Java-technology based projects

TietoEnator Corporation Oyj
Simo Vuorinen
simo.vuorinen@tietoenator.com
Agenda

✓ Java: language, architecture, platform?
✓ Javan promises and problems
✓ Enterprise-APIs
✓ Java-component architectures: JavaBeans and EJB
✓ Application servers
✓ Experiences and experiments
✓ Close-up and future visions
Java: language, architecture, platform?

Programming language:

✓ Easy-to-use object-oriented programming language, which has been cleaned of features that used to cause problems in C and C++.

✓ Designed for Internet- and WWW-use.

✓ Via Web-browser Java gives the possibility to use applications from anywhere in the world.

✓ Java is distributed, interpreted architecture- and platform-neutral programming language that can be used in addition to traditional application building also applets (small applications running in WWW-browser). Applets can be downloaded over the network from anywhere in the world.
Architecture generally:

✓ Architecture (application architecture, system architecture) defines the guidelines, which together help to achieve the information system goals without the need for ad hoc-compromises during the implementation of the system.

✓ Architecture defines
  • general usability
  • performance and scalability
  • availability and robustness
  • security

✓ Architecture can be divided into
  • logical architecture:
    • what is to be done; db-, UI-, bl-services and utilities
  • technical architecture:
    • with what tools; os, db, mw, development tools
Java: language, architecture, platform?

JavaOS

Logical Architecture

JECF

Business logic

Technical Architecture

Java WebServer

Java Application Server

Java Enterprise APIs
Java-platforms

- Desktop OS
  - Enterprise Servers
  - Mainframes
- Java AE
- Personal Java AE
- Embedded Java AE
- SmartCards
  - Java Card

Figure 1

- RTOS Vendors
  - Set-top Boxes
  - PDAs
  - Screen Phones
  - Hi-end mobile phones
  - Internet TV
  - Car navigation
- Industrial Controllers
  - Instrumentation
  - Automotive
  - Printers
  - Hi-end Pagers
  - Mid-range phones
Where does Java run?
Java’s promises and problems

- Ease of use
- Feature richness
- Desktop-features
- Performance
- Matureness
Java’s promises and problems

WWW: Applets

✓ more functionality
✓ “real” applications in WWW
✓ “write once, run anywhere”

✗ Problems with firewalls
✗ problems with browsers and their versions
✗ applets tend to be big; downloading is slow
Java’s promises and problems

Java-applications

✓ Same application runs anywhere that you can find a VM
✓ Programmers don’t make as many errors as they used to
✓ made network-ready
✓ threads are easily implemented

✗ Interpreting slows down the application
✗ Critical real-time-applications require good memory management; GC is in important role

✗ desktop features have been deficient for enterprise usage; printing, drag-drop, db-handling
Java’s performance

✓ Has been slow
✓ But is getting better
✓ JIT-compilers
✓ native compilers
✓ HotSpot
**Java Enterprise-APIs**

**Enterprise JavaBeans Architecture**
The Enterprise JavaBeans specification defines an API that will make it easy for developers to create, deploy and manage cross-platform, component-based enterprise applications that work within the framework of the systems currently in use.

**JavaServer Pages**
JavaServer Pages combines easy-to-use server-side tags with the power of JavaBeans™ components to provide a solution that cleanly separates the presentation of dynamic content from the generation of that content. Neutral to development platforms and Web servers, it's a Write Once, Run Anywhere™ solution.

**Java Servlet**
Java Servlet provides a uniform, industry-supported interface for extending a web server with cross-platform and cross-server components written in the Java Programming Language.

**Java Naming and Directory Interface**
Provides uniform, industry-standard, seamless connectivity from the Java platform to business information assets, thus allowing developers to deliver Java applications with unified access to multiple naming and directory services across the enterprise.

**Java Interface Definition Language (IDL)**
Provides interoperability with CORBA, the industry standard for heterogeneous computing. Java IDL includes an IDL-to-Java compiler and a lightweight ORB that supports IIOP.
Java Enterprise-APIs

**JDBC**
Provides programmers with a uniform interface to a wide range of relational databases, and provides a common base on which higher-level tools and interfaces can be built.

**Java Message Service (JMS)**
The Java Message Service specification provides developers with a standard Java API for enterprise messaging services such as reliable queuing, publish and subscribe communication and various aspects of push/pull technologies.

**Java Transaction (JTA)**
Java Transaction API (JTA) defines a high-level transaction management specification intended for resource managers and transactional applications in distributed transaction systems.

**Java Transaction Service (JTS)**
The Java Transaction Service (JTS) API technology ensures interoperability with sophisticated transaction resources such as transactional application programs, resource managers, transaction processing monitors and transaction managers. Since these components are provided by different vendors, JTS provides open, standard access to these transaction resources.
Java Enterprise-APIs

JavaMail
The JavaMail API provides a set of abstract classes that models a mail system. The API is meant to provide a platform independent and protocol independent framework to build Java-based mail and messaging applications.

RMI/IIOP
RMI-IIOP provides developers an implementation of the Java RMI API over the Object Management Group's industry-standard Internet Inter-Orb Protocol (IIOP). With it, developers can write remote interfaces between clients and servers, and implement them just using Java technology and the Java RMI APIs.
Java-components: JavaBeans and EJB

JavaBeans (Client-component architecture):

JavaBeans-component is composed of three parts:

Properties
Methods
Events

MyBean
...  
getXXX
setXXX
doStuff()
fireEvent(Event ev)

Register Event Listener

SomeClass

Event Object

Fire Event

SomeClass
Java-components: EJB

EJB Clients:
- Client access is controlled by the container
- EJB’s are located through JNDI
- RMI and CORBA are means of accessing a bean over a network

Web Container:
- Runtime Environment for JSP and servlets

J2EE Server:
- Naming and Directory
- Authentication
- HTTP
- EJB

EJB Container:
- Runtime Environment for an EJB
- Transaction mgmt.
- Security
- Remote client connectivity
- Life cycle mgmt.
- DB Connection Pool

EJB Components:
- Specialized Java class
- Distributed over a net
- Transactional & Secure
- Server vendor tools provide: distribution, transaction, security
CASE: Jaguar CTS

Data

Application Layer

Jaguar CTS

Business Logic Layer (Java Classes)

Client

Service Layer

http

JDBC

TDS

Oracle DB

getComp

DBTransactionC

http

GUI Layer (Java)

BugInfo-applet

DBAction

JagDBAction

TDSTDS
CASE: Servlets and Java

Application Layer
- DB Layer (Java Classes)
- Business Logic Layer (Java Classes)
- Service Layer (Servlets)

Data
- FileNet
- Oracle DB

Client
- HTTP Server
- HTML

Servlets
- Collector
- GUI Layer (Java)

Business Logic
- M2DB
- M2User

Service Layer
- JDBC
- M2Html Gen
What does Java offer?

- Client-component architecture (JavaBeans)
- Server-component architecture (EJB)
- 4 different Java-application environments
- 11 enterprise-APIs
- 8 core-APIs (JFC, Security, JavaBeans, 2D, ...)
- Lot of extensions, frameworks, etc.
- Platform-neutrality in reasonable scale
Where can Java be chosen?

- BL requires integration with Internet-technology
- Old systems C/C++ Client-Server-systems
- application is meant to be deployed on more than one platform
When not to choose Java?

- System is complex, business-critical and must be in production in six months
- Lot of integration is needed with legacy-systems
- Existing 3rd-party products fill the need
Mistakes in WWW-technology projects can be caused by:

- Lack of Internet- and WWW-knowledge
- Too big of a desire of experimenting new tech
- Lack of knowledge in Internet security
- Not making a proof-of-concept
- Not testing enough: profiling, load testing, etc.
- Lack of inspections
- Lack of architecture lining
Mistakes in Java-technology projects can be caused by:

- Component recognizing
- C-style, procedural programming
- Disregarding internal security
- Complex object-object relations
- Mixing objects with components