# History of Enterprise Java


- At first: Sun focused on the Java Development Kit (JDK)
  - Remember that Java is a spec, not a technology
  - Different vendors can implement Java
  - The JDK became the de-facto reference implementation of the Java platform
  - No enterprise features yet

# History of Enterprise Java


- Then, Sun released new specifications for “Java extensions”
  - Naming and lookup services (Java Naming and Directory Interface)
  - Transactions (Java Transaction Service)
  - Server-side components (Enterprise JavaBeans)
  - Together, Sun named this the Java Platform for the Enterprise (JPE)
  - Sun also began writing Java extensions for consumers devices
- Idea: A middleware vendor would implement these Java extension specifications and ship it as a product

# History of Enterprise Java

- Alas, there were problems with the JPE
  - Ambiguities
  - Poor synchrony between Enterprise APIs
  - Moving Target
  - No way to test middleware compatibility
  - No reference implementation

# Then came platform separation


- Sun broke up Java into 3 parts:
  - Java 2 Platform, Micro Edition (J2ME)
    - Consumer devices
  - Java 2 Platform, Standard Edition (J2SE)
    - The old JDK
  - Java 2 Platform, Enterprise Edition (J2EE)
    - The Java Platform for the Enterprise (JPE)

# The results of platform separation


- Platform separation had the following benefits:
  - Fewer ambiguities
  - Enterprise API synchrony
  - Locked-down specification revisions
  - A test suite for each edition of Java
  - A reference implementation for each edition of Java

# What is in the J2EE Ecosystem?


- Sun provides:
  - J2EE Platform Specifications
  - J2EE Compatibility Test Suite
  - J2EE Blueprints
  - J2EE Reference Implementation
- From other vendors, you get
  - 30+ vendor implementations of the J2EE specs
  - Tool vendors (UML modeling, IDEs, testing tools, …)
  - Component vendors
  - J2EE-specific web hosting companies
  - Prof services (Consultants, deployers, trainers, …)

# APIs and SPIs

- There are 2 parts to the J2EE interfaces defined by Sun:
  - Application Programming Interfaces (APIs)
  - Service Provider Interfaces (SPIs)
- You write your application to the API
- The J2EE vendor writes a container to the SPI
- The container provides services to your application
- The API/SPI duality is the key to J2EE
- It allows your applications to work with a variety of middleware (in theory…)

# J2EE Containers / Servers


- The J2EE products (from say IBM or BEA) are called containers or servers)
- These provide you with many services:
  - Web services
  - Business logic services
  - Database services
  - Transactions
  - Security
  - Much more
- Some services are more mature than others.
- In general, if you can use the services, development is more rapid than building this yourself.

# J2EE Services

- Java 2 Platform, Standard Edition (J2SE)
- Remote Method Invocation (RMI) and RMI-IIOP
- Java Naming and Directory Interface (JNDI)
- Java Database Connectivity (JDBC)
- JavaMail
- Java Transaction API (JTA)
- Java Transaction Service (JTS)
- Java Messaging Service (JMS)

- + More (cont)...

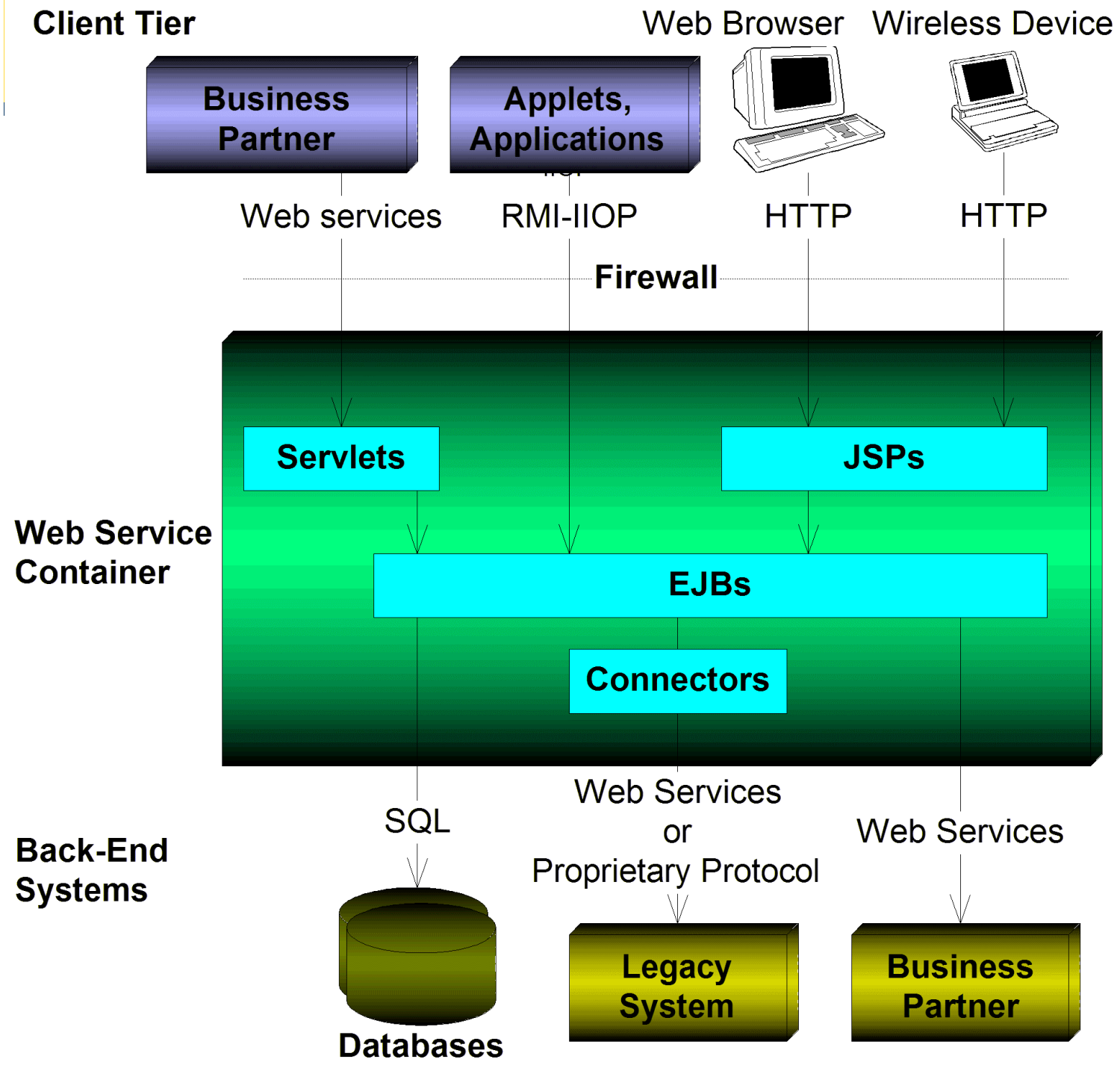
# And More…

- Servlets
- Java Server Pages (JSP)
- Enterprise JavaBeans (EJB)
- Legacy Connectors Specification
- CORBA integration (RMI-IIOP and Java-IDL)
- XML Support

**+ all of Java 2 Standard Edition (J2SE)!!**

# Portability

- Enforced by J2EE test suite
- Must exercise judgment and stick with basic J2EE platform to assure portability
- Not all vendors will license the test suite
- Proprietary extensions exist and are sometimes necessary to use


Client Tier
Web Browser
Wireless Device
Business Partner
Applets, Applications
Web services
RMI-IIOP
HTTP
HTTP
Firewall
Servlets
JSPs
Web Service Container
EJBs
Connectors
SQL
Web Services or Proprietary Protocol
Web Services
Back-End Systems
Databases
Legacy System
Business Partner