SSO, PKI, Jennifer, WAS, WEB Server, 3-Tier

2021년 8월 2일 월요일 오후 1:52

SSO : Single Sign on

- 하나의 로그인 정보를 이용해 여러 애플리케이션에 접근
- 생산성, IT 모니터링 및 관리, 보안통제 측면에서 유리 비밀번호분실이나 취약 비밀번호의 위험도를 낮춰준다.

장점:

- 직원 퇴사시 다양한 내부 애플리케이션에 로그인 할 수 있는 기능을 한번에 해제할 수 있다. (효율적 관리)
- 공인인증서 등으로 추가인증을 할 수 있다 (보안성 강화)
- 아무래도 로그인을 한번만 하면 되니까 편하다 (사용자 편의성 증가)

단점:

- 서버가 단일실패지점(SPoF): ID/PW 노출 시 전체 시스템 위험
- SSO 서버 침해시 모든 서버의 보안 침해
- SSO 개발 및 운용 비용
- 자원별 관한관리 미비

PKI : Public Key Interface

- 인증기관(CA)에서 공개키와 개인키를 포함하는 인증서를 발급받아 네트워크 상에서 안전하게 비밀통신을 가능하게 하는 구조

중요성

- 개인키 관리 및 인증을 위한 CA의 운영을 통해 대행키(비밀키)암호 방식의 운영상 한계를 극복
- 키의 분실이나 분배상의 어려움에 대한 해소방안
- 웹상에서 비즈니스나 안전한 거래를 보증하기 위한 대안

절차

1. 사용자가 등록대행기관(은행) 에게 발급요청
2. 등록대행기관(은행)이 CA(인증서발급기관)에 발행요청
3. 유효성 확인 VA(검증기관)
4. 사용자에게 인증서 발급
5. 사용자가 온라인 상점에 인증서/전자서명
6. 온라인 상점이 VA(검증기관)에 인증서확인

Jennifer - 국내기업

모든 트랜잭션이 웹 애플리케이션 서버에 들어오는 순간부터 처리되기 전까지 전 과정을 모니터링

- 직관적인 그래프

  모든 트랜잭션의 응답 시간을 개별 점 그래프로 표현

WEB 서버

정적인 컨텐츠를 요청 받아서 처리합니다.

WAS란

동적인 컨텐츠(JSP, ASP, PHP 등) 요청 받아 처리합니다.

웹 브라우저와 같은 클라이언트로부터 웹 서버가 요청을 받으면 애플리케이션의 로직을 실행하

여 다시 반환

WEB 서버와 WAS의 동작과정:

1. Client가 Web Server로 HTTP Request 요청
2. Web Server가 WAS로 애플리케이션 로직 실행 요청
3. WAS가 Database로 데이터 요청
4. Database가 WAS로 데이터 반환
5. WAS가 Web Server로 로직 처리 결과 반환
6. Web Server가 Client로 HTTP Response.

왜 WEB서버와 WAS를 나누는가
사실 WAS의 경우 웹 서버+ 웹 컨테이너의 개념이라 웹 서버가 없더라도 웹 서버의 역할을 동시에 수행할 수 있다.

그래서 웹 서버를 사용하지 않더라도 웹 서버와 WAS를 나눠서 사용합니다.

1. 데이터 처리 방식

위에서 말씀드린 것처럼 웹 서버는 정적인 컨텐츠를 처리하고 WAS는 동적인 컨텐츠 처리

만약 부하가 적은 웹 서비스라면 두가지의 요청을 하나의 WAS에서 처리하면 되지만,
부하가 많다면 굳이 빠른 시간에 처리할 수 있는 정적 컨텐츠를 WAS에서 처리하여 부하를 줄 필요가 없다.

2. 보안

사용자들에게 WAS는 공개되어질 필요가 없습니다.
위의 동작가정을 보시면 아시겠지만 사용자에게 요청은 웹 서버가 받고 그 요청을 웹 서버가 WAS에 전달합니다
그리고 WAS의 경우 DB서버에 대한 접속 정보가 있기 때문에 외부로 노출될 경우 보안상 문제가 될 수 있습니다.

1. 오픈소스

WAS의 종류
WAS의 경우 웹 서버와 마찬가지로 오픈소스와 상용 소프트웨어가 있습니다.

아파치 Tomcat
아파치와 같이 아파치 재단에서 관리하는 WAS

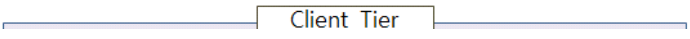
Jetty
이클립스 재단에서 관리하며, 임베디드 자바 애플리케이션에서 웹 서비스를 제공하고 이클립스를 IDE컴포넌트로 사용

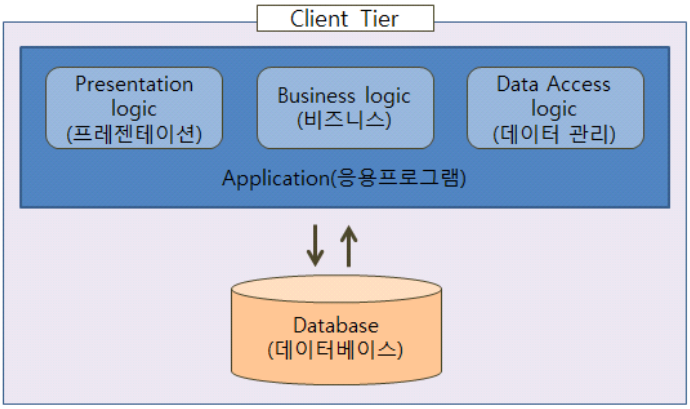
2. 상용 소프트웨어

WebLogic
오라클에서 관리하는 WAS입니다.
Jboss
레드햇에서 관리하는 WAS
Jeus
WebToB와 같이 국내 회사인 티맥스소프트에서 관리하는 제품으로써 국내에선 많이 사용.

3계층 구조 (3 Tier Architecture)

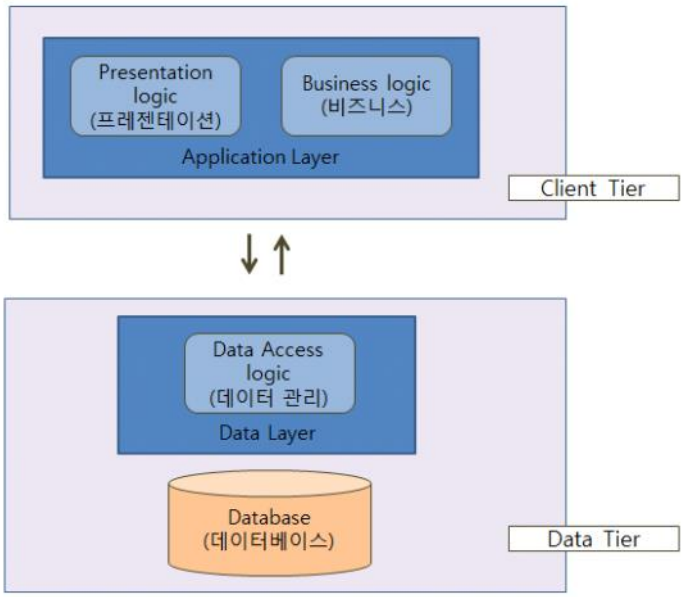
1계층 구조(1 Tier)





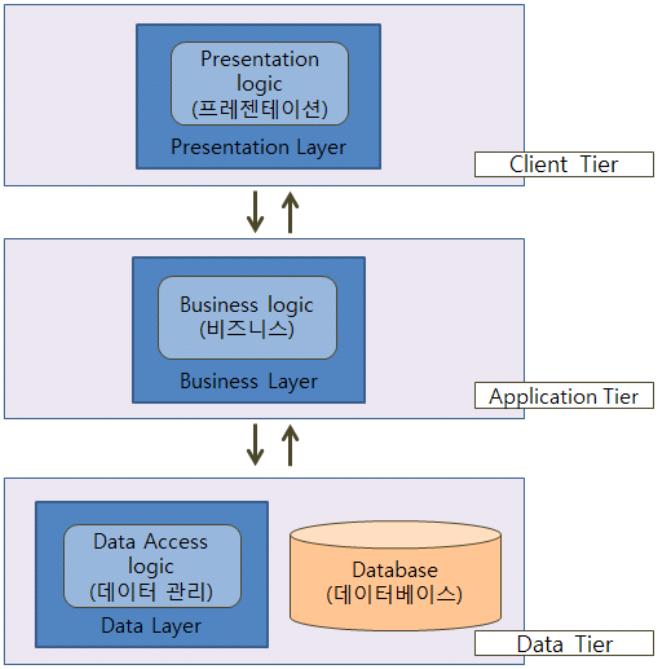
위 그림은 한 클라이언트 컴퓨터에 3가지 로직을 다 구현한 것을 표현한 그림. 한 클라이언트 서버에서 모든걸 지원하므로 새로운 컴퓨터를 사용하고자 할 경우 모두 새로 변경해야 한다는 단점(한가지 로직을 바꾸려면 다른 로직의 변경도 필요하다)이 있다.

2계층 구조(2Tier)



Client Tier와 Data Tier로 2개의 물리적 컴퓨터로 구분된다. 클라이언트와 서버를 분리하여 어플리케이션과 데이터베이스가 분리되어 있기 때문에 데이터베이스의 변경이 편리한 장점을 가지고있다.

3계층 구조



3계층 구조에서 각 계층은 물리적으로도 독립적이며 각 계층의 변경이 다른 계층에 의존하지 않는다..

1. 프레젠테이션(클라이언트) 계층

프레젠테이션 계층은 응용 프로그램의 최상위에 위치하고 있는데 이는 서로 다른 층에 있는 데이터 등과 커뮤니케이션을 한다.

- 사용자 인터페이스를 지원한다. (인터넷 브라우저의 정적인 데이터를 제공한다.)
- 이 계층은 GUI, 또는 front-end도 불린다.
- 비즈니스 로직이나 데이터 관리코드를 포함해서는 안된다.
- 주로 웹서버를 뜻한다(물리적 : WEB서버)

Ex) HTML, javascript, CSS, image

2. 애플리케이션 계층

이 계층은 비즈니스 로직 계층 또는 트랜잭션 계층이라고도 하는데, 비즈니스 로직은 워크스테이션으로부터의 클라이언트 요청에 대해 마치 서버처럼 행동한다. 차례로 어떤 데이터가 필요한지를 결정하고, 메인프레임 컴퓨터 상에 위치하고 있을 세 번째 계층의 프로그램에 대해서는 마치 클라이언트처럼 행동한다.

- 정보처리의 규칙을 가지고 있다.(동적인 데이터를 제공한다)
- Middleware 또는 back-end로 불린다.
- 프레젠테이션코드나 데이터관리 코드를 포함해서는 안된다.
- 주로 어플리케이션 서버를 뜻한다(물리적 :WAS서버)

Ex) JavaEE, ASP.NET, PHP

3. 데이터 계층

데이터 계층은 데이터베이스와 그것에 액세스해서 읽거나 쓰는 것을 관리하는 프로그램을 포함한다. 애플리케이션의 조직은 이것보다 더욱 복잡해질 수 있지만, 3계층 관점은 대규모 프로그램에서 일부분에 관해 생각하기에 편리한 방법이다.

- 데이터베이스를 주로 뜻한다.
- DB 또는 File System를 접근 및 관리한다.
- Back-end라고도 불린다
- 주로 DB서버를 뜻한다(물리적 :DB서버)

Ex) MYSQL DB, Oracle DB