

Web Services Support in Middleware Platforms (J2EE)

Workflows and Web Services Kapitel 5

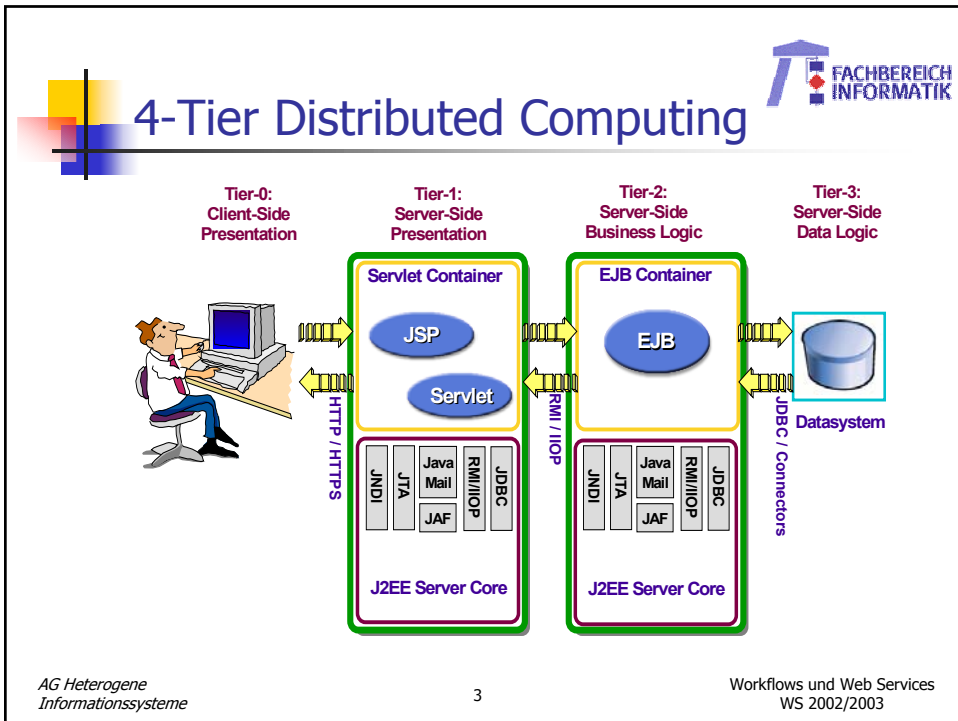
Workflows und Web Services
WS 2002/2003


1

What is J2EE?



- Platform that enables solutions for developing, deploying and managing multi-tier server-centric applications.
- Defines a **standard** architecture that is delivered in the following elements
 - J2EE Application Programming Model - A standard programming model for developing multi-tier, thin-client applications
 - J2EE Platform - A standard platform for hosting J2EE applications, specified as a set of required APIs and policies
 - J2EE Compatibility Test Suite - A suite of compatibility tests for verifying that a J2EE platform product is compatible with the J2EE platform standard
 - J2EE Reference Implementation - A reference implementation for demonstrating the capabilities of J2EE and for providing an operational definition of the J2EE platform
- There are numerous vendor implementations of J2EE



- # J2EE – Overview
- 
- Component Technologies
 - Web components
 - Java Server Pages (JSP), Servlets
 - Enterprise JavaBeans
 - Service Technologies
 - Java Transaction API (JTA)
 - Java DataBase Connectivity (JDBC)
 - Java Naming and Directory Service (JNDI)
 - Java Message Service (JMS)
 - JavaMail
 - Java Connector Architecture
 - Java Authentication and Authorization Service (JAAS)
 - Communication Technologies
 - Internet protocols
 - HTTP, TCP/IP, SSL
 - Remote Object protocols
 - Java RMI, RMI/IIOP, Java IDL
- AG Heterogene Informationssysteme
- 4
- Workflows und Web Services
WS 2002/2003

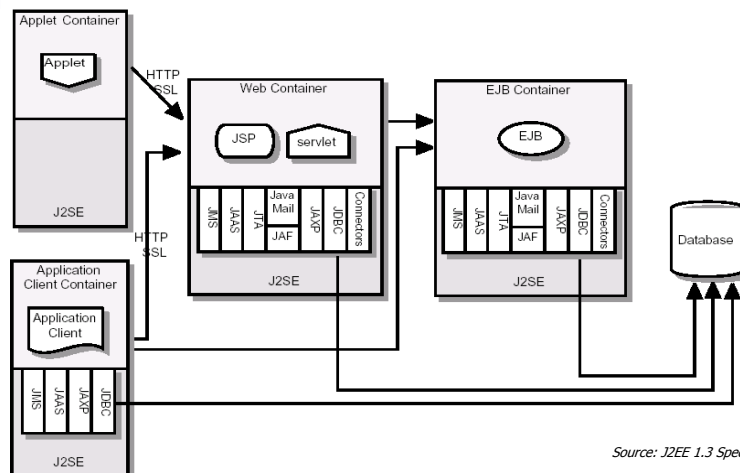


J2EE – Overview (2)

- Containers
 - Types: web container, EJB container, client application container, applet container
 - Controls component life cycle
 - E.g., creates new instances of EJBs
 - Provides run-time services
 - Uniform access of technologies and APIs used by components
 - Routes (client) requests to (server) components
 - Declarative services
 - Service configuration during deployment
 - E.g., transactional behavior
- Deployment
 - Process of preparing a component for execution in a J2EE runtime environment
 - Involves declarative specification of technical aspects
 - Transactions, security, naming, ...
 - Requires deployment descriptor
 - XML file



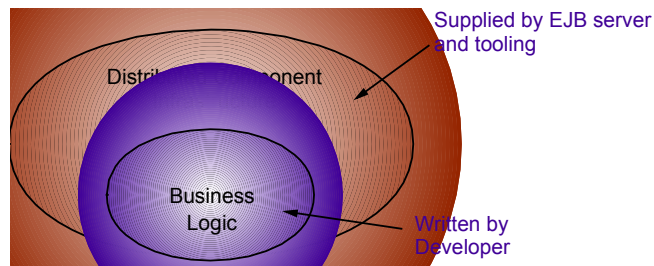
J2EE Architecture



Source: J2EE 1.3 Specification

Enterprise JavaBeans

- Portable, server-side components written in Java
 - Transactional, Secure, etc.
- Widespread industry support
- Allow focus on business logic, not infrastructure



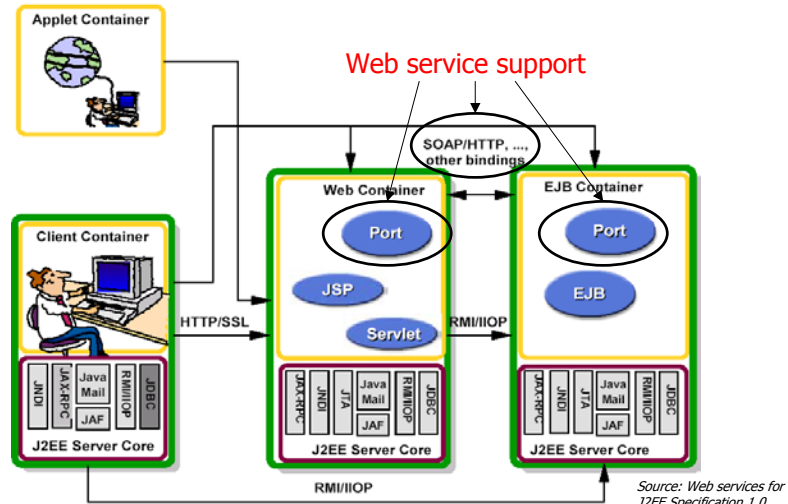
Types of EJBs

- **Session** beans contain information that will disappear after the user has closed the connection to the server (Internet Session ends).
 - **Stateful** session beans exist for the duration of a single client/server session.
 - **Stateless** session beans are pooled by the container to handle multiple requests from multiple clients.
 - Access databases using standard JDBC/SQLJ
 - obtain DB connection using JDBC data source (connection pooling)
- **Entity** beans contain persistent data that can be saved across sessions in various datastores.
 - **Bean-Managed persistence (BMP)**: Entity beans that manage their own persistence (e.g., using JDBC, SQLJ)
 - **Container-Managed persistence (CMP)**: Entity beans that delegate their persistence to the EJB container
- **Message-driven** Bean eases integration with existing applications
 - Asynchronous, message-oriented (JMS)

Web Services for J2EE Specification (WS4J2EE)

- Sun specification (JSR109), included in J2EE 1.4
- Defines "a service architecture that leverages the J2EE component architecture to provide a client and server programming model which is
 - portable and interoperable across application servers,
 - provides a scalable secure environment, and yet
 - is familiar to J2EE developers"
- Objectives (among others)
 - Simple model for defining and implementing a new Web service and deploying this into a J2EE application server
 - Build on evolving industry standards (WSDL 1.1, SOAP 1.1, ...)
 - Leverage existing J2EE technology
 - Inter-operability of vendor implementations
 - Minimize new concepts, interfaces, file formats, etc.

J2EE Architecture





JAX-RPC

- WS4J2EE requires JAX-RPC support
- Java API for XML RPCs (JAX-RPC)
 - APIs for supporting XML based RPC for the Java platform
 - Define web service
 - Use web service
 - Defines
 - WSDL/XML to Java mapping
 - Java to XML/WSDL mapping
 - Core APIs
 - SOAP support (including attachments)
 - Client and Server Programming models involving generated stub classes
- Client side invocation (standard programming model)
 - Application invokes web service through generated stub class
 - JAX-RPC runtime maps the invocation to SOAP, builds the SOAP message, processes the HTTP request
- Server side processing
 - JAX-RPC runtime processes HTTP, SOAP message, maps to RPC and dispatches to target (class implementing the web service)



JAX-RPC (2)

- Mapping WSDL to Java – Example
 - WSDL port type definition

```
<!-- WSDL Extract -->
<message name="getLastTradePrice">
  <part name="tickerSymbol" type="xsd:string"/>
</message>
<message name="getLastTradePriceResponse">
  <part name="result" type="xsd:float"/>
</message>
<portType name="StockQuoteProvider">
  <operation name="getLastTradePrice"
    parameterOrder="tickerSymbol">
    <input message="tns:getLastTradePrice"/>
    <output message="tns:getLastTradePriceResponse"/>
  </operation>
</portType>
```
 - Corresponding Java service endpoint interface:

```
//Java
public interface StockQuoteProvider extends java.rmi.Remote {
  float getLastTradePrice(String tickerSymbol)
    throws java.rmi.RemoteException;
}
```



Client Programming Model

- Client can be
 - J2EE application client
 - Web component
 - EJB component
 - Another web service
- Client view of web service
 - Set of methods that perform business logic
 - Service endpoint interface
 - Stateless, i.e., there is not state information that persists across method invocations
- Uses the WS4J2EE runtime to access and invoke the methods of a web service
 - JNDI lookup to access a Service object
 - Factory to obtain a stub/proxy that implements the service endpoint interface
 - Invoke web service method on the stub object implementing the service endpoint interface



Client Programming Model (2)

- Client developer works only with the Service and Service endpoint interfaces, which may have been
 - supplied by the web service provider, or
 - generated using tools based on WSDL provided by WS provider
- Client developer does NOT generate stub/proxy class during development
 - Will be generated during deployment of the client component
 - Can be specific to the vendor runtime used on the client
- Developer can also use dynamic invocation interface (DII) of Service
 - Generic methods for invoking the web service methods
 - Useful if WS details are not known at development time
- Web services client deployment descriptor contains additional information about web service
 - Service reference name used for JNDI lookup
 - WSDL, port information, ...



Server Programming Model

- Two methods for implementing a web service
 - Java class running in a web container
 - Actually defined in the JAX-RPC specification
 - Stateless session EJB running in an EJB container
- Port Component
 - Defines server view of web service
 - Services
 - a location defined by WSDL port address
 - a set of operation requests defined by a WSDL PortType
 - Has a
 - Service Endpoint interface
 - Java mapping of the WSDL port type and binding
 - Service Implementation Bean
 - In general a Java class implementing the methods of the service endpoint interface
 - Differences based on the type of container (web or EJB)



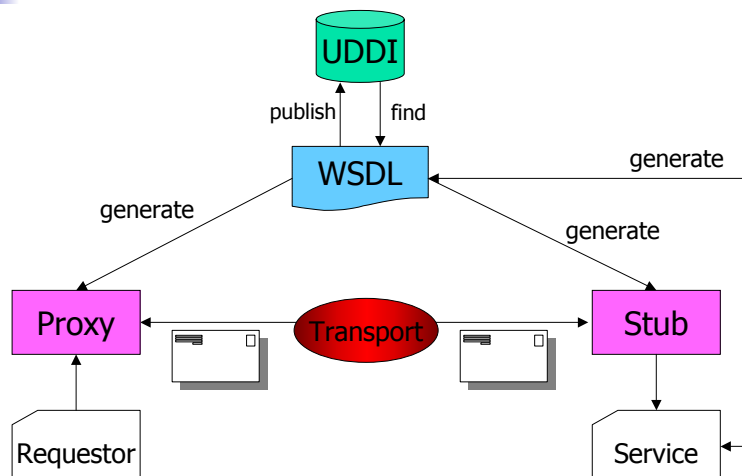
Implementation Methods

- Stateless session bean used to implement a web service
 - EJB container takes care of multi-threaded access to web service
 - Requirements more or less as defined for stateless EJB by EJB specification
- Existing stateless EJB can be exposed as a web service
 - Service endpoint interface methods can be a subset of the EJB remote interface methods
 - Transaction attribute MANDATORY is not permitted
 - Existing transaction context will be suspended by container during execution of a web service
- Web container component
 - Implementation can be
 - single-threaded
 - Class implements `javax.servlet.SingleThreadModel`
 - Container responsible for synchronizing access
 - multi-threaded
 - Implementation class must be stateless

Container Responsibilities

- **Listening** on a well known port or on the URI of the Web service implementation (as defined in the service's WSDL after deployment) for SOAP/HTTP bindings.
- **Parsing** the inbound message according to the Service binding.
- **Mapping** the message to the implementation class and method according to the Service deployment data.
- **Creating the appropriate Java objects** from the SOAP envelope according to the JAX-RPC specification.
- **Invoking** the Service Implementation Bean handlers and instance method with the appropriate Java parameters.
- **Capturing the response** to the invocation if the style is request-response
- **Mapping** the Java response objects into SOAP message according to the JAX-RPC specification.
- **Creating the message envelope** appropriate for the transport
- **Sending** the message to the originating Web service client.

Tooling Principles





Additional Concepts

- **Handlers**
 - A means for intercepting and processing the raw SOAP request
 - Can examine and probably modify a request before it is processed by a web service component
 - Can also process/modify the response
 - May run on server as well as client side
 - Usage scenarios
 - Message logging
 - SOAP header processing/generation
 - Processing parts of the SOAP body
- **Security**
 - Authentication: BASIC-AUTH, symmetric HTTPS
 - Authorization: J2EE container support
 - Integrity and confidentiality: HTTPS
 - Non-repudiation: recommended, but not defined
 - ... a lot is left for future work



Relationships to other Java specs for XML

- Only JAX-RPC (JSR 00101) support is required!
- JAX-M (JSR 00067): XML messaging and the Java language.
- JAX-R (JSR 00093): Java interfaces to XML registries, like JNDI, ebXML and UDDI.
- JAX-P (JSR 00005 and 00063): APIs for parsing XML
- XML Trust (JSR00104): APIs and protocol for a "Trust Service" to minimize the complexity required for using XML Signatures.
- XML Digital Signature (JSR 00105): APIs for XML digital signature services.
- XML Digital Encryption (JSR 00106): APIs for encrypting XML fragments.
- Java APIs for WSDL (JSR00110): APIs for manipulating WSDL documents.