• Nearly all comments on project proposals have been emailed ...
J2EE Application Server Architecture

- Servlets
- JavaServer Pages (JSP)
- Java API for XML Processing (JAXP)
- JavaMail
- Java Authentication and Authorization Service (JAAS)

**Support for Communication and Presentation**

- Enterprise JavaBeans (EJB)
- Java transaction API (JTA)
- Java Message Service (JMS)
- Java Naming/Directory Interface (JNDI)

**Support for the Application Integration**

- Java DataBase Connectivity (JDBC)
- Java 2 Connector Architecture (J2CA)

**Support for Access to Resource Managers**
What is a bean?

- Java Object
- Container provides many standard services
  - location / binding
  - life cycle
  - persistence
  - transactions
  - etc
Classes of Beans

- **Session**
  - embodies business logic associated with a session / business txn
- **Entity**
  - represents a real-world object
- **Message-driven**
  - asynchronous processing using JMS queues
EJB Classes and Stubs

- Client
  - Client Stub
  - Client Interface
  - Client i/f Impl

- Container
  - Bean Impl

- Home Interface
  - Home Stub
  - Home Impl

- Business Methods/Remove
- Create/Find/Remove

- Notify

- Create/Find/Remove

Client | Server
Deployment

- XML deployment descriptors
  - identify programmer-provided code
  - dependencies
  - transactional behavior
  - security properties
  - etc
- Container & tools generate stubs
- Modify code / create wrappers based on properties specified in deployment descriptors
Session Beans

- embodies business logic associated with a session’s / business txn
- always given a session context
- not persistent
- stateful
  - invoked only thru creating session
- stateless
  - use by concurrent sessions allowed
Entity Beans

- represents a real-world entity
- may be shared by multiple sessions
- persistent beyond session or EJB container lifetime
- persistence may be *bean-managed* (JDBC/JTA) or *container-managed*
Bean-Managed Persistence

• Container notifies bean on activation/passivation
• Bean implementation responsible for serializing state to database
• Bean implementation responsible for transaction behavior (JTA)
  • distributed txns may be supported
Container-Managed Persistence

- Container manages serializing state to database and retrieving
- Container manages transactional behavior
  - declarative
Declared Transaction Attributes

- NotSupported
- Supports
  - use txn if present
- Required
  - create txn if none
- RequiresNew
  - create txn
- Mandatory
  - error if no txn
- Never
Managing a Transaction Context
Message-Driven Beans

- Communication by queueing (JMS) rather than RPC
- Container manages
  - creation of MDB when message arrives
  - invoke onMessage method with each arriving message
  - remove idle beans
- Equivalent to dequeue-process loop
Message Driven Beans

MDB A → Entity Bean C → Session Bean D → Entity Bean E → MDB B

Txn 1

Txn 2
Introducing Web Services

- [ACKM04] Ch 5, 6
Definition

• A standardized way of integrating Web-based applications, using
  • XML to tag data
  • SOAP to transport data
    • Simple Object Access Protocol
  • WSDL to describe available services
    • Web Services Description Language
  • UDDI to list available services
    • Universal Description, Discovery and Integration
Manual Integration Between Companies

customer

- internal procurement requests
- internal infrastructure

supplier

- web server
- internal infrastructure

warehouse

B2B interactions occur by accessing Web pages, filling Web forms, or via email.
In Principle - Middleware a Service

third party

- WfMS
  - WfMS adapter
  - message broker

a "global" workflow is executed here

the combination of message broker and adapters enables interoperability

customer
- customer's adapters
- internal procurement requests
- internal infrastructure

supplier
- supplier's adapters
- internal infrastructure

warehouse
- warehouse's adapters
- internal infrastructure

customer's adapters

third party

supplier's adapters

warehouse's adapters
Why not?

- Autonomy?
- Trust?
- Confidentiality?
How About Point-to-Point?

- Note middleware must interoperate
Middleware Explosion ...

customer

warehouse

another party (XYZ)

yet another party (ABC)

supplier

middleware for supplier-customer interaction

middleware for supplier-warehouse interaction

middleware for supplier-XYZ interaction

middleware for supplier-ABC interaction

supplier's adapters

supplier's adapters

supplier's adapters

internal infrastructure

middleware for integrating the middleware
Goals for B2B Integration

- Service-Oriented Architecture
- Redesign of middleware protocols
- Standardization
Service-Oriented

• Functionality always exposed as services
• Loosely coupled
• Invoked by programs as well as users
Middleware Redesign

- Peer-to-Peer
- Between trust domains
- Compatibility with Internet
Web Services Integrating Between Companies

- **Customer**
  - Internal procurement requests
  - Internal infrastructure

- **Supplier**
  - Internal infrastructure
  - Web service

- **Warehouse**
  - Internal infrastructure
  - Web service

- Interactions based on protocols redesigned for peer to peer and B2B settings
- Internal functionality made available as a service
- Languages and protocols standardized, eliminating need for many different middleware infrastructures (need only the Web services middleware)
A Sophisticated Wrapper - Expose Apps on Web

Company A (provider)

Middleware

Internal service

Internal service

Web service

Web service

Web service

Wide area network (Internet)

Client

Middleware

Internal service

Internal service

Company B (client)
Can Use WS Protocols Internally

Company A (or a LAN within Company A)

- Integrating application (contains the composition logic)
- Web service-enabled broker
- SmartQuotation
- DBMS applications
- SmartForecasting
- Sendmail application
- XYZ

Assumes all back-end systems are accessible as Web services.
Two Facets of Web Services Architecture

- Internal
  - run conventional apps
  - expose them as web services
- External
  - global services (like DNS)
Internal Architecture

Company A (service provider)

Web service interface
access to internal systems

Web services middleware (internal)

service interface
integration logic

conventional middleware (includes middleware services)

other tiers

Conventional middleware provides lots of services (load balancing, transaction support, etc). Current Web services middleware is quite poor in this respect.

clients from other companies
Company A (service requester)

1. publish the service description
2. find
the abstraction and infrastructure provided by the registry are part of the external middleware
3. interact

Service descriptions

Company B (service provider)

Web service

Web services middleware (internal)

other tiers

Company C (directory service provider)

Web services middleware (internal)

other tiers
• Is service discovery the only component of Web Services middleware?

• No but external middleware needs to run in peer-to-peer fashion with minimal trust requirements ...
External Architecture With Middleware

Company A (service requester)
- Web service client
- internal middleware
- other tiers

external middleware
- transaction mgmt
- other protocol infrastructure
- composition engine

Company B (service provider)
- Web service
- internal middleware
- other tiers

Company C (directory service provider)
- service descriptions

---

Other tiers:
- other tiers
- other tiers

Transaction mgmt:
- transaction mgmt
- transaction mgmt

Other protocol infrastructure:
- other protocol infrastructure
- other protocol infrastructure

Composition engine:
- composition engine
- composition engine

Service descriptions:
- service descriptions
- service descriptions
Minimalist requirements:
- communicate (SOAP)
- describe services - IDL (WSDL)
- directory service (UDDI)
Minimalist Infrastructure ...

![Diagram showing the interaction between service requestor and service provider.]

- **Service Requestor**
  - Application object (client)
  - SOAP-based middleware

- **Service Provider**
  - Application object (service provider)
  - SOAP-based middleware

SOAP messages exchanged on top of HTTP, SMTP, or other transport.

Converts procedure calls to/from XML messages sent through HTTP or other protocols.
Using WSDL Specification

<operation name="orderGoods">
  <input message = "OrderMsg"/>
</operation>

Note all WSDL “processing” happens at development time.
Using UDDI Registry

- **service requestor**
  - application object (client)
  - stub
  - SOAP-based middleware
  - SOAP messages (to look for services)

- **service provider**
  - application object (service provider)
  - skeleton
  - SOAP-based middleware
  - SOAP messages (to publish service description)

- UDDI registry
  - service descriptions
  - SOAP-based middleware
  - SOAP messages
Simple Object Access Protocol

• Specifies:
  • message format for 1-way comms
  • specification for SOAP RPC
  • rules for processing SOAP messages
  • rules for transport - HTTP and SMTP
A SOAP Message

- SOAP envelope
  - SOAP header
    - header block
  - SOAP body
    - body block
Document vs RPC

(a) Document-style interaction

SOAP envelope
SOAP body
PurchaseOrder document
- product item
- quantity

SOAP envelope
SOAP body
Acknowledgement document
- order id

(b) RPC-style interaction

SOAP envelope
SOAP body
method name
orderGoods
input parameter 1
product item
input parameter 2
quantity

SOAP envelope
SOAP body
method return
return value
order id
Intermediate Processing - Roles

- Header blocks only ...
- none: no node processes this block
- next: every node processes the block
- ultimateReceiver: only last node in path
Intermediate Processing - Roles

- Note “next” role in header
SOAP over HTTP
Simple Implementation

**service requestor**

- **client implementation**
  - invokes the service as a local call
- **client stub**
  - invoke SOAP engine to prepare SOAP message
- **SOAP engine**
  - packages SOAP into HTTP and passes it to an HTTP client that sends it to the provider
- **HTTP engine**

**service provider**

- **service implementation**
  - invokes the local procedure of the service implementation
- **server stub**
  - the router parses the message, identifies the appropriate stub, and delivers the parsed message
- **SOAP router**
  - passes the content of the HTTP message to the router
- **HTTP server**