Introduction to J2EE

Dr. Alejandro Zunino
ISISTAN – UNICEN

Main Topics

• Introduction
• What is J2EE?
• Motivations
• The J2EE Architecture

Motivations

• In the beginning, there was darkness and cold. Then, ...

Motivations

• In the 90’s, systems should be client-server

Motivations

• Today, enterprise applications use the multi-tier model

Motivations

• In the 90’s, systems should be client-server

Motivations

• In the beginning, there was darkness and cold. Then, ...

Motivations

• Today, enterprise applications use the multi-tier model

Motivations

• In the 90’s, systems should be client-server
**N-tier Complexity**

- Multitier applications have several independent components.
- An application server provides the infrastructure and services to run such applications.

**Application Servers**
**What is J2EE?**

- The Java 2 platform, Enterprise Edition reduces the cost and complexity of developing ... multi-tier services, resulting in services that can be rapidly deployed and easily enhanced.
- It is a public specification that embodies several technologies.
- Current version is 1.4.
- J2EE defines a model for developing multi-tier, web-based, enterprise applications with distributed components.

**What is J2EE?**

- J2EE means:
  - The J2EE platform:
    - An **standard** for executing enterprise applications.
    - Specified as a number of APIs by JDC (>200 companies).
  - J2EE compatibility tests:
    - For testing J2EE servers.
    - **Low dependency on the vendor!**
    - A free reference implementation
  - The J2EE blueprints:
    - Style guides
    - Norms
    - Patterns
    - Examples

**What for?**

- Programming in the large:
  - Many users
  - Applications should last many years
  - Heterogeneity
  - Versioning
  - Large development teams
  - Easy to maintain, flexible, reusable, scalable...
  - Load balancing
  - Large volumes of data
  - RDBMS independent
  - Secure

**J2EE Benefits**

- High availability
- Scalability
- Integration with existing systems
- Freedom to choose vendors of application servers, tools, components
- Multi-platform

**But...**

- Expensive
- Difficult to learn
- Applications are tied to the J2EE API (server...)

**J2EE Multi-tier Model**
The J2EE Framework

- Components Based Architecture
- Applications are built from components
  - Java class files
  - Configuration files (usually XML)
  - Data files (html, images, directories,...)
- Components are packaged into archives for deployment
  - jar: Java Application Resource
  - war: Web Application Resource
  - ear: Enterprise Application Resource

J2EE Container

J2EE Technologies

- JNDI
  - namespaces, ldap
  - database connectivity
  - JAAS authentication
  - JTA transactions
  - JMX management & monitoring
  - JavaMail
  - EJB business logic & data

- JMS
  - messaging
- JDBC
  - SOAP
  - web services
- JSP remote method invocation
- Servlets
- web logics
- XML
  - web presentation
  - portability

J2EE

- 'Programming in the large' and 'enterprise computing':
  - Lots of users
  - The applications is supposed to last forever
  - Many operating systems
  - Versioning
  - Large and heterogeneous development teams
  - Maintainability, flexibility, reusability, scalability
  - Load balancing
  - Huge volumes of data
  - DBMS independent
  - Security

JNDI: Java Naming & Directory Interface

- Provides namespace support for J2EE
  - Applications use JNDI to locate objects, such as environment entries, EJBs, datasources, message queues
  - Application attributes located by name within application local namespace
- Controlled access from external servers
- Access to LDAP directories

JDBC: Java Database Connectivity

- Java equivalent of ODBC (Open DataBase Connectivity)
- Java API that enables Java programs to execute SQL statements. This allows Java programs to interact with any SQL-compliant database
- Enables to write RDBMS independent code!!!
**JAAS: Java Authentication and Authorization Service**

- Supports user-based authorization
- Access control at every level
- Authentication via userid/password or digital certificates
- Role-based authorization limits access of users to resources (URLs, EJB methods)
- Embedded security realm

**JTA: Java Transaction API**

- Specifies standard interfaces between a transaction manager (e.g., JBoss) and the parties involved in a distributed transaction system:
  - the resource manager
  - the application server
  - and the transactional applications
- Provides "ACID" transaction support
- Two types of transaction paradigm
  - Declarative for entity & session beans
  - Procedural for session beans & servlets

**JMS: Java Message Service**

- Point to point messaging
  - each message has only one consumer
  - used to decouple applications
- Publish/subscribe messaging
  - each message may have multiple consumers

**EJB: Enterprise Java Beans**

- Bean
  - originally a Java class with get() and set() methods
  - e.g.: `getFirstName()`, `setFirstName()`
- EJBs come in 3 flavors:
  - ENTITY BEAN
  - SESSION BEAN
  - MESSAGE DRIVEN EJB

**EJB – Entity Bean**

- Represent actual data items (e.g., rows in a result set)
- Two forms of Entity Bean
  - Container managed persistence (CMP):
    - DB interactions handled by the J2EE environment (no programming!!!)
  - Bean managed persistence: requires that the bean carries out DB interactions itself
- May be called across network connection (RMI)

**EJB – Session Bean**

- Model a process or task
- Represent resources private to the client
- May update shared data
- Two forms of Session Bean
  - Stateful: state maintained between method calls
  - Stateless
- One client per session bean instance
**EJB – Message Driven Bean**
- Used in conjunction with Java Messaging System
- Activated on JMS message arrival
- No state maintained between activations

**JMX - Java Management Extensions**
- Consulting and changing application configuration
- Collecting statistics about application behavior and making the statistics available
- Notification of state changes and erroneous conditions

**Servlets**
- Used for web applications
- Extend functionality of an HTTP server
- Replace CGIs
- Servlet support defined by specification
- Filters are part of the spec
- Consist of get and post methods for requests

**JSP – Java Server Pages**
- A JSP combines Java code and template HTML in a single file.
- This is similar to the way PHP/ASP work.
- Scripting elements are used to provide dynamic pages
- Each Java server page is compiled into a servlet before it can be used

**An Example: MyStore**
- Table **Supplier** records details of those suppliers (many) who sell different materials to MyStore as request is sent to these suppliers from MyStore manager as need arises.
- Table **Manager** records details of managers who run MyStore, currently there is only one manager.
- Table **Customer** records details of those customers (many) who have bought some items at least once.
- Table **Items** maintains an inventory of available/non-available items (many).
- Table **StoreAccess** records the authentication details of all customers, suppliers and manager for on-line access of MyStore.
Conclusions

- J2EE is a very good alternative for medium and large systems
- Hard to learn, but very productive
- Very robust
- Lots of supporters
- Result of years of R&D (both academy and industry)

References

- J2EE:
  - http://java.sun.com/j2ee/
- Free J2EE Containers:
  - http://jonas.objectweb.org
  - http://www.jboss.org
- Development Tools:
  - http://jakarta.apache.org
  - http://www.eclipse.org