통 합 구 현

대용량 서비스 레퍼런스 아키텍처 1

한국기술교육대학교
온라인평생교육원

학습내용

- 레퍼런스 아키텍처
- Access Layer (웹 캐시, 리버스 프록시, API 게이트웨이)
- Access Layer (계정 관리, 시스템 연동)

학습목표

- 레퍼런스 아키텍처를 구성하는 각 레이어의 역할을 설명할 수 있다.
- Access Layer를 구성하는 웹 캐시, 리버스 프록시, API 게이트 웨이의 각 요소를 설명하고 사용할 수 있다.
- Access Layer를 구성하는 계정 관리, 시스템 연동의 각 요소를 설명하고 사용할 수 있다.

1. 레퍼런스 아키텍처

1) 대용량 서비스 레퍼런스 아키텍처 개요

(1) 대용량 서비스 레퍼런스 아키텍처

대용량 서비스 레퍼런스 아키텍처의 정의

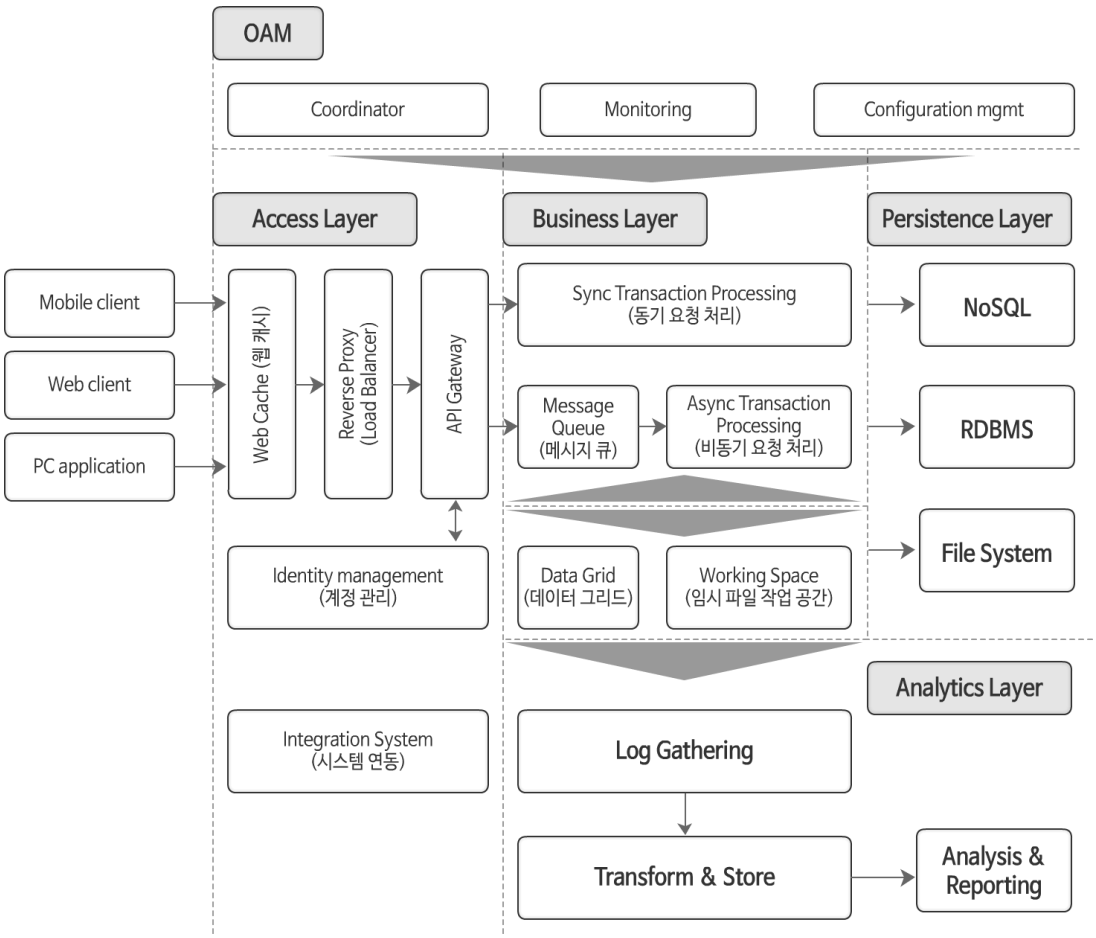
- **대용량 서비스를 위한 플랫폼에 대한 레퍼런스 아키텍처로 대부분의 온라인 서비스 처리에서 사용되는 구조**

- SOA를 기반으로 하는 아키텍처
- 요구사항이나 시스템의 활용 방법에 따라서 변형 가능
- 웹 사이트나 모바일 애플리케이션들이 클라이언트로서 API를 사용해 서버 플랫폼과 통신하여 비즈니스 로직을 처리하는 형태

1. 레퍼런스 아키텍처

1) 대용량 서비스 레퍼런스 아키텍처 개요

(2) 전체 시스템 계층 구조



[대용량 서비스 레퍼런스 아키텍처]

1. 레퍼런스 아키텍처

1) 대용량 서비스 레퍼런스 아키텍처 개요

(3) 개요

- 서버 측 트랜잭션 처리 : Access Layer, Business Layer, Persistent Layer
  - 사용자 요청에 따른 비즈니스 로직 처리 및 저장
  - Access Layer : 사용자 요청에 대한 관문 역할 및 외부 시스템과의 연동
  - Business Layer : 들어온 사용자 요청에 대한 비즈니스 로직을 처리 및 응답
  - Persistent Layer : 비즈니스 로직에 의해 처리되는 데이터 저장
- 시스템 관리 및 운용 : OAM(Operation Administration Monitoring; OSS-Operation Support System)
- 분석 : Analysis Layer (BSS - Business Support System)

2. Access Layer - 1

1) 개요

- 클라이언트로부터 사용자 요청을 처음으로 받는 계층
- 사용자 요청에 대한 사용자 인증 및 권한 인증을 수행하여 뒷단의 비즈니스 로직으로 전달하는 역할

Web Cache

Reverse Proxy

API Gateway

계정 관리(IDM)

시스템 연동(EAI 아키텍처)

2. Access Layer - 1

2) Web Cache

(1) 개요

- 역할 : 웹에서 사용하는 정적인 자원, 자바스크립트, 이미지, HTML 등을 캐싱해 전체 시스템의 부하를 줄임
- Access Layer의 맨 앞 단에 위치하는 계층
- 다양한 웹 캐시 솔루션 존재

CDN(Contents Delivery Network)

- **분산 웹 캐시 서비스**

  - 전 세계에 웹 캐시 서버(에지 서버)를 배치해두고 서비스해 로딩 시간을 줄임
  - 서비스 지역이 넓을 때 용이
  - 근래에는 클라우드에서 CDN 서비스 제공

3) Reverse Proxy

(1) 개요

- 역할
  - 웹 서버 역할을 하며 정적 콘텐츠에 대한 서비스 제공
  - 필요에 따라 HTTP 요청에 대한 인증 수행
  - 뒷단 비즈니스 로직을 처리하는 컴포넌트로 라우팅하는 역할
- 아파치 httpd, Nginx, HAProxy 등 주로 사용

아파치 httpd	Nginx	HAProxy
• 웹 서버 엔진 • 안정성 • SSL, 프록시, 캐시, 인증 등 기능 다양	• 웹 서버 엔진 • 아파치에 비해 성능 월등히 뛰어남 • 버그 문제	• 성능 가장 뛰어남 • 로드 밸런싱 개념만 가짐 → 소프트웨어 로드 밸런서로 활용

2. Access Layer - 1

4) API Gateway

(1) 정의

① API Gateway의 정의

API Gateway

- **클라이언트가 사용하는 API에 대한 엔드포인드를 통합하고 몇 가지 추가적인 기능을 제공하는 미들웨어**

- 시스템 특성에 따라 API Gateway나 ESB(Enterprise Service Bus)를 선택적으로 사용
  - ESB : API Gateway기능을 포함하며, 레거시 시스템이나 다른 프로토콜을 커버할 수 있음
  - API Gateway : REST API 기반의 API를 처리할 수 있도록 최적화되고, 군더더기가 적음
- 외부로 서비스를 제공하는 API나 REST기반 API에 대하여 API Gateway를 시스템 앞단에 배치해서 사용하고, 내부 컴포넌트나 레거시 시스템간의 통신은 ESB를 사용하는 이원화된 구성도 가능

2. Access Layer - 1

4) API Gateway

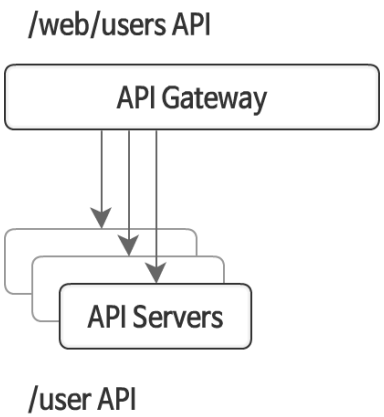
(2) 기능

① API 인증 처리 및 API 키 관리

- 인증을 위해 API 키 발급, 혹은 근래에는 Oauth 기반의 인증 제공
- 사용자 인증을 위해서 SAML이나 Oauth 등 기타 표준을 이용해 계정 관리 시스템(IDM) 혹은 사용자 정보 저장소(LDAP)등에 직접 연결
- API Gateway는 API 키(토큰) 생성 부터 업데이트, 파기까지 전체 생명 주기 관리

② 로드 밸런싱

- 뒷단에 같은 기능을 하는 여러 개의 API 서버들이 있을 때 API Gateway는 단일 엔드 포인트에서 여러 개의 API 서버들로 부하를 분산해주는 역할
- Reverse Proxy와 다소 중첩되는 기능



[API Gateway를 이용한 API 간 부하 분산]

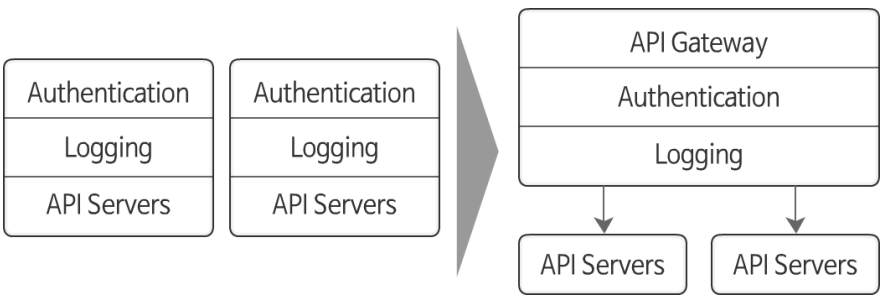
2. Access Layer - 1

4) API Gateway

(2) 기능

③ 공통 기능 처리

- API 로그 처리 혹은 인증 처리 등 API 서버들이 공통적으로 가지는 기능에 대해 API Gateway단으로 통합해 관리
- 공통 기능 변경 시 API Gateway부분만 재배포 해 API 비즈니스 로직에만 집중할 수 있음



[API Gateway를 이용한 공통 기능 처리]

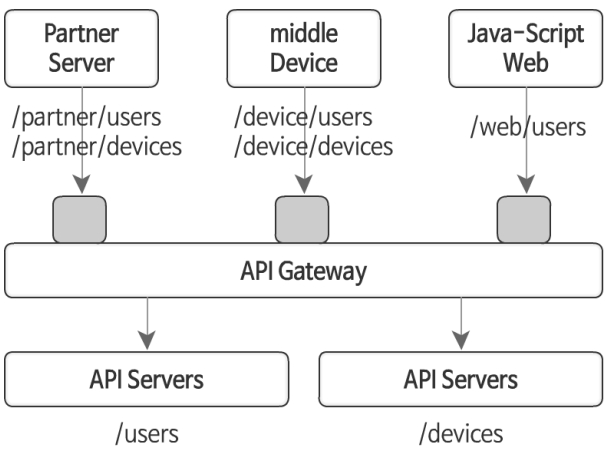
2. Access Layer - 1

4) API Gateway

(2) 기능

④ 다수 엔드 포인트 제공

- API Gateway를 이용해 다수의 엔드 포인트(URL)로 API 제공 가능
- 뒷단의 API 서버는 여러 종류가 있다고 해도, API Gateway를 통해 클라이언트 종류별로 다른 URL 로 API 제공 가능
  - 뒷단 API서버: /users /devices
  - 클라이언트: partner 서버, mobile device, web client
- 같은 API라도 각각의 엔드 포인트(Client)에 서로 다른 속성 부여 가능



[API Gateway를 이용한 다양한 엔드 포인트 제공]

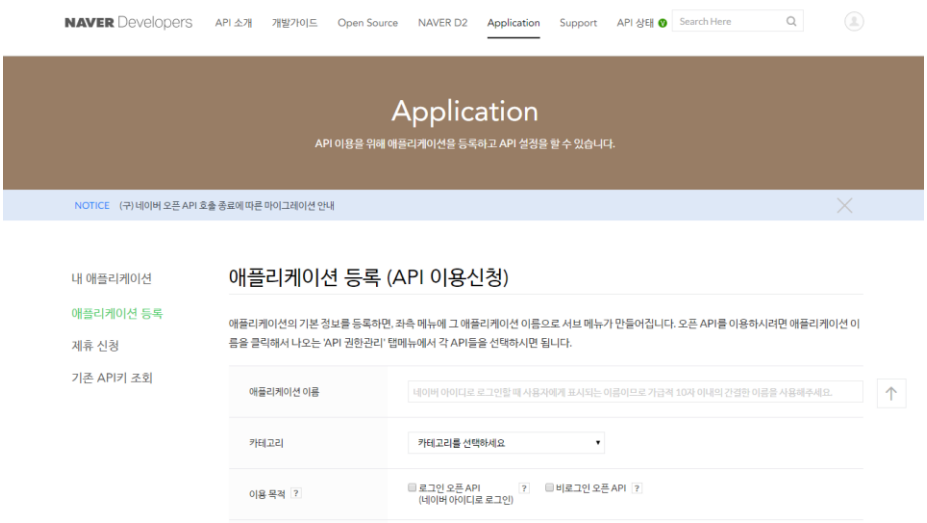
2. Access Layer - 1

4) API Gateway

(2) 기능

⑤ 개발자 포털

- API를 외부로 서비스할 때 개발자와 소통할 수 있는 인터페이스
- API 사용자가 API 사용 가이드 등을 볼 수 있는 사이트(ex) 구글 API, 페이스북 API, 네이버 개발자 포털 등)
- 기능 : 토큰 발급, API 모니터링, API 매뉴얼 제공, API 테스트 베드 제공
- Naver Developers
  - https://developers.naver.com/



2. Access Layer - 1

4) API Gateway

(2) 기능

⑥ 변환 로직

• API에 변환
- 프로토콜 변환
예) HTTP -> TCP, Protocol Buffer(구글)
- 메시지 변환
예) 상호 메시지 포맷 변환 (ex) JSON -> XML)
- MEP(Message Exchange Pattern: 메시지 교환 패턴) 변환
예) 메시지 큐 등을 이용한 비동기 API 일 경우, 동기형으로 변경
예) API Gateway를 통한 요청을 여러 API 서버로 메시지 패턴 변경해 분산 (1:N spawning)

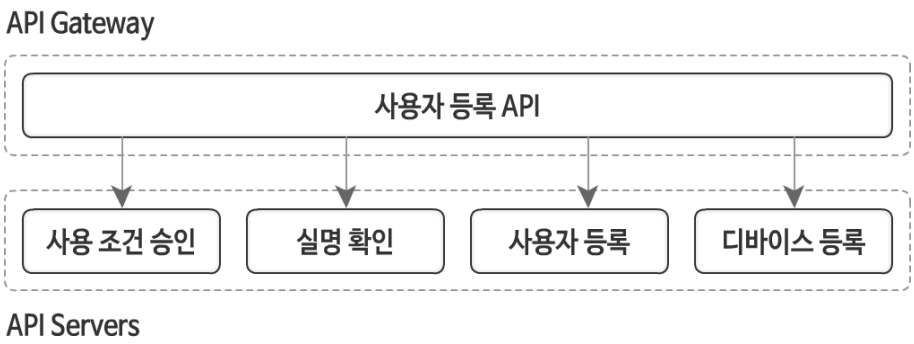
2. Access Layer - 1

4) API Gateway

(2) 기능

⑦ 매시 업

- Mash up = Aggregation = Orchestration
- 여러 API를 묶어 새로운 API를 만드는 기능
  - API Gateway 단에서 API를 만들어 서버 단의 다른 API들을 호출해 사용하는 형태
- 해당 로직이 복잡할 때는 API Gateway에서 하는 것보다 오케스트레이션 계층을 서버 단에 하나 더 두어 작업하는 것이 효율적임
  - API Gateway는 프록시 성격이 강해 고속으로 처리해줘야 하기 때문



[API Gateway를 이용한 매시업]

⑧ QoS 컨트롤

- Quality of Service
- API에 대한 호출 통제
  - API 키 별로 호출 회수 제한
  - 특정 사용자에게서 들어온 요청에 대한 우선순위 처리 (API 유료 서비스에 필수적인 기능)

- 오픈소스 : WSO2
- 상용 : CA 사의 Layer 7
- 클라우드 형 서비스 : apigee.com, 3scale

3. Access Layer - 2

1) 계정 관리(IDM)

(1) 개요

- 일반적인 서비스의 경우 RDBMS에 사용자 계정을 저장하고 애플리케이션에서 사용자관리와 권한인증 등을 통제
- 시스템의 규모가 커지고 사용자의 종류가 많아지면서 여러 시스템에서 공통 계정 체계를 사용하기 위해 별도의 독립된 공통 컴포넌트인 IDM(Identity Management System) 사용
  - 사용자 종류 : 일반 사용자, 관리자, 중간 관리자, 파트너 등

3. Access Layer - 2

1) 계정 관리(IDM)

(2) 기능

① 사용자 관리

- 사용자 계정 정보 및 사용자 신원 정보 관리

② 사용자 계정 생명 주기 관리

- 계정 생성, 승인, 휴면, 폐기 등의 생명 주기 관리
- 대부분 웹 사이트의 경우 사용자 계정 생성이 단순하나, 인터넷뱅킹과 같은 경우 복잡한 사용자 계정 생성절차 거침
  - 비즈니스 프로세스가 복잡한 시스템일 수록 복잡한 절차가 적용

③ 사용자 로그인 정보 관리

- 시스템 로그인 계정과 비밀번호 관리
- 부가적인 인증 방식 (인증서, 지문 정보 등)을 함께 저장 및 관리
- 비밀번호 : Salted Password 방식 사용
  - 비밀번호를 그대로 저장하는 것이 아니라 해시화한 값 저장

④ 사용자 정보 관리

- 로그인 정보 외 부가적인 프로필 정보 기록 (이름, 소속, 주소, 이메일 등)
- 인증보다는 상세 정보를 파악하기 위해 주로 조회 목적으로 사용하기 때문에 사용자 로그인 정보와 분리해서 일반적인 RDBMS에 저장하기도 함

⑤ 사용자 역할 관리

- 시스템 내 다양한 역할을 정의 및 관리

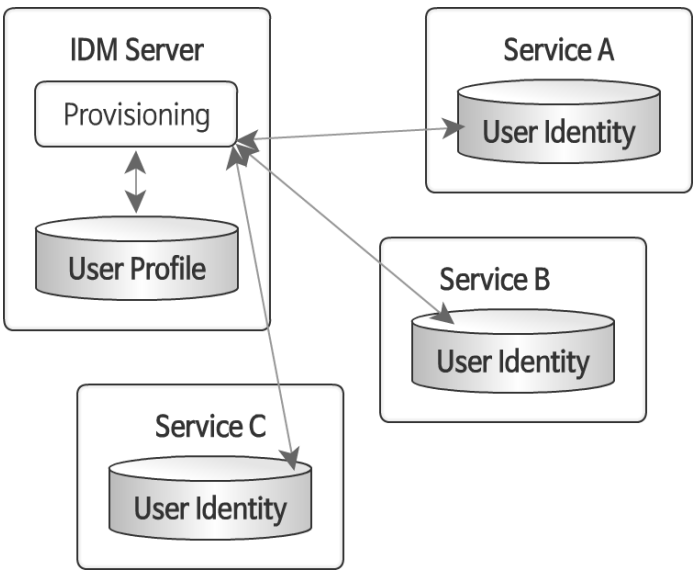
3. Access Layer - 2

1) 계정 관리(IDM)

(2) 기능

⑥ 계정 정보 프로비저닝

- 계정 정보 변경 시 변경된 정보를 연계 시스템에 전달해주는 기능
- 많은 종류의 서비스가 연동되는 경우 전송 메시지 타입이나 프로토콜 등 연동 스펙이 다를 수 있기 때문에 다양한 연동 방식을 지원하며 변화를 신속하게 반영
- 관련 표준 기술 : SCIM, WS-Provisioning



[IDM을 이용한 각 서비스 시스템으로의 계정 정보 프로비저닝]

3. Access Layer - 2

1) 계정 관리(IDM)

(2) 기능

⑦ 접근 제어

- 웹 사이트와 같은 접근 대상이 되는 시스템과 그 시스템이 가진 기능에 대한 접근을 제어하는 것
- 단순하게 사용자 계정으로 로그인만 되면 사용할 수 있는 시나리오부터 사용자가 가진 권한에 따라 접근 수준을 제어하는 시나리오까지 필요
  - 시스템 접근 수단(웹, PC 애플리케이션, 모바일 디바이스 등)에 따라 다른 접근 제어 메커니즘 사용
  - 사내 및 사외 연계 시스템까지의 접근 제어

⑧ 사용자 인증(Authentication)/권한 인가(Authorization) 처리

- 인증 : 올바른 사용자인가 확인
- 권한인가 : 특정기능에 접근할 권한이 있는지 확인
- 관리 권한 위임(Delegated Admin) : 시스템 별로 관리자를 나눠 관리 권한 위임
  - 전체 시스템에 대한 모든 관리 권한을 가진 슈퍼 관리자가 이메일 관리자를 지정해서 이메일 시스템만 관리하도록 이메일 시스템 관리 권한 위임

⑨ SSO(Single Sign On)

- 하나의 계정으로 한 번의 로그인으로 다른 독립된 시스템들을 같이 사용
  - https://mail.google.com/ 로그인
    → https://drive.google.com/, https://www.google.co.kr/ 등의 사이트에도 별도의 로그인 없이 해당 계정으로 접근
- SAML(Security Assertion Markup Language), Oauth2.0 등의 표준 기술 사용

3. Access Layer - 2

1) 계정 관리(IDM)

(2) 기능

⑩ 계정 정보 페더레이션

- Idp와 Sp를 연계해주는 작업
  - Idp(Identity Provider : 계정 정보 제공자)
  - Sp(Service Provider : 계정 정보 사용자)
- 페이스북이나 카카오톡 계정을 이용한 서드파티 게임 서비스의 경우 Idp는 페이스북이나 카카오톡이고, Sp는 게임서비스임
- SSO와는 달리 로그인이 된 상태에서 연계된 시스템의 사용자 인증 시 재로그인이 필요할 수 있음

⑪ 타 서비스 연동

- 서로 다른 표준과 기술을 사용하는 분리된 시스템들을 연동해주는 인터페이스
- 계정 인증 표준 기술 : SAML2.0, OpenID, Oauth 1.0, 2.0
- 권한 인가 연동 기술 : XACML
- 표준을 따르지 않거나 특별한 기술이 사용된 레거시 시스템들의 경우 자체 인증 규격에 맞는 어댑터 사용 필요
- 각 요소 시스템이 사용하는 계정 관련 기술이나 계정 체계가 모두 다르기 때문에 전체 시스템의 계정 체계에 대한 일관성 있는 체계 정립 후 시스템을 연동해야 함

⑫ 감사와 리포팅

- 시스템의 모니터링 관점에서 사용자 행위를 추적하고 시스템 사용에 대하나 리포트를 생성하는 기능
- 감사(audit) : 사용자의 시스템 접근 기록을 저장 및 조회
- 리포팅(Reporting) : 시스템에 대한 사용 패턴을 분석 하는 등 표나 그래프 형태로 각종 통계 자료 리포팅

3. Access Layer - 2

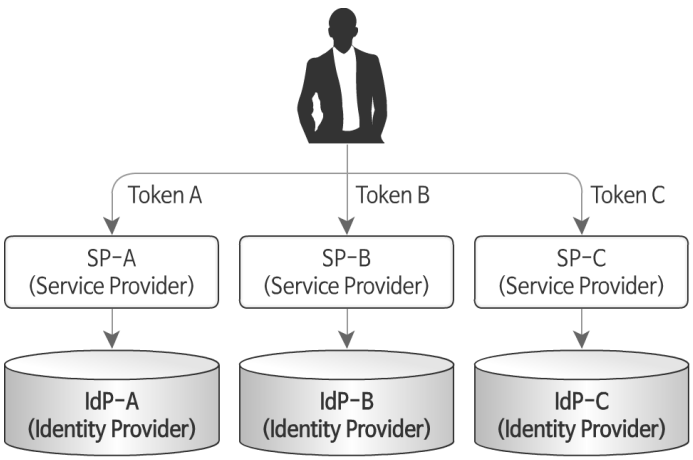
1) 계정 관리(IDM)

(3) 계정 관리 모델

① 분산된 시스템 간의 연계 방법에 따른 분류

개별 분산 모델(Isolated)

- 각각 독립된 서비스로 각기 다른 계정 체계 사용 (N개의 사이트에 N개의 사용자 계정으로 접근)



[개별 분산 계정 관리 모델]

3. Access Layer - 2

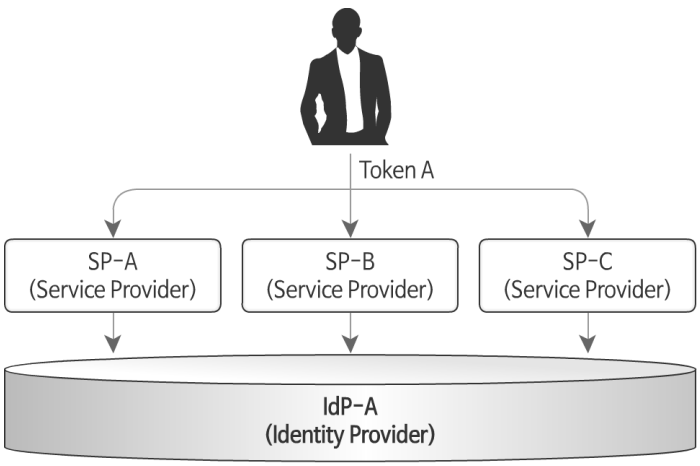
1) 계정 관리(IDM)

(3) 계정 관리 모델

① 분산된 시스템 간의 연계 방법에 따른 분류

중앙 집중형 모델(Centralized)

- 모든 시스템이 하나의 계정 체계 사용
- 구축이 어려움 : 개발 분산 모델 구축 → SSO 도입 → 페더레이션모델로 개선



[중앙 집중형 계정 관리 모델]

3. Access Layer - 2

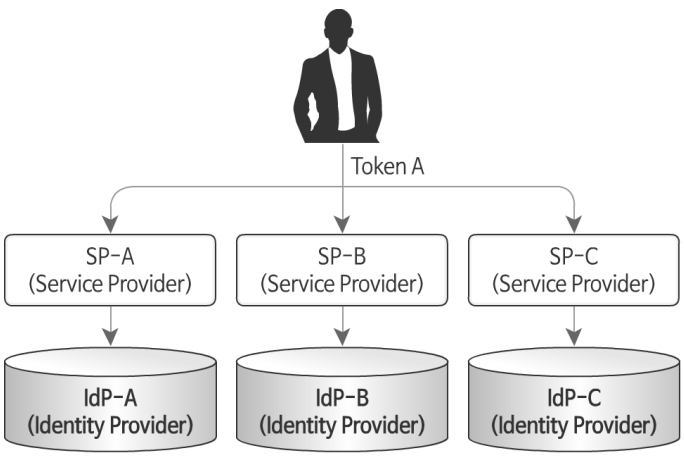
1) 계정 관리(IDM)

(3) 계정 관리 모델

① 분산된 시스템 간의 연계 방법에 따른 분류

페더레이션 모델(Federated)

- 별도 계정 시스템 사용하지만 페더레이션 통해 다른 계정을 연동해 전체 시스템에 접근



[페더레이션 형태의 계정 관리 모델]

3. Access Layer - 2

1) 계정 관리(IDM)

(4) 계정 관리 시스템 솔루션

- 상용 제품 : Oracle, CA.com, Forgerrock.com
- 오픈 소스 : WSO2의 Identity server 제품
- 사용자 인증 기능만 제공 : SAML 기반의 simpleSAMLPhp, 자바 기반의 shibboleth.net
- 페더레이션 솔루션 : pingidentity.com 서비스
- 클라우드 기반 계정 관리 기능
  - MS 클라우드(Azure)의 AD 서비스
  - stormpath.com 서비스

3. Access Layer - 2

2) 시스템 연동(EAI 아키텍처)

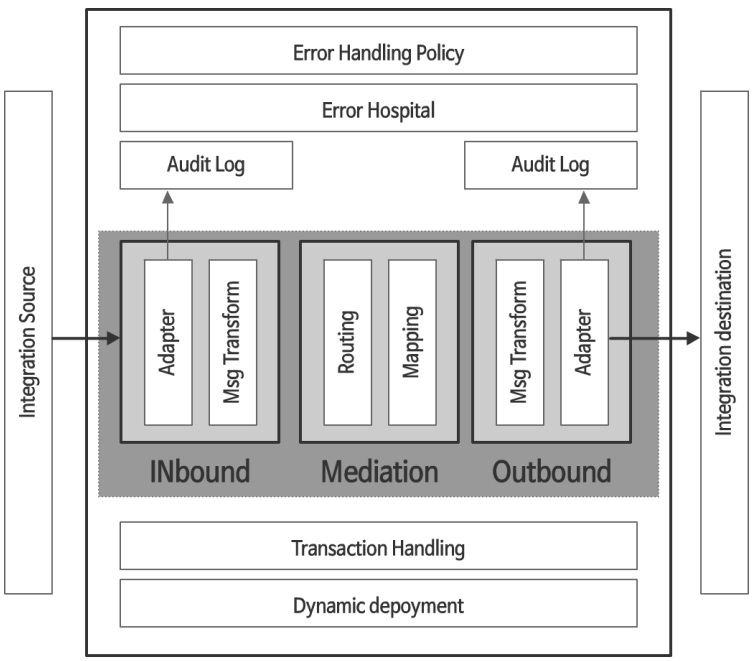
(1) 개요

EAI(Enterprise Application Integration)

- **대외 시스템 연계 및 대내 시스템 간의 연동을 수행하는 시스템(통합 시스템)을 구현한 엔터프라이즈 아키텍처로 다음 세가지 기능 통합 지원**

- 대내 처리 통합 : 내부 시스템 간의 업무 통합, 실시간성 처리가 많음
- 대외 처리 통합 : 기업간의 업무처리 정의. 기업 업무처리 특성 상 분산 트랜잭션을 이용한 트랜잭션 보장이 불가능하기 때문에 로그 저장이 중요
- 배치 처리 통합 : 대용량 데이터를 송신 시스템에서 수신 시스템으로 전송하는 요건으로 배치 처리와 정보계로 구분
  - 배치 처리 : 다른 업무 시스템 간의 처리 정보 통합 (전달 보장 요건이 중요)
  - 정보계 : 데이터 분석해 경영에 필요한 리포트를 뽑아내는 웨어하우스나 BI성 업무

(2) EAI 시스템의 아키텍처



[EAI 시스템의 아키텍쳐]

3. Access Layer - 2

2) 시스템 연동(EAI 아키텍처)

(3) 인터페이스 모듈

인터페이스 모듈의 정의

- **EAI의 가장 기본적인 아키텍처 모듈로 송수신 시스템을 통합하는 부분에 대한 아키텍처**

- Inbound : 송신 시스템과 연동해 요청을 받고 송신 시스템에 응답을 보내는 역할
  - 어댑터 : 다양한 플랫폼으로부터 메시지 읽는 진입점(entry point)
  - 메시지 변환
    - 어댑터에 의해 요청된 메시지의 다양한 형태(XML, 텍스트, 바이너리 등)를
    - 공통적인 데이터 구조로 변환
    - 상용 솔루션에서는 메시지 유연성 위해 xml을 사용
    - 인하우스 개발의 경우 성능 최적화 위해 HashTable형태의 java POJO object 사용
- Mediation - 메시지 가공
  - 라우팅 : 입력된 메시지를 내용에 따라 적절한 수신 시스템으로 라우팅 (1:1, 1:N, N:1)
  - 매핑 : 입력된 메시지를 수신 시스템에서 요구하는 형태로 매핑
  - 메시지 교환 패턴 처리 : MEP(Message Exchange Pattern)를 메시지 큐잉 시스템을 이용해 구현
- Outbound - 수신 시스템과 연동
  - 수신 시스템 플랫폼의 네이티브 메시지 형태로 변환해 어댑터를 통해 전달

3. Access Layer - 2

2) 시스템 연동(EAI 아키텍처)

(4) 모니터링 및 장애 관리

① 거래 추적 및 에러 처리 방식

- 감시 로그 (Audit log)
  - 송수신 내용을 확인하고 EAI 시스템 장애 시 송수신 시스템과 거래 내용을 맞춰 복구하는데 사용
  - 감시 로그 ID는 전사 표준 전문의 헤더에 정의하며, 송신에서 수신 시스템까지 공통된 ID 사용
  - 저장 장소 및 저장 방식
    - 파일(IO성능 빠르고, DB장애에 대한 의존성 적지만, 거래 추적 시 파일 일일이 검색)
    - DB(특정 거래 및 조건에 대한 검색은 쉬우나 별도 DB 하드웨어 필요 및 DB장애에 의존적)
  - 분산형 로그 수집 솔루션 (Logstash 등)을 활용한 로그 작성
  - 감시 로그는 IO를 유발하기 때문에 성능에 큰 영향을 주는 부분으로 세밀한 설계와 검증 필요
- 에러 처리 로직
  - 장애 감지
  - 장애 원인 리포팅
  - 장애 해결 : 장애 처리 정책(Fault-Policy)이 인터페이스마다 정의되어 인터페이스별로 장애를 처리할 수 있음

3. Access Layer - 2

2) 시스템 연동(EAI 아키텍처)

(4) 모니터링 및 장애 관리

① 거래 추적 및 에러 처리 방식

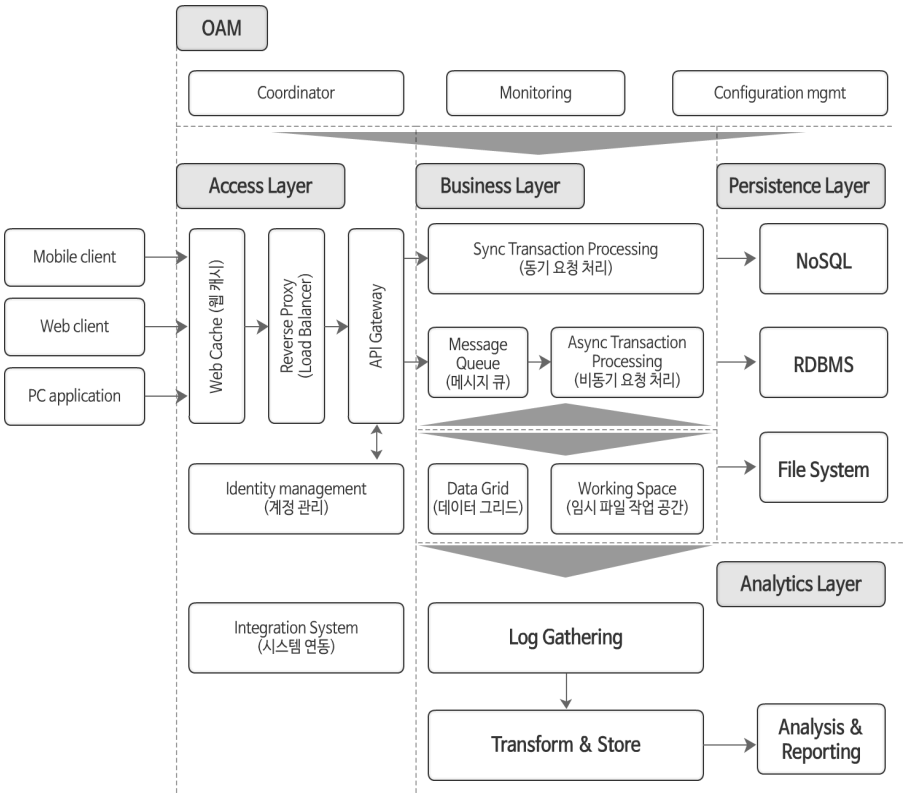
• 장애 처리 정책
- 장애가 발견되고 리포팅되면 모든 장애를 Error-Hospital이라는 곳에 집결하고 장애처리 정책에 따라 처리
- 장애 해결의 일반적인 4가지 정책

정책	설명	노트
Ignore(무시)	메시지 실패 시 무시	
Report (보고)	메시지 전달 실패 시 이메일, 메신저 등을 통해서 보고	
Retry (재시도)	메시지 전달 실패 시 지정된 시간 후에 자동으로 재처리	지정된 횟수만큼 재처리를 시도했을 시 모두 실패했을 때에 대한 정책 정의 필요
Manualhandling (수동 처리)	시스템 운영자에게 통보한 후 실패 메시지를 운영자가 수동 처리 하도록 함	

핵심정리

1. 레퍼런스 아키텍처

- 대용량 서비스 레퍼런스 아키텍처 개요
  - 대용량 서비스를 위한 플랫폼에 대한 레퍼런스 아키텍처로 대부분의 온라인 서비스 처리에서 사용되는 구조



핵심정리

2. Access Layer

- 개요
  - 클라이언트로부터 사용자 요청을 처음으로 받는 계층
  - 인증 및 권한 인가을 수행하여 뒷단의 비즈니스 로직으로 전달하는 역할
- Web Cache
  - 웹에서 사용하는 정적인 자원, 자바스크립트, 이미지, HTML 등을 캐싱 해 전체 시스템의 부하를 줄임
- Reverse Proxy
  - 웹 서버 역할을 하며 정적 콘텐츠에 대한 서비스 제공하며, 뒷단 비즈니스 로직을 처리하는 컴포넌트로 라우팅하는 역할
  - Httpd, Nginx, HAProxy 등
- API Gateway
  - 클라이언트가 사용하는 API에 대한 엔드포인드를 통합하고 추가적인 기능을 제공하는 미들웨어
  - 기능 : API 인증 처리 및 API 키 관리, 로드 밸런싱, 공통 기능 처리, 다수 엔드 포인트 제공, 개발자 포털, 변환 로직, 매시 업, QoS 컨트롤
- 계정 관리(IDM)
  - 여러 시스템에서 공통 계정 체계를 사용하기 위해 별도의 독립된 공통 컴포넌트인 IDM(Identity Management System) 사용
  - 기능 : 사용자 관리, 사용자 계정 생명 주기 관리, 사용자 로그인 정보 관리 등
  - 모델 : 개별 분산 모델(Isolated), 중앙 집중형 모델(Centralized), 페더레이션 모델(Federated)
- 시스템 연동(EAI 아키텍처)
  - 대외 시스템 연계 및 대내 시스템 간의 연동을 수행하는 시스템(통합 시스템)을 구현한 엔터프라이즈 아키텍처
  - 역할 : 대내 처리 통합, 대외 처리 통합, 배치 처리 통합
  - 모니터링 및 장애 관리 : 감시 로그, 에러 처리 로직, 장애 처리 정책(무시, 보고, 재시도, 수동 처리)