통 합 구 현

메시징 시스템 개요

한국기술교육대학교
온라인평생교육원

학습내용

- 애플리케이션 통합의 필요성 및 통합 패턴
- 애플리케이션 통합 스타일
- 메시징 시스템의 개요

학습목표

- 애플리케이션 통합에 필요한 통합 패턴을 구별할 수 있다.
- 메시징 시스템을 구성하는 요소들을 설명할 수 있다.

1. 통합 패턴

1) 애플리케이션 통합의 필요성

(1) 필요성

- 일반적으로 애플리케이션(혹은 서비스)은 단독으로 운영되지 않음
- 각각의 애플리케이션들 간의 연결과 통합을 통해 하나의 트랜잭션 처리
- 고객 주문 트랜잭션
  - 고객 아이디 확인
  - 고객 상태 확인
  - 재고 확인
  - 주문처리
  - 배송 견적
  - 세금 계산
  - 청구서 전송 등 처리
- 애플리케이션들 사이에 공통 비즈니스를 처리하고 공통 데이터를 공유하려면 애플리케이션들을 통합할 필요가 있음

애플리케이션 통합 시

효율적 | 신뢰 가능 | 안전한 데이터 교환 제공

1. 통합 패턴

1) 통합 패턴

(1) 통합의 정의

통합

- 컴퓨터 시스템들과 회사들과 사람들을 연결하는 일
- 연속되고 통일된 기능 집합을 만들기 위해 서로 다른 애플리케이션들을 함께 엮는 작업

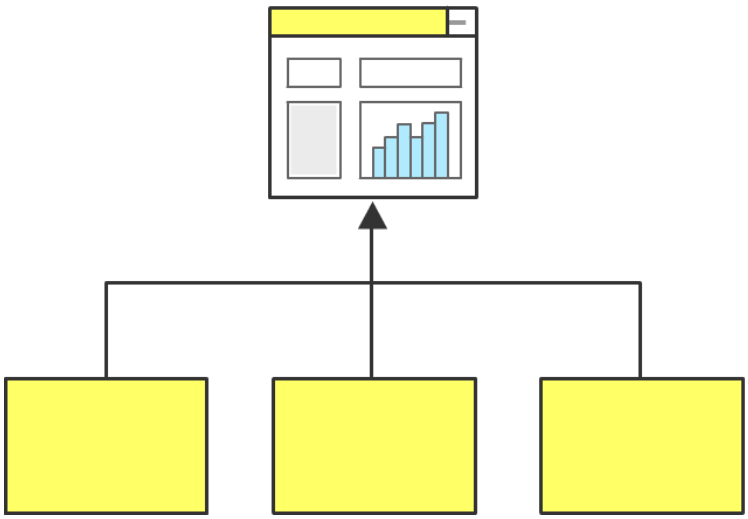
- 이전에 해결했던 문제와 새로운 문제를 비교하면서 통합에서 발생하는 문제 해결 → 패턴(유형)

정보포털	데이터 복제
공유 비즈니스 기능	서비스 지향 아키텍처
분산 비즈니스 프로세스	비즈니스 대 비즈니스 통합

1. 통합 패턴

1) 통합 패턴

(2) 정보포털

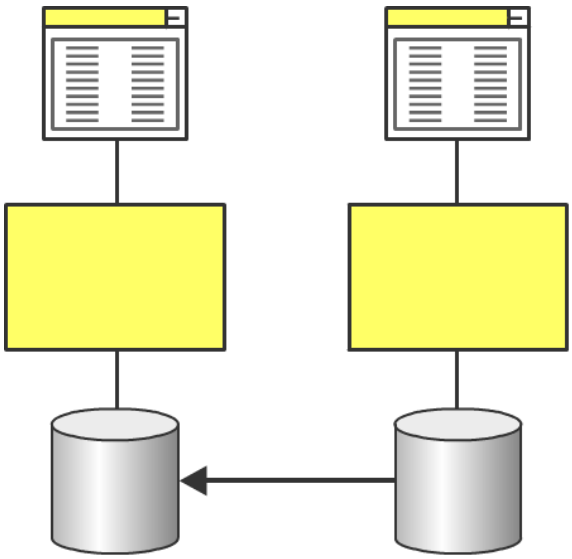


- 여러 소스에서 정보를 수집해 한 화면으로 표시
- 화면을 여러 영역으로 나누고 화면 영역마다 다른 시스템의 정보 표시
- 영역들의 상호작용도 제공 가능

1. 통합 패턴

1) 통합 패턴

(3) 데이터 복제

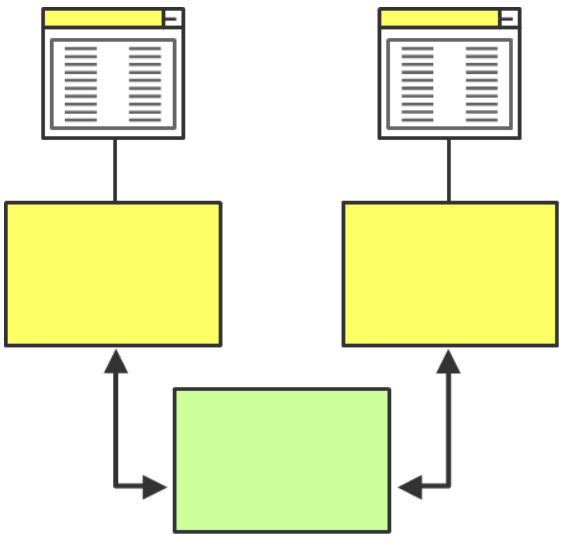


- 각 시스템마다 독자적인 데이터 저장소를 가진 경우 데이터 복제 필요
- 고객정보를 시스템 마다 독립적으로 가지고 있을 경우 한 시스템에서 고객정보를 변경하면 다른 시스템들에 고객정보 변경

1. 통합 패턴

1) 통합 패턴

(3) 공유비즈니스 가능

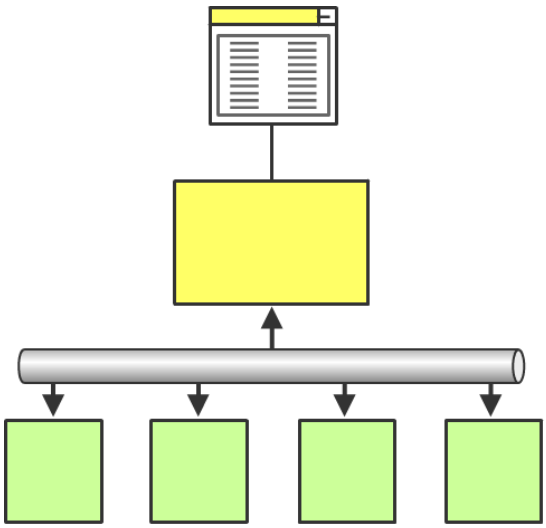


- 각 시스템 마다 동일한 기능을 별도로 구현하지 않고 공유 비즈니스 기능으로 한 번만 구현해서 다른 시스템에 서비스로 노출

1. 통합 패턴

1) 통합 패턴

(4) 서비스 지향 아키텍처

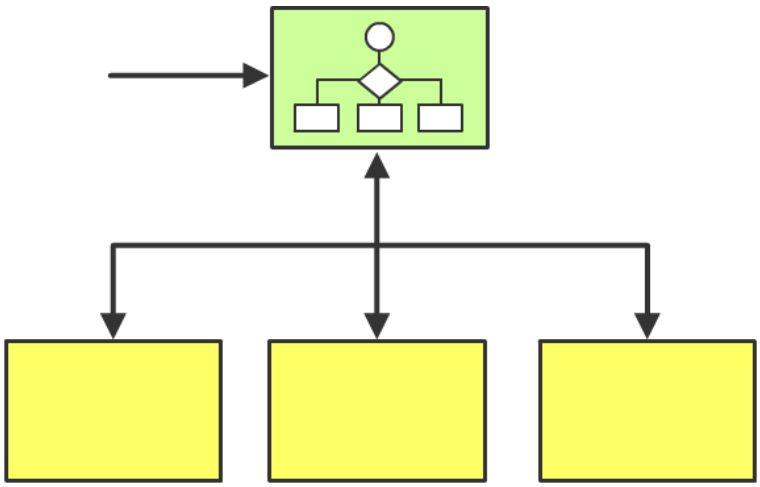


- 서비스: 사용가능하고 사용자의 요청에 응답하는 잘 정의된 기능
- 서비스 디렉토리, 서비스 발견, 협상 등을 통해 서비스 이용

1. 통합 패턴

1) 통합 패턴

(5) 분산 비즈니스 프로세스

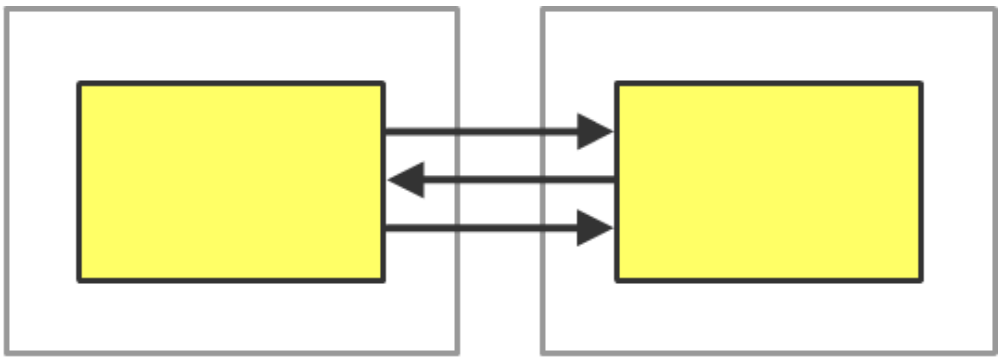


- 단일 트랜잭션이 여러 시스템에 분산되어 처리
- 여러 시스템에 걸쳐 처리되는 비즈니스 기능을 한 비즈니스 기능으로 실행하려면 기존 시스템들의 비즈니스 실행들을 관리하는 비즈니스 프로세스 관리 컴포넌트 필요

1. 통합 패턴

1) 통합 패턴

(6) 비즈니스대 비즈니스 통합



- 기업과 기업간의 비즈니스 통합
- 통신 프로토콜과 보안 문제 야기

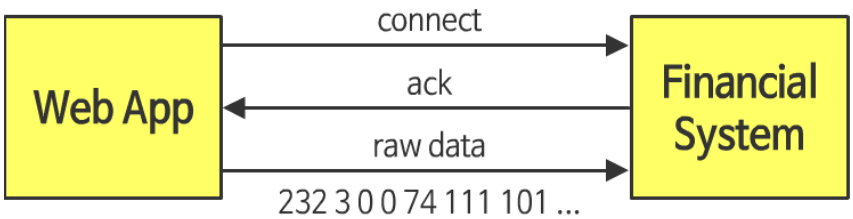
1. 통합 패턴

2) 간단한 통합 구현

(1) 기능 및 문제점

① 온라인 은행 거래 시스템의 계좌 이체 기능

- Front-end : 웹 애플리케이션
- Back-end : 금융 시스템



- TCP/IP 소켓을 사용하고, 하드코딩 한 시스템 주소를 사용하고, 원시 데이터 전송

② 문제점

- 작고, 빠르고, 저렴하지만 단단한 결합(tightly coupling)으로 불안정한 시스템

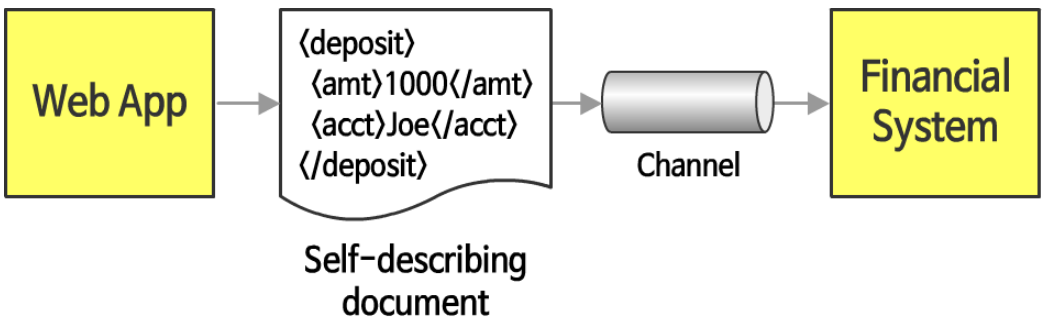
1. 통합 패턴

2) 간단한 통합 구현

(1) 기능 및 문제점

① 온라인 은행 거래 시스템의 계좌 이체 기능

- Front-end : 웹 애플리케이션
- Back-end : 금융 시스템



- 자기 기술적인 공통 데이터 포맷, 채널을 통한 비동기 통신, 포맷 변환기 등을 활용

② 문제점

- 느슨한 결합(loosely coupling)으로 구성
  → 유연성, 확장성 등이 증가하나 복잡성 증가

2. 통합 스타일

1) 통합 기준

(1) 애플리케이션의 통합 시 고려해야 되는 기준

① 애플리케이션 결합도(Application Coupling)

- 애플리케이션들간의 상호 의존성을 최소화

② 통합 단순성 (Integration Simplicity)

- 애플리케이션의 변경을 최소화하고, 통합을 위한 코드의 양도 최소하기 위해 노력

③ 통합 기술 (Integration Technology)

- 통합 기술마다 요구하는 소프트웨어와 하드웨어가 다름

④ 데이터 포맷 (Data Format)

- 애플리케이션간의 교환하는 데이터 포맷에 대한 동의(포맷 변환기가 필요할 수 있음)

⑤ 데이터 적시성 (Data Timeliness)

- 애플리케이션 간의 데이터 전파시간 최소화

⑥ 데이터 아니면 기능 (Data or Functionality)

- 단순 데이터의 공유가 아닌 기능의 공유
- 다른 애플리케이션의 기능호출로 지역 호출 뿐만 아니라 원격호출도 가능해야 함

⑦ 비동기성 (Asynchronicity)

- 성능향상을 위한 비동기성 지원으로 효율적인 솔루션을 만들지만 설계, 개발, 디버깅은 복잡해짐

2. 통합 스타일

2) 통합을 위한 스타일

- 모든 기준을 고루 만족하는 통합 접근 방법은 없음
- 데이터 전송 방식에 따라 4가지 스타일로 구분

파일전송

공유 데이터베이스

원격 프로시저 호출

메시징 시스템

2. 통합 스타일

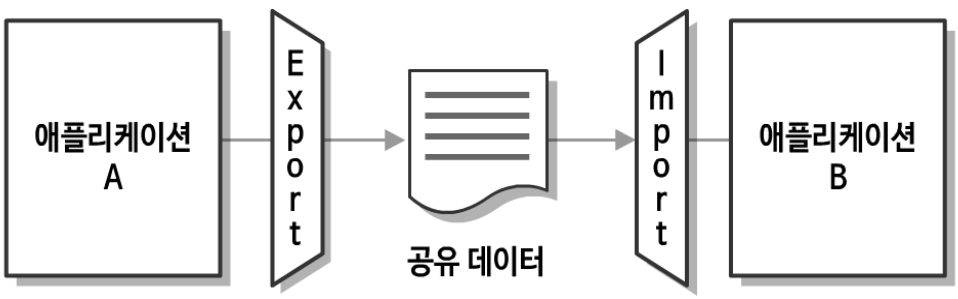
3) 파일 전송

(1) 파일 전송의 정의

파일 전송

- 애플리케이션들을 통합하기 위한 가장 간단한 통합 방식
- 사전에 결정된 파일 포맷을 이용하여 상호 통신

- 애플리케이션은 다른 애플리케이션들이 소비할 정보를 포함한 파일을 생성
- 통합자가 파일을 다른 포맷으로 변환
- 파일은 비즈니스 성격에 따라 정기적으로 생성



(2) 파일 전송의 장단점

① 장점

- 통합자가 애플리케이션 내부 구조를 알 필요가 없음
- 추가적인 도구나 통합 패키지가 필요하지 않음

② 단점

- 파일 생성과 소멸에 대한 파일 life cycle 관리
- 파일 갱신이 불규칙하게 발생할 경우 시스템의 동기화 불일치 문제 (갱신주기 설정)
- 애플리케이션들간의 결합도는 낮추지만 적시성이 부족하고, 시스템들의 협업이 느림

2. 통합 스타일

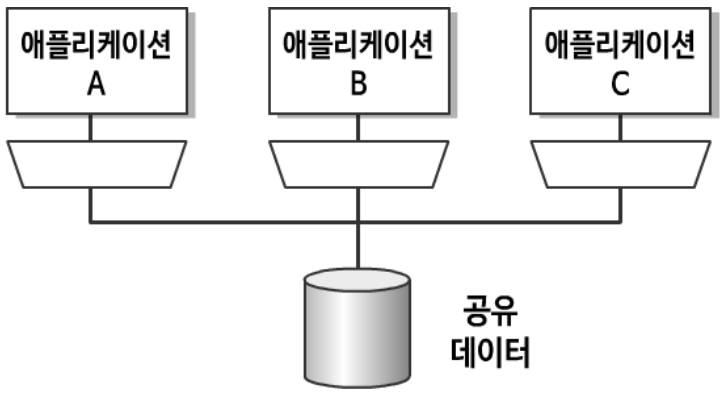
4) 공유 데이터베이스

(1) 정의

공유 데이터베이스란?

- 데이터베이스(DB)를 이용하여 애플리케이션들 간의 데이터를 공유하는 방법

- 데이터를 하나의 공유 데이터베이스에 저장하고 애플리케이션들의 모든 요구사항들을 처리할 수 있는 데이터베이스 스키마를 정의해 애플리케이션들을 통합



(2) 공유 데이터베이스의 장·단점

① 장점

- 데이터 베이스를 사용함으로써 데이터 일관성 유지
- 트랙잭션 관리 시스템으로 동기화 문제 해결

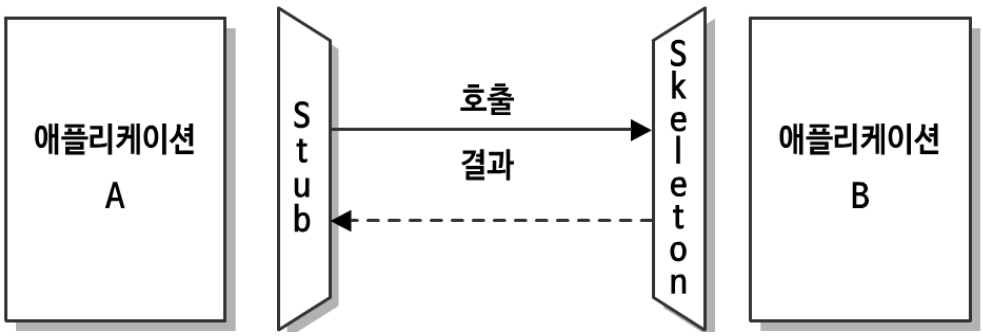
② 단점

- 여러 애플리케이션의 요구를 충족하는 공통 스키마를 찾기가 어려움
- 외부 패키지 사용 시 공유 데이터 베이스 활용이 불가능할 수 있음
- 데이터 베이스의 성능저하, 교착상태(deadlock), 잠금 충돌(lock conflict)등의 문제 야기 가능성이 있음
- 적시성이 좋고, 데이터들간 결합도가 높으나 애플리케이션간 협업이 어려움

2. 통합 스타일

5) 원격 프로시저 호출

- 애플리케이션이 다른 애플리케이션의 기능을 호출
- 다른 애플리케이션의 데이터를 요청하거나 수정을 지시할 수 있음
- 애플리케이션을 캡슐화 된 데이터를 처리하는 큰 규모의 객체나 컴포넌트로 개발하고, 다른 애플리케이션들과 상호작용 할 수 있는 인터페이스 제공



(1) 원격 프로시저 호출의 장·단점

① 장점

- 애플리케이션들간의 기능공유 가능

② 단점

- 성능과 신뢰성의 문제 발생 가능
- 애플리케이션간의 결합도가 높음

2. 통합 스타일

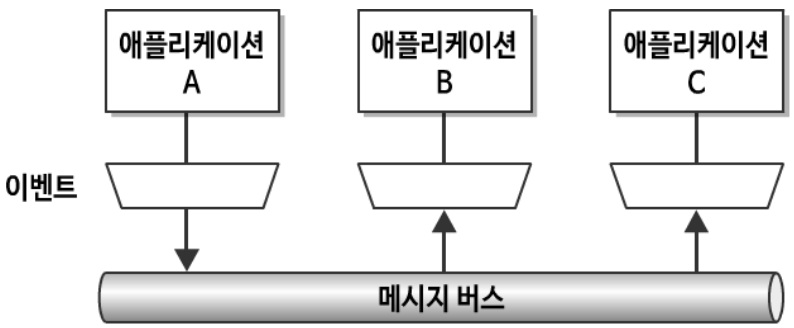
6) 메시징

(1) 정의

메시징이란?

- **애플리케이션간에 빠르고 신뢰할 수 있는 통신을 비동기 방식으로 가능하게 하는 전송기술**

- 다양한 포맷의 데이터 패킷을 메시징을 이용해 전송
- 메시징은 메시지를 자주, 즉시, 안정적으로 전송하는 비동기 기술



(2) 메시징의 장·단점

① 장점

- 애플리케이션들간의 결합도를 낮추어서 개발과 통합을 분리 할 수 있음
- 메시지 전송을 통해 데이터 공유와 협업 가능

② 단점

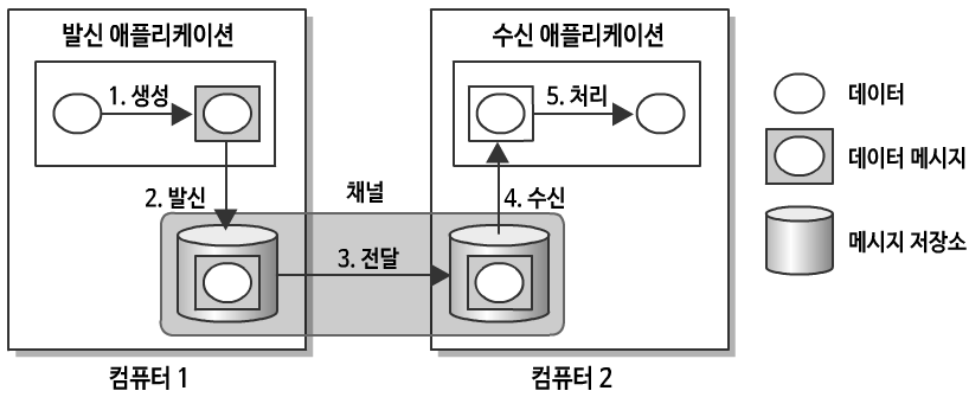
- 애플리케이션 구현의 복잡도 증가

3. 메시징 시스템

1) 메시징 시스템이란?

(1) 메시징 시스템

- 메시지 버스(채널)를 이용하여 수신자와 발신자가 분리되어 통신
- 메시지라 불리는 데이터 패킷을 전송함으로 상호간 통신
- 메시지 자체는 일종의 데이터로 헤더와 본문으로 구성
  - 헤더 : 메타정보로 메시징 시스템에서 사용되는 데이터
  - 본문 : 애플리케이션에 사용할 데이터



[메시지 전송 절차]

3. 메시징 시스템

2) 메시징을 이용하는 이유

(1) 이유

① 원격통신(Remote Communication)

- 분리된 애플리케이션간의 통신 및 데이터 전송이 가능하고, 데이터 전송시 데이터 변환을 메시징 시스템이 담당

② 플랫폼 언어 통합(Platform/Language Integration)

- 시스템마다 언어, 기술, 플랫폼이 다르고, 메시징 시스템은 각 시스템간의 중립지대로 범용 변환기로 활용

③ 비동기 통신(Asynchronous Communication)

- 메시지를 발신하고 메시징 시스템 메시지를 전송하는 동안 다른 작업 수행

④ 시간조절(Variable Timing)

- 비동기 통신으로 수신자/발신자가 자신의 진행속도에 따라 작업수행

⑤ 흐름조절(Throttling)

- 동시 다발적 수신자 요청을 메시징 시스템의 큐에 저장함으로써 수신자의 요청 처리 속도를 적절히 조정하여 과부하 방지

⑥ 신뢰통신(Reliable Communication)

- 메시징 시스템이 메시지 전달 보장

⑦ 비접속 작업(Disconnected Operation)

- 네트워크가 끊긴 상황에서 수행한 작업을 네트워크가 다시 연결되면 동기화 수행

⑧ 중재(Mediation)

- 메시징을 사용하는 애플리케이션들간의 중재로 고가용성, 부하분산, 우회연결, 이중화 등 제공 가능

3. 메시징 시스템

3) 비동기 메시징의 과제

① 복잡한 프로그래밍 모델(Complex Programming Model)

- 이벤트 기반 프로그래밍 모델

② 순서문제(Sequence Issues)

- 메시지가 순서에 맞지 않게 전송될 수 있음

③ 동기 시나리오(Synchronous Scenarios)

- 동기식 애플리케이션도 존재하므로 동기식 메시징 지원

④ 성능(Performance)

- 통신 부하 발생

⑤ 제한된 플랫폼 지원(Limited Platform Support)

- 메시징 시스템을 모든 플랫폼에서 지원하는 것은 아님

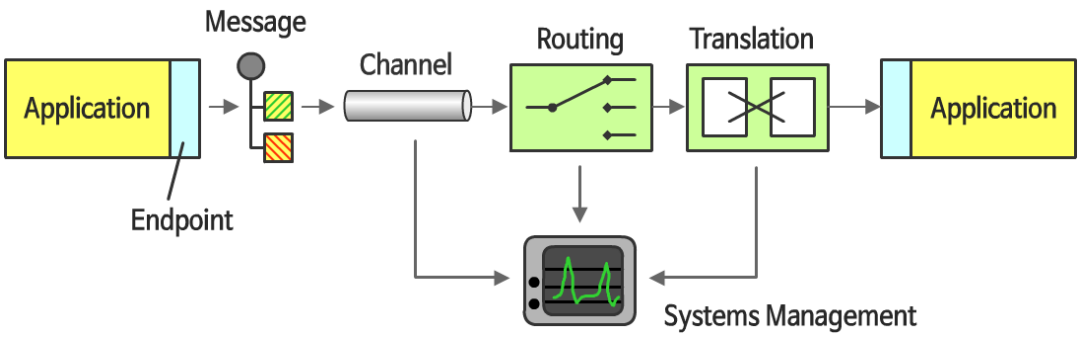
⑥ 벤더 의존성 (Vendor Lock-in)

- 대부분의 메시징 시스템은 독점적 프로토콜에 의존

3. 메시징 시스템

4) 메시징의 기본 개념

- 채널 : 수신자와 발신자를 연결하는 가상의 파이프
- 메시지 : 전송할 데이터
- 파이프필터 : 전송 중 메시지의 검증이나 변환
- 라우팅 : 전송 경로 찾기
- 변환 : 수신자와 발신자 사이의 상이한 메시지 포맷 변환
- 엔드포인트 : 애플리케이션과 메시징 시스템을 연결해주는 인터페이스

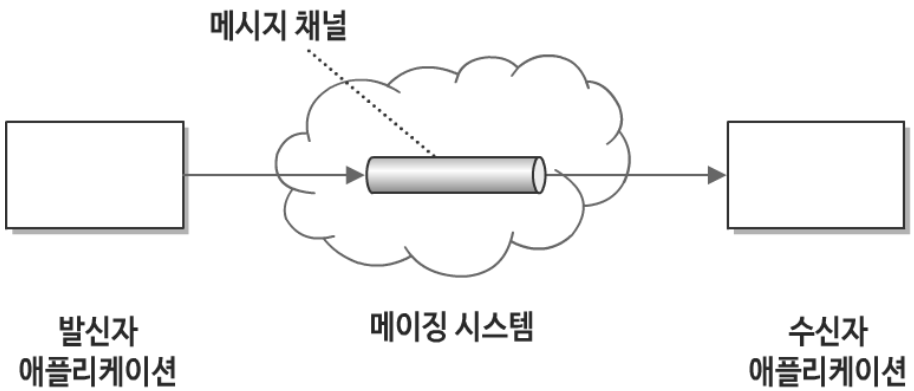


3. 메시징 시스템

4) 메시징의 기본 개념

(1) 채널

- 메시징 시스템내의 메시지 채널을 활용하여 애플리케이션들간 통신
- 채널은 메시징 시스템의 논리주소
- 채널은 다양한 방식으로 구현
- 메시지 채널을 사용해 애플리케이션들을 연결하고, 애플리케이션이 정보를 채널에 기록하면 다른 애플리케이션은 채널에서 정보를 읽음

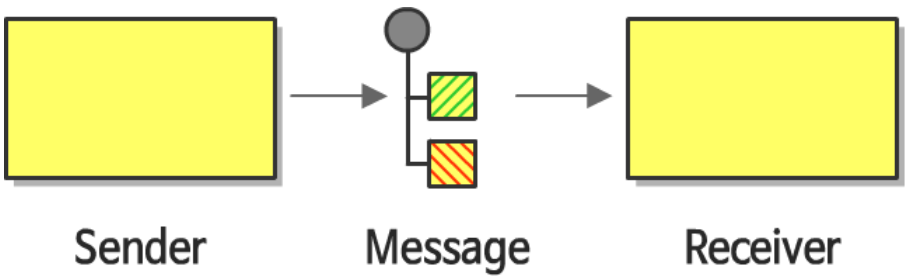


3. 메시징 시스템

4) 메시징의 기본 개념

(2) 메시지

- 메시징 시스템에서 통신이 이루어지는 데이터의 구성
  - 헤더 : 데이터의 기원 목적지 등 전송되는 데이터 설명
  - 본문 : 전송되는 데이터
- 데이터가 큰 경우 작은 메시지로 분할하여 순서대로 전송
- 메시징 시스템에 따라서 메시지를 표현하는 방법 다양

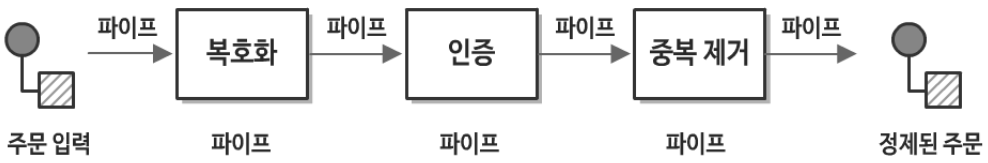


3. 메시징 시스템

4) 메시징의 기본 개념

(3) 파이프 필터

- 메시지 처리를 위한 개별 처리 단계를 구현하여 필터로 사용하고, 필터는 메시지 채널(파이프)로 연결
- 많은 처리 단계들을 포함하는 프로세스는 파이프 필터 아키텍처 스타일을 사용
  - 이 아키텍처 스타일은 프로세스를 연속되는 소규모 독립 처리 단계(필터)들로 나누고 각 처리 단계를 채널(파이프)로 연결함

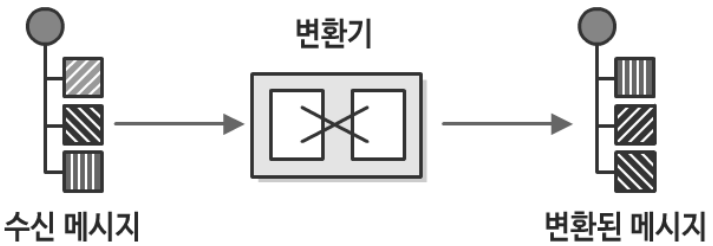


3. 메시징 시스템

4) 메시징의 기본 개념

(4) 메시지 변환기

- 서로 다른 데이터 포맷을 사용하는 애플리케이션들을 지원하기 위해 메시지 변환기를 사용하여 전송하려 하는 메시지 포맷을 변경하는 역할
- 필터나 어플리케이션들 사이에 특별한 메시지 변환기를 사용해 데이터 포맷을 변환



- 변환 수준

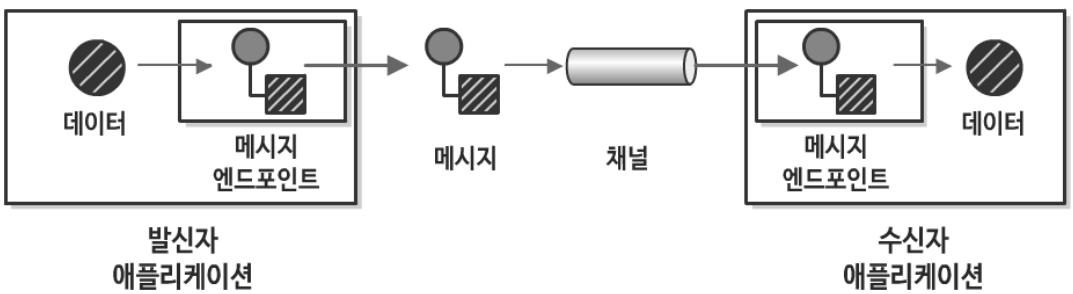
계층	처리대상	변환요구(예)
데이터구조 (애플리케이션계층)	개체, 연관성	다대다 관계를 축약집계
데이터 형식	필드 이름, 자료형, 값 도메인, 제약 조건, 코드 값	• ZIP 코드를 숫자에서 문자열로 변환 • 이름 필드와 성필드를 합쳐서 하나의 이름 필드로 구성 • 미국 주 이름을 두 문자 코드로 변경
데이터 표현	데이터 포맷(XML, 이름/값 쌍, 고정 길이 데이터 필드 , 업체 독자 포맷 등) 문자집합(ASCII, UniCode) 암호화/압축	• 데이터 표현을 해석해 다른 포맷으로 변경 • 암호화/복호화
전송	통신 프로토콜 : TCP/IP 소켓, HTTP, SOAP, JMS 등	메시지 내용에 영향없이 데이터를 이동

3. 메시징 시스템

4) 메시징의 기본 개념

(5) 메시지 엔드포인트

- 메시징 시스템과 애플리케이션의 서로를 접속시키는 부분
- 메시지 엔드포인트를 사용해 어플리케이션과 메시징 채널을 연결
- 메시지 엔드포인트는 어플리케이션이 메시지를 발신하고 수신하는데 사용하는 메시징 시스템의 클라이언트



핵심정리

1. 통합 패턴

- 통합의 정의
  - 연속되고 통일된 기능 집합을 만들기 위해 서로 다른 애플리케이션들을 함께 엮는 작업
  - 이전에 해결했던 문제와 새로운 문제를 비교하면서 통합에서 발생하는 문제 해결
- 애플리케이션의 통합 시 고려해야 되는 기준
  - 애플리케이션 결합도(Application Coupling)
  - 통합 단순성(Integration Simplicity)
  - 통합 기술(Integration Technology)
  - 데이터 포맷(Data Format)
  - 데이터 적시성(Data Timeliness)
  - 데이터 아니면 기능(Data or Functionality)
  - 비동기성(Asynchronicity)

2. 통합 스타일

- 파일전송
  - 애플리케이션들을 통합하기 위한 가장 간단한 통합방식
- 공유 데이터베이스
  - 데이터베이스(DB)를 이용하여 애플리케이션들 간의 데이터를 공유하는 방법
- 원격 프로시저 호출
  - 애플리케이션이 다른 애플리케이션의 기능을 호출
- 메시징 시스템
  - 애플리케이션 간에 빠르고 신뢰할 수 있는 통신을 비동기 방식으로 가능하게 하는 전송 기술

핵심정리

3. 메시징 시스템

- 메시징 시스템
  - 메시지라 불리는 데이터 패킷을 전송함으로써 상호간 통신을 하는 시스템
  - 원격통신, 플랫폼 통합, 비동기 통신, 시간조절, 중재 등 기존 통합 스타일 보다 높은 효율성을 보임
  - 복잡한 프로그래밍, 순서 문제, 제한된 플랫폼 통신부하 등의 문제점을 가지고 있음