Java Enterprise

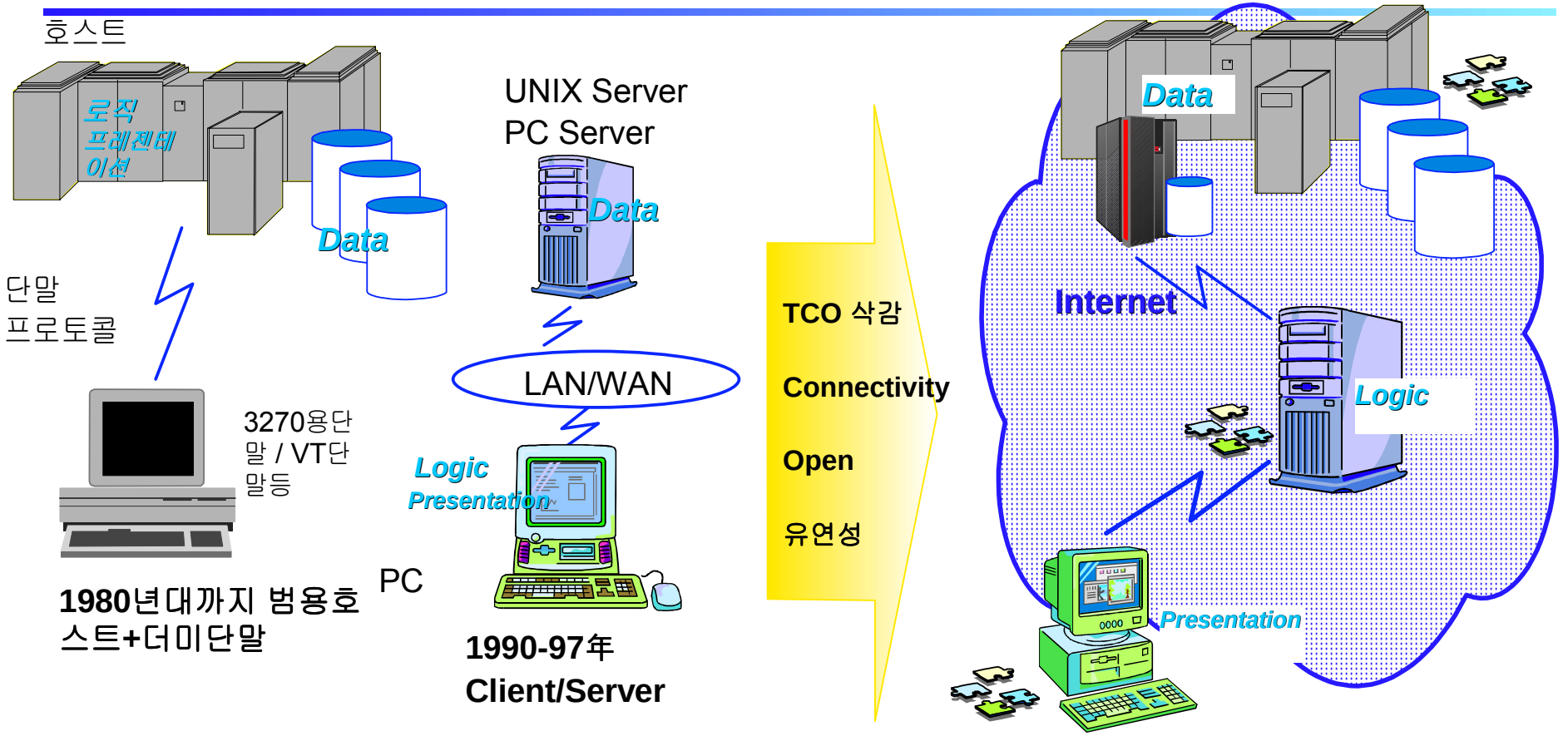
Kim, Good hyun

Web Computing의 기본

- 원칙: 서버는 HTTP를 받아 HTML을 되돌린다.
  - 종래의 C/S에 비해 심플하고 배포가 용이
  - 로직과 프레젠테이션이 깔끔히 분리
- HTTP와 HTML만으로 무리라면?
  - Business Logic이 담긴 Applet
    - 화려하고 친숙한 UI를 제공 : Intranet에 적합
    - 그러나 Scalablity를 압박할 우려가 있다.
  - DHTML, JavaScript, 적당한 Applet의 Thin Client化로.
    - Web Computing의 메리트를 살리도록

IBM

2계층에서 3계층으로..

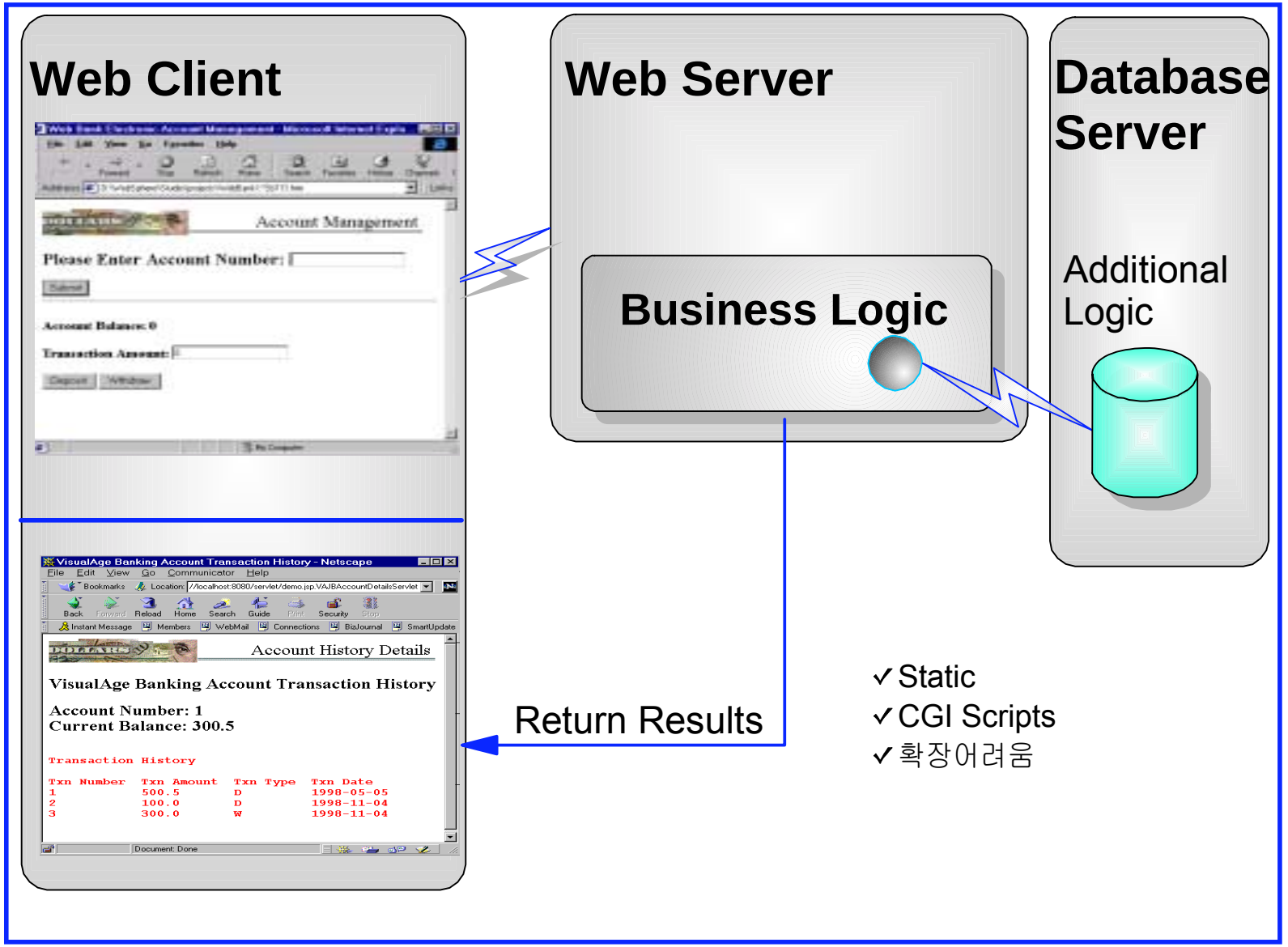


2 계층 시스템

3 계층 시스템

IBM

Web Computing의 기본



Web Application의 요건

- Interactivity
  - 「2003년까지, 인터넷의 소비자 구성이 변화함에 따라, 기업의 마케팅정보 발신에만 중점을 두는 웹사이트로는, 트랜잭션량이 지금까지의 최소가 되고, 이 사이트를 찾는 고객의 수도 급격히 떨어진다. (가능성 = 0.9) 」 (Gartner Group 1999、Oct.)
    - 정보 Publishing(정적 Contents)에서 One-to-One마케팅(동적 Contents)으로
- 고속 처리
  - Web Application은 '유저수'를 가늠할 수 없다.
    - 어떤 부하에도 견디는 테크놀로지

IBM

Web Application의 요건

- Scalibility
  - 「앞으로 3년간, 기업이 고객으로부터 받는 (전자메일 및 Web form에 의한) 질의의 양은 연평균 35% 증가하게 된다 (가능성 =0.7) 」 (Gartner Group 1999、Oct.)
    - 투자와 리스크를 최소화하면서 애플리케이션 부하의 증대를 견디도록
- 개발 생산성
  - Server-side program을 효율적으로 개발 / Debug하는 방법은?
    - 웹 전체를 통합하는 개발툴

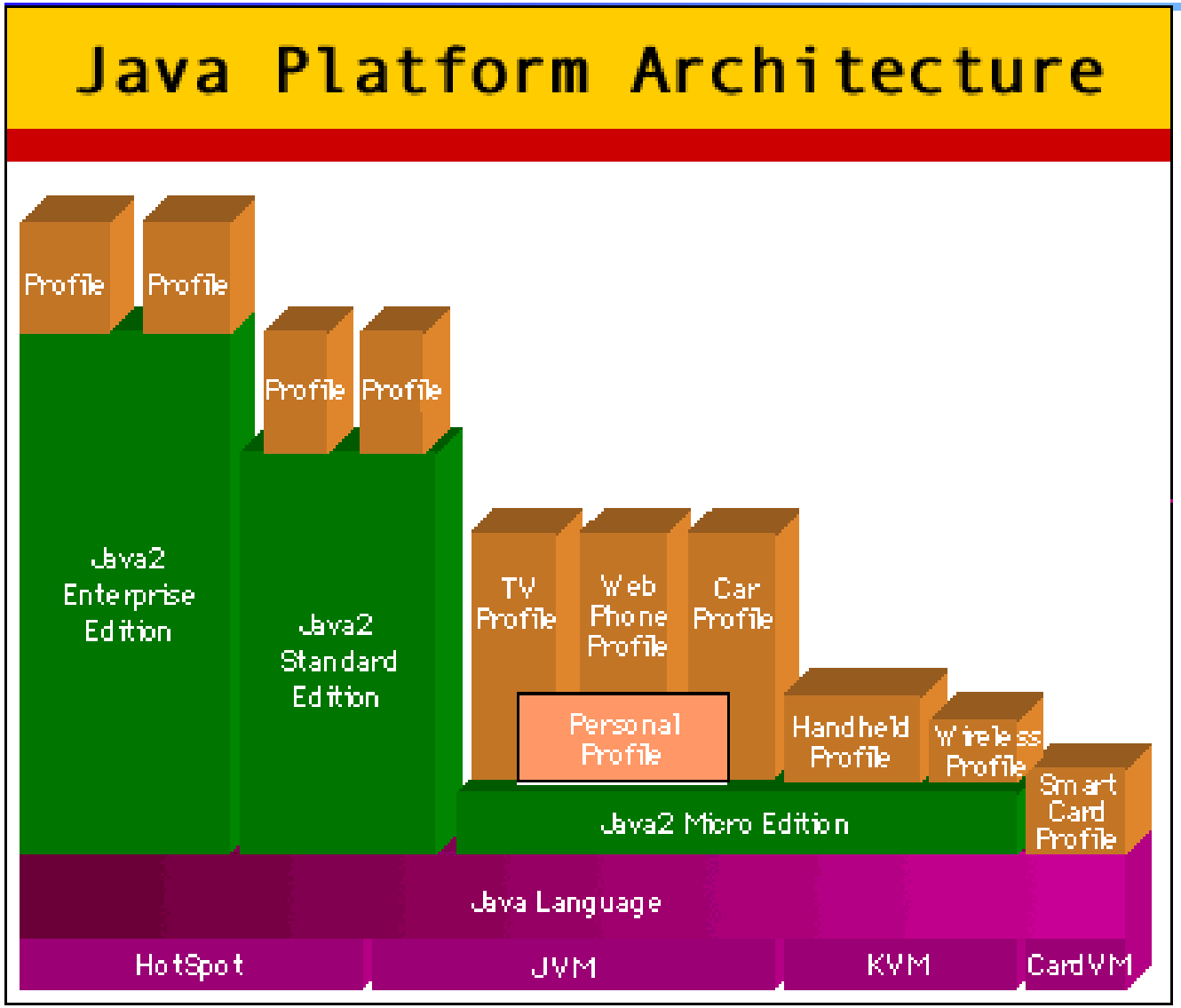
IBM

Web Application의 요건

- 신뢰성 & 가용성
  - Network 저편의 보이지 않는 고객으로부터 신용을 얻으려면
    - 장애시의 fail-over, Availability
- Connectivity
  - 변화하는 Business speed를 따라잡고, 앞서기 위해서는 기존 IT 자산의 활용이 필수 과제.
    - 기존 Data, Host System과의 Connector

IBM

Java2 Platform @ JavaOne '99



IBM

Web Application Server ?

- Web Application에 대한 니즈가 증가함에 따라 등장한 신개념의 Middleware
- Transaction / Database 중시의 종래 시스템과 Document 중시의 Web 사이의 간격을 매우려면..
- Java에 의한 애플리케이션 구축 기능
  - Servlet, JSP, EJB

IBM

Web Application Server

- IBM Websphere
- BEA Weblogic
- Inprise Application Server (JBuilder, Visibroker...)
- IONA / Symantec (OrbixHome + Visual Cafe)
- Sun / Netscape (NetDynamics + NAS)
- Microsoft (not Java)

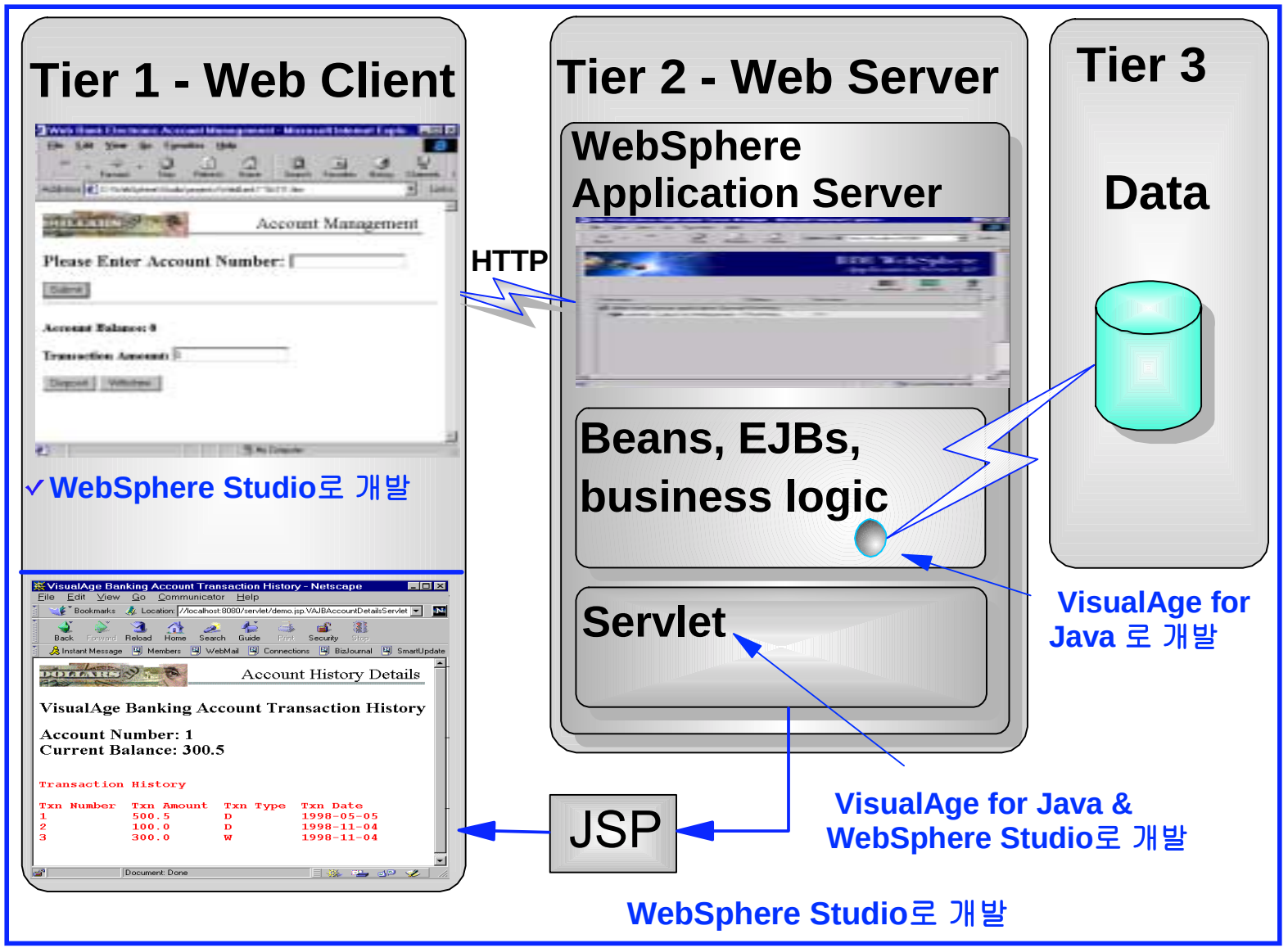
IBM

JavaOne '99의 IBM Technology

- San Francisco framework
  - 차기판에서는 Component Architecture를 EJB로 이행. EJB판은 Websphere상에서 가동
- Websphere studio v 3.0
  - 최초의 JSP Visual 개발툴
- VA for Java Pro 3.0
  - SQLJ, DB2 stored procedure 구축기능
  - Websphere 연계 강화
- Linux판 VA for Java

IBM

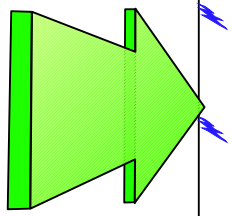
Web Application: WebSphere



Web application의 기술 변화

現行Web Application 기술의 문제

- CGI의 한계
  개발 생산성, Portability, Performance
- Not so interactive
- 보수나 관리의 문제
- 매력적인 웹사이트 구축의 어려움
- 기간업무에 축적된 방대한 데이터를 웹에서도 이용하고 싶다.
- 액세스가 집중하면 반응이 느려진다



진화하는 Web Application에 필요한 기능과 기술

- **Java Enterprise** 에 의한 Web Application 개발
- **Java Servlet/JSP** 실행환경
- Coding하지 않는 개발환경의 제공
- **개발->실행->운용**의 전 사이클을 최소한의 부담만으로 지원하는 통합된 제품군
- 개개의 유저에 대해 Personalize된 **Dynamic Contents**를 손쉽게 작성
- 기간업무와의 **Connector**제공
- Web Server의 **負荷分散**기능 제공, 나아가 WLM에 의한 Application level의 부하분산

IBM

IBM WebSphere의 장점

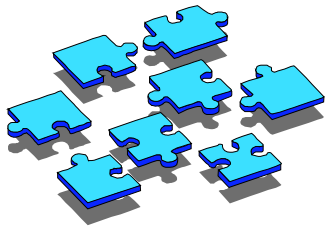
- ☑ 고객의 요건에 맞는 풍부한 소프트웨어 제공
- ☑ 웹 애플리케이션 구축을 위한 개발운용환경을 토탈 서포트
- ☑ 투자를 보호하는 업계 표준, 오픈스탠다드에 준거한 **Component Architecture**
- ☑ **Host** 연계, 대규모 트랜잭션, 애플리케이션 통합에 있어서의 실적, 그리고 최신 테크놀로지
- ☑ ＣＲＭ,ＥＣ,ＳＣＭ 등 모든 **e-business**의 기반

Why Java? Why Websphere?

- **e-business** 혁신
  - 업무 시스템의 **World wide**화, **Realtime**화, **netrwork**화
  - 급격한 성장과 시간 단축

- **Java**의 확대
  - 급속한 파급
  - 높은 생산성, **Network, Security, Multi-platform**

- **Component** 지향
  - 조립 **vs** 처음 부터 개발
  - **Open vs** 독자
  - **OO Technology**의 성과



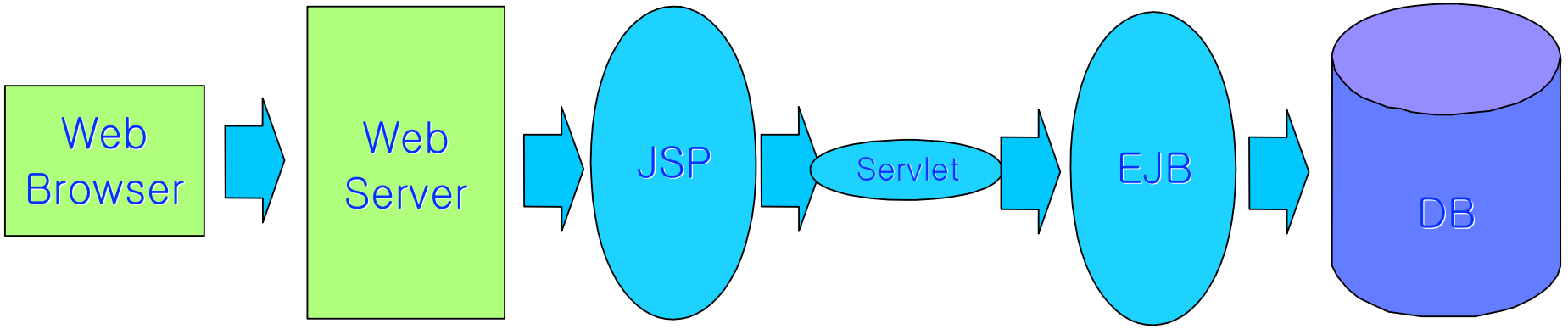
IBM

서버측 Java 아키텍쳐 (J2EE)
3가지 키워드

- Servlet
- JSP(Java Server Pages)
- EJB(Enterprise JavaBeans)

IBM

Java Web Computing System Architecture



IBM

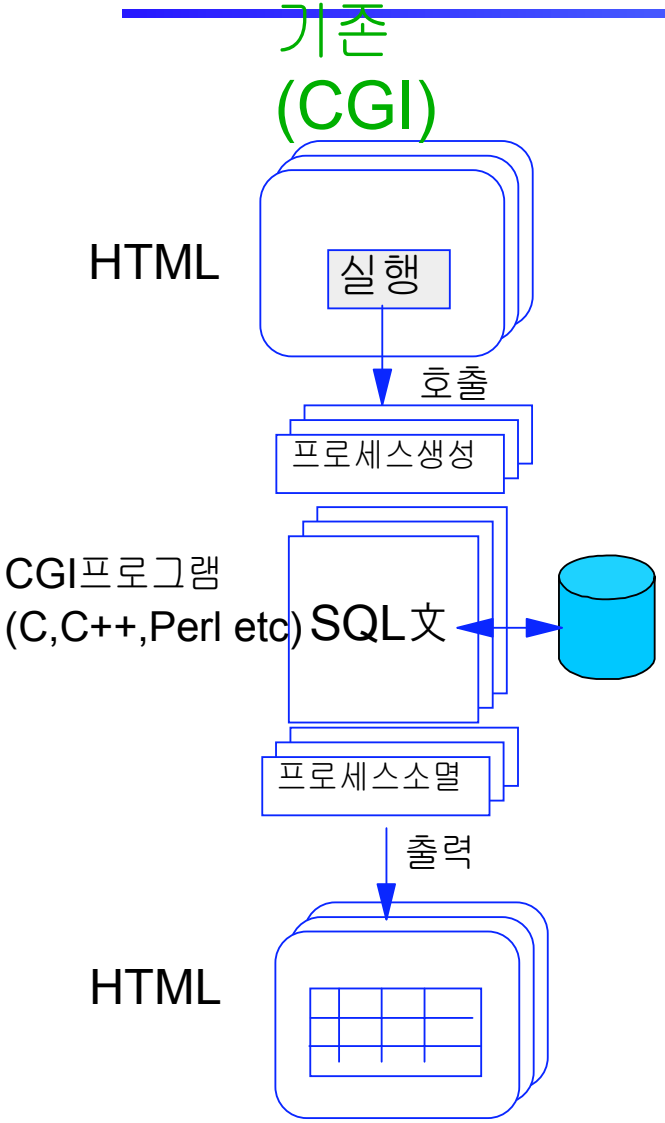
Servlet

- 브라우저에서 불러낼 수 있는 서버측 Java 프로그램.
- CGI와 유사. 종래 CGI에서 개발했던 내용을 이행하는 예도 있다.
  - CGI : Perl, C 등으로 Standard I/O, 표준 Unix Command를 wrapping. 나름대로의 장점.
  - CGI는 Request마다 프로세스가 뜨고, DB 접속시도 매회 세션을 잡아줘야 한다.
- Servlet은 메모리에 상주. 자연스러운 세션관리

IBM

Before Servlet

기존(CGI)의 문제점



- × 프로세스의 생성/실행/소멸을 반복한다.
- × 반복되는 액세스마다 프로세스가 생성

액세스가 집중되면 서버의 부하가 너무 높아진다.

반응이 느려져 유저가 짜증낸다.

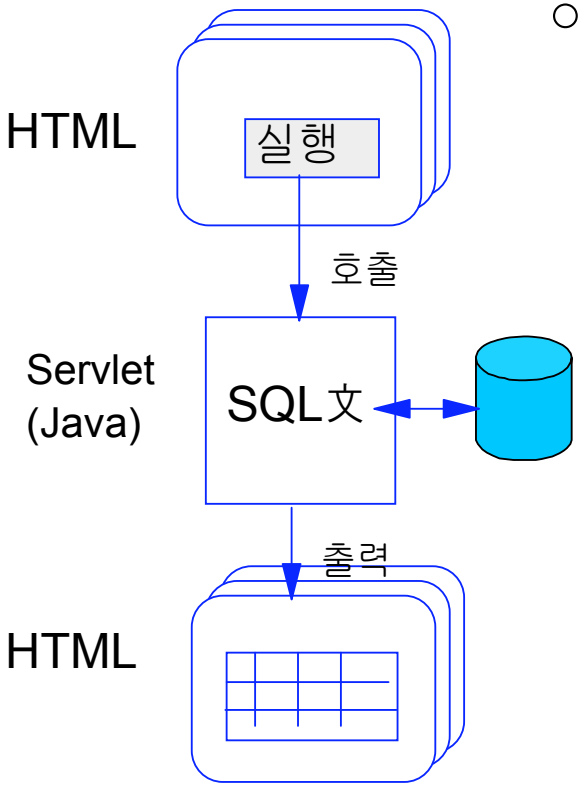
IBM

Servlet

WebSphere도입후(Servlet)

Servlet

- 메모리상주형이므로 프로세스의 생성/소멸의 부하가 없다.
- 서버관리가 용이



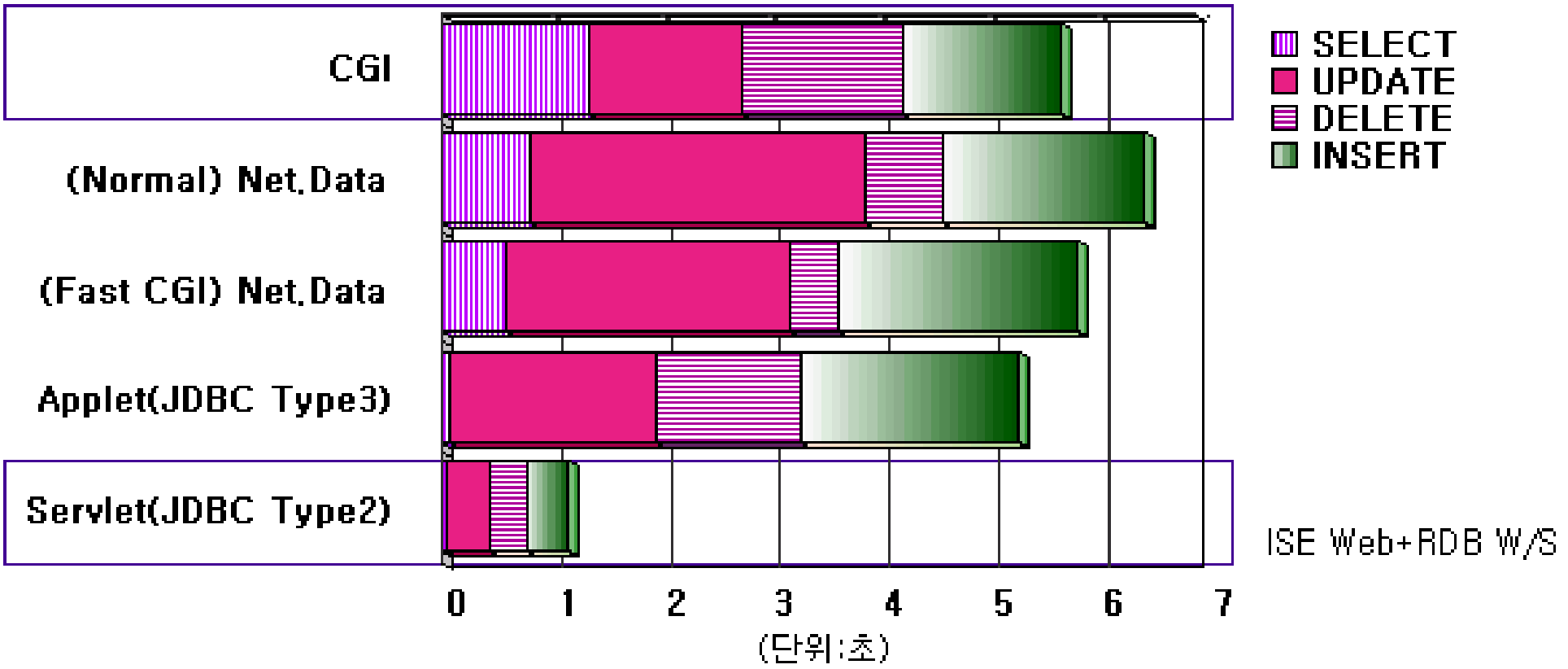
액세스가 집중해도 견딜 수 있다.

단, Servlet에 프레젠테이션 로직과 비즈니스로직이 뒤엉켜 개발이 번잡하다.

IBM

CGI/Servlet 퍼포먼스 비교

Test Case : 사원정보 테이블(10만건) 에 대한 검색/갱신/삭제/삽입(100건씩)
Test 환경 : RS/6000 43P AIX4.2.1, Domino Go Webserver4.6.1,WebSphere 1.1,UDB5.0



Servlet은 CGI에 대해 전체적으로 약5배、
검색만이라면 약20배빠르다

Servlet만의 시스템에서는...

- 소스 여기저기에 HTML을 print하는 코드가 산재
- 디자인 변경시 소스를 수정하고 재컴파일
- HTML이 print문안으로 삽입되어 알아보기 힘든 코드 생성
  →Wysiwyg HTML 에디터등 사용불가

JSP에 의해 Presentation을
Business Logic과 분리

IBM

JSP (Java Server Pages)

- Server측에서 작동하는 Script언어. 동적으로 HTML 페이지를 생성해 내는 기술.
- HTML페이지 안에 삽입되어 서버에서 처리
- 사용자에게는 처리후의 깨끗한 HTML만 전달
- Microsoft의 ASP와 유사
  - Not VB but Java
  - 다양한 Web Application Server가 지원
  - NT뿐만 아니라 다양한 중대형 서버에서도 작동

IBM

Servlet 없는 JSP만의 시스템에서는...

- JSP Script들 간의 데이터 교환, 제어가 힘들다
- Multithread에 의한 Script의 병행 처리 불가
- HTML안에 삽입된 Script가 비대해져 알아 보기 힘든 코드 생성 →Sphagetti Code
- Not So Object Oriented

Servlet이 JSP를 통해 HTML 표현
Logic은 Servlet에 이관

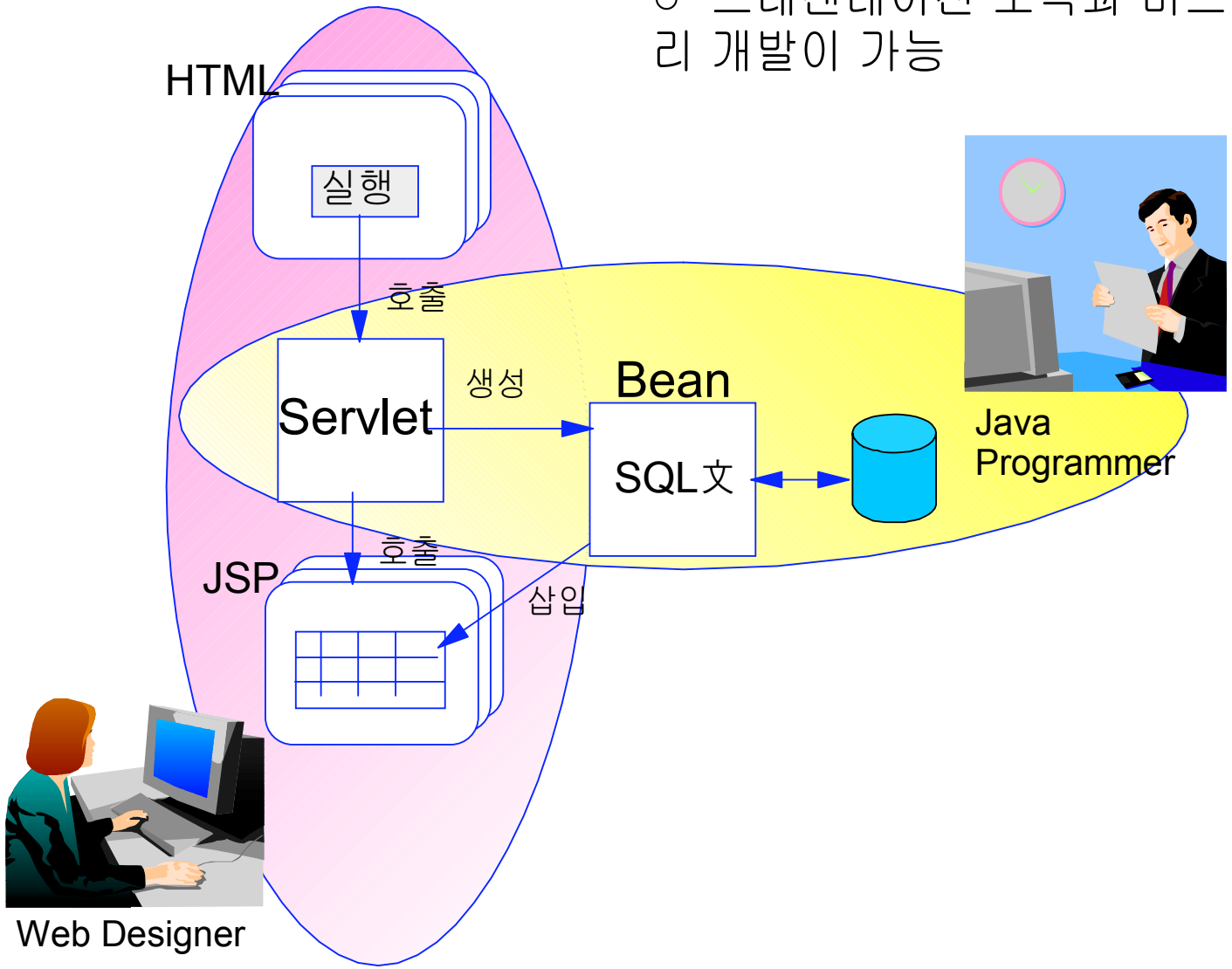
IBM

Servlet과 JSP

WebSphere도입후 (JSP)

JSP

○ 프레젠테이션 로직과 비즈니스 로직의 분리 개발이 가능



IBM

JSP와 Servlet의 공존 포인트

- Servlet에서 print 메소드로 HTML을 한줄한줄 출력하는 대신 forward() 메소드로 JSP로 전송
- Servlet 프로그래머는 HTML로 출력할 데이터를 Java 오브젝트(JavaBeans)에 저장
- JSP/HTML 디자이너는 <jsp:useBean> 으로 Java 오브젝트를 끼워넣고 <jsp:getProperty>로 속성값 취득
- Servlet과 JSP 제작자는 각자의 업무에 전념하고, 별도로 버전업도 가능하다.

IBM

Servlet 만으로 Logic을 만들자면..

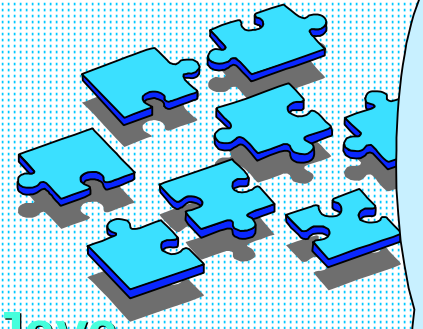
- 프로그래머는 모든 로직에 대해 처음부터 하나하나 코딩하지 않으면 안된다
- 모두 Java로 짜여졌더라도 코드 수준에서의 재사용은 힘들다
  →남의 코드는 읽기 힘들다
- 서로 다른 프로젝트간에 공유가 가능하고, 분산화된 오브젝트를 실현하기 위한 환경이 필요

실행 환경 및 Component Model로서의 EJB 대두

IBM

EJB

Web Application Server
(EJB없이)



Java Beans

- 비즈니스 로직
- 트랜잭션 처리
- 세션 관리

.....프로젝트마다 서로 다른 스타일의 프로그램이 만들어 진다. 모처럼의 Java 코딩의 Portability가 떨어 진다.

WebSphere Application Server

EJB Server

Advanced Ver.

EJB Container

Enterprise Java Beans

- **Bean**은 비즈니스 로직만

모든 프로젝트에서 공유될 인프라(실행환경 +컴포넌트 모델)를 **EJB** 서버와 **EJB** 컨테이너가 제공

- Standard판에서는 백엔드의 자원에 액세스하기 위해서 커넥터 제품을 써야만 했다.
- Advanced판에서는 백엔드의 자원에 액세스하는 표준적인 방법 (ＥＪＢ)을 제공
- Enterprise Java Beans는 급속히 보급되고 있는 분산 오브젝트 표준

IBM

EJB (Enterprise JavaBeans)

- Business Logic의 부품화, 규격화
- Software의 자산화
- 개발한 모듈을 다른 프로젝트에서 이용할 수는 없을까?
- Component Model의 필요성
- 미들웨어(EJB 서버/컨테이너)가 부품을 끼워 넣는 틀을 제공하여 생산성과 재사용성을 높임
- 그러나 아직은 산업으로서의 기반이 취약

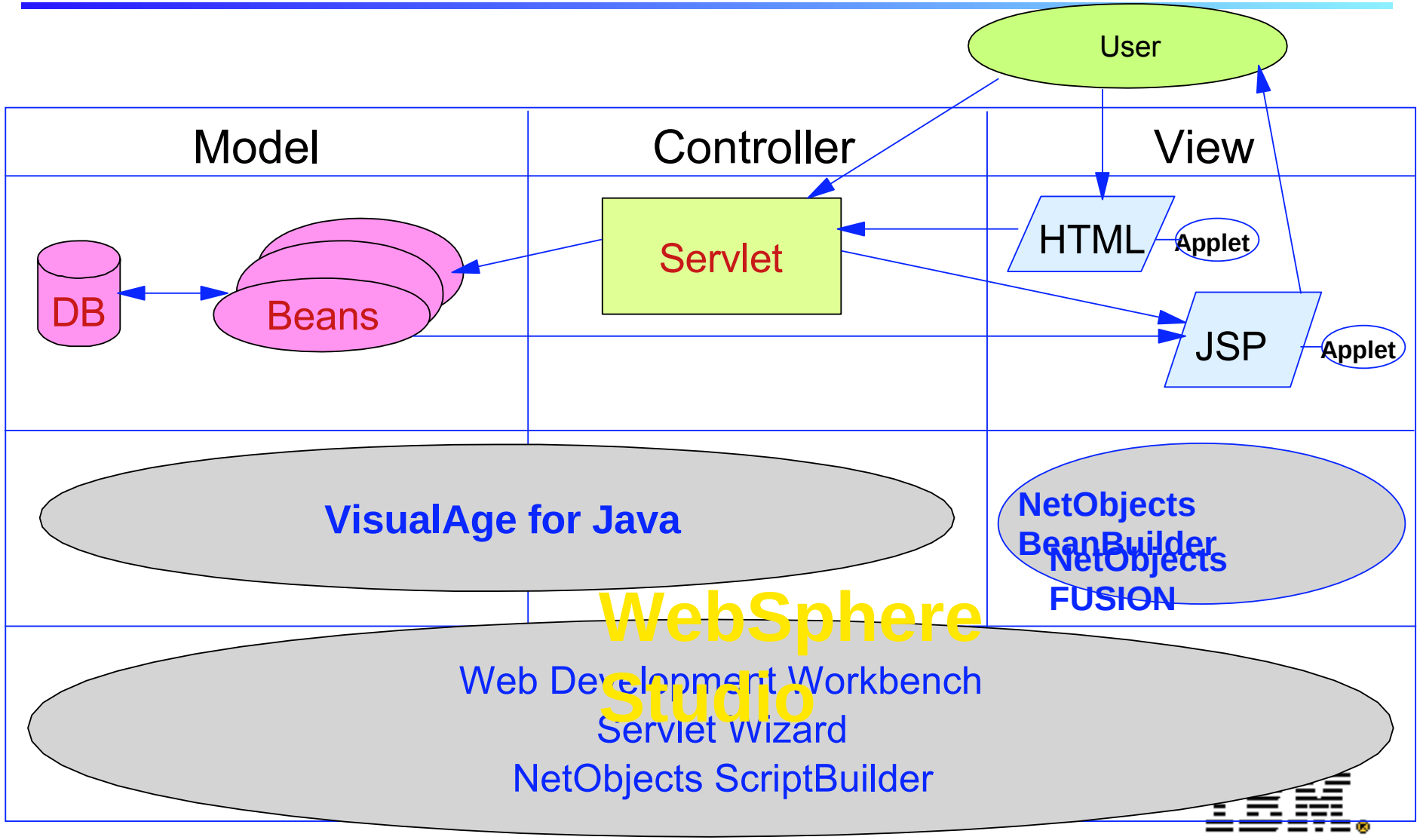
IBM

MVC Model과 Java

- MVC(Model / View / Controller)
- JSP : View
  - 어떻게 보일지를 규정한다.
- Servlet : Controller
  - 처리를 기술한다 (계좌 잔고에 대한 query)
- EJB : Model
  - 로직을 기술한다 (계좌 잔고 계산하는 로직)

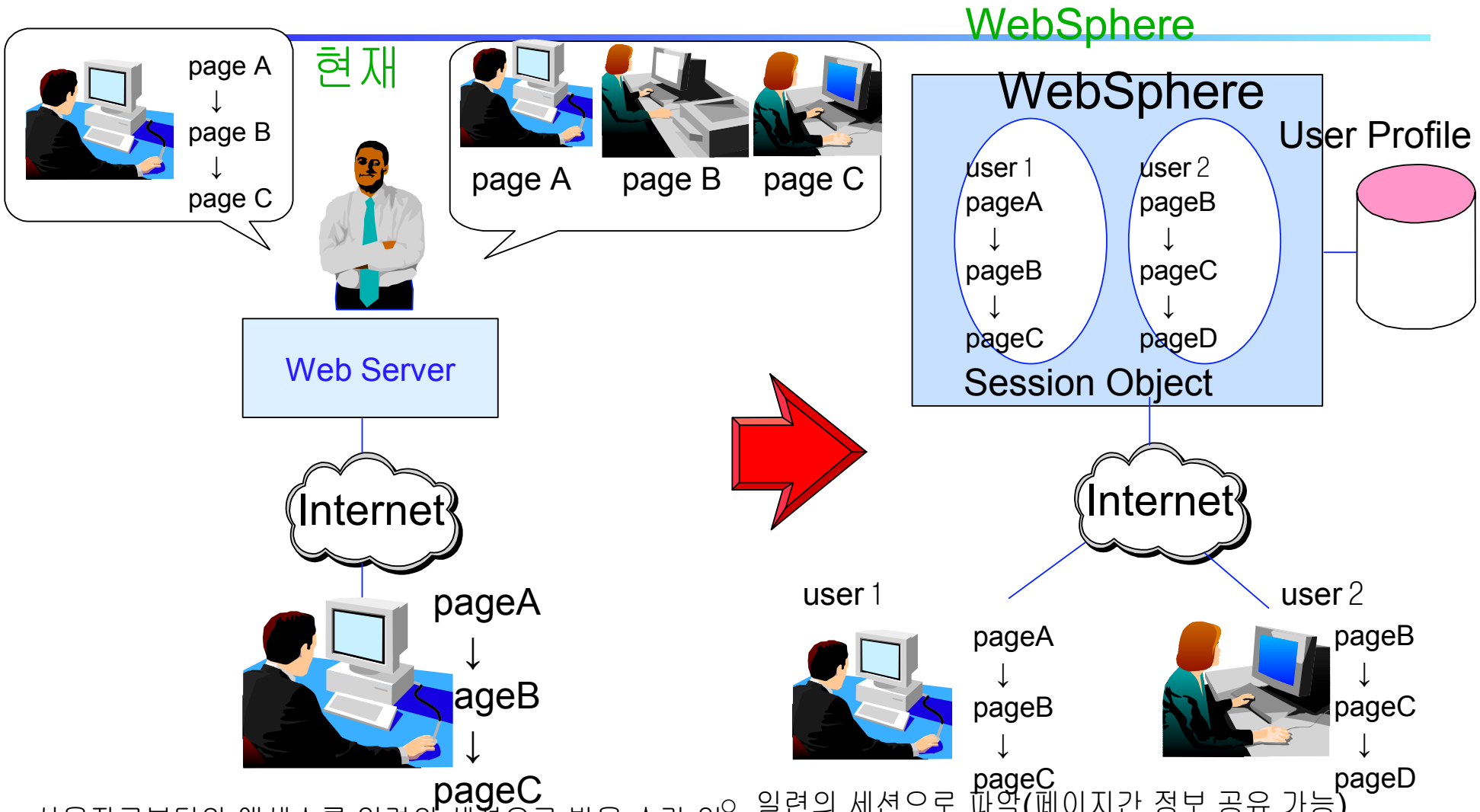
IBM

Web Application 의 설계와 개발



기타 유용한 개념①

: Session 관리 기능



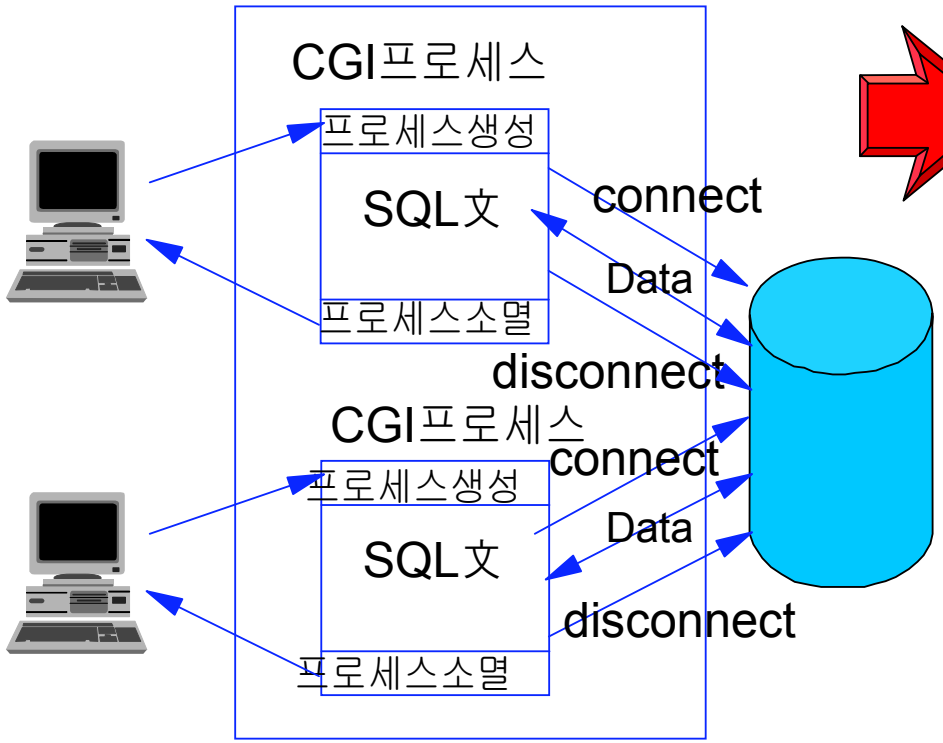
× 사용자로부터의 액세스를 일련의 세션으로 받을 수가 없다.

○ 일련의 세션으로 파악(페이지간 정보 공유 가능)
○ 화면 진행을 제어 가능(BackButton 문제등)
○ 유저마다 커스터마이즈한 페이지를 제공가능 (=Personalization)

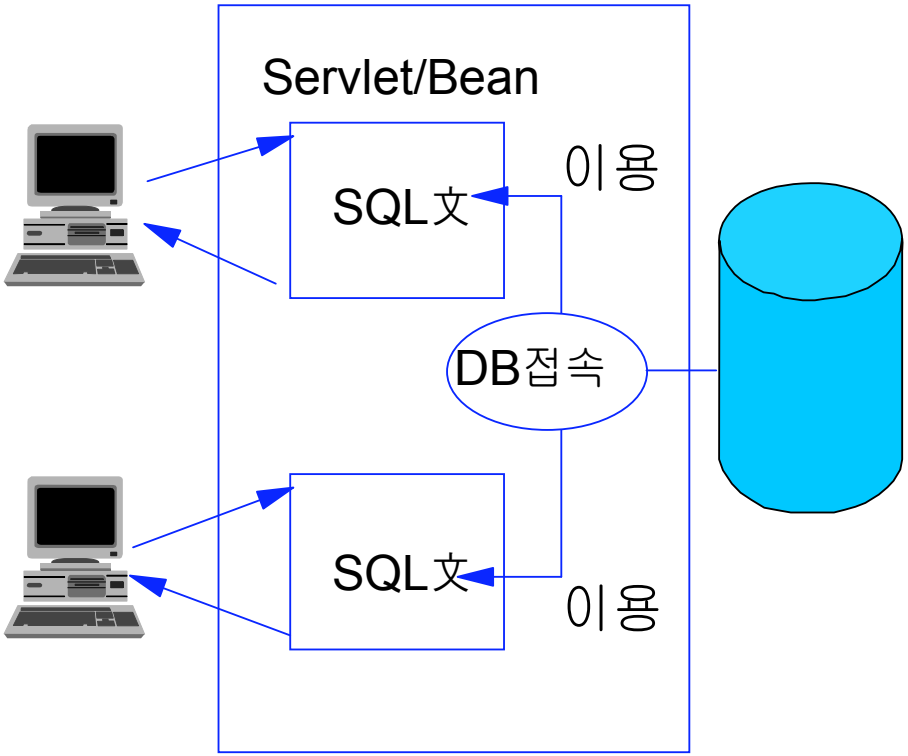
기타 유용한 개념 ②:

DB 접속 Pool 기능

기존



WebSphere



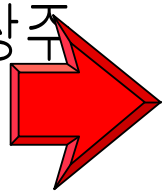
× 유저로부터의 request마다 발생하는 CGI process로 DB로 Connect、Disconnect를 반복한다.

○ Connect, Disconnect 의 부하가 줄어 퍼포먼스가 향상된다.

IBM

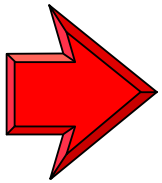
WebSphere Application Server의 가시적 이점

- Servlet: Multithread, Memory상주
- DB Connection Pooling



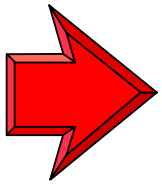
퍼포먼스 향상

- JSP(JavaServer Page)



디자인과 로직의 분리 개발

- Session 관리 기능



사용자마다 Customize된 화면 제공가능
사용자별로 액세스 통계
화면 flow 제어가 가능

IBM