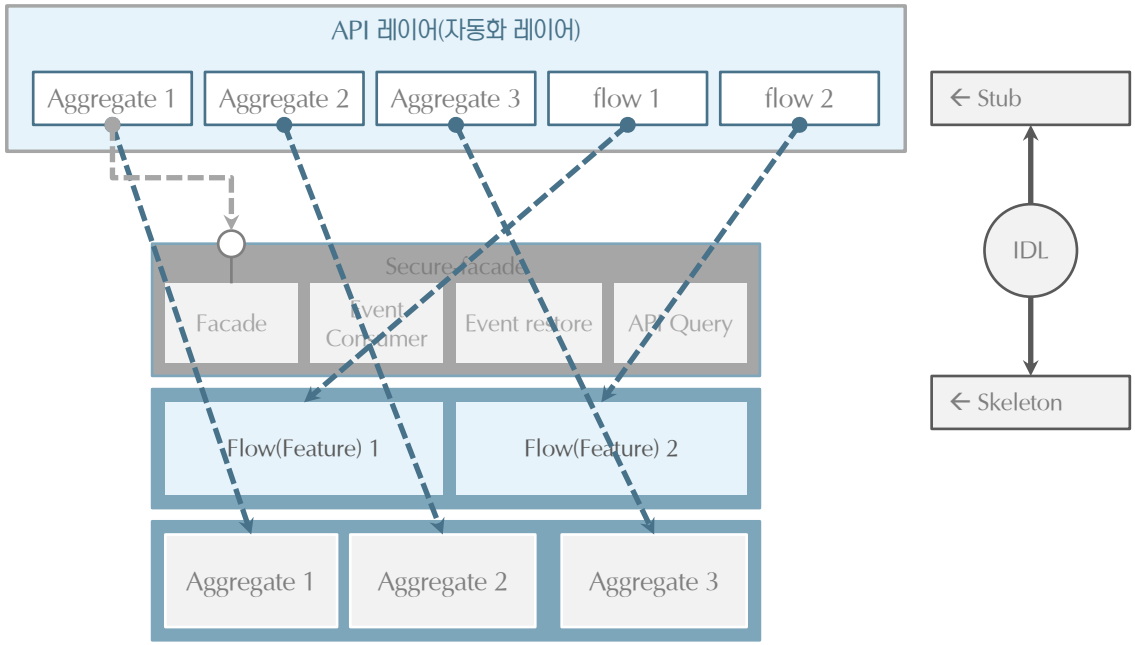


[Single Page APP based on Layered Style]

2. 레이어 설계(6/8) – API Stub

posco ict

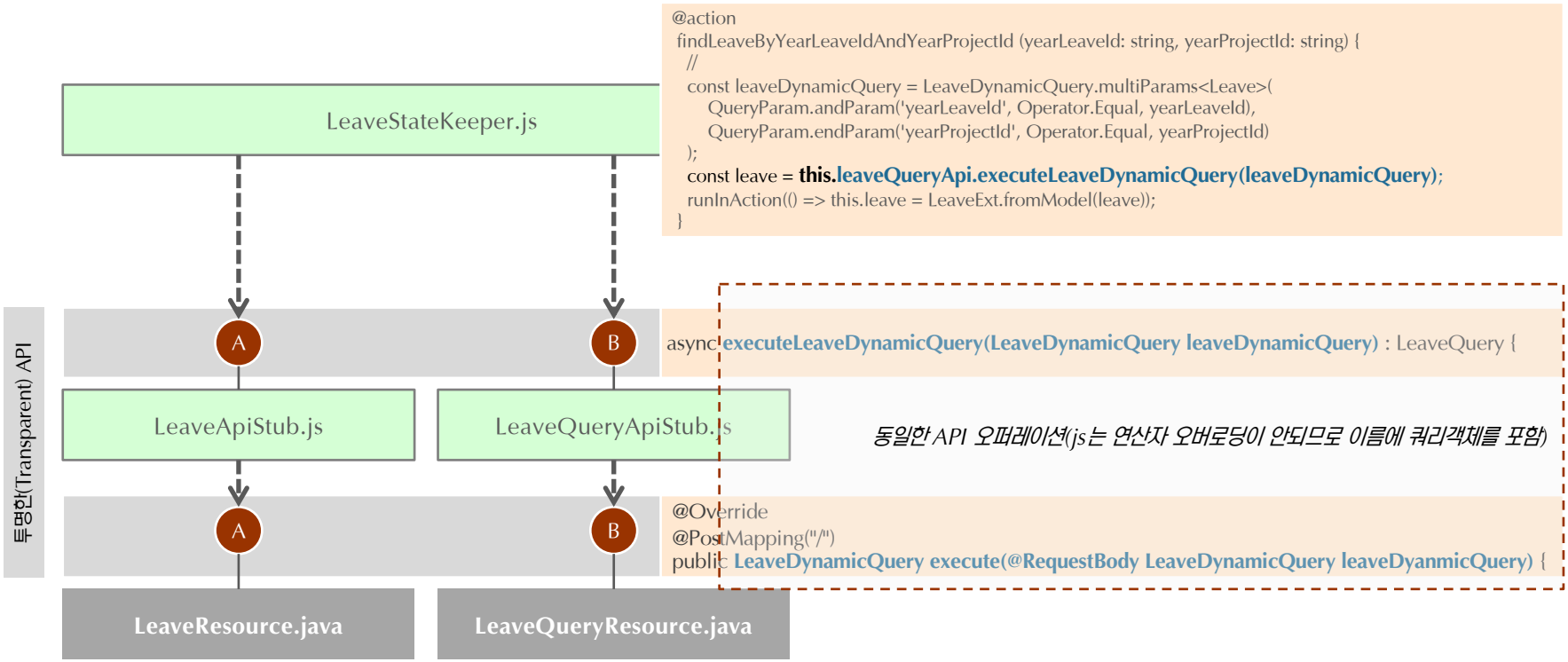
- ✓ API 레이어의 모듈은 서비스 백엔드의 모듈과 1:1로 매핑하는 방식으로 구성합니다.
- ✓ API 레이어는 서비스 Stub 역할을 가지고 있으며, 백엔드 RESTful API로부터 생성하는 것을 목표로 합니다.
- ✓ 전통적인 통신 인프라 소프트웨어를 설계할 때, 인터페이스 파일(IDL)로부터 Stub과 Skeleton은 자동으로 생성하였습니다.
- ✓ 동일한 개념을 적용하되, 백엔드에 미리 생성한 RESTful Resource를 바탕으로 API Stub 생성을 목표로 합니다.



2. 레이어 설계(7/8) – API Stub

posco ict

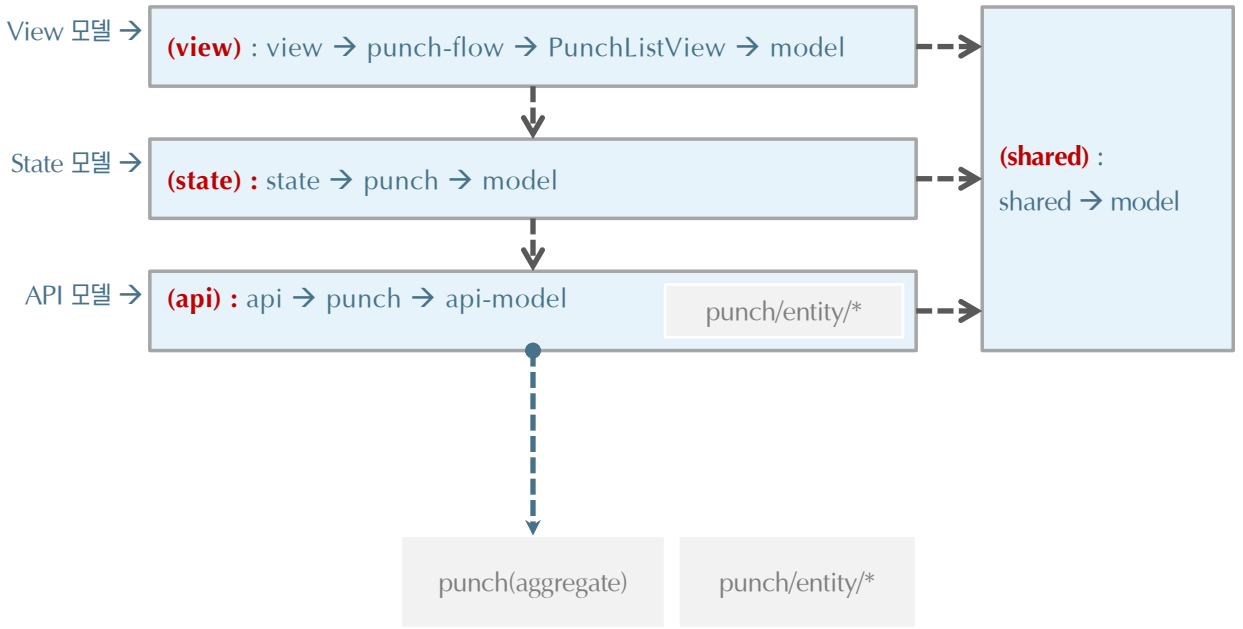
- ✓ API 레이어의 모듈 이름을 api이라고 하는 이유는 백엔드 API와 동일하다는 것을 의미합니다.
- ✓ 따라서, ApiStub 안에는 어떤 로직도 둘 수 없습니다. LeaveStateKeeper는 ApiStub이 원격에 있는 REST API라고 생각하고 동일한 인터페이스 오퍼레이션을 사용하여야 합니다.
- ✓ JavaScript/TypeScript에서 연산자 오버로딩이 없는 관계로 오퍼레이션 이름에 패러미터/리턴타입 이름을 추가합니다.



2. 레이어 설계(8/8) – 레이어별 모델

posco ict

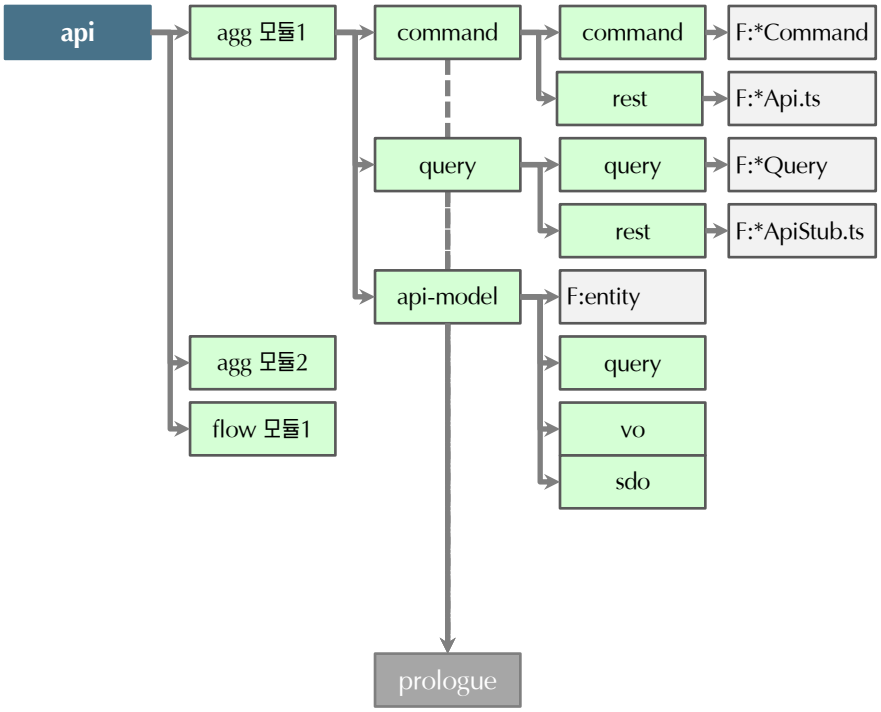
✓ 각 레이어는 레이어 모델을 갖고 있습니다. 레이어의 목적에 어울리는 정보 모델을 정의하여 사용합니다.
✓ view 레이어의 각 개별 UI 컴포넌트(*View) 별로 내부에서 사용할 목적으로 모델을 정의합니다.
✓ state 레이어의 각 모듈 안에는 API 모듈과 일치하는 모듈 또는 사용자 정의 모듈이 있고 그 속에 모델을 정의합니다.
✓ api 레이어의 모듈은 백엔드의 모듈 정보와 일치하는 모델을 정의하여 사용합니다. ← 백엔드로 부터 생성 가능

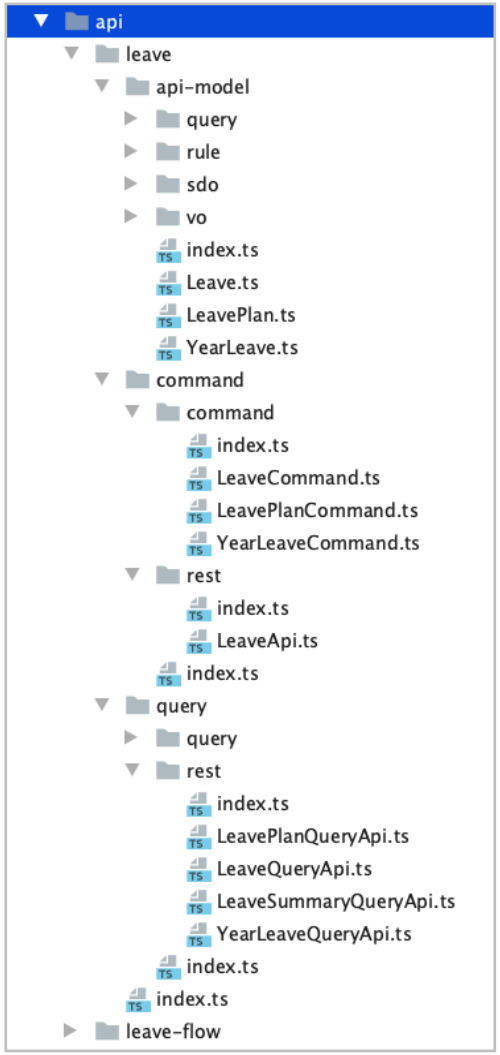


3. 패키지 구조 설계(1/6) – api 패키지

posco ict

- ✓ API는 백엔드의 모듈 단위로 묶음을 구성합니다. → aggregate, flow 동일 계층
- ✓ 단위 모듈 아래, api-model, command, query 서브-패키지를 둡니다.
- ✓ 서브-패키지 구조는 백엔드의 Java 패키지 구조와 동일하게 구성합니다.
- ✓ API는 모두 코드 자동화 대상입니다. 확장이 필요할 경우, 위 레이어에서 수행합니다.

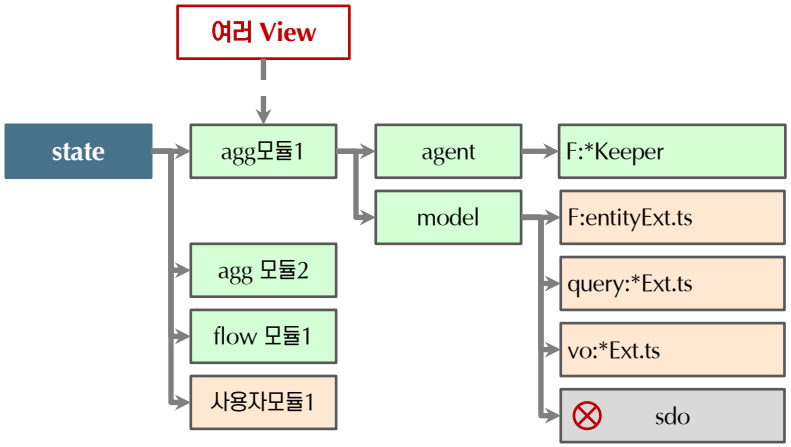




3. 패키지 구조 설계(3/6) – state 패키지

posco ict

- ✓ state 레이어의 묶음 기준은 우선 API의 모듈 단위, 다음으로 사용자 모듈, 두 가지 유형이 있습니다.
- ✓ api 기준 모듈의 경우, 처음에는 자동으로 생성되고, 확장 파일을 직접 추가할 수 있습니다.
- ✓ state 모듈(State Container, State Controller, State Manager)은 주요 로직으로 역할 이름을 Keeper로 정합니다.
- ✓ state 모듈은 여러 View 컴포넌트에서 사용할 수 있으며, 일부 자동화가 가능합니다.

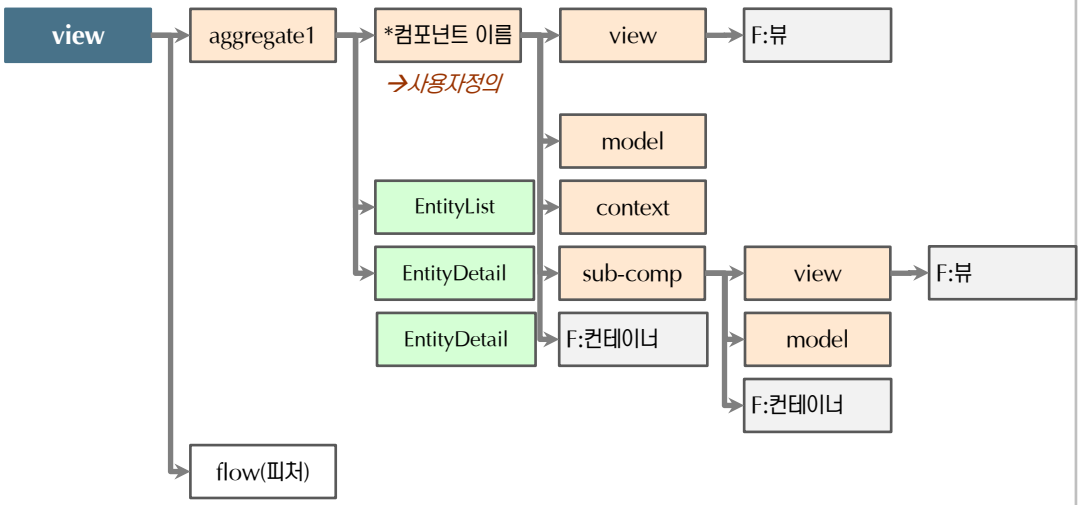


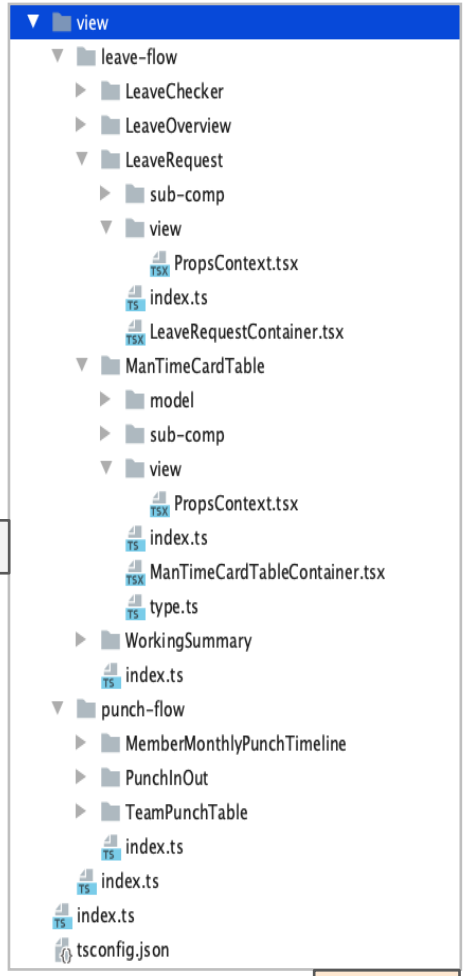
*수작업 블럭
*자동화 블럭

3. 패키지 구조 설계(4/6) – view 패키지

posco ict

- ✓ view 레이어는 Aggregate과 피처 단위(flow 또는 feature)로 묶음을 만듭니다.
- ✓ 각 Aggregate와 피처 속에 UI 컴포넌트 단위로 패키지를 구성합니다.
- ✓ UI 컴포넌트의 패키지 이름은 파스칼 케이스를 적용하여 정의합니다.
- ✓ UI 컴포넌트 패키지의 이름이 곧 외부에서 인식하는 컴포넌트 이름이 됩니다.





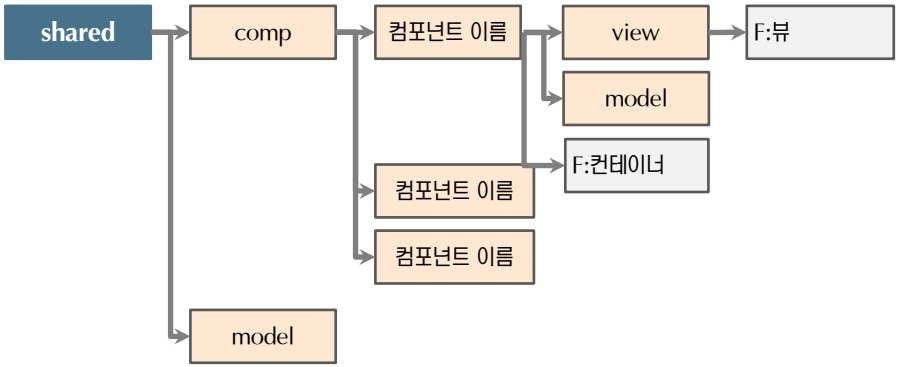
*수작업 블럭

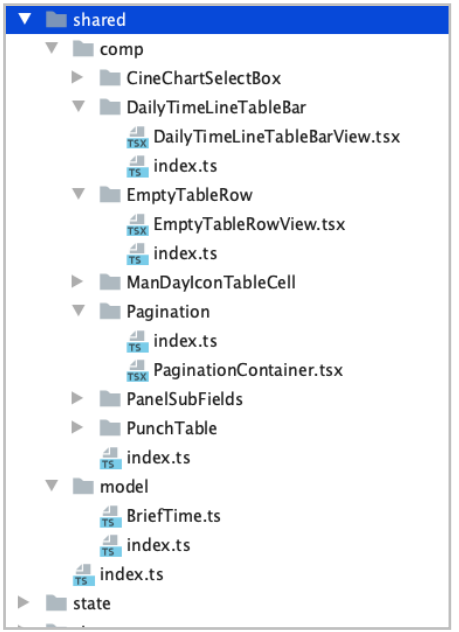
*자동화 블럭

3. 패키지 구조 설계(5/6) – shared 패키지

posco ict

✓ shared 패키지는 comp(UI 컴포넌트 패키지)와 모든 공유 model(UI 컴포넌트가 공유하는 모델) 패키지로 구분합니다.
✓ 공유 UI 컴포넌트는 카멜 케이스로 표현한 컴포넌트 이름을 가진 패키지에 둡니다.
✓ 컴포넌트 구조는 다른 UI 컴포넌트와 마찬가지로 view와 model 패키지로 구성합니다.



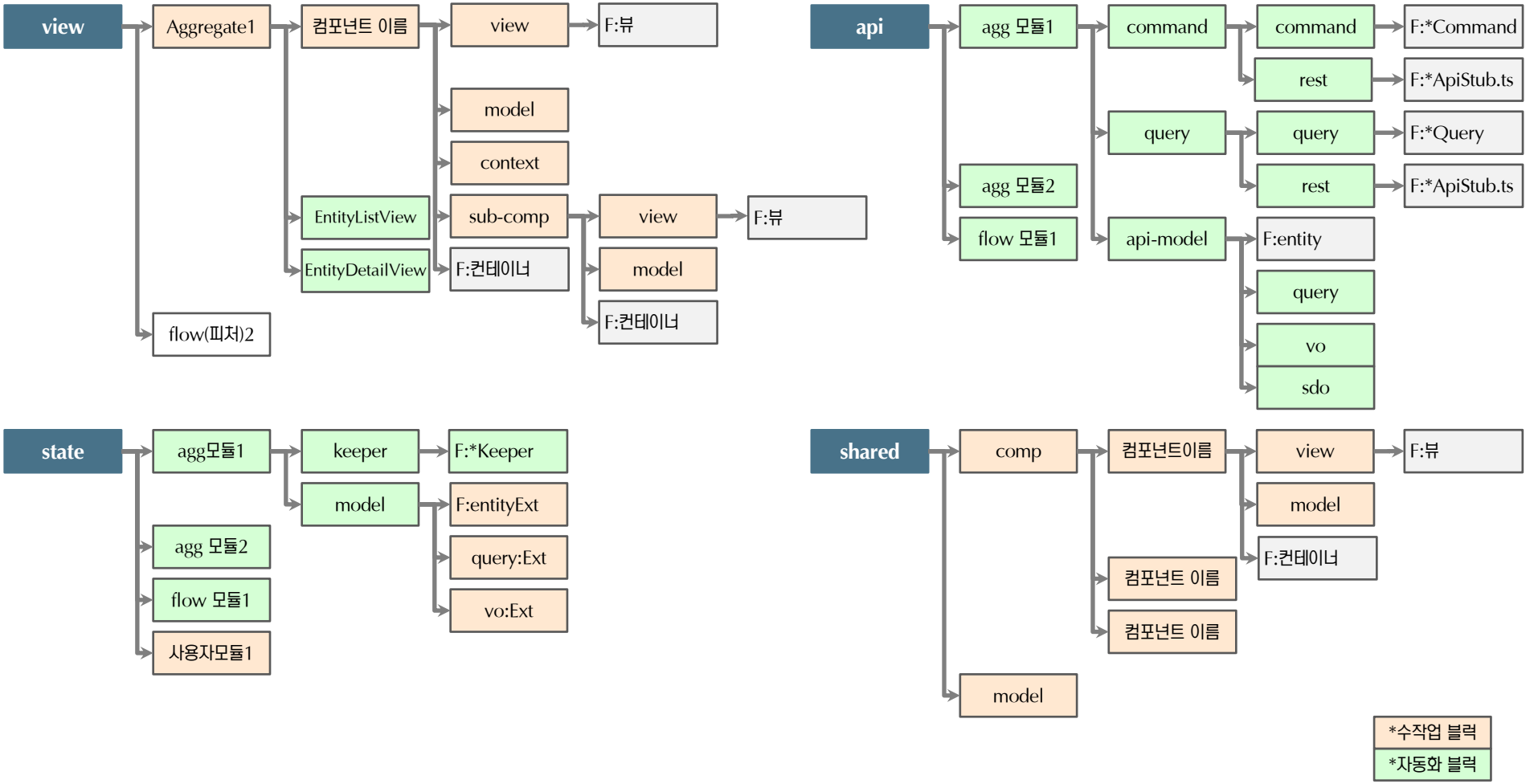


*수작업 블럭
*자동화 블럭

3. 패키지 구조 설계(6/6) – 모든 패키지

posco ict

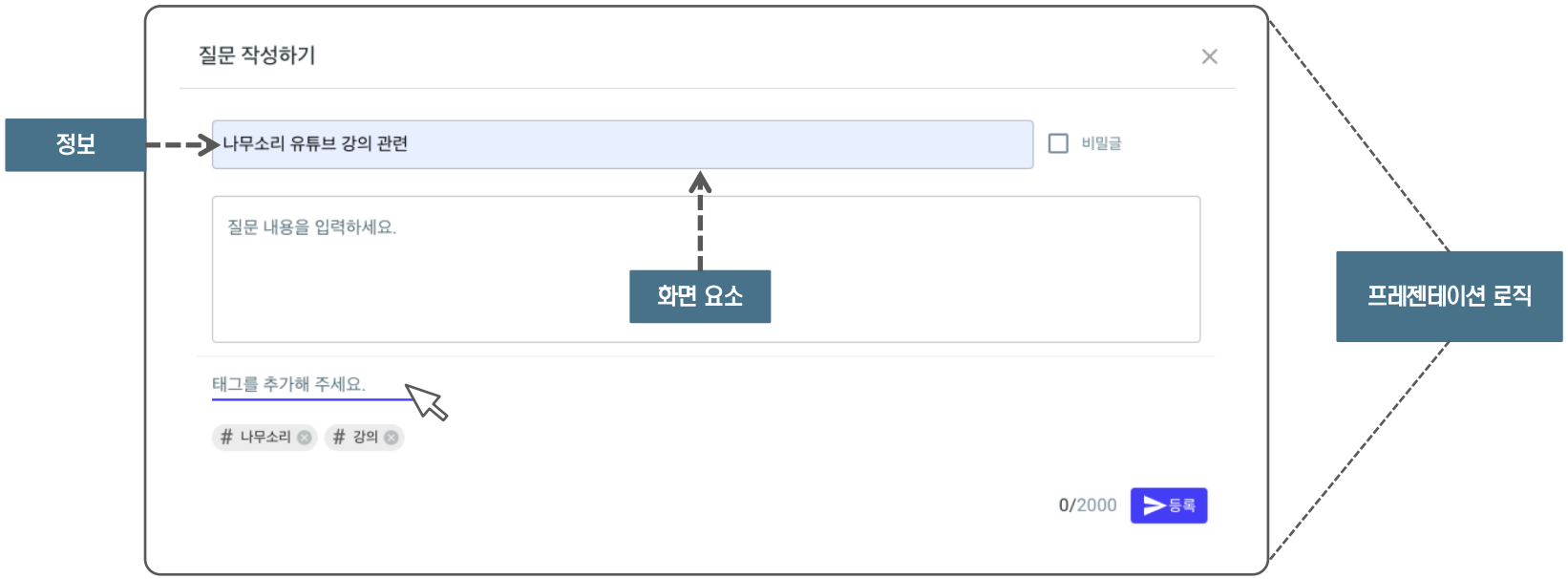
✓ 모든 패키지를 모아 보면 다음과 같습니다.



4. UI 컴포넌트 설계(1/2)

posco ict

- ✓ UI 컴포넌트의 구성은 정보, 화면 요소, 프레젠테이션 로직 세 가지로 볼 수 있습니다.
- ✓ 정보는 상태 관리 모듈의 영역이기 때문에 UI 컴포넌트는 상태 관리 모듈(StateKeeper)로부터 정보를 읽고, 액션을 전달하여 정보의 변화를 야기합니다.
- ✓ 화면 요소는 백엔드 또는 프론트엔드 자체 저장소의 정보를 표현하기 위한 것으로, 구조를 잡는 DOM과 style을 말합니다.
- ✓ 프레젠테이션 로직은 사용자의 액션 이벤트와 데이터를 받아 화면을 변화시키거나 상태 관리 모듈로 액션을 위임합니다.



4. UI 컴포넌트 설계(2/2)

posco ict

- ✓ UI 컴포넌트 내부에서 화면 요소를 표현하는 부분은 view, 프레젠테이션 로직은 container로 분리였습니다.
- ✓ view는 가능하면 자체적으로 state 관리하지 않습니다. 예외적으로 한 영역 안에서 단지 보여 주기 위한 속성일 경우에는 사용하기도 합니다.
- ✓ Container는 StateKeeper를 주입 받아 사용하거나 Context Provider가 되거나 UI를 위한 로직을 수행합니다.