Modernizing the Mainframe with Java

Gregg Lippa
Senior Technical Analyst
Themis Education
Themis, Inc.

glippa@themisinc.com
Visit us at: www.themisinc.com
Also: www.themisinc.com/webinars

Objectives

Upon successful completion of this webinar, you will be able to:

– Describe the advantages of using Java on the mainframe

– Explain where can Java be used in the mainframe environment

– Discuss how Java is supported in DB2, CICS, and in batch jobs

– Describe how to access z/OS-based files from Java programs

– Discuss ways in which Java code can interact with COBOL modules

– Explain how the WebSphere server plays into mainframe modernization

– Describe the role played by message queues and XML in this effort

– Explain how Java can be used as a Web Service provider and consumer

Themis

Legacy System Lifecycle

- Application lifecycle
  - Starts with initial development and implementation
  - Followed by **maintenance, modernization and replacement**
  - The modernization and replacement phases are particularly relevant for **Java applications on z/OS**
- Eventually, ongoing maintenance is no longer acceptable
  - Obsolete user interfaces
  - Incompatible operating environments

Themis

Screen Scraping

- One **simple approach:** implement a **wrapping layer**
  - Provide newer interfaces
  - Technique known as screen scraping
- Advantages of **screen scraping**
  - Tools exists to facilitate transforming existing applications
  - Protects the investments in existing applications
  - Less risky than replacement with a newer system
- However…
  - **Underlying applications are not modernized**
  - Makes these solutions **difficult to maintain**

Themis

Legacy System Replacement

- **Replacement –** last phase of legacy system lifecycle
  - Legacy system is completely replaced with a new system
- Involves **greater risk**
  - No guarantee that new system will provide better service to users in spite of good intentions and design of the new application
- Legacy systems must not be dismissed lightly
  - Have been supporting the business successfully for many years
- **New systems** may need to access resources based on z/OS
  - Such systems may also benefit from Java batch components
  - Opportunities to modernize application portfolio exist here as well

Themis

Considerations for Replacement

Factors to consider when replacing a legacy application:

1. **Availability of skills**, especially looking toward the future
   - COBOL developers are a dwindling resource
   - Many Java developers coming out of our universities
   - More tools to increase developer productivity in the Java realm
2. **Integration with other systems** over the network
   - Proven integration frameworks
3. **Performance** as measured by application throughput
   - Java has been improving in this regard (e.g.: JIT compiler)
4. **Portability** allows execution in other operating environments
   - Applications may be moved off of z/OS (e.g. to Unix or Linux)
   - Java: *write once, run anywhere*

Themis

Why Use Java?

- **Object oriented** programming (OOP)
  - Many benefits
- **Portability**: compiled code can run on any OS
  - Java compiles to **bytecode**, not native machine language
  - A Java Virtual Machine (JVM) interprets the bytecode
- **Complete** language
  - Supports all programming constructs
    - Including support for multi-threading
- Full support for dynamic **website development**
  - Using Java Enterprise Edition (Java EE aka J2EE)
- **Widespread support** and many available add ons

Themis

Other Benefits of Using Java

1. **Functionality**
   - E.g. ability to send an e-mail or update a remote database
   - Availability of numerous existing reusable objects
2. **XML processing**
   - Java has mature frameworks for processing XML data
   - Also full support for Web Service clients and providers
3. **Reusing** Java EE online **program logic**
   - Batch jobs on z/OS often manage complex and heavy workloads
4. **Migrating Java applications** for QoS / server consolidation
   - z/OS is the platform of choice in many situations today
5. **zIIP processor** can run most instructions in Java applications
   - COBOL programs can only use a General Purpose processor
   - Can result in significant cost savings

Themis

Batch Java

Batch Applications

- Features of batch applications
  - **No terminal or browser** waiting for job output
  - Typically process a **large workload**
    - Repeat a task thousands or even millions of times
  - Can result in more efficient processing
    - Throughput can be increased via batch execution
  - Utilize **workload management** techniques

Java Batch Application Scenarios

- Some scenarios that lend themselves to considering Java:
  1. Processing **large volumes of data** in batch
     - Flat files, VSAM data sets or DB2 data
  2. Communication with other environments via **messaging**
     - Allows an application to integrate with other environments
  3. Generating and printing **reports**
     - Large amounts of print work to be completed in batch
  4. Independent Software Vendor (ISV) applications
     - Software solution providers may deliver Java batch applications to support the industry-specific needs of your organization

Java Batch Capabilities

What you can do in a Java on z/OS batch environment:

- **Call DB2 stored procedures**
  - Using Java Database Connectivity (JDBC)
- **Read from and write to** various types of files including **MVS data sets,** VSAM data sets and UNIX files
- **Call programs in CICS or IMS**
  - Using Java Connector Architecture (JCA)
- **Access the network** using protocols such as TCP/IP and HTTP
- Perform CPU-intensive calculations
- **Integrate with other programming languages** using
  - Java Native Interface (JNI)
  - Language Environment (LE)

APIs for Java Batch Processing on z/OS

- IBM uses a common code base for the JVM on all its platforms
- Java APIs provided by a z/OS extension support:
  1. **MVS data set and VSAM access** to interact with z/OS specific data
  2. Condition code passing for integration of Java batch jobs into z/OS job nets
  3. z/OS Catalog search
  4. Interaction with the MVS Console
  5. **Conversion of COBOL/ASM data types to Java types**
  6. **Invoking DFSORT** to effectively sort data
  7. Access to z/OS Access Method Services (IDCAMS)
  8. **RACF APIs** to integrate Java into the z/OS security model
  9. **Writing of Log streams** (e.g. as an Appender to log4j)
  10. Submission of Jobs from Java

Data Access with Java on z/OS

- Data access is a central component of batch processing
  - Data on z/OS typically resides in DB2 on z/OS
    - JDBC supports **access to relational data** (more to come)
  - Still, much z/OS data is often stored in sequential files, partitioned data sets and VSAM clusters
- Java APIs exist to access these z/OS specific data stores
  - Data access support is provided by the **JZOS toolkit library**
- JZOS supports Java access to mainframe file systems
  - Ordinary Java APIs do not support access to these types file

JZOS Toolkit Library

- **JZOS toolkit** library is available for Java applications
  - Supports **access to mainframe file systems**
    - Not available via the standard Java I/O library
  - JZOS provides thin wrappers for z/OS C/C++ Library functions to access MVS data sets
- JZOS allows Java access to any MVS data sets, including:
  - **Sequential files**
  - Partitioned Data Set (**PDS**) and Partitioned Data Set Extended (PDSE)
  - Virtual Sequential Access Method (**VSAM**) including KSDS, RRDS, and ESDS

Themis

Creating Java Record Classes

- z/OS datasets often contain records described by **COBOL copybooks**
  - Contain fields that conform to the set of valid COBOL types
    - Alpha, alpha-numeric, unscaled numeric, packed decimal, binary, etc.
  - A Java program may need to process this kind of dataset
    - **May be difficult to convert from COBOL to Java data types**
- **JZOS offers a complete set of field converter types**
  - Operate on a byte array
  - Convert from COBOL types to Java types

Themis

Generating Java Record Classes

- **Build Java record class programatically from a copybook**
  - **Automated process** uses input from the COBOL copybook
- JZOS includes development time tools for doing this
  - It is a **development time tool** and is not required at runtime
  - Uses the ADATA output of z/OS Enterprise COBOL compiler
  - **Generates a Java class to map a selected copybook record**
- Client-side script is used to create Java record classes
  - Submits a job to generate an **ADATA file** for a COBOL copybook
    - Machine readable file **describing structure** and content of compilation
  - Downloads the file and runs **RecordClassGenerator**
    - Which uses ADATA file to **generate the associated Java class**
  - Generated Java record classes can be used like any other Java class

Themis

Java Interoperability with IMS

Themis

Java Interoperability with COBOL & PL/I

- Two ways to achieve interoperability between Java and other languages:

  1. **Using functionalities of the runtime environment:**
     - CICS, IMS, DB2, WebSphere Application Server (WAS), or JES
     - Depends heavily on the selected run time
  2. **Creating direct language calls**
     - Not bound to a runtime environment
     - Java methods can be invoked from object-oriented (OO) COBOL
       - Supported by using Java Native Interface (JNI) technology
     - COBOL programs can also be invoked from Java code

Java in IMS

- IMS provides a **batch processing** option
  - Exploits the reliability provided by IMS through specific methods for **checkpoint, restarting, and rollback**
- IMS Java batch applications support access to data residing in
  - **IMS** databases
  - **DB2** databases
  - **VSAM** files
  - GSAM files
- Java Message Processing (JMP) and Java Batch Processing (JBP) applications can access IMS databases
  - Uses the **IMS hierarchical database interface for Java**
    - Requires the IMS Base API for Java to convert the Java calls to DL/I calls

Java Batch Processing in IMS

- Batch processing with Java in an IMS environment is supported
  - Programs in JBP regions can access **IMS message queues for output**
  - They can be started using JCL (or using TSO)
- The **JBP container provides transactional services** as well as supporting checkpoint / restart processing
  - Data changes are bundled into transactions to ensure data integrity
    - Changes are either executed completely or not at all
- Batch processing transactions are established by IMS checkpoints
  - At a checkpoint, IMS commits all data changes since the last checkpoint
  - A batch process can be restarted if does not successfully complete its work
- The IMSTransaction class supports calls to **checkpoint(), restart() and rollback()**

Themis

Java and DB2

Java in DB2 for z/OS

- DB2 for z/OS can also be used as a Java batch run time
  - By using **DB2 Java stored procedures**
- Allows Java programs to benefit from DB2 these functionalities:
  - **Transactional processing**
    - Allows a Java program to use the transaction context of the calling program
      - Including **two-phase commit support**
  - Java Virtual Machine (JVM) instantiated only once
    - JVM for a DB2 Java stored procedure is started only when it is first called
    - **DB2 reuses the started JVM for subsequent calls to a stored procedure**
  - Reuse of DB2 thread
    - Stored procedures use the same DB2 thread for processing SQL statements as the process that called the stored procedure
  - Scalability
    - Several stored procedures can run at the same time within one address space
    - Integrated Workload Manager can start additional address spaces if required

Themis

Static SQL vs Dynamic SQL

- DBAs prefer static SQL; developers may prefer Dynamic SQL
  - Static SQL
    - **Compiled and bound to the database** during application development
  - Dynamic SQL
    - **Compiled at runtime**, requiring SQL to be optimized when run
    - Compile at run time **increases total statement execution time**
    - Errors in the SQL statement won't be detected until runtime
- Benefits of **Static SQL**
  - Compile time bind runs optimizer at development time
  - **Security can be managed better**
- Disadvantages of Static SQL
  - Need to bind before runtime
  - **JDBC does not provide support**
    - Except through its ability to call Stored Procedures

Static SQL vs Dynamic SQL

- Benefits of **Dynamic SQL**
  - IDEs and APIs such as Eclipse and JDBC support it
  - **Better statistics** – because the statement is compiled at runtime, it uses the latest statistics available
    - **May result in a better execution plan**
- Disadvantages of Dynamic SQL
  - **Compile at runtime** can be a problem for several reasons:
    - Every time a statement is executed, it needs to be compiled
      - **Increases the total statement execution time**
      - **May slow overall performance** instead of improving it
      - Errors in the SQL statement will not be detected until runtime

Themis

Stored Procedures

- **Pre-coded SQL** usually contained within programming logic
- **Highly recommended** and widely used in many organizations
  - Less vulnerable to attacks
  - Can be written with static SQL
- **JDBC can be used** to call Stored Procedures (SPs)
  - Writing the actual Stored Procedures may also be done with Java
- Each DBMS provides its own language support for writing SPs
  - DB2 SPs can be written in commercial languages, such as COBOL
    - SQL/PL is preferred for DB2 stored procedures today

JDBC

- Framework composed mainly of interfaces
  - Defined by Oracle
  - **Implemented by database drivers**
  - Same functionality regardless of driver
  - Switching databases requires minimal code changes
- Java Application Programming Interface (API)
  - Provides **access to any type of tabular data**
  - Normally used to access **Relational Databases**
  - Vendor independent
- Allows Java applications to easily
  - Connect to data source
  - Create and execute queries and updates
  - Retrieve and process query results

Themis

JDBC Basics

- Java Database Connectivity (JDBC) is a standard for accessing relational databases using vendor independent java commands
  - It **utilizes SQL** and **supports calling stored procedures**
- Vendors create JDBC drivers for their database products
  - Insulate Java developers from the idiosyncrasies of their specific implementation of relational database specification
- **Basic Steps of JDBC Processing**
  1. Load a driver or access a Connection Pool
  2. Open a Connection to the database
  3. Create a Statement (QUERY, UPDATE, INSERT, DELETE)
  4. Execute the Statement (may be a stored procedure call)
  5. Process your ResultSet (for SELECT statements)
  6. Close your Connection

Themis

WebSphere XD Compute Grid

Themis

Java Batch Applications

- Today's z/OS batch workloads often use COBOL
  - Contain complex business logic
  - Process large amounts of input and output data
- Java batch applications can run on z/OS
  - However, a JVM is created for every batch job
    - **More CPU usage**
    - **Additional resource consumption**
    - **Greater overall cost**
  - Managing JCL via a workload scheduler can help to mitigate these costs

Themis

WebSphere Can Help

- WebSphere Application Server for z/OS
  - Provides many container services
  - Allows Java execution to utilize the **On Line Batch Processing paradigm**
- WebSphere Extended Deployment (XD) **Compute Grid**
  - Provides a **batch container** for Java EE applications
  - Extends the existing WebSphere core functions
  - Uses the same administrative console as WebSphere Application Server for z/OS

WebSphere XD Compute Grid

Advantages of Java EE platform:

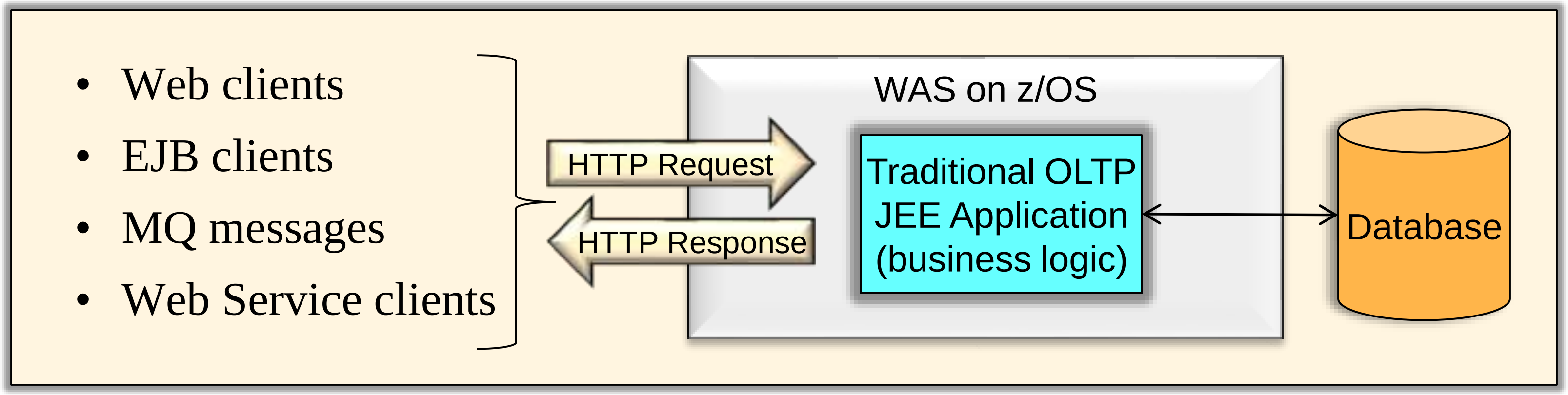
- Supports **concurrent Java batch and OLTP** workloads effectively
- **Optimizes performance** of business-critical applications
  - **Maximizes scalability** for high volume transaction processing
- Centralizes WebSphere and non-WebSphere server management
- Simplifies and **improves management** of complex system operations
  - Real-time visualization tools
  - Application **versioning support**
  - Autonomic capabilities such as **health management**
  - Reduced cost and complexity of managing WebSphere resources
- Allows greater focus on WebSphere to configure resources
  - Helps increase enterprise’s **adaptability to business change**
- **Improves customer service levels**
- Takes advantage of existing Java skills and resources

Java Enterprise Edition (Java EE)

- **WebSphere** has become **increasingly important**
  - Popular environment for modern, enterprise-class business applications
    - Especially OLTP applications
- **Java EE skill base is rapidly growing** in the industry
  - The Java language and Java EE servers offer many advantages
    - **Now provide a good option for batch** workloads
- Today, Java can reduce application development cost over older languages
  - Because it is object oriented, **Java can also reduce development time**
- Many powerful Java applications and frameworks are available
  - Further reduces application development cost and time requirements
- Additionally, **JVM performance is continually improving**
- This makes Java attractive for developing **new business applications**
  - And for **re-engineering existing applications**

The Java EE Environment on z/OS

- WebSphere Application Server **(WAS) for z/OS**
  - Supports enterprise-class business applications written in Java
  - **Typical OLTP workload in WAS** includes various types of clients:



  - WAS-based applications are **often transaction-oriented**
    - Business logic written in Java is invoked to create a response for the client
- WAS is able to manage all of the transactional processing

Container Services

- WebSphere provides many system services (**container services**)
  - **Security** management
  - **Transaction** support
  - **Web service** support
  - Session and connection **pooling**
  - **Caching**
  - Messaging
- The container services provide these **benefits**
  - Simplifies Java application design
  - Accelerates development time
  - Simplifies application management
  - Increases application security, availability, performance, and stability
- **However…**
  - WAS-based business logic is **not accessible from traditional batch**
  - Integrating traditional runtime environments and WAS is based on connectors and messaging

Themis

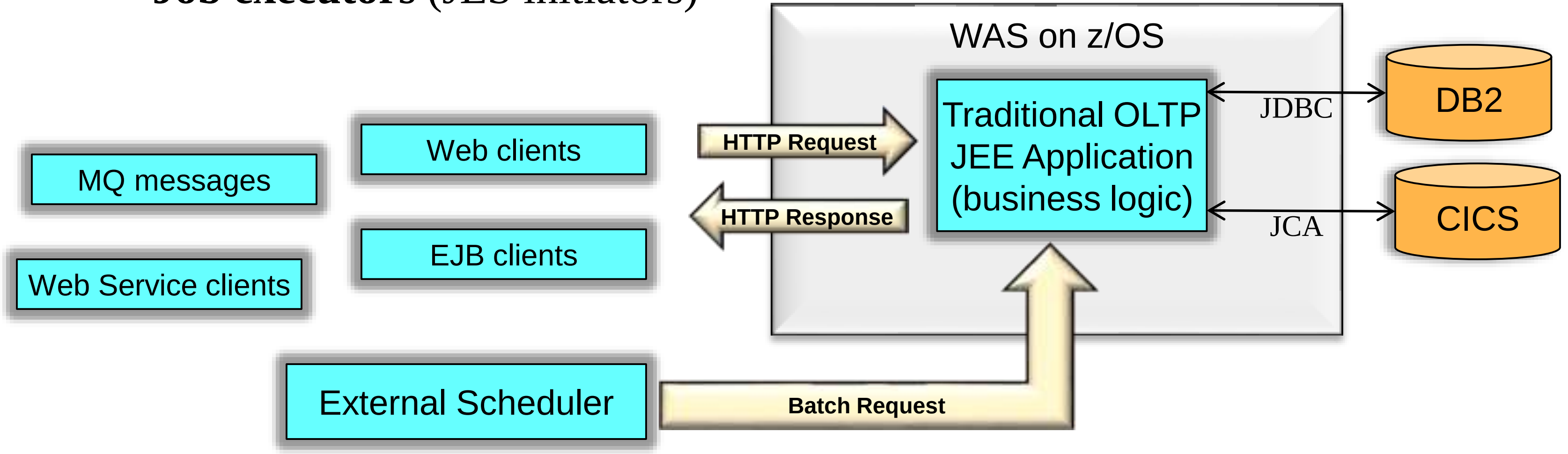
Batch Workloads for Java EE

- WebSphere XD Compute Grid supports batch workloads
  - Able to **run concurrently** – like traditional batch on z/OS
  - Expands on existing z/OS support for WebSphere
  - Able to **run both transactional and batch Java** workloads
  - Provides a batch container
    - **Makes existing WAS-based business logic accessible to traditional batch**
      - Can be seamlessly integrated into the traditional batch execution environment
      - **Can be shared across batch and OLTP applications**

Themis

WebSphere XD Compute Grid Overview

- WebSphere XD Compute Grid for z/OS
  - Builds on existing Java EE programming model and container services
  - **Provides a job scheduler and an execution environment**
    - Includes additional features designed for long-running batch applications
  - Provides an environment for deploying Java **batch applications**
  - Modeled after the z/OS Job Entry Subsystem (JES) using
    - Job Control Language (**JCL**)
    - A **job dispatcher** and
    - **Job executors** (JES initiators)



Compute Grid Components

- WebSphere XD Compute Grid components are Java EE applications:
  - **xJCL –** XML based job control language
  - **Job Scheduler** (JS) – a job dispatcher – job entry point to Compute Grid
  - **Grid Execution Endpoints** (GEE) – multi-threaded job executors
  - **Parallel Job Manager** (PJM) – governs execution of parallel batch jobs
- Components can be placed in one JVM or in a **clustered environment**
  - Support WAS security, scalability, availability, and life cycle management
- Other components used in a Compute Grid environment include:
  - Job Management Console (JMC)
  - Scheduler tables – job information stored in a database
  - Container tables – checkpoint information stored in a database

Compute Grid Component Detail

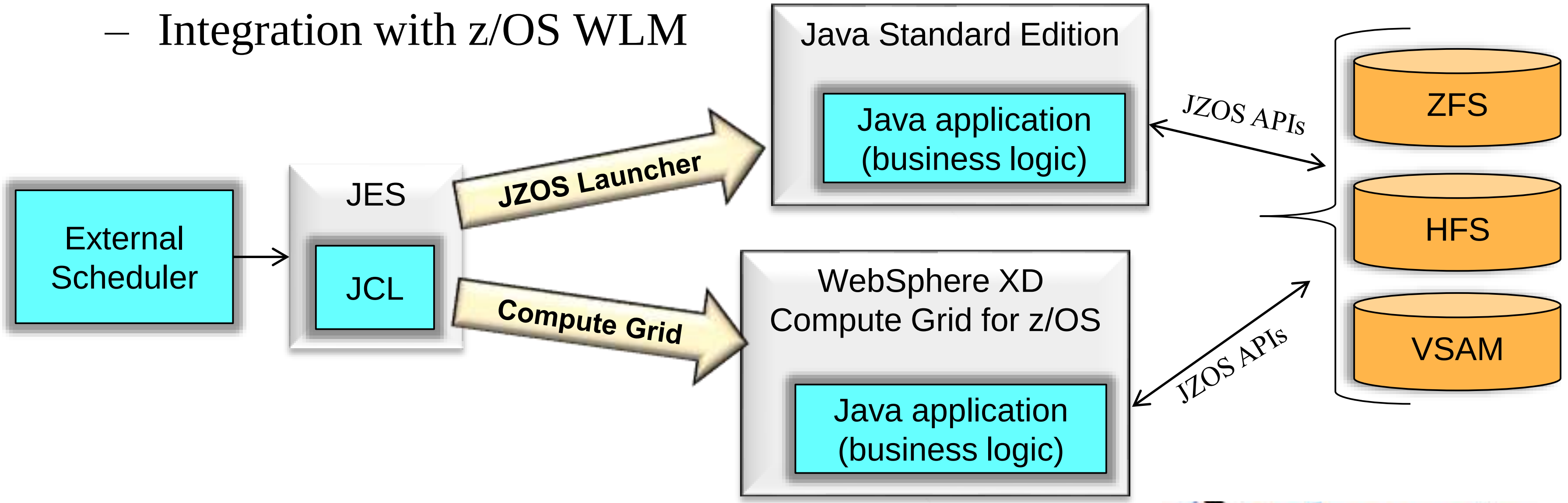
- Components used in a WebSphere XD Compute Grid environment:
  - **Job scheduler (JS)** – job entry point to XD Compute Grid
    - Provides job **life-cycle management** functions (submit, cancel, restart) and monitoring
    - Maintains a **history of all jobs** (waiting, running, or completed) and job usage data
    - Dispatches workload to either the PJM or GEE and also hosts the JMC
  - **Batch container** – Grid Execution Endpoint (GEE) component
    - Executes the actual business logic of the batch jobs
    - A WebSphere cell can have any number of batch containers
  - **Java EE batch application** – WebSphere applications deployed as an EAR file
    - Contains implementations of one or more Java batch applications
  - **JDBC** – provides connectivity to the scheduler and container tables
  - **Parallel Job Manager (PJM)**
    - Breaks large batch jobs into partitions for parallel execution
    - Not a required component in a WebSphere XD Compute Grid environment

JZOS Launcher

- JZOS delivers both the **JZOS Launcher and Java z/OS APIs**
  - The Launcher allows a stand-alone Java program to be invoked using JCL
  - It also plays a role in a WebSphere XD Compute Grid environment
    - Its APIs provide an **integration point for Java and traditional z/OS**
- The JZOS launcher has several **drawbacks making it inefficient** for the high volume of batch jobs that run within batch window
  - **It lacks these capabilities**:
    - Security, transaction, or connection management facilities
    - Checkpoint / restart facility for batch jobs
    - High availability and other quality of service (QoS) provided by WAS
    - A persistent, reusable JVM – each job step requiring Java reloads the JVM

Compute Grid vs. JZOS

- WebSphere XD Compute Grid offers all necessary Quality of Service and other services including:
  - Persistent, reusable JVM and execution container
  - Security, transaction, and connection management
  - Thread pooling
  - High availability
  - Prioritization
  - Integration with z/OS WLM


External Scheduler
JES
JCL
JZOS Launcher
Compute Grid
Java Standard Edition
Java application (business logic)
WebSphere XD Compute Grid for z/OS
Java application (business logic)
JZOS APIs
JZOS APIs
ZFS
HFS
VSAM

Programming with Compute Grid

- WebSphere XD **Compute Grid batch container**
  - Contains and **controls the Java batch application**
  - Uses a container-managed thread defining the unit of execution
  - Carries out the job's life cycle as it runs in the batch container
- A batch application is made up of these components:
  - Batch **job step**
    - Java class providing the **business logic to execute** as a step in the job
    - Invoked by the batch container during job processing
  - Batch **data stream**
    - Provides the job step with **access to various types of data** including
      - Relational databases
      - File systems
      - Message queues
      - J2C connectors to other types of data

Batch Container Responsibilities

- During life cycle of a job step, batch container is responsible for open, close, and checkpoint-related callbacks on batch data streams
  - Job step calls methods on batch data stream to get and put data
- Batch applications may include either or both of these components:
  - **Checkpoint** algorithm
    - Container provides mechanism to **support job restart**
    - Compute Grid provides **two pre-built checkpoint algorithms**
      - One supporting a **time-based** checkpoint interval
      - One supporting a checkpoint interval **based on record-count**
  - **Results** algorithm
    - A **return code is supplied** upon completion of each job step
    - The **overall return code** for the job as a whole is returned
      - A provided results algorithm returns the highest step return code as the overall job return code

Themis

JCICS – Using Java with CICS

Using Java with CICS

- **Java has been available** as a programming language in CICS since CICS Transaction Server (TS) V1.3
  - Support has steadily improved over a number of releases
  - CICS TS V4.2 is of particular significance for modern workloads
- **CICS TS V4.2 fully utilizes the JVM server model**
  - Embraces modern dynamic application development technologies through the **Open Services Gateway initiative (OSGi)** technology
  - Introduces a rich client development platform via the **CICS Explorer**
  - Provides capacity and workload benefits via **64-bit storage for JVM**
    - Delivers significant **gains in performance and scalability** over previous versions of CICS Java support
  - Further benefits are realized by embracing the **CICS bundle** resource for packaging, deployment, and controlling the application lifecycle
    - Further improves manageability and deployment of CICS Java applications

JVM Server Replaces Pooled JVMs

- More powerful **JVM server model** introduced in CICS TS V4.1
  - Previous "pooled JVM" model is no longer strategic to CICS
    - Still available and supported in CICS TS V4.2
    - Removed entirely in CICS TS V5.1
- **JVM server is strategic direction** for Java CICS applications
  - More standard Java behavior of the JVM
  - Supports a **multi-threaded** architecture
  - Migration of applications from pooled JVM to JVM server recommended
    - Primarily a “repackage and redeploy” operation
  - With pooled JVM, a “new” JVM is started for each task
    - JVM Server, with a single JVM, **requires Java code to be thread safe**

Open Services Gateway initiative

- Open Services Gateway initiative (**OSGi**)
  - A module system and service **platform for Java applications**
    - Provided by the OSGi Alliance (http://www.osgi.org)
  - **Addresses excessive costs of class loading and class management**
    - Prior to CICS TS V4.2, code changes required the JVM to be restarted to enable reloading new classes, resulting in application downtime
  - **OSGi allows applications to be developed into components**
    - Dependencies can be made explicit and resolved ahead of time
    - **Applications can be dynamically refreshed or versioned** and run side-by-side in the same JVM
- OSGi implements a complete and dynamic component model
  - Applications or components (in the form of **bundles** for deployment) can be remotely installed, started, stopped, updated, and uninstalled without requiring a JVM restart

CICS Explorer SDK

- CICS Explorer is an **Eclipse-based system management** tool
  - Initially released as a SupportPac in 2008
  - Eclipse-based graphical user interface (GUI) application
  - Runs on the Windows or Linux operating systems
  - Connects to either a CICSPlex or to a single CICS region
  - **No-charge downloadable** application
  - Can be extended with **plug-ins** for
    - CICS **Performance Analyzer**
    - CICS **Interdependency Analyzer**
    - CICS **Configuration Manager**
    - CICS **Transaction Gateway**
  - Supports development of Java applications to run in CICS
  - Supports creation and deployment of CICS Java OSGi applications

Themis

Support for Axis2 Technology

CICS can act as either a **web services provider or requester**

- CICS **implements the pipeline concept**
  - Service requests, inbound or outbound, go through a **series of handlers** to **process the web service** being called or to call another web service
  - The pipeline is the bridge between the web service request, typically in XML, and the CICS resources needed to process or make the request
  - These pipelines in CICS call a series of **handler programs to process the incoming or outgoing SOAP messages**
  - In previous versions of CICS, none of these handlers were written in Java
- CICS TS V4.2 supports use of the Axis2* **Java based SOAP engine** to process web service requests in provider and requester pipelines
  - Axis2 is an open source web services engine from the Apache foundation provided with CICS to process SOAP messages in a Java environment
  - Since **Axis2 uses Java**, the SOAP processing is **eligible for** offloading to System z Integrated Information Processor (**zIIP**) it runs in a JVM server

Interfacing With Language Structures

- CICS is a well established mature environment
  - Millions of lines of application code written using **COBOL or PL/I**
  - **Java code may need to interact with these** applications
    - Java programs **must be able use structured data** from those applications
      - E.g., link to a COBOL program passing a commarea (or a channel) where the data inside the channel or commarea is defined in a COBOL copybook
    - Typically used where VSAM file records or database records are accessed
  - Copybooks must be converted into Java constructs that Java programs can easily manipulate
    - CICS supports tools which can be used to import copybooks
    - Allows use of structured data from other programming languages in Java code

Themis

Java Development Using JCICS

CICS Java class library (JCICS)

- Used by Java applications to access CICS services
- Is the Java equivalent of the **EXEC CICS** application programming interface
  - API provided for other CICS supported languages, such as COBOL
- Can integrate with programs written in other languages
- Supports most of the functions of the **EXEC CICS** API
- Is included with CICS and with the CICS Explorer® SDK

Arguments for Passing Data

- Pass data between programs using a **communication area** (COMMAREA) or by using **channels and containers**
  - COMMAREA supports passing a maximum of 32 KB at a time
  - A channel and containers supports passing more than 32 KB
- The COMMAREA or channel, and any other parameters, are passed as arguments to the appropriate methods
  - For example, the Program class includes various link() methods
    - One version passes a **CommArea**, optionally including a length
    - Another version passes a **Channel** with one or more **Containers**

Java and Messaging

Message-Oriented Middleware

- Message-Oriented Middleware (MOM)
  - Software or hardware infrastructure
  - Supports sending and receiving **messages** between distributed systems
  - Allows **distributing application modules** over heterogeneous platforms
  - Simplifies development of applications that span multiple operating systems and network protocols
  - Creates a **distributed communications** layer
    - Insulates developers from operating systems and network interface details
  - Support **asynchronous calls** between the client and server applications
- Enterprise Messaging System (EMS)
  - A set of published enterprise-wide **standards**
    - Allows organizations to send messages between computer systems
  - Promote **loosely coupled** architectures
    - Allow message formats to change with minimum impact on message clients

Java Messaging Service (JMS)

- JMS is a **standard API** that is part of the J2EE platform
  - Used to access enterprise **messaging systems**
  - Enables **loosely coupled** (asynchronous) communication
    - A JMS provider delivers messages to a client as they arrive
    - Clients need not request messages in order to receive them
- Similar to JDBC in that it allows access to many different enterprise messaging systems, such as IBM MQ
  - Previously called WebSphere MQ … and MQ Series before that
- Applications that use JMS for sending or receiving messages are portable across JMS vendors
- Java EE Servers are required to have support for JMS

Themis

JMS Terminology

JMS clients	Java applications that use JMS
JMS providers	Messaging systems; route and deliver messages
JMS application	Business system using JMS clients & providers
Messages	Communicate information between JMS clients
Producer	JMS client that sends a message
Consumer	JMS client that receives a message
Administered objects	Preconfigured JMS objects - Created by an administrator for the use of clients - Two types: **destinations** and **connection factories**

Themis

Java , XML and Web Services

Themis

What is XML?

- eXtensible Markup Language
  - Simple, flexible text format
  - Derived from SGML
    - Standard Generalized Markup Language
      - An ISO standard (ISO 8879)
- The XML specification is based on recommendations by the World Wide Web Consortium (W3C)
- Fully supported in IDEs such as Eclipse and RAD
- Full Java support for processing XML data

Themis

Use of XML Documents

- XML is playing an increasingly important role in the exchange of a wide variety of data on the Web
  - E.g. business-to-business **e-commerce**
- Can be used to **store data** in files or databases
- Can be used as a means for **exchanging** platform independent **data** for distributed computing
- Used to describe **application configuration**

A Well-Formed XML Document

```
<?xml version="1.0" encoding="UTF-8" ?>
<class  title="DB2 for z/OS Advanced SQL">
   <date>July 25 – 29, 2016</date>
   <instructor>David Simpson</instructor>
   <location>ABC Training Center</location>
   <city>Westfield</city>
   <state>NJ</state>
   <student>
      <name>Susan Lee</name>
      <company>Sun Star Bank</company>
   </student>
   <student>
      <name>Edward Rice</name>
      <company>West Telco</company>
   </student>
 </class>
```

Themis

Web Service

- Definition
  - Interface that describes a **collection of operations** that are **network accessible** through standardized XML messaging
  - Accessed via common Internet protocols and data formats
    - HTTP
    - TCP/IP
    - XML
    - SOAP

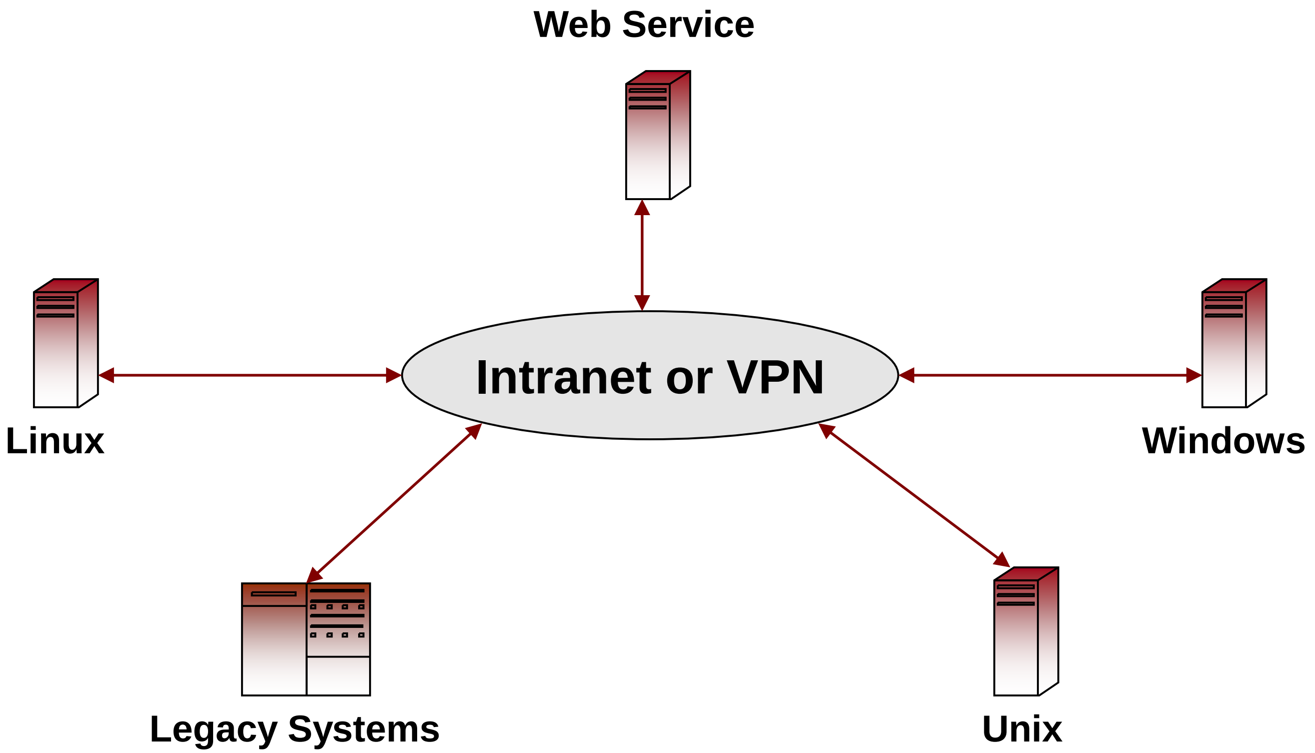
Hides implementation details so that it can be used independently of the hardware or software platform on which it is implemented and regardless of the programming language in which it is written

- Benefits
  - Separate systems can communicate and share services over the Internet
  - Technology is not tied to a specific platform or implementation language
  - Facilitates the **reuse of legacy systems** in new applications
  - **Loosely coupled** services
    - Interact only through stable interfaces (contracts)
  - Based upon industry standard protocols
    - Results in **universal support**

Themis

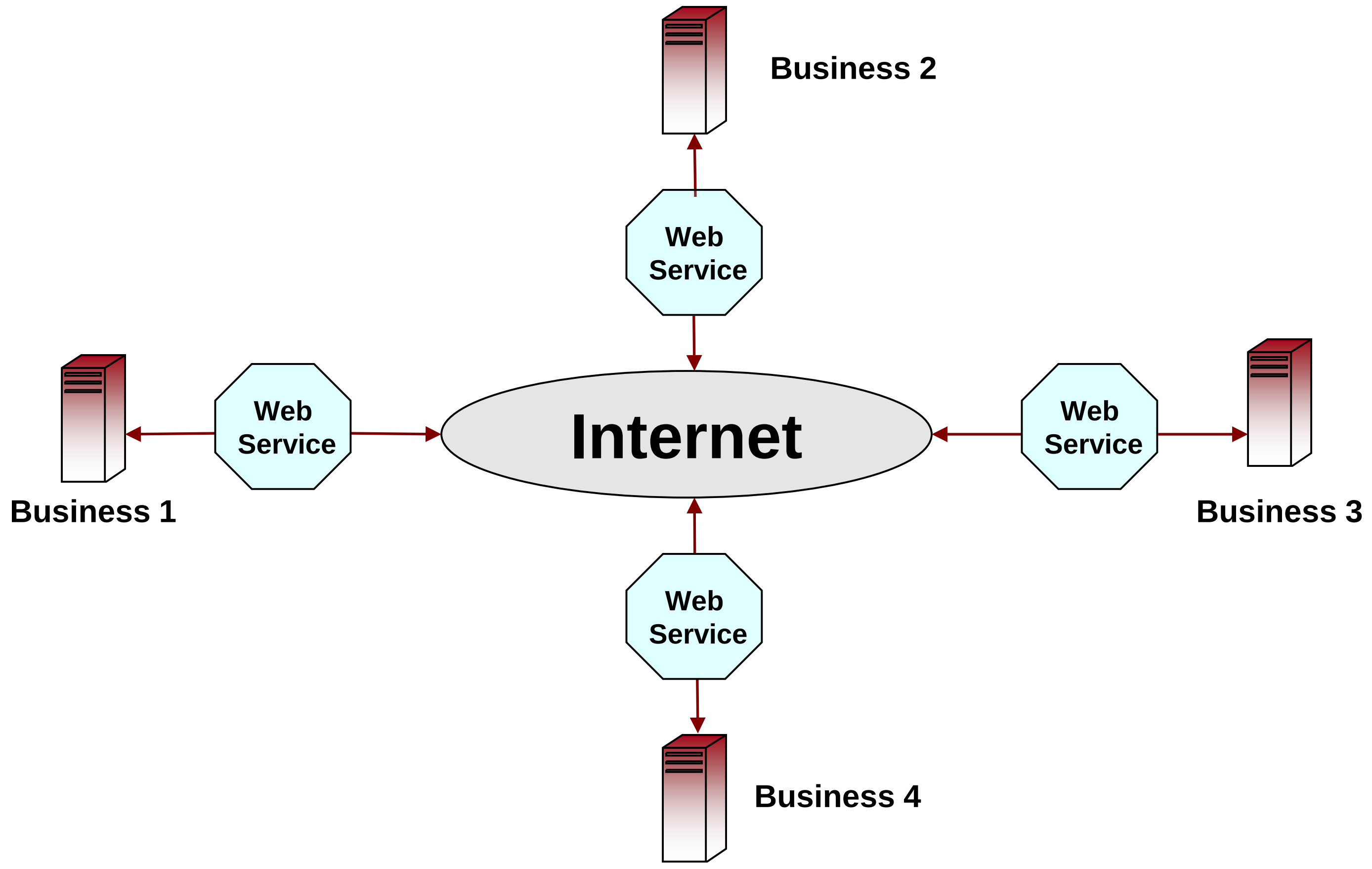
Roles of Web Services

- Integration of application functionality **within an organization** over an intranet, internet or extended virtual private network



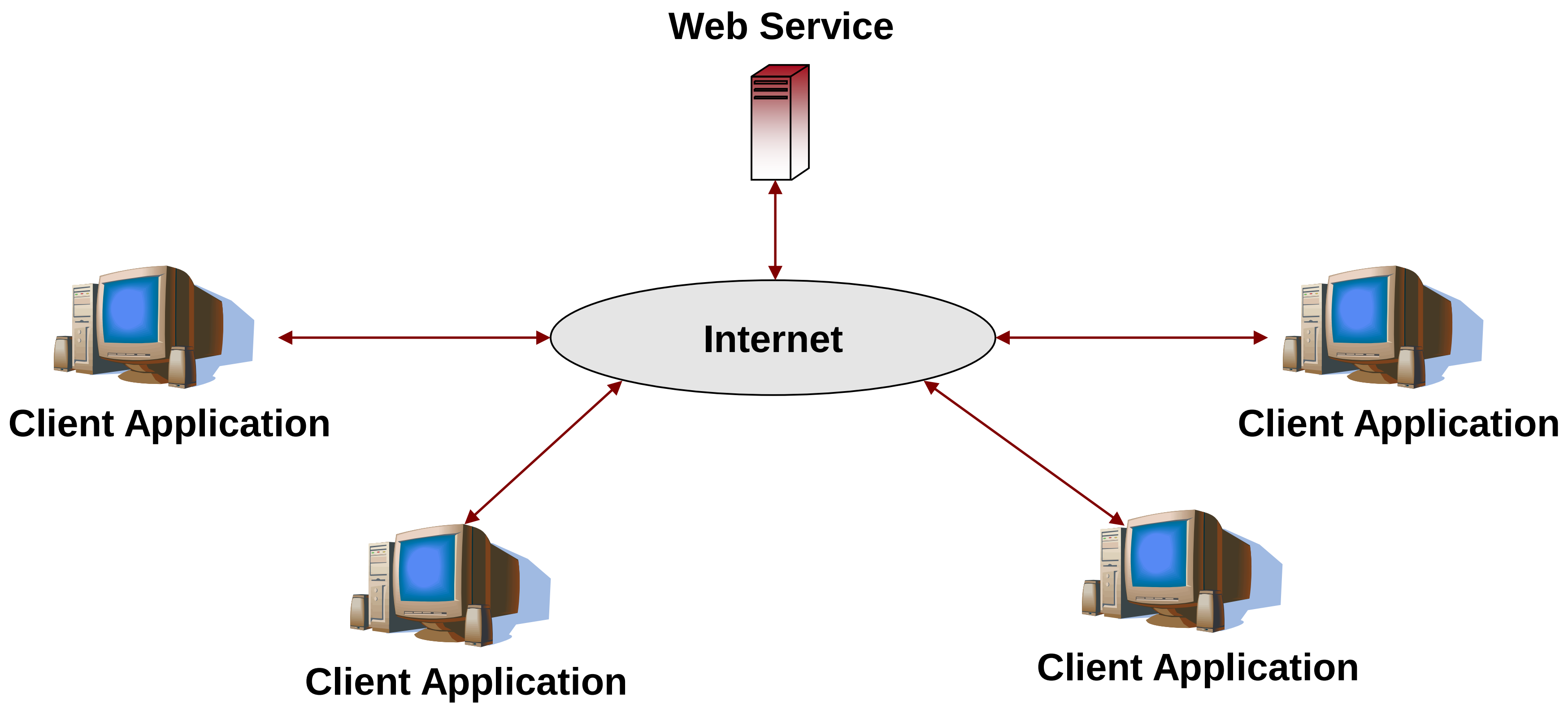
Roles of Web Services

- Business to Business (B2B)
  - Integration of applications **between business partners** via the internet



Roles of Web Services

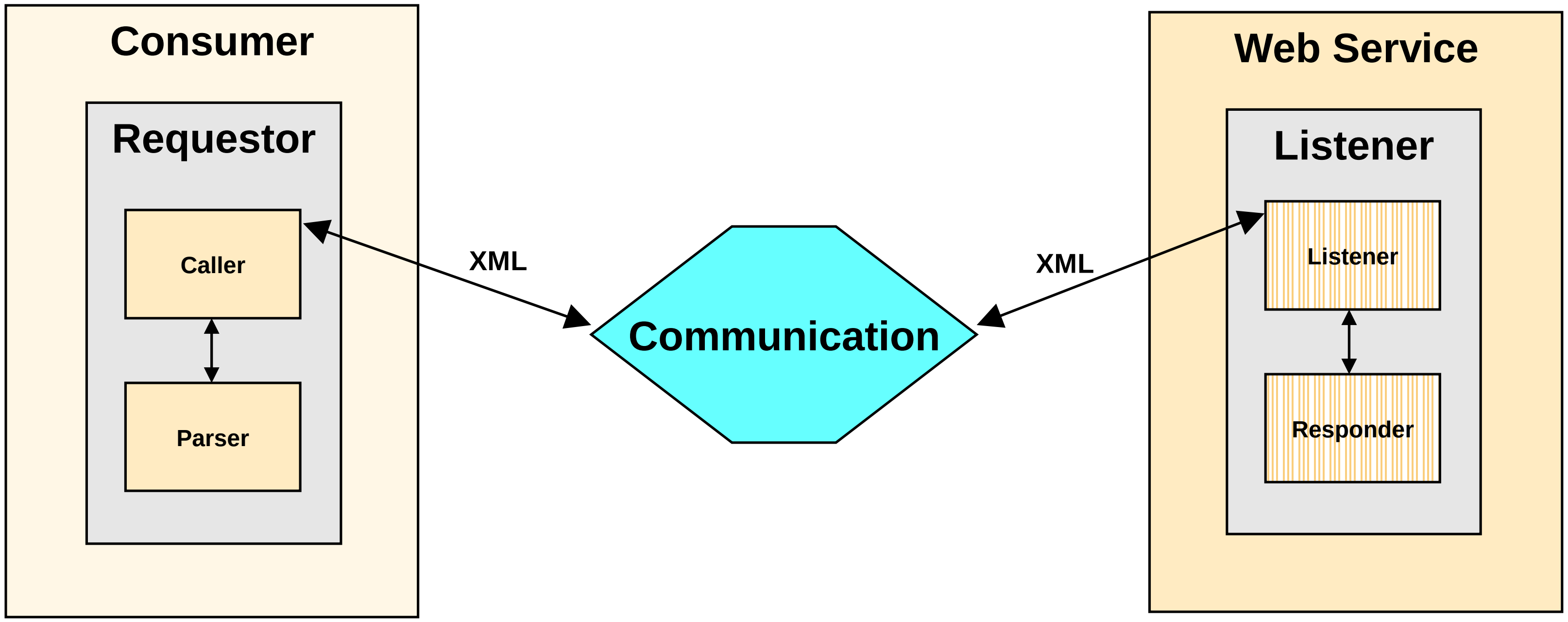
- Business to Customer (B2C)
  - Integration of business services with client applications



Web Services Communication Architecture

Three main components

- Consumer
- Communication
- Web service



Java Web Services (JWS)

- Hosted in **J2EE Web Container**
  - Each Java application in a container has a **Context Root**
    - Uniquely defines the **location of its services**
- **Exposes methods** of a Java class using annotations
- Rational Application Developer (RAD) provides **tools**
- Java bindings allow mapping an XML message to a java class
- Can create a Java Web service from the **bottom up**
  - Turn existing Java component into a Web service
- Or from the **top down**
  - Generate a Java bean skeleton for a Web service from a WSDL
    - Web Service Descriptor Language defines details of a web service

Themis

Web Service Client

- Any application that uses (invokes) a Web service
  - Client could even be a Web service itself
- Needs to know how to access the Web service
  - Defined by the WSDL for that Web service
- RAD can use a WSDL file as input to
  - Automatically generate a **Java proxy class** to simplify communication with the Web service defined by the WSDL
  - **Handles remote communication** with the Web service
- The Web service **client needs to obtain the WSDL**
  - May be found using a UDDI directory, via a URL, or privately

Themis

A RESTful Alternative

- REST: Representational State Transfer
  - A way to **define and access resources** on a network
    - Resources are identified by a URI
  - **REST is data-centric**, where SOAP is API-centric
  - **Very simple interface**
    - Simple URIs with HTTP methods such as GET and POST
    - Much simpler than SOAP-based messaging
- Sample REST service performing a catalog lookup
  - To lookup an item by its ID, simply issue a GET request to this URI

```
http://www.themis.com/catalog/rest/items/12345
```

  - The REST service will returns the item with that ID
  - Various options of format of return data (text, JSON, XML)
  - No WSDL, no SOAP – quite simple

Themis

REST vs. SOAP

- REST is **simpler than SOAP**
  - Uses common web concepts and existing HTTP infrastructure
  - Shorter learning curve – design similar to browser-based web requests
- REST is **HTTP based**
  - SOAP is capable of using other transports
    - e.g. SOAP over JMS (rare)
- REST is **point-to-point**
  - SOAP supports intermediary nodes (handlers)
- REST has **no standards** for security
  - Leverages existing HTTP standards
  - SOAP has many additional capabilities (WS-*)
    - E.g. WS-Security and WS-ReliableMessaging
- REST is **newer than SOAP**
  - Both are mainstream SOA approaches popular today

WS-* is a shorthand for referring to the myriad additional SOAP-based Web service standards

Thank you for coming

Themis inc.

www.themisinc.com

US 1-800-756-3000
Intl. 1-908-233-8900

On-site and Public
Instructor-led
Hands-on
Training
Over 400 IT Courses
Customization Available