# Using Java for Integration with CICS

Steve Fowlkes – fowlkes@us.ibm.com

Leigh Compton – lcompton@us.ibm.com

## Abstract

- There's never been a better time to take advantage of Java technology in CICS.
- This topic discusses the ways in which Java applications can be used to facilitate connectivity and interoperation between CICS and applications running on different platforms. The JCICS and JCICSX classes can be used in any CICS Java program, including Liberty Profile environments.
- While every effort is made to ensure the information presented is correct, always consult the IBM documentation for CICS.

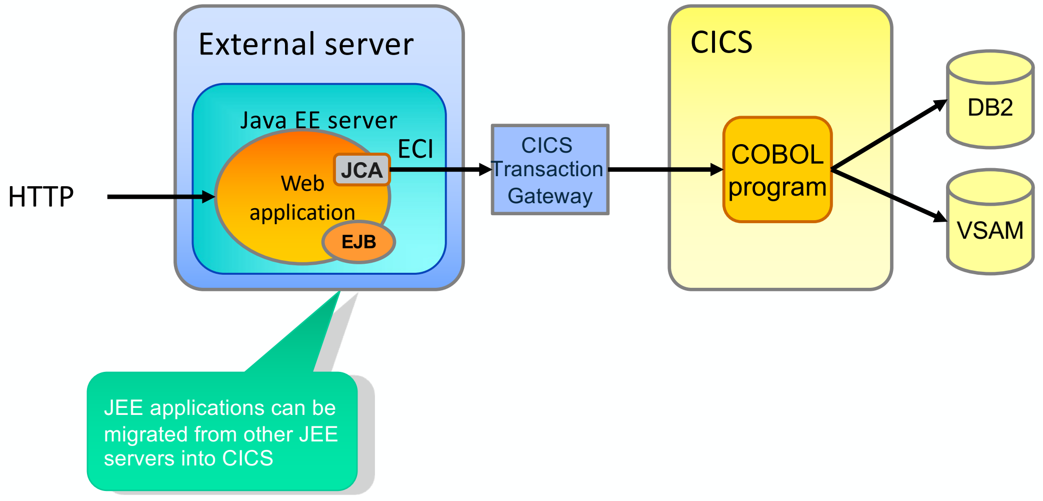
## Agenda

- Integration patterns with Java
- Java in CICS
    - Access patterns to CICS applications
    - JVMSERVER resource definition – Java virtual machines
        - OSGi and Liberty JVM servers
    - The JCICS and JCICSX classes
    - Link-to-Liberty with @CICSProgram annotation

## Integration Patterns with Java

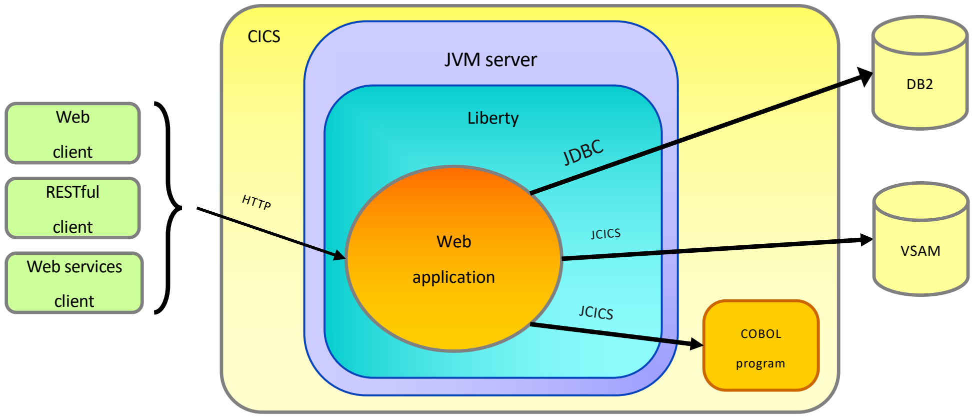
# Java running in external server

- Java application providing user interface or service enablement
- CICS program is backend



## Java running in CICS

- WebSphere Liberty Profile runs in a CICS JVM server
- Supports JEE 8, JEE 9, and Spring Boot



## Java in CICS

## Java support in CICS

- CICS provides the tools and runtime to develop and run Java applications in a CICS JVM server
  - Java applications can interact with CICS services and applications written in other languages
  - You can develop applications using the IBM CICS SDK for Java, Maven modules, or Gradle modules

- The CICS JVM server
  - Eligible Java workloads can run on specialty engine processors
    - reducing the cost of transactions
  - Different types of work such as threadsafe Java programs and web services
  - Application life cycle can be managed in the OSGi framework
    - no need to cycle the JVM server
  - Java applications that are packaged using OSGi can be ported more easily between CICS and other platforms
  - Java EE applications can be deployed into the Liberty JVM server

## JVM servers in CICS

- CICS uses a JVMSERVER resource to define the properties of a Java Virtual Machine
- Types of JVM servers:
  - OSGi
  - Liberty Profile
  - Classpath
    - AXIS2 capable
    - Security Token Server (STS) capable
    - Batch capable
    - Mobile capable

## Where can you use Java in CICS?

- JEE style programs (Liberty Profile environment)
  - Web applications
    - Servlets, Java Server Pages
  - JAX-RS applications (REST services)
  - JAX-WS applications (SOAP services)
  - Target of an EXEC CICS LINK
    - Target of EXCI call or DPL including ECI
    - Target of a CICS pipeline Web Service request
    - Pipeline Handler

- CICS style programs (OSGi environment)
  - Initial program of a transaction
  - Started by a user at terminal or EXEC CICS START
  - Program named in EXEC CICS HANDLE ABEND command
  - Target of an EXEC CICS LINK
    - Target of EXCI call or DPL including ECI
    - Target of a CICS pipeline Web Service request
    - Pipeline Handler

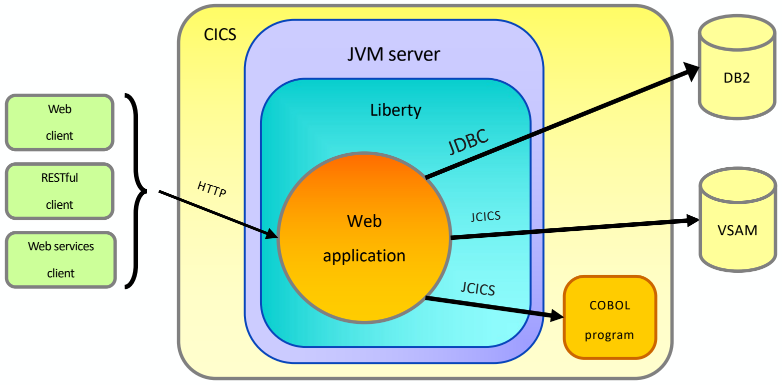
## What is WebSphere Liberty Profile?

***A lightweight, dynamic, composable JEE server runtime***

- Lightweight
  - Server install is only about 55 MB
  - Extremely fast server starts – typically under 5 seconds
- Dynamic
  - Available features are user selected and can change at runtime
  - Restarts are not required for server configuration changes
- Composable
  - Features are implemented as loosely coupled components with lazily resolved optional and mandatory dependencies
  - The availability of features and components determines what Liberty *can do and what is available to applications*
  - Focuses on having an easily configurable opt-in customization model, giving you full control over your configuration

## Why use Liberty in CICS?

- Porting web application to z/OS
  - JEE Web profile and JCA local ECI - ‘lift and shift’ porting from other JEE servers
- New integration logic for existing CICS services
  - Restful services or SOAP Web services interfacing existing CICS components
- Java business logic in CICS
  - Access to DB2 data – JDBC, EJBs, JPA or VSAM/JCICS



## Java EE 8 & Jakarta EE 8

- Provided by convenience feature javaee-8.0
- Takes advantage of HTTP 2.0 technologies like servlet-4.0
- Includes Java EE Security-1.0 API – JSR 375 (appSecurity-3.0)- which is a fully portable/standard Security API
- Provides new versions of: beanValidation-2.0, cdi-2.0, jaxrs-2.1, jpa-2.2, jsf-2.3, jsonp-1.0, servlet-4.0
- Introduces jsonb-1.0, a standard binding layer for converting Java objects to/from JSON

## CICS Servlets and JSP support

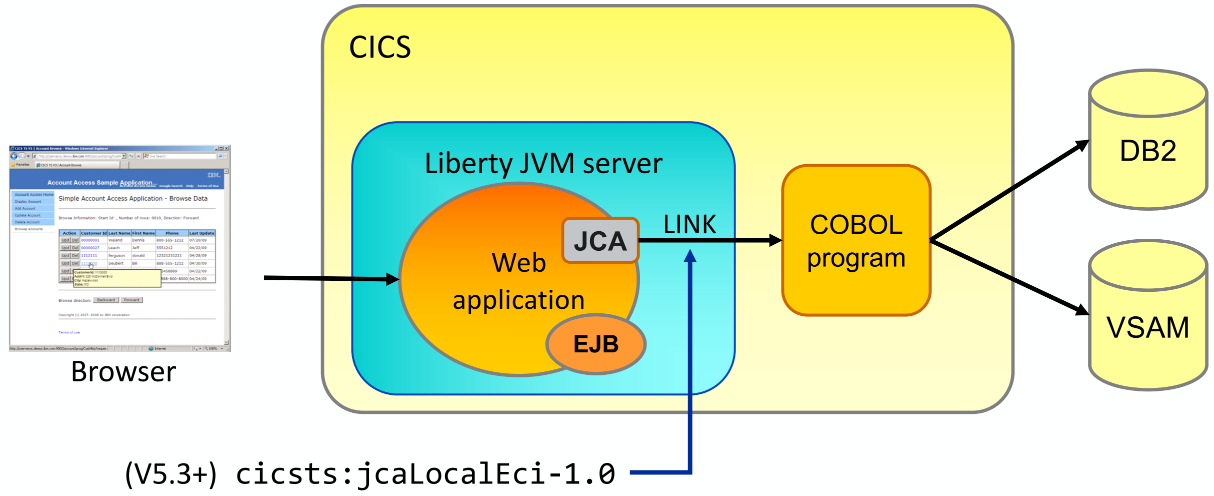
- Liberty JVM Server
  - Uses Java 8+ (64-bit)
  - Can use CICS basic authentication
  - Can run under Liberty security
- By default, task runs under the CJSA transaction
  - Can add a URIMAP and TRANSACTION resource to a CICS bundle if you want to run under a specific transaction

## Java Servlets and JSP support…

- Supports Java servlets and JavaServer Pages (JSPs) and RESTful clients, connectivity to DB2 (type 2 driver)
- CICS TS also supports distributed transactions, a type 4 JDBC driver, and Liberty security
- CICS's support built on the WAS Liberty profile technology
- Highly configurable
- Can use the JCICS and JCICSX classes along with JEE 8 and 9 and Spring Boot framework.
- Can use Eclipse, Rational Application Developer (RAD), and IBM Developer for System z (Idz) as well as other IDEs to develop servlets and JSPs
- Define BUNDLE resource for your application
- Several samples

# Liberty in CICS:

## Web UI for CICS programs

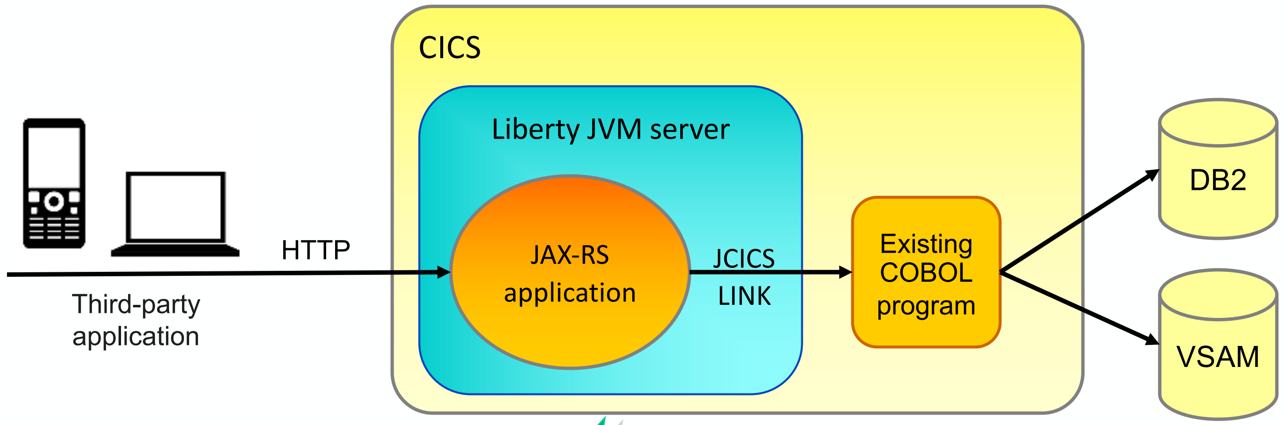


## REST/JSON in CICS using Liberty Server

- Liberty Profile Server executes within CICS
  - Dynamic profile of WAS
    - Provision only those Java features needed for application
  - Initially delivered web application server capabilities to CICS
- JAX-RS
  - Java API for RESTful Web Services
  - Simplifies development and deployment of service clients and endpoints
- JSON4J
  - Set of classes for handling JSON data
  - Simple model for transforming JSON data to other representations
- JAX-RS and JSON4J are well accepted in the Java community

# Liberty in CICS:

## REST and API enablement of existing CICS programs



RESTful (JAXRS) applications can be easily built to provide new service interfaces to existing business process applications

## CICS Liberty – Web Service support

- CICS TS V5.2+: JAX-WS and JAXB (in the Liberty profile)
- See an example in the CICSdev community article
  https://www.ibm.com/developerworks/community/blogs/cicsdev/entry/jax_ws_and_jaxb_support_in_cics_ts_v5_2_open_beta_liberty_profile?lang=en (Mark Cocker)
  - Article contains the steps necessary to get a sample web service running in:
    – Liberty on your desktop
    – CICS Liberty profile
  - Gets some CICS APIs running in the page that displays the web service output (CICS APIs only work in CICS)
  - Also tests the web service in the Eclipse Web Service Explorer

## CICS-Liberty – Web Services Support

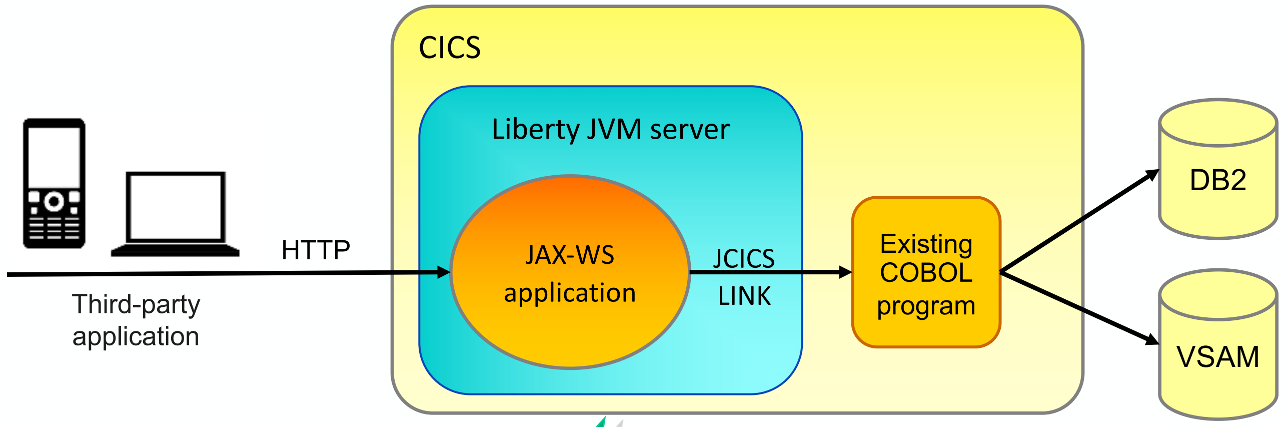
- Web Service Hello World (very, very, simple)

```
package com.ddw.verysimple.web.service;

import javax.jws.WebService;
import javax.jws.WebMethod;

@WebService
public class HelloWebService {
    private String message = new String("Hello, ");
    public HelloWebService() { }

    @WebMethod
    public String sayHello(String name) {
        return message + name + ".";
    }
}
```

- Eclipse wizards generate everything else:
  - Implementation
  - WSDL

The Liberty profile also supports JAXB
(JAXB can be used to prepare a string of XML that
makes up the body of your web service)

# Liberty in CICS:
## SOAP enablement for existing CICS programs



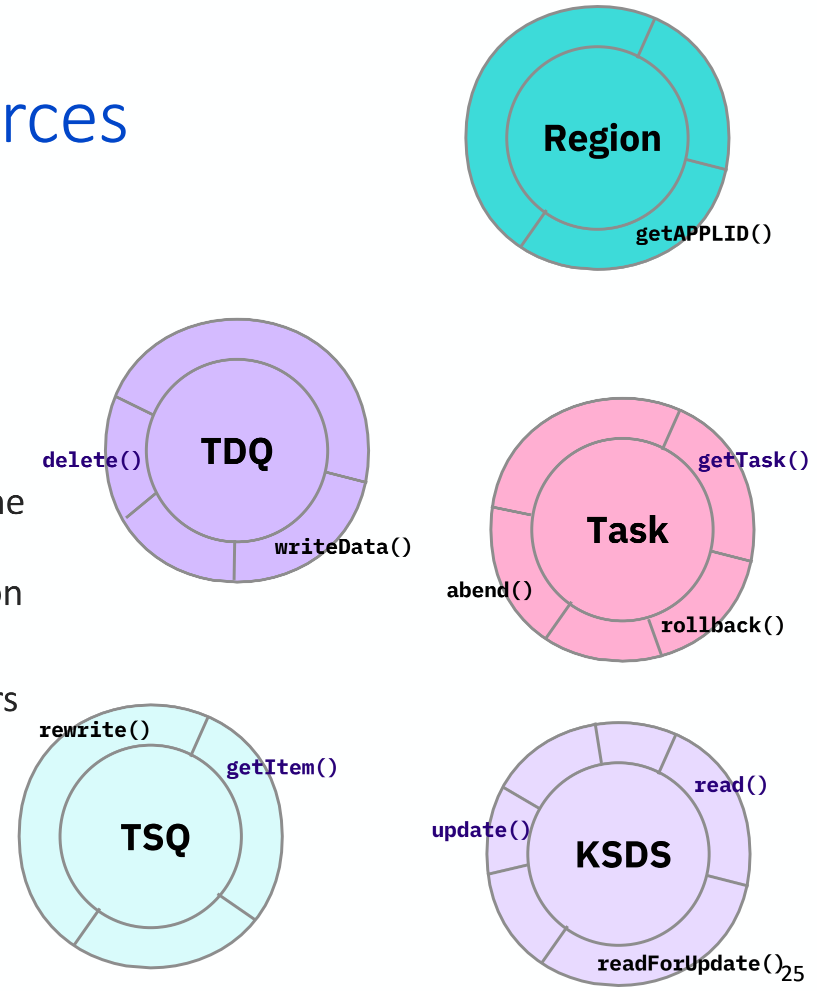
Web service (JAXWS) applications can be easily built to provide new service interfaces to existing business process applications

## Java program entry

- Servlet
  - At the doGet(), doPost(), doPut(), doDelete() method
- Web services and REST services
  - At the requested method or annotation
- Link to Liberty
  - At the @CICSProgram annotation

## Accessing CICS resources

- CICS resources are represented by objects of the appropriate type
  - Files
  - Programs
  - Temporary Storage
  - etc.
- Use setName() to specify resource name
  - Set other characteristics as appropriate
- Invoke method specific to desired action
- Data to and from CICS is in byte arrays
- The byte arrays from CICS are in Holders



# LINK from COBOL or Java

```
 MOVE 'Weiand'     TO COMM-EMP-NAME.
     MOVE 1             TO COMM-EMP-ACCOUNT-NO.
*
*      more moves here
*
     EXEC CICS LINK PROGRAM('EMPPROG1')
            COMMAREA(COMM-DATA)
            LENGTH(SIZE OF COMM-DATA)
            RESP(RESP-FLD) RESP2(RESP2-FLD)
            END-EXEC.
     EVALUATE RESP-FLD
         WHEN DFHRESP(NOTAUTH)
*       ---> code for not authorized here
         WHEN DFHRESP(PGMIDERR)
*       ---> code for Program id error here
     END-EVALUATE.
```

```
EMPPROGCommarea ec = new EMPPROGCommarea();
ec.setEmployeeName("Weiand");
ec.setEmployeeAccountNumber(1);
//
// more sets here
//
Program aProg = new Program();
aProg.setName("EMPPROG1");
try {
   aProg.link(ec.getBytes());    // Data Binding
   System.out.println("Link was successful");
} catch (InvalidProgramIdException a) {
   System.out.println("Program not found");
} catch (NotAuthorisedException b) {
   System.out.println("Not authorized to use pgm");
} catch (CicsConditionException cce) { // catch the rest
   System.out.println("Unexpected exception: "+cce);
}
```

## JCICS classes, part 1

- CANCEL, RETRIEVE, START
- Temp Storage
- Transient Data
- ENQUEUE, DEQUEUE
- APPC mapped conversations
- TRACE
- SYNCPOINT, ROLLBACK
- Asynchronous API
  - AsyncService
  - ChildResponse
- BMS and Terminal Control
  - converse(), receive(), send(), sendControl(), sendText()
  - no SEND MAP, RECEIVE MAP, HANDLE AID or WAIT TERMINAL
- Document API
- Common equivalents of ASSIGN, ADDRESS, INQUIRE
- FILE Control, including BROWSE
- LINK
  - no SUSPEND

## JCICS classes, part 2

- SIGNAL EVENT
- TRANSFORM
  - XmlTransform
- INVOKE SERVICE
- WS-Addressing
- Channels and Containers
- IsCICS – Test whether we are in a CICS environment
- WEB
  - HttpSession
  - HttpRequest
  - HttpResponse
  - HttpHeader
  - HttpClientRequest
  - HttpClientResponse
  - CertificateInfo

## Restrictions, limitations, and unsupported functions

- Journal services
- Storage services
  - no GETMAIN – use normal Java storage management
- Timer services
  - START & CANCEL are supported
  - no POST, DELAY, WAIT EVENT
- System Programmer Interface (INQUIRE/SET/PERFORM, etc)
- Must not use System.exit() method
- BMS Send Map and Receive Map **
- APPC unmapped conversations
- CICS Business Transaction Services (BTS)
- XCTL
- DUMP services

## JCICX

- Benefits
  - easy mocking and stubbing
  - can be run remotely in development environments
  - syntax is simplified and natural with more recent Java constructors
  - Code written using the JCICSX API classes can execute without change, both in remote development mode and when deployed to run in CICS
  - compatible with the JCICS API

- JCICSX API introduced in CICS TS V5.6
- JCICSX API classes extend parts of the JCICS API with the capability of remote development and mocking
- The JCICSX API classes support only a subset of CICS functionality
  - focused on linking to CICS programs using channels and containers
- Client-side tooling is available to enable Liberty users to use JCICSX to access CICS from a servlet

## JCICSX

- Note, there is no support for LINK passing Commarea
- Mocking out the CICS calls enables you to independently unit test the logic of your application
  - There are many mocking frameworks you can use
  - Example in documentation shows how to use Mockito to return some mocked contents of a container

- Supported functions
  - Channels and Containers
    - Create Channel
    - Create Containers
    - PUT data into Containers
    - GET data from Containers
  - Program LINK
    - LINK to program passing Channel
    - LINK to program with no data

## Link to Liberty

- Java application runs under the same unit-of-work (UOW) as the calling program
  - updates made to recoverable CICS resources are committed or backed out when the transaction ends
  - when the Java application is invoked, there is no JTA transaction context
  - If the application starts a JTA transaction, a syncpoint is performed to commit the CICS UOW, and create a new one
- Java EE application is required to contain a plain old Java object (POJO)
  - packaged as a web archive (WAR) or enterprise archive (EAR) file
- A method in the Java EE application can be made a CICS program by use of the @CICSProgram annotation
- CICS creates the program resource defined by the @CICSProgram annotation
- Data passed between non-Java and Java programs using Channels and Containers
  - COMMAREA is not supported

## Summary

- Integration patterns
  - External servers
    - Connectivity to CICS over MQ or CICS TG
  - JVM servers in CICS
- Java in CICS
  - Patterns and application styles
    - Modern user interface (support for browsers)
    - SOAP-based web services
    - REST and RESTful services
  - Using CICS resources
    - JCICS and JCICSX classes
    - Link-to-Liberty