Introduction to component-based software development

Nick Duan
8/31/09

Overview

• What is a component?
• A brief history of component software
• What constitute the component technology?
  – Components/Containers/Platforms
  – Standardization
• The enterprise challenge
  – The characteristics of enterprise applications
  – The multi-tier architecture of J2EE
  – The separation of responsibilities
• Component software techniques covered in class
What is a component?

- Component in general
  - A self-contained entity
  - Well-defined interfaces
  - Pluggable into a base
  - Support composition (component hierarchy)
- Software Component
  - Unit of independent deployment
  - Unit of third-party composition
  - No external observable state

A brief history of component software

- Machine-level programming language
- High-level programming language
- Structured programming
- Object-oriented programming (module level)
  - Modeling software according to real-world entities
  - The 5 basic concepts of OO paradigm
- Component software (application level)
  - Declarative programming
  - Composition intensive
  - Standardization
The Definition of Software Components

• Szyperski: A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third-parties.
• The making of component software in distributed computing
  – Client/server, RPC, DOC, Application Servers

Component, Containers and Platforms, Architecture

• Component – Self-contained object
• Container – Environment where components live
• Platform – Language/HW/SW specific environment for language-specific components
  – JavaBeans, J2EE, .Net, COM, DCOM
• Architecture – Language independent environment specifications
  – CORBA
Issues dealt with in component software

• Interfaces
• Component-container contract
  – Controlling the life-cycle components
• Component state and interactions
  – Context of component
• Declarative Configuration
• Packaging and deployment
• Standardization

The enterprise challenge

• Component frameworks are usually developed for application-specific domains
  – GUI: AWT/Swing Components as Java Beans
  – Web: Java Servlet and JSPs
  – Enterprise: EJBs, DAOs, Connectors
• Customized Component Frameworks
Enterprise Application Domain

• Utilities and infrastructure support in a distributed environment
  – Naming
  – Persistence/Database Connectivity
  – Transaction Management
  – Security
  – Performance and Process Monitoring
• The real challenge: incorporating the plumbing functions into the component framework
  – Via standardization

Multi-tiered J2EE applications

• What is J2EE?
  – A platform for implementing enterprise applications in a distributed environment using Java
• Three-tier Architecture
  – Presentation tier (Web)
  – Business tier (EJB)
  – Integration (JDBC, JCA)
Types of J2EE applications

• Support of different computation models or architecture
  – Web applications (Servlet/JSP)
  – Distributed computing (Session and Entity EJBs)
  – Messaging (JMS, MDBs)
  – How about web services?
• JCP Standards in J2EE
  – Java Servlet, JSP, EJB, JMS, JDBC, JNDI, JCA, JTA,
    Java Security, etc.

Types of J2EE Components

• Web Components
• Business/EJB Components
  – Local
  – Remote
  – Web Service End Points
• Applications Client components
• What types of containers does J2EE provide?
Separation of Responsibilities

- Component Developers
  - Developing application components
- Application Assemblers
  - Configuring applications for specific environment
- Application Deployers
  - Deploy/Install applications onto specific environment
- Container Developers
  - Develop containers for standard components
- Platform/Server Developers
  - Develop server utilities and common framework functions
- System Administrators
  - Application administration and monitoring on production

What we will learn in this class

- Focus on J2EE, specially EJB, through the class project
- Investigate RMI for creating our own component frameworks from basics
- Understand the concept and methodologies for both use and create component frameworks
  - What problems to be addressed?
  - Why the issues are important to be standardized?
  - How to deal with real-world situations?