#SHAREorg

SHARE

# Web Apps using Liberty Profile Technology in CICS TS v5

Ian J Mitchell,
IBM Distinguished Engineer, CICS Portfolio Architect
IBM Hursley

Thursday 13th March 2014
Session Number : 14832

SHARE in Anaheim

# Abstract

CICS TS V5.1 offers a fast and lightweight Java™ web container, providing developers with the rich features of Java Servlet and JavaServer Pages (JSP) specifications, and fast local access to your existing CICS applications and data. Built on WebSphere® Application Server Liberty profile technology, this web container runs in the CICS JVM server environment. A wide range of Java development tools can be used to develop web applications, such as WebSphere Application Server Developer Tools for Eclipse (WDT), and Rational® Developer for System z. This session will demonstrate these features and show the integration between the web container and CICS resources.

# Agenda

- Java Update for CICS TS
- What is Liberty?
- Liberty in CICS
- Deeper look at Liberty Technology in CICS
- Future Direction and Summary

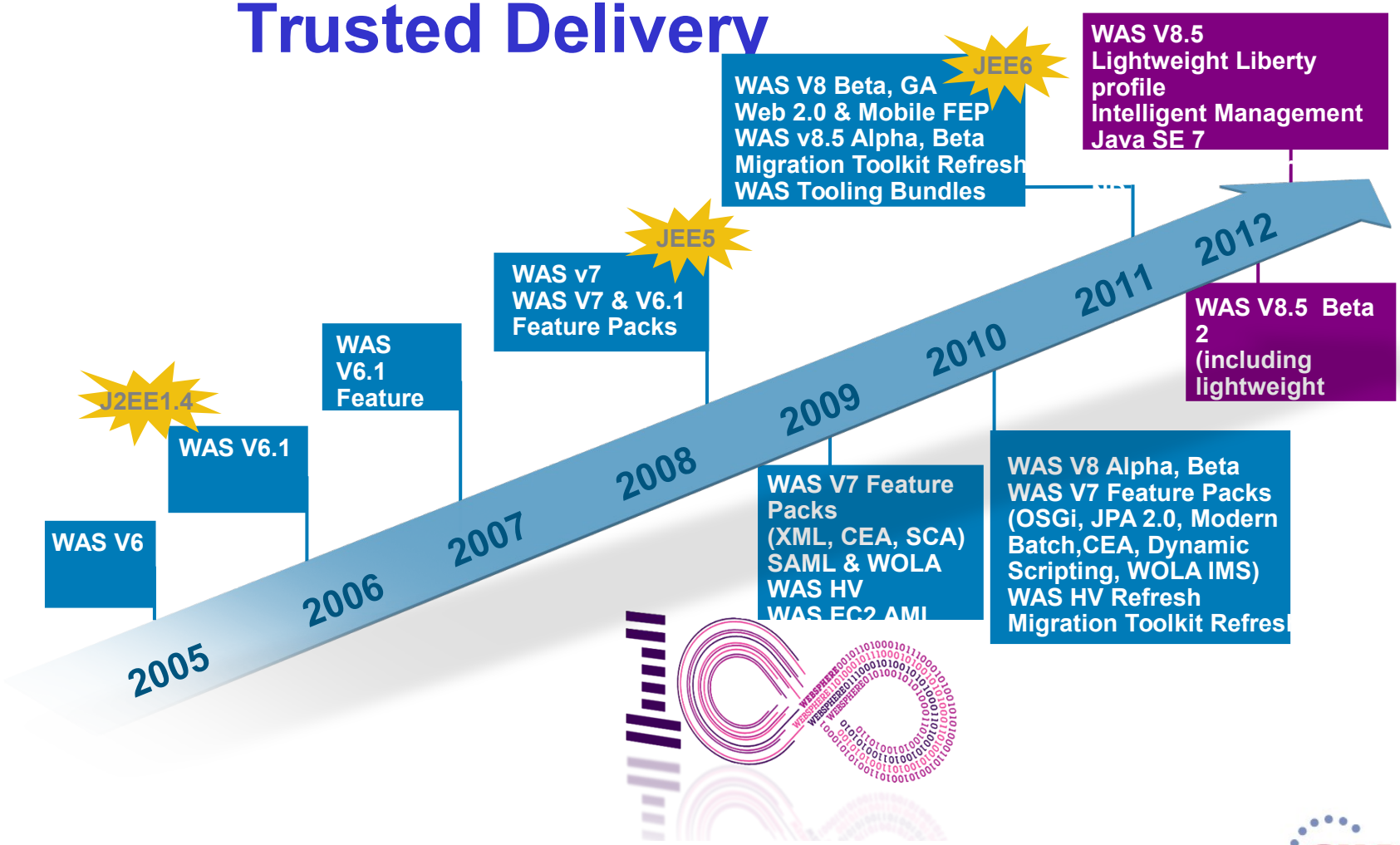
# Java Update

- Java 7 (64-bit) JVMServer
- Equinox 3.7 as the OSGi framework.
  - Implements the OSGi R4.3 specification
- WAS Liberty Profile 8.5.0 based Web Container
- IBM CICS SDK for WebSphere Application Server Liberty profile v5.1
- Eclipse 3.6.2

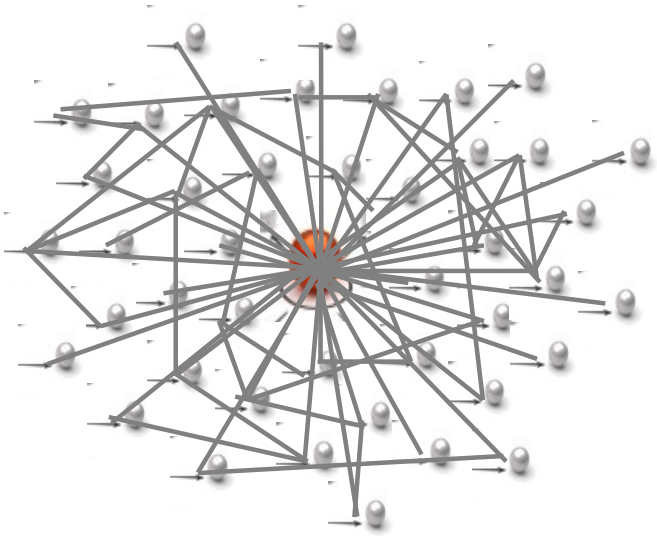
# Introduction to the CICS Java Web Container based on WAS Liberty technology

# What's Liberty?

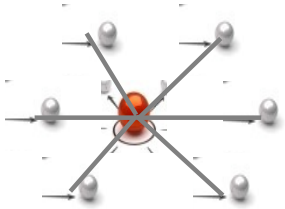
# WebSphere Application Server: 15 Years of Leadership and Trusted Delivery



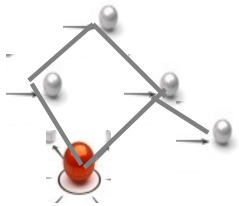
If this is tWAS...

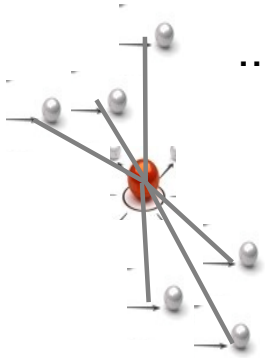


...this is Liberty (WAS)



...so is this





...or even this!

# What is the ‘Liberty’ Profile?

***A lightweight, dynamic, composable runtime***

**Lightweight**

Server install is only about 55 MB

Extremely fast server starts – typically well 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’s available to applications*

# Configuration by Exception

- This is the entire configuration needed to run Liberty as a Web-container with Servlet support.

```
<server description="new server">

    <featureManager>
        <feature>servlet-3.0</feature>
    </featureManager>

    <application id="BasicWeb" location="BasicWeb.war"
      name="BasicWeb" type="war"/>

</server>
```

# Lightweight Configuration

```
<server description="tradeLiteServer">
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>jdbc-4.0</feature>
    </featureManager>


    <logging consoleLogLevel="INFO" />


    <application type="war"
                 id="tradelite"
                 name="tradelite"
                 location="${shared.app.dir}/webcontaine


    <include location="jdbc-drivers.xml" />
    <include location="${user.home}/custom.xml" optional="


    <dataSource id="jdbc/DerbyTradeDataSource"
                jndiName="jdbc/TradeDataSource"
                jdbcDriverRef="DerbyEmbedded">
        <properties databaseName="${shared.resource.dir}/data/tradedb" />
    </dataSource>
</server>
```

Features control what's available in the runtime.

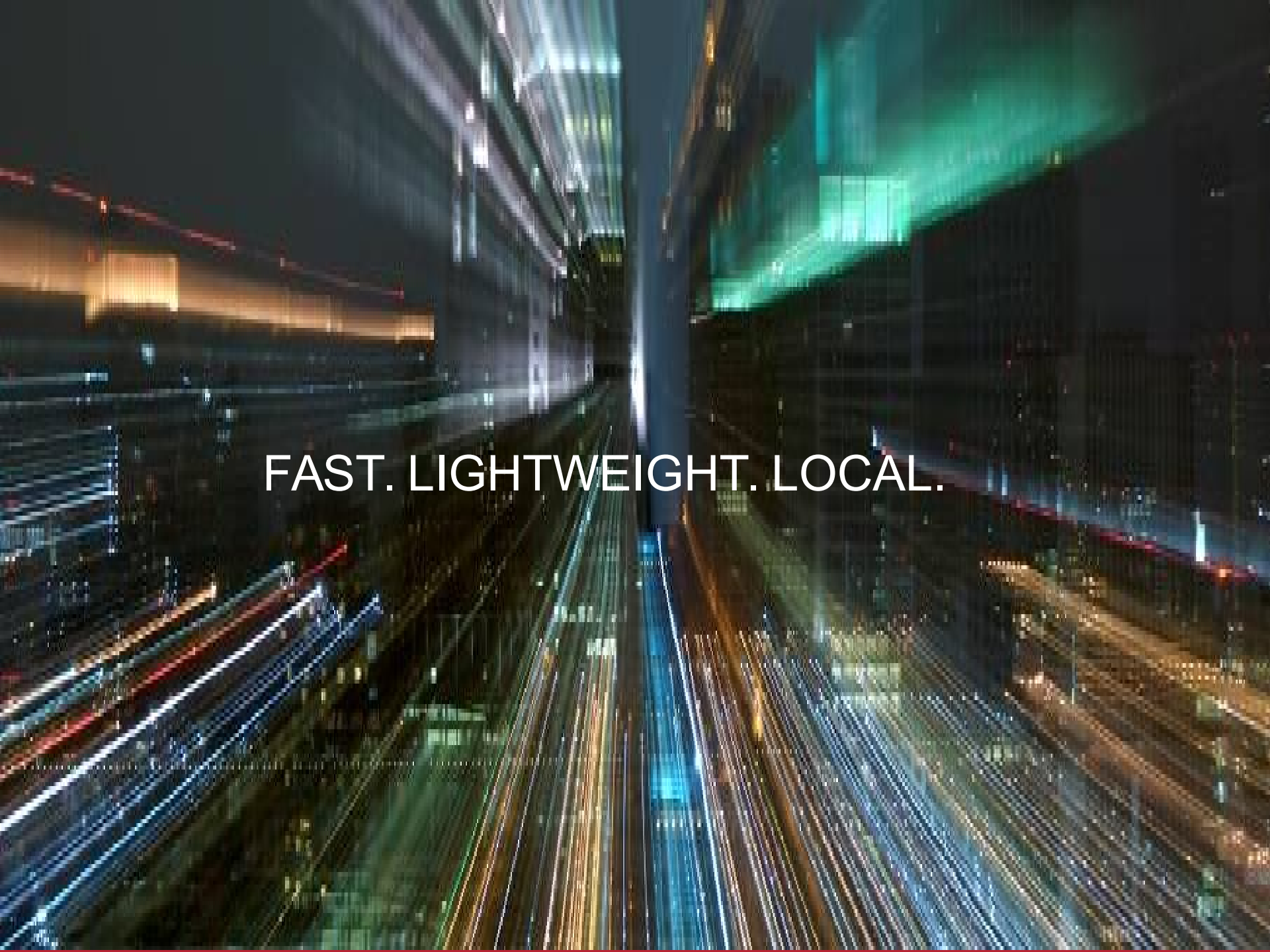
*Singleton* configurations specify properties for runtime services when there's only one instance

*Instance* configurations allow multiple instances of resources and applications to be declared

*Includes* can be used to implement an extensible configuration model

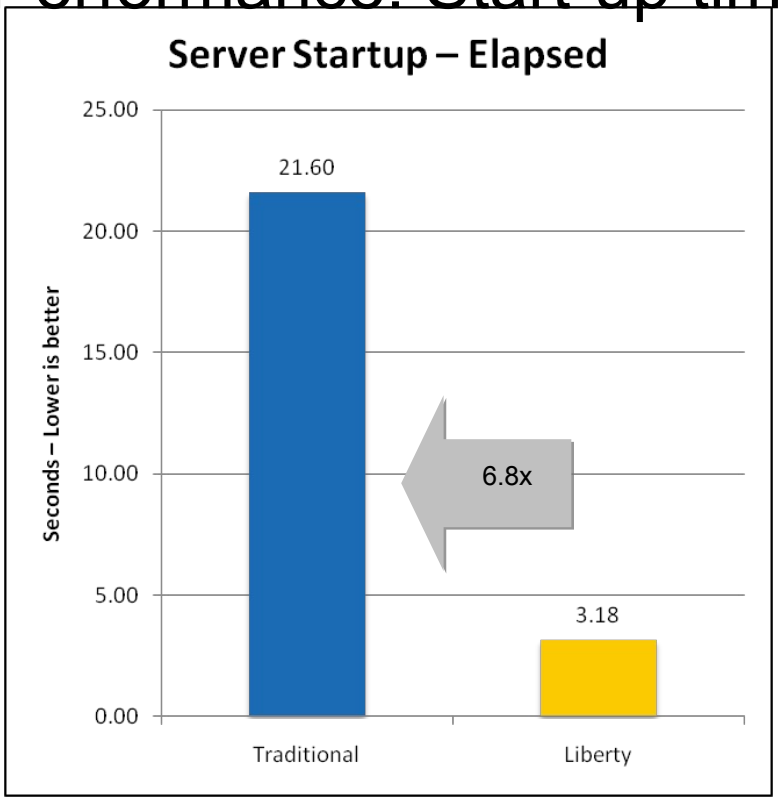
*References* can be used in multiple elements to point to and share a common definition

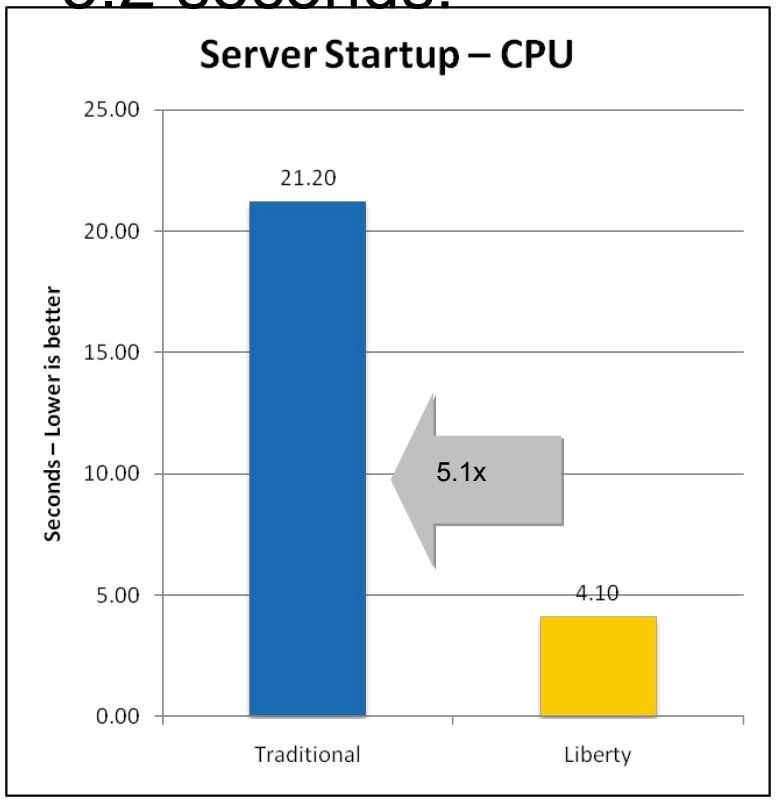
- “CICS TS V5.1 offers a fast and lightweight Java web container, providing developers with the rich features of the Java Servlet and JavaServer Pages (JSP) specifications, and fast local access to your existing CICS applications and data. Built on WebSphere Application Server Liberty technology, this web container runs in the CICS JVM server environment. A wide range of Java development tools can be used to develop web applications, such as WebSphere Application Server Developer Tools for Eclipse (WDT), and Rational Developer for System z. “


FAST. LIGHTWEIGHT. LOCAL.

# Liberty on z/OS – start-up time

## Performance: Start-up time – 3.2 seconds!

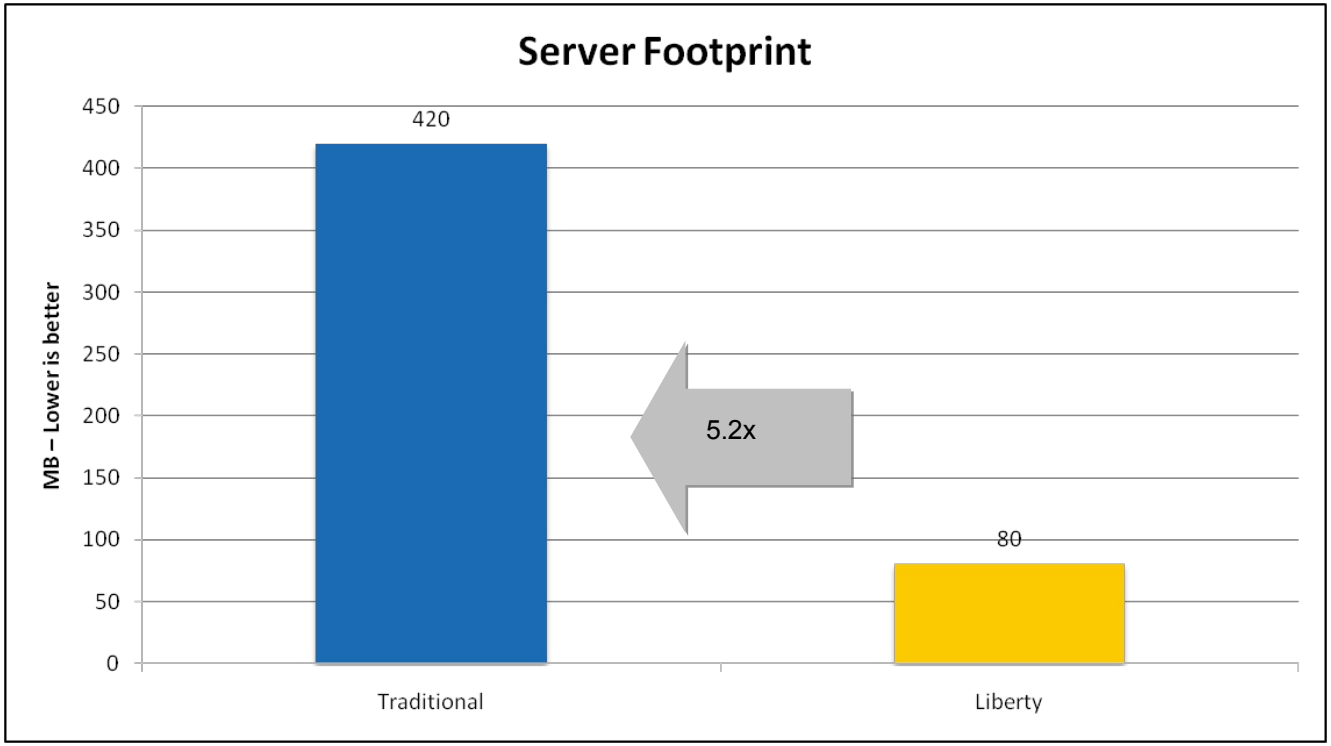


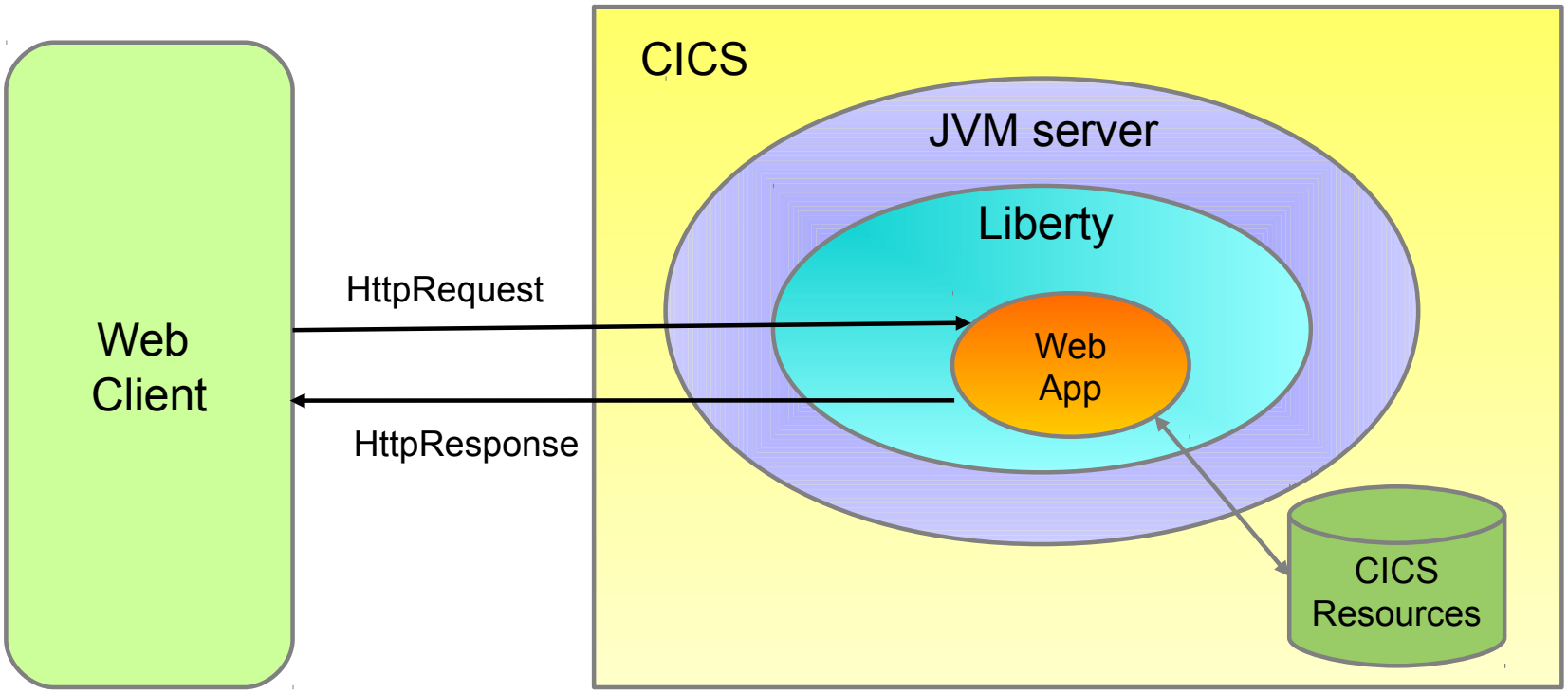

- Liberty – 64bit IBM Java 6.0.1, 64/64MB min/max heap, 60MB shared class cache, TradeLite installed
- Traditional – 64bit IBM Java 6.0.1, 1SR,128/256MB min/max CR heap, 256/512MB min/max SR heap, 75MB CR shared class cache, 75MB SR shared class cache, no applications installed

# Liberty on z/OS – memory footprint

Performance: Memory footprint – 80% reduction




- Liberty – 64bit IBM Java 6.0.1, 64/64MB min/max heap, 60MB shared class cache, TradeLite installed
- Traditional – 64bit IBM Java 6.0.1, 1SR,128/256MB min/max CR heap, 256/512MB min/max SR heap, 75MB CR shared class cache, 75MB SR shared class cache, no applications installed

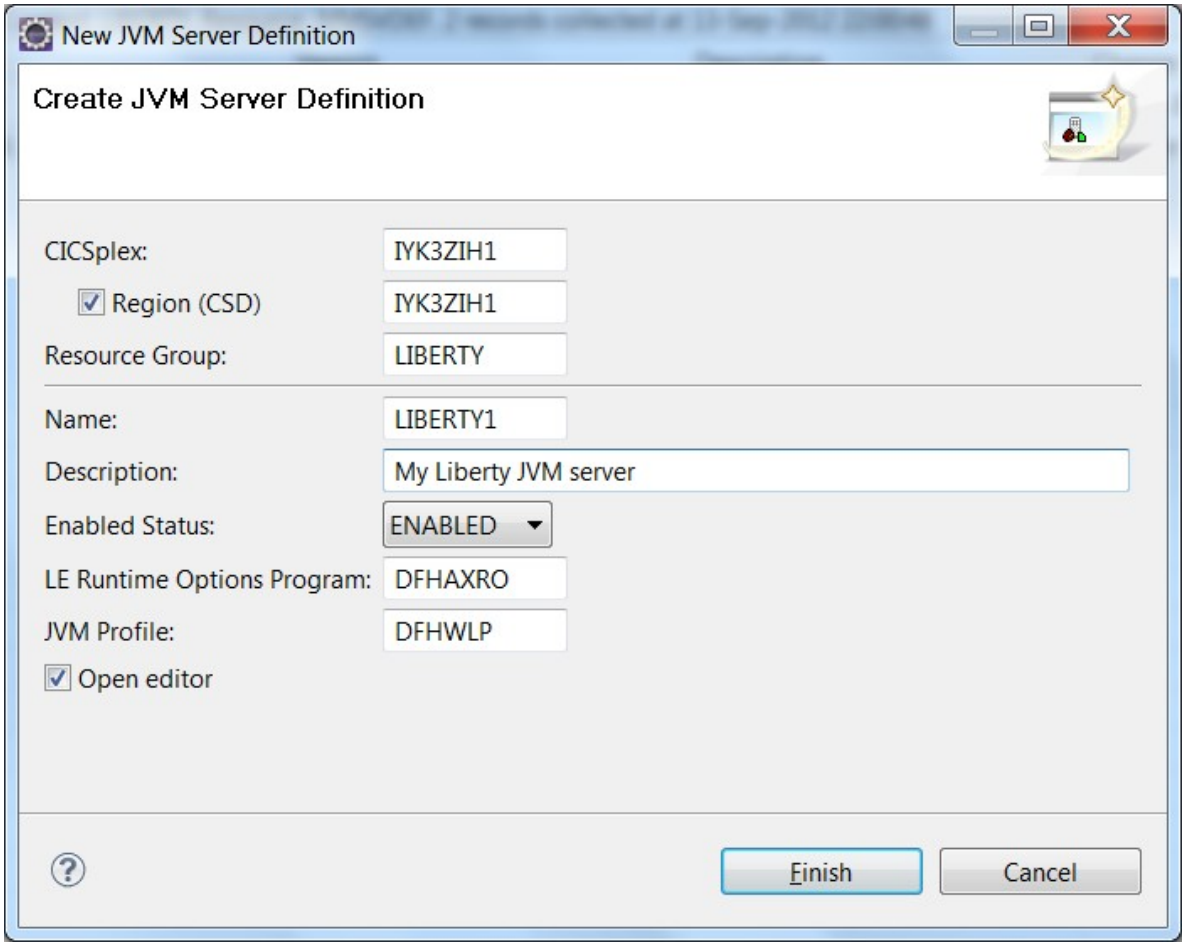


# Benefits for CICS

- Provides “off the shelf” Web-server capabilities (JSPs and Servlets)
- Potential to re-use even more WebSphere technology in CICS.
- JSP and Web servlets have direct, local, access to CICS data and resources.
- Servlets can take advantage of existing CICS OSGi applications to provide a Dynamic Web front end.

# Nought to Web-App

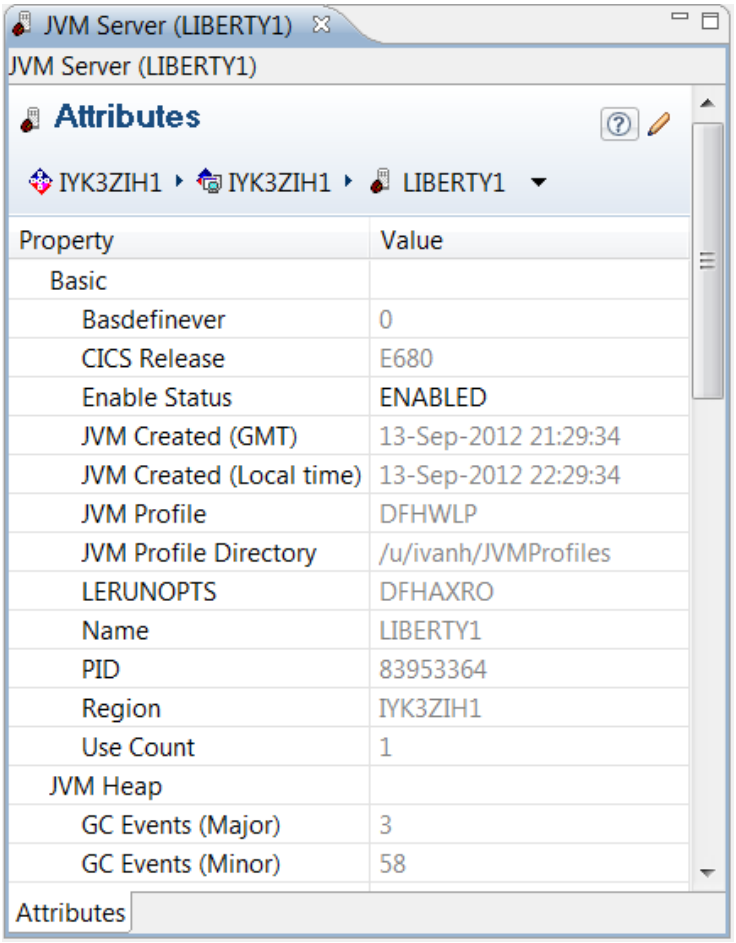
# Create a JVM server resource in Explorer, CEDA, or CPSM.



# Configure the JVMProfile

- Copy the sample DFHWLP
- Check JAVA_HOME is correct.
- Uncomment the WLP_SERVER_HTTP_PORT and choose a unique port number.
- Point your JVM server definition at the new JVMProfile

# Enable the JVM server



# Liberty is running! (check the logs).

```
Server defaultServer created.

Launching defaultServer (wlp-1.0.0.20120428-1251/websphere-kernel_1.0.0) on
   IBM J9 VM, version pmz6470sr1-20120302_01 (SR1) (en_US)

[AUDIT   ] CWWKE0001I: The server defaultServer has been launched.

[AUDIT   ] CWWKG0028A: Processing included configuration resource:
   file:/u/ivanh/IYK3ZIH1/LIBERTY1/wlp/usr/servers/defaultServer/installedApp
   s.xml

[AUDIT   ] CWWKG0028A: Processing included configuration resource:
   file:/u/ivanh/IYK3ZIH1/LIBERTY1/wlp/usr/servers/defaultServer/cicsSecurity
   .xml

[AUDIT   ] CWWKZ0058I: Monitoring dropins for applications.

[AUDIT   ] CWWKF0011I: The server defaultServer is ready to run a smarter
   planet.
```

# Foundation is Eclipse

Install Eclipse 4.2.2 (Juno) preferably JEE version, but Classic will suffice.

- Windows
- Linux

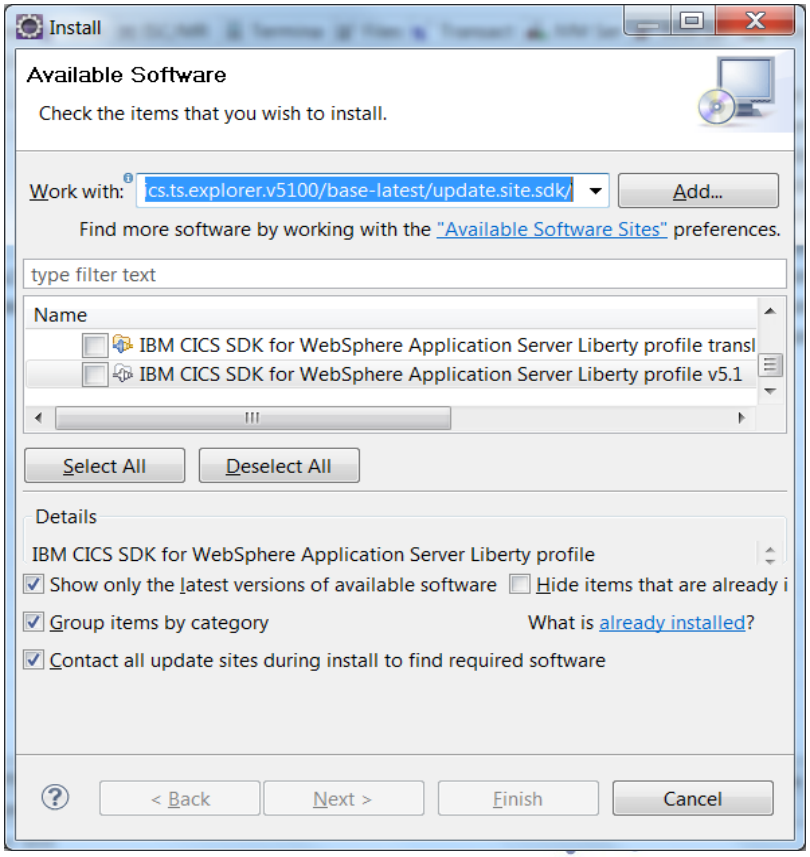
http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/junosr2


# Install IBM CICS SDK for WebSphere Application Server Liberty profile v5.1

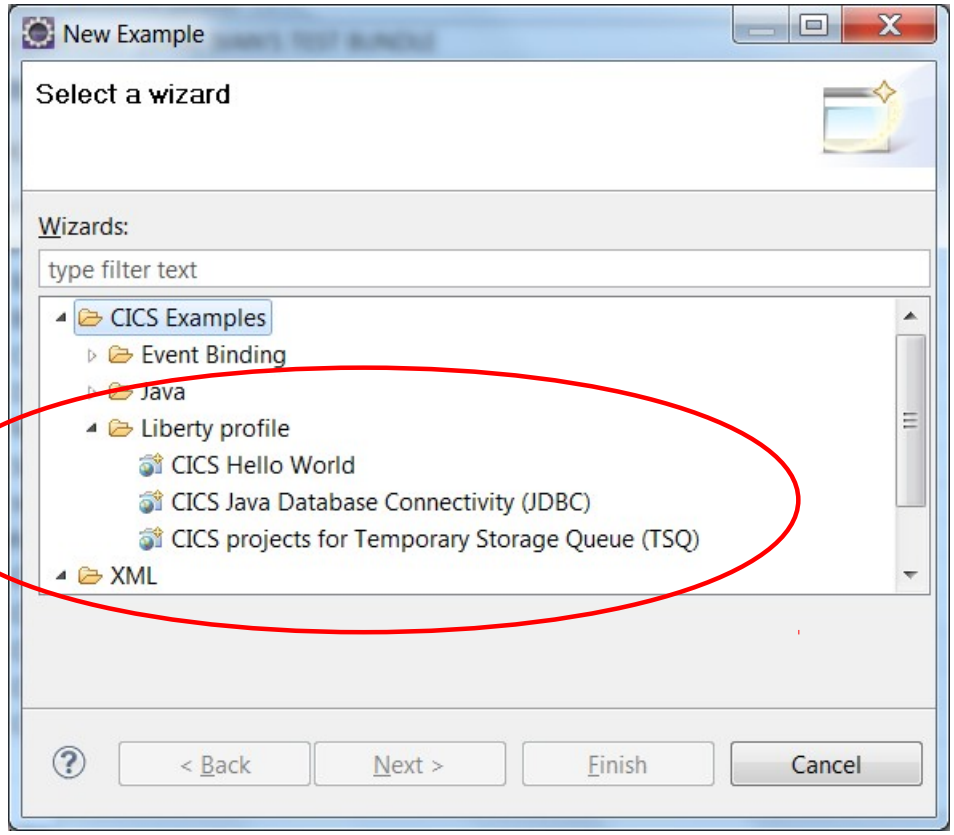
http://www.ibm.com/support/docview.wss?rs=1083&uid=swg24033579

Add CICS SDK to Eclipse.

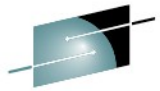
- Via Installation Manager (IM)
- Via Eclipse “Help->Add New Software” (P2)
- Direct download

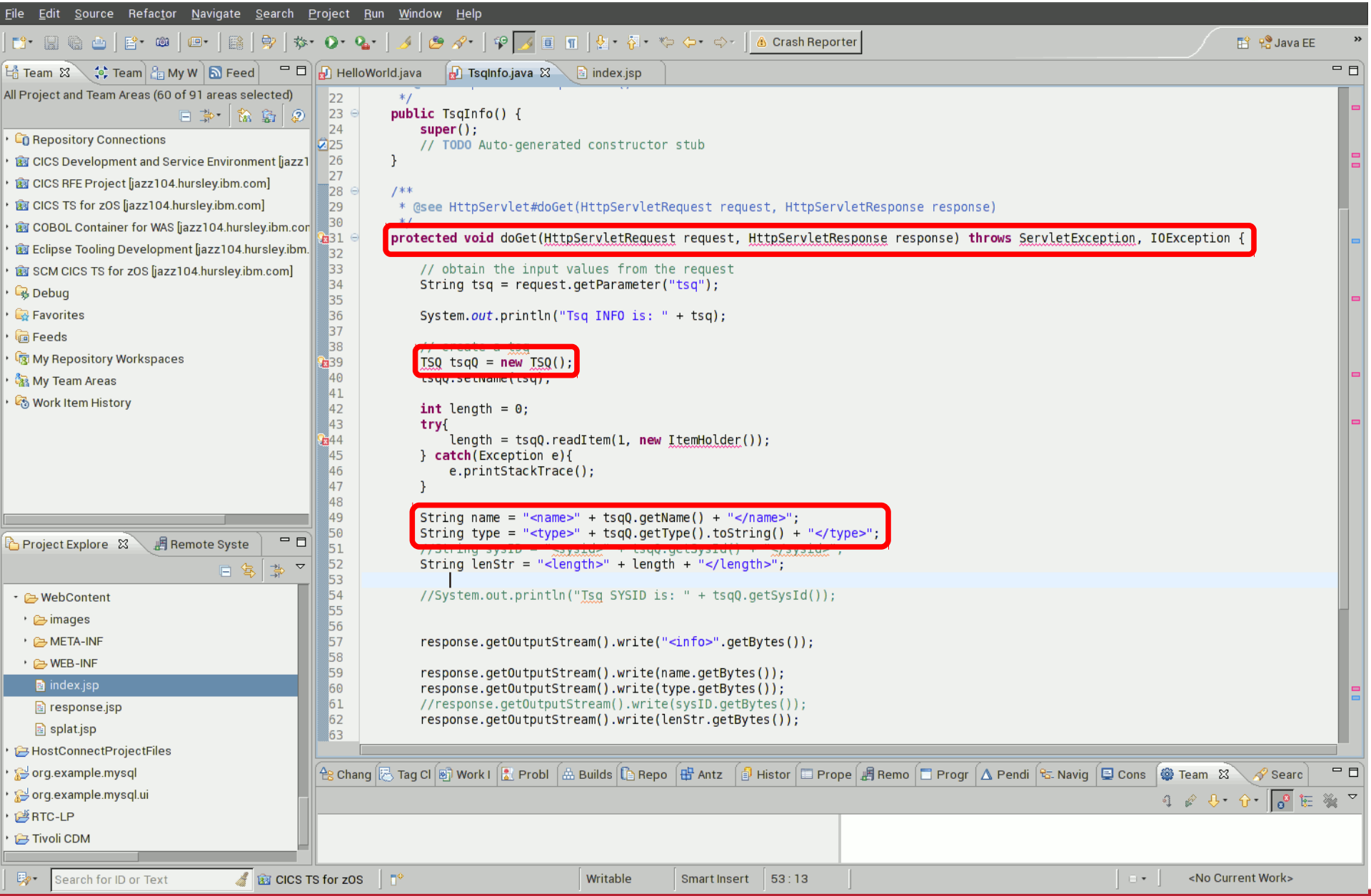


# Create a Dynamic Web Project, or choose one of the Examples

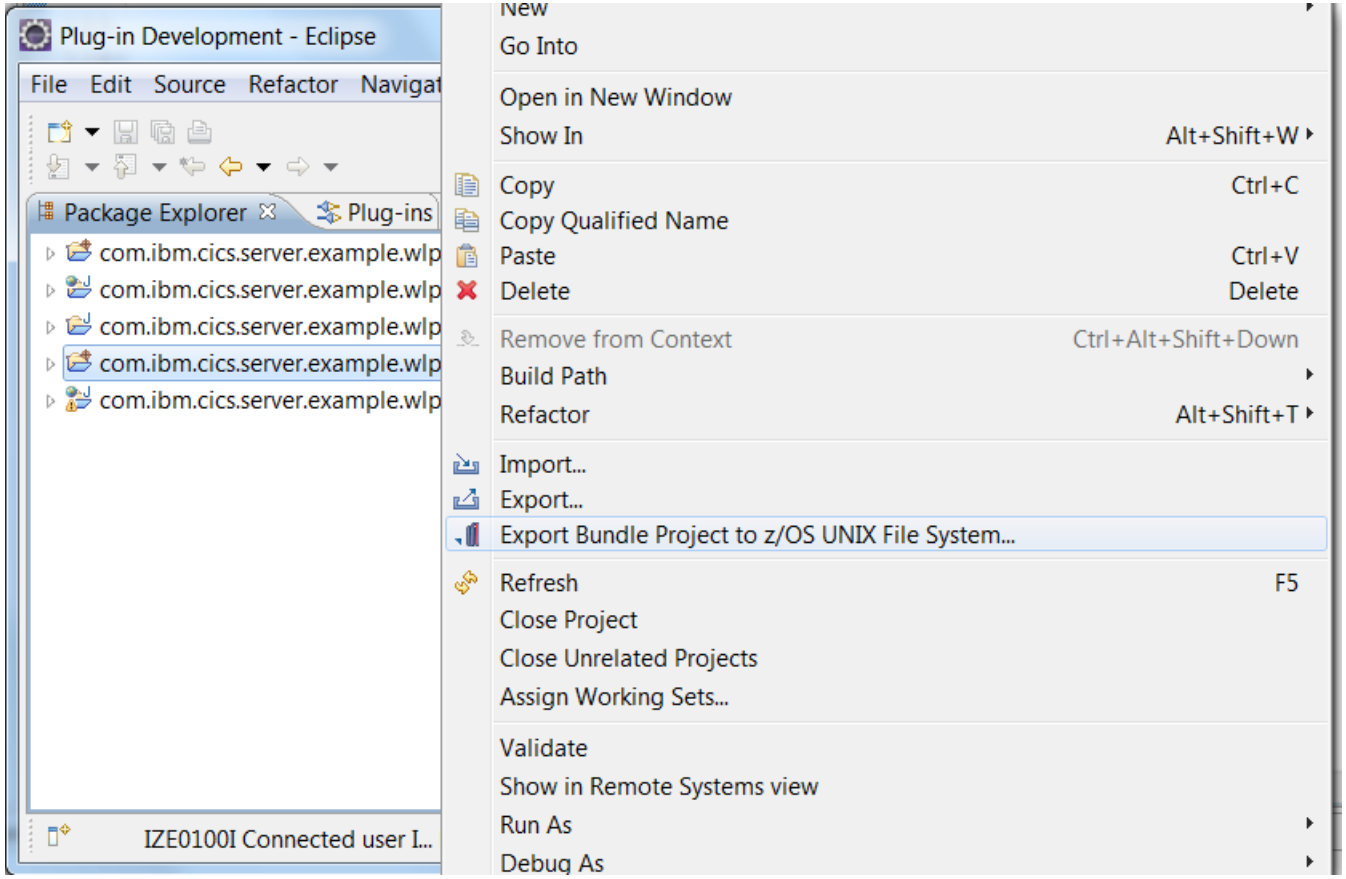


# JSP/Servlets plus JCICS/JDBC/Cobol

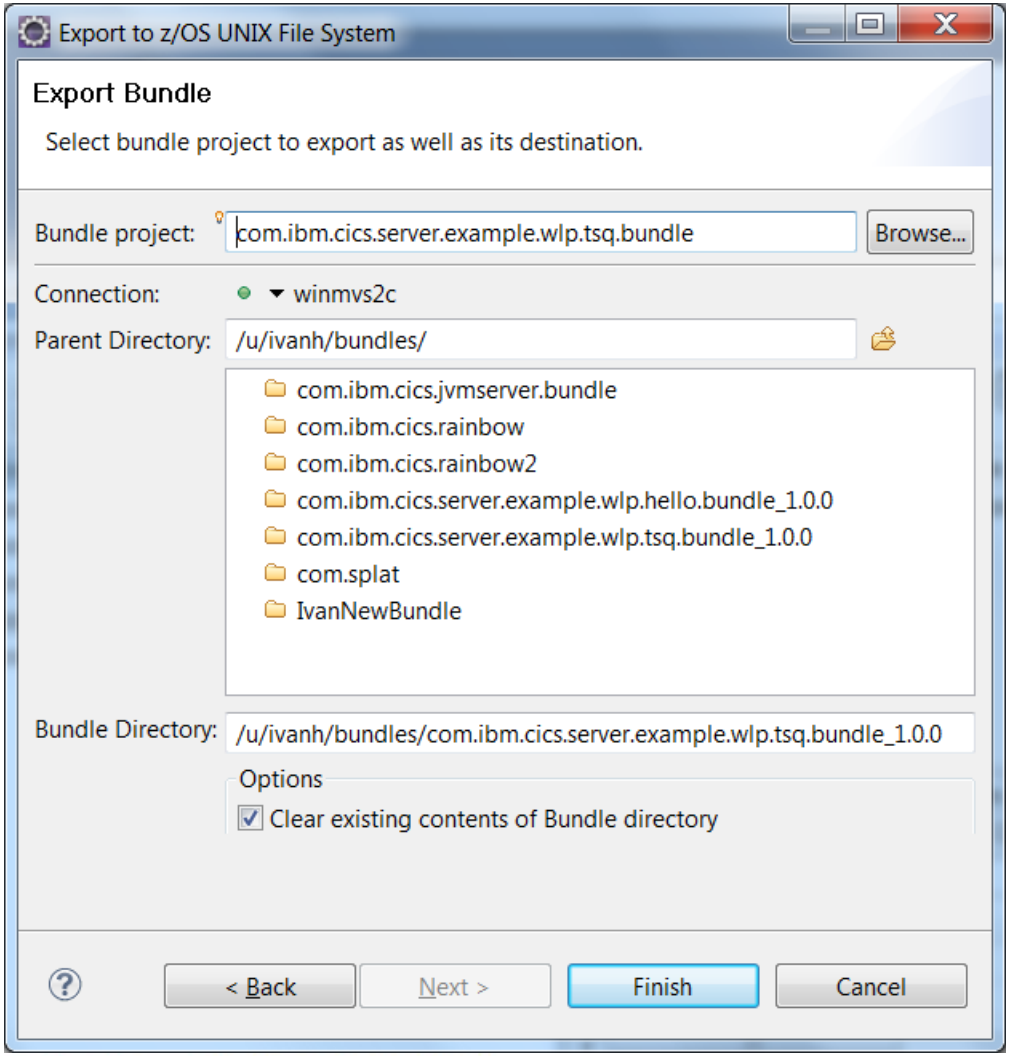




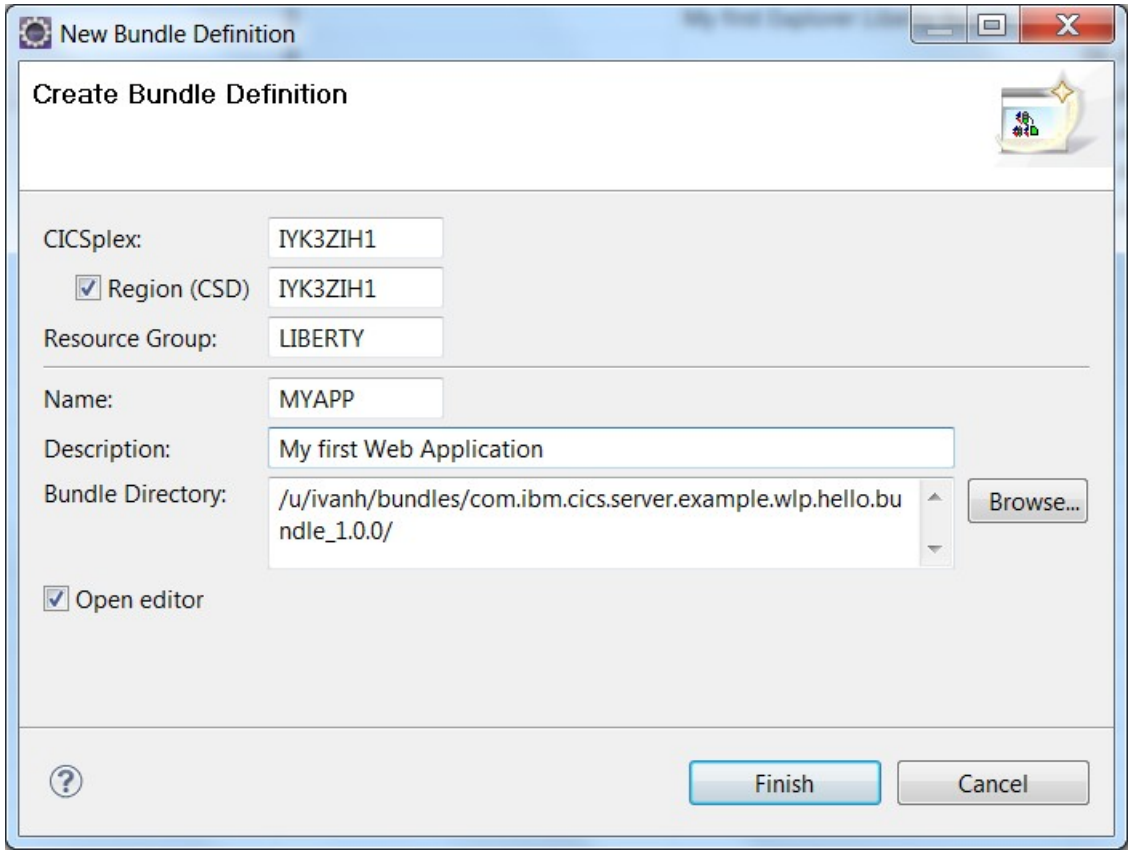
# Export the CICS bundle project



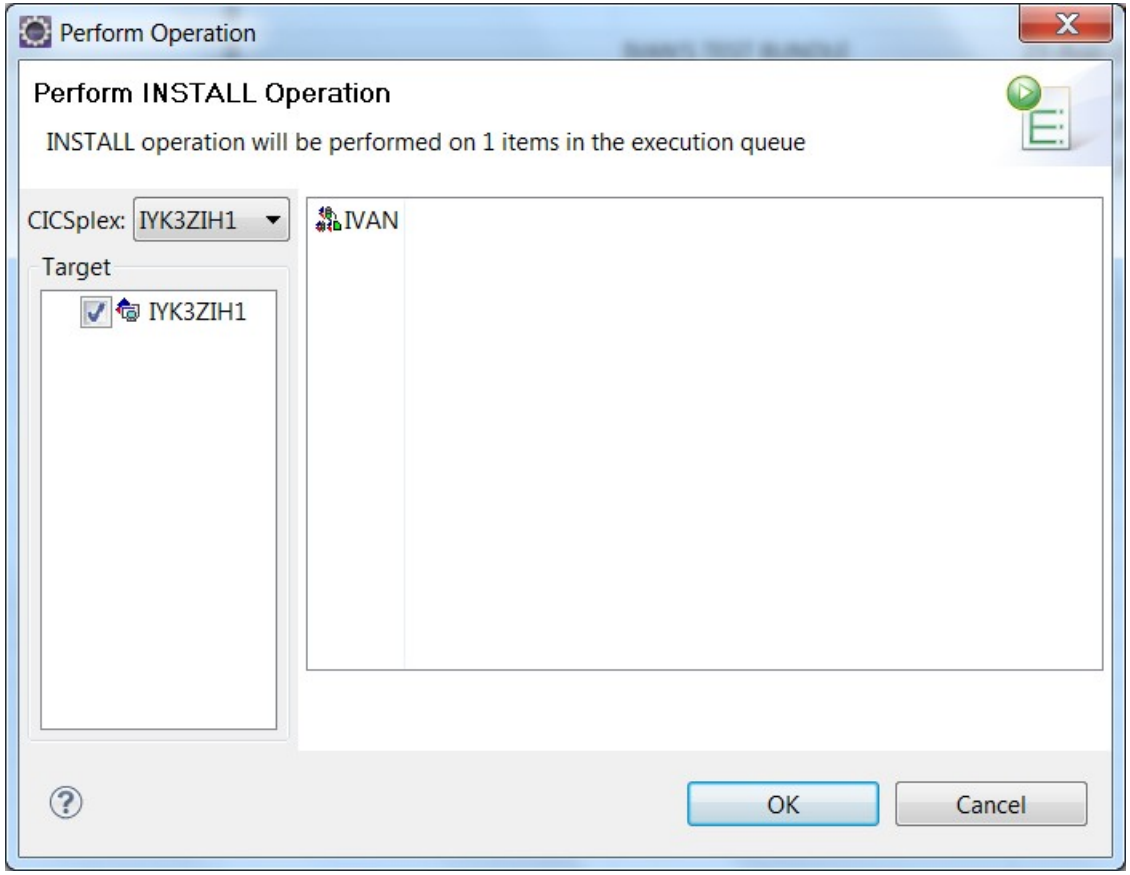
# Pick a zFS location for the CICS bundle project



# Create a CICS bundle definition to control the life-cycle of the Application



# Install the CICS bundle definition

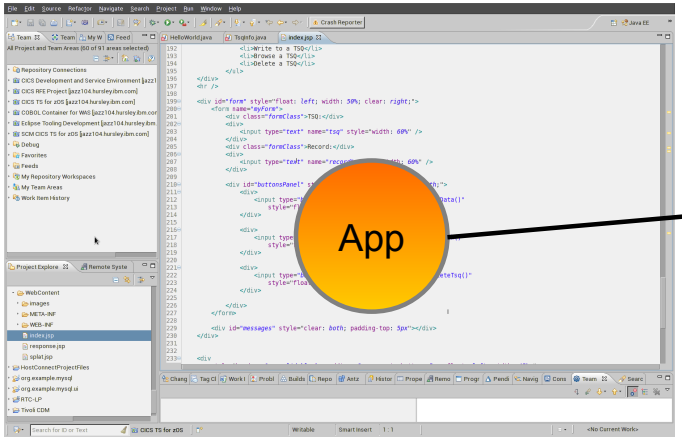


# Run the application!

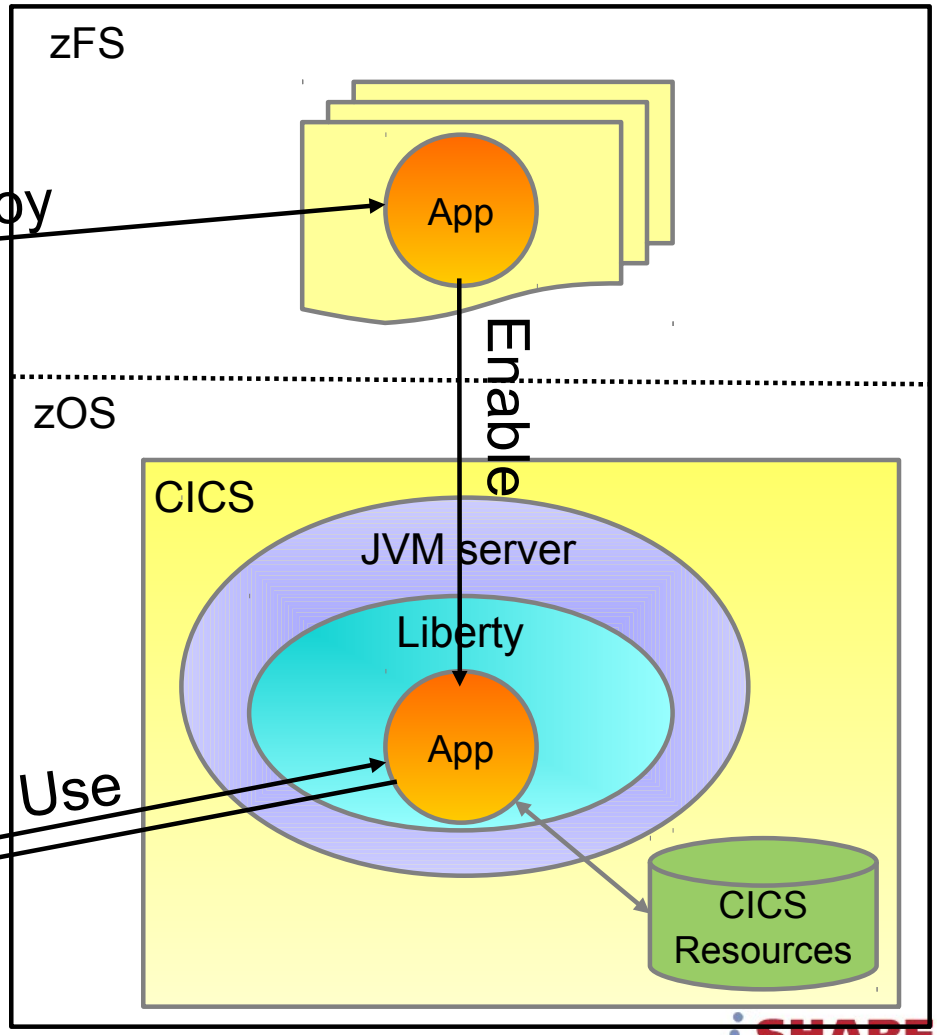
[AUDIT ] CWWKT0016I: Web application available (default_host):
http://winmvs2c.hursley.ibm.com:27245/com.ibm.cics.server.example.wlp.tsq.web/

# Putting it all together

Eclipse with Liberty Tools







# The Technology

# Principles

As little customization as we can get away with.

- Do things the Liberty way first, and if appropriate, only the Liberty way.
- Ensure Server.xml can be configured dynamically by the user.
- Support Liberty monitored drop-ins directory for applications.

Provide CICS enhancements only where absolutely necessary (Security, Tasks, JDBC, MQ)

Provide End-to-end Development and Deployment experience to enable non-mainframe professionals to develop for CICS.

Fully compatible with existing CICS OSGi Java applications running within the same JVM server.

# Specifications and Standards

Java 7 (64-bit)

Equinox 3.7 as the OSGi framework.

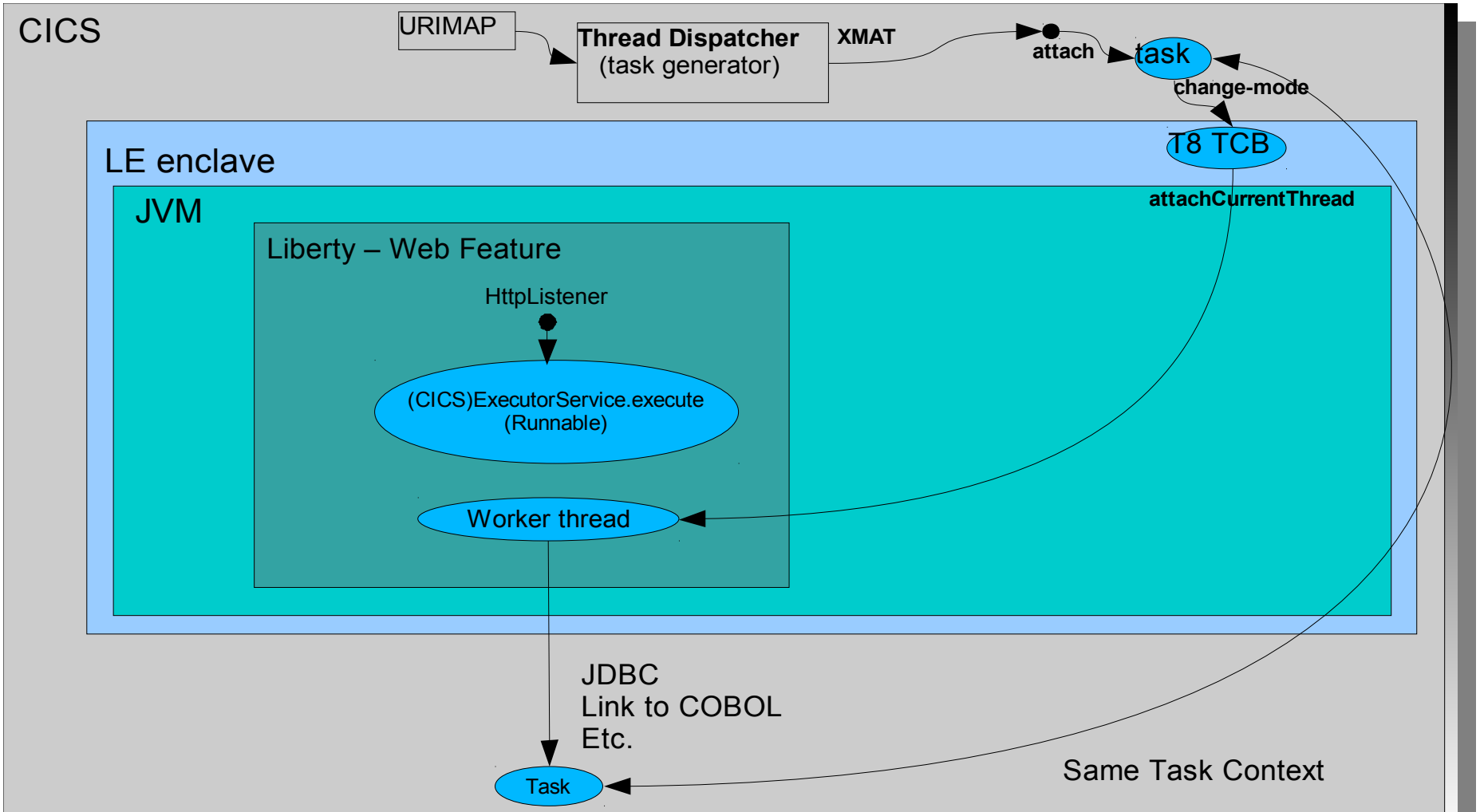
Implements the OSGi R4.3 specification

WAS Liberty Profile 8.5.0

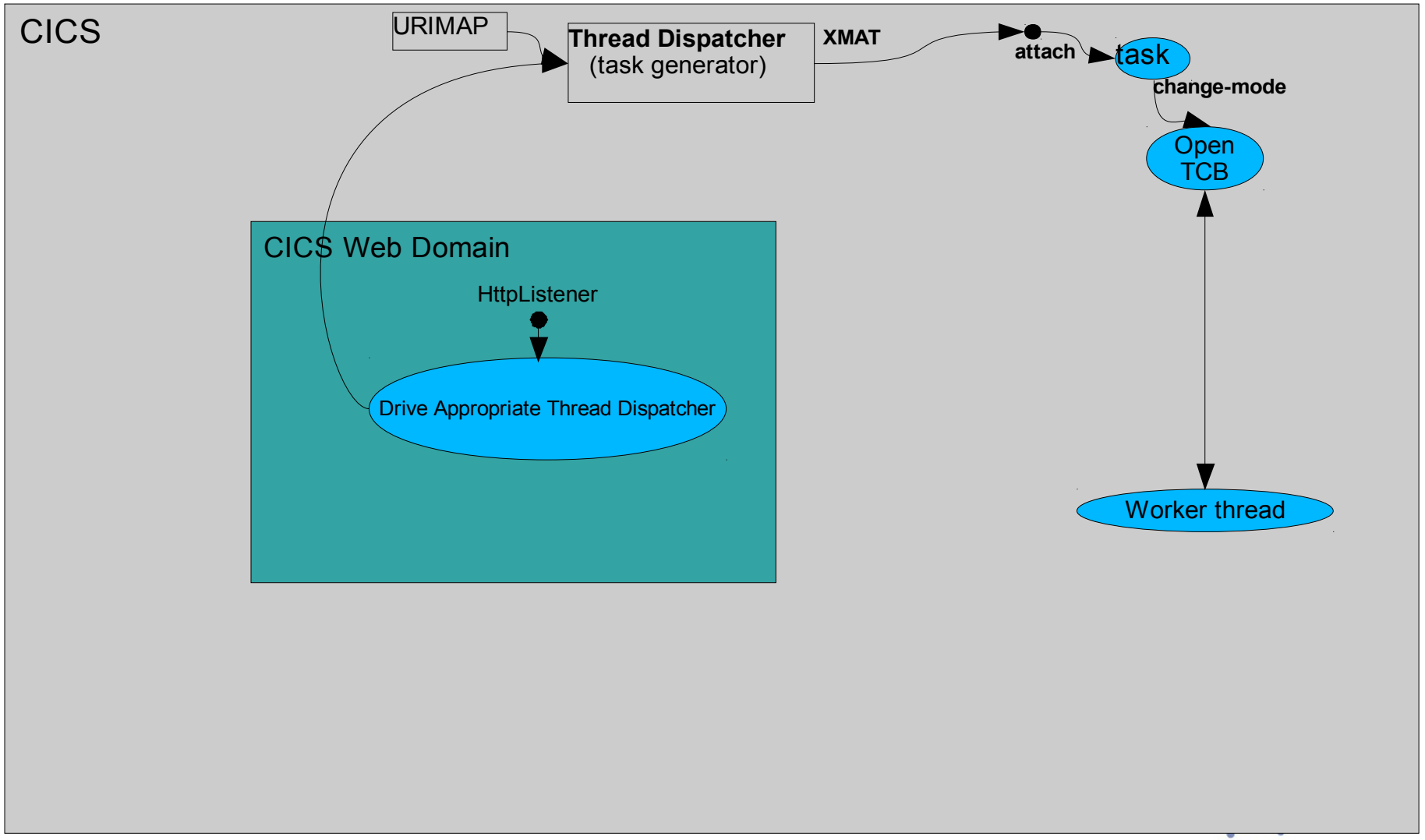
IBM CICS SDK for WebSphere Application Server Liberty profile v5.1

Eclipse 3.6.2

# Hybrid Threads



# 'Standard' CICS Listener Pattern



# JCICS – ExecutorService

- Thread.start equivalent (from Java concurrency package)
- A standard Java pattern for dispatching runnable code to threads.
- CICS provides “CICSExecutorService” - to create CICS capable threads.
- CICSExecutorService registered with OSGi registry, can be obtained and used by 'vendor' products and applications.
- A convenience method provided called “CICSExecutorService.runAsCICS()”
- Liberty requests an ExecutorService from the OSGi service registry.When running in CICS JVM server, it is given the CICSExecutorService which produces JCICS enabled threads for Liberty to run servlets on.

# Benefits of Hybrid Threads

- Each 'Invocation' (think Servlet Request) on a Hybrid Thread is also a CICS Transaction (Has a Tranid, Task Context etc).
- This gives you
  - A single common Transaction (UOW) and CICS Managed JDBC
    - Which can cross between Java and Cobol
  - Full JCICS API Access
    - In particular, LINK and access to VSAM
  - WLM (CICS WLM, Performance Classes etc).
  - Monitoring / Statistics
  - CICS Transaction Tracking / Association Data

# CICS Security with Liberty

Servlets run under default transaction CJSA with CICS

**SEC=YES** turns Security ON.

Basic-auth only (http or https) – Client cert not yet supported.

**Client Application**: Web.xml needs <security_constraint> to run with Security

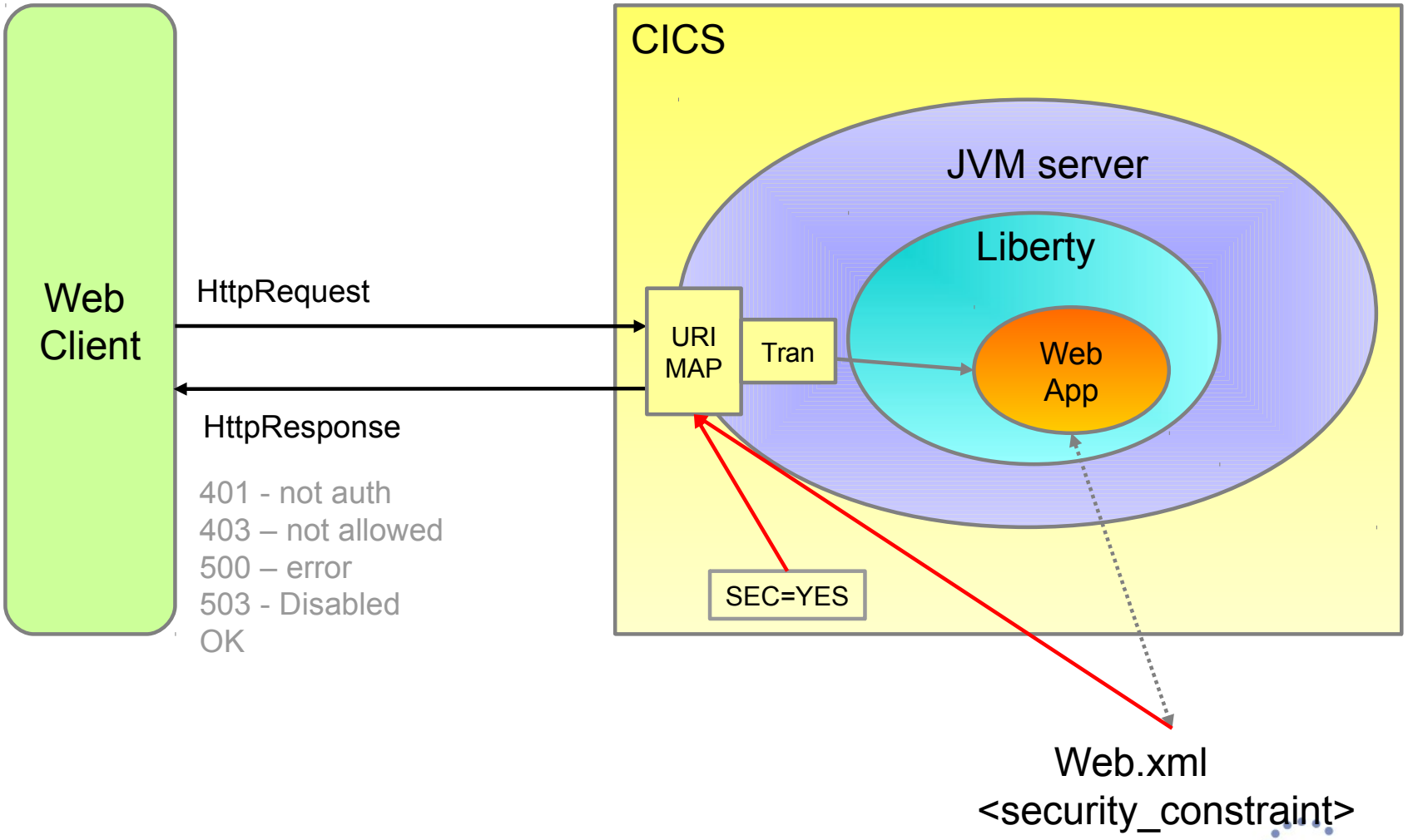
**Liberty**: Server.xml will be updated by CICS automatically

- <application-bnd>

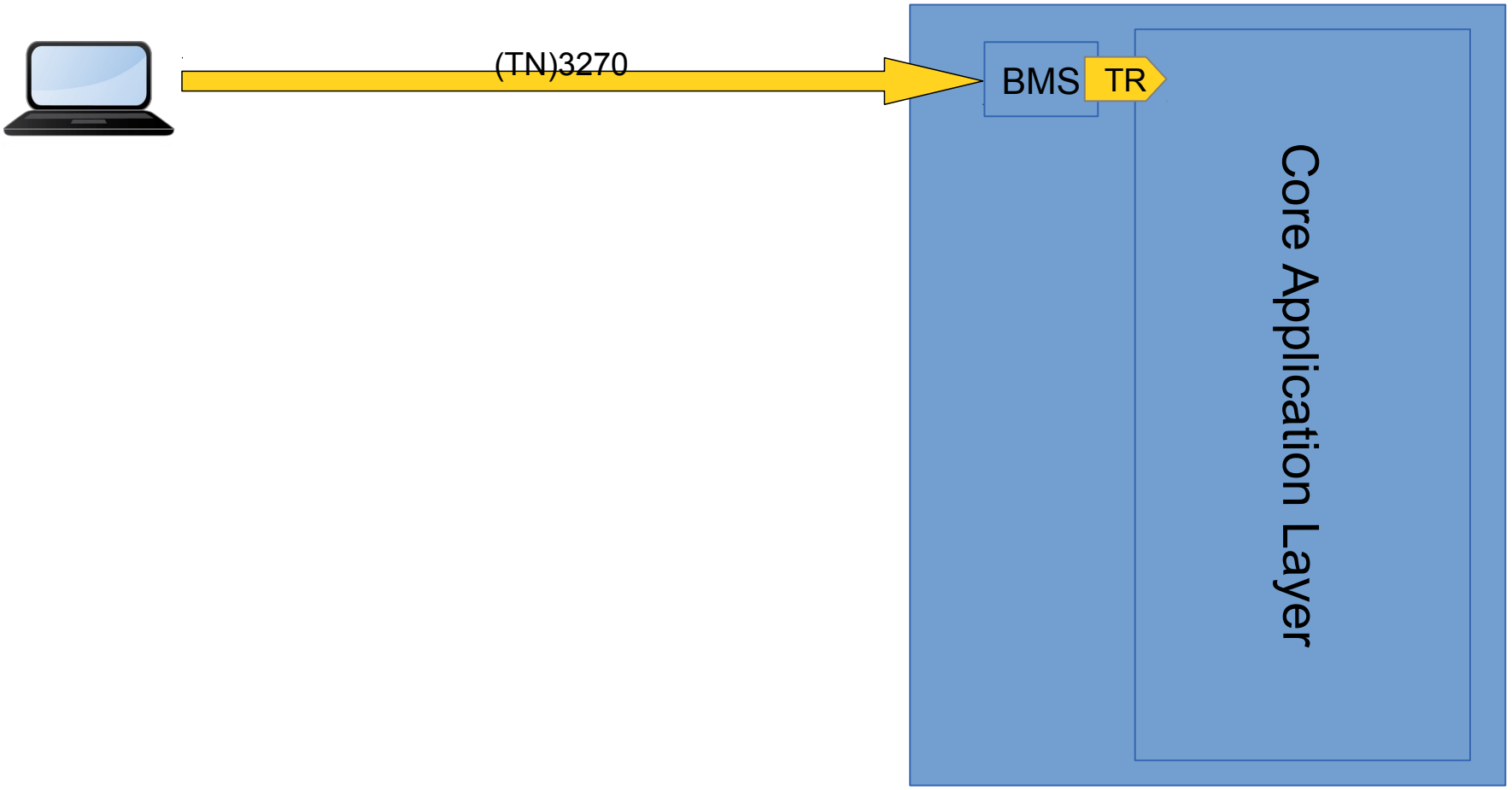
Role based Security not supported.

# URIMAP enhancements for Liberty

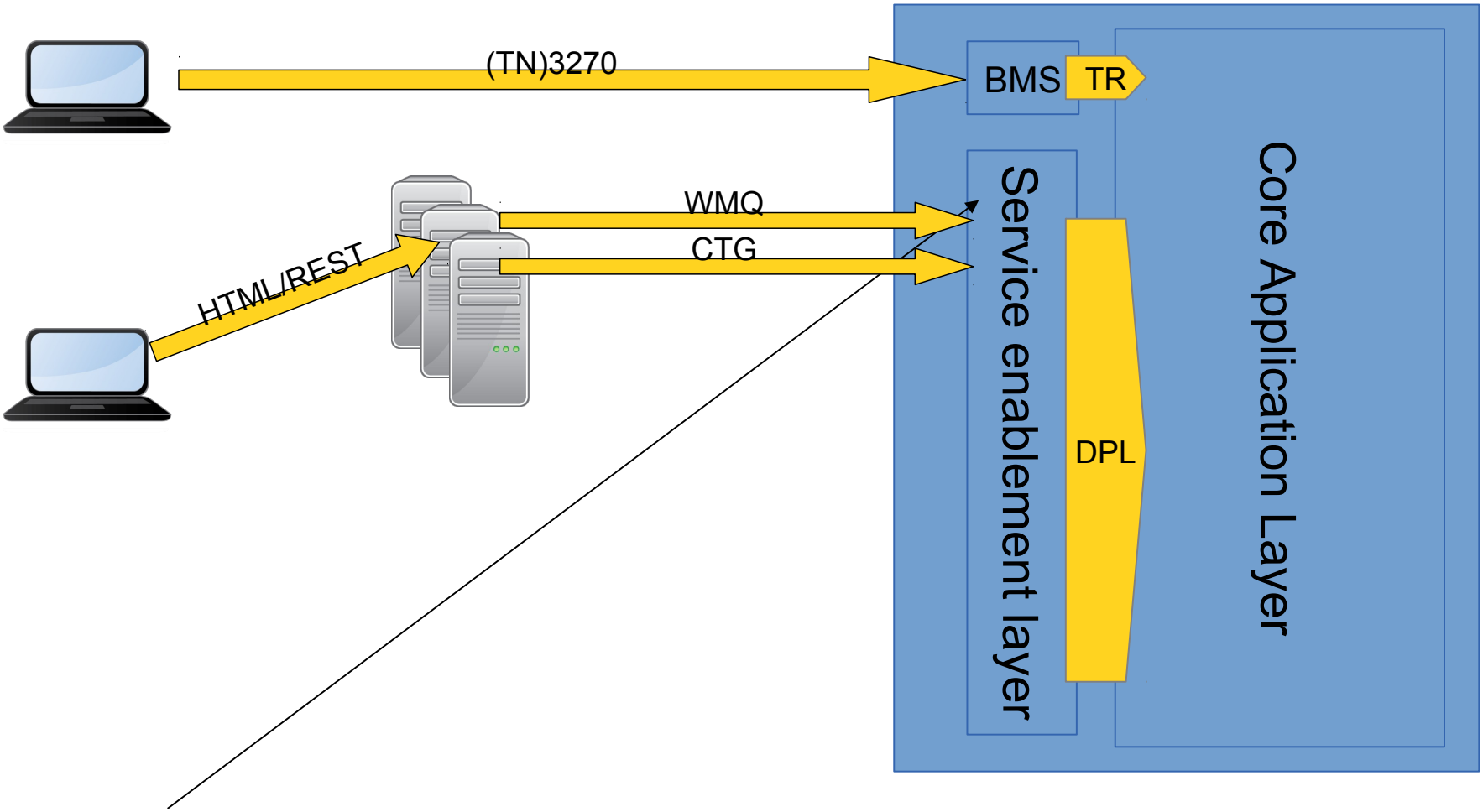
- URIMAP provides CICS authorisation via Transaction Security
- URIMAP allows context switch to a 'user' transaction
  - Transaction Security (URL mapped to transaction)
  - monitoring and audit purposes.
  - “Transaction class” support



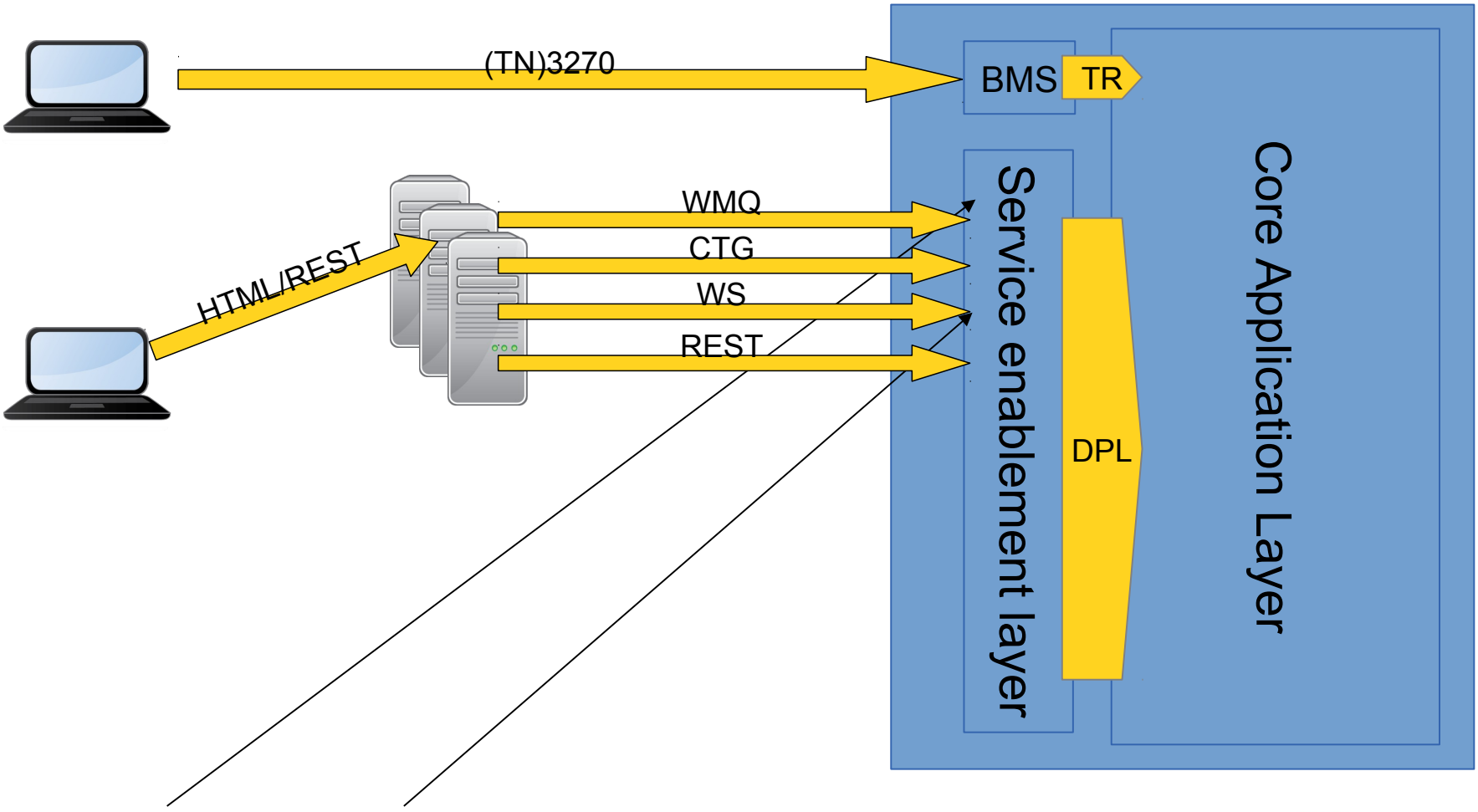
# Adding the Web container to the Connectivity landscape



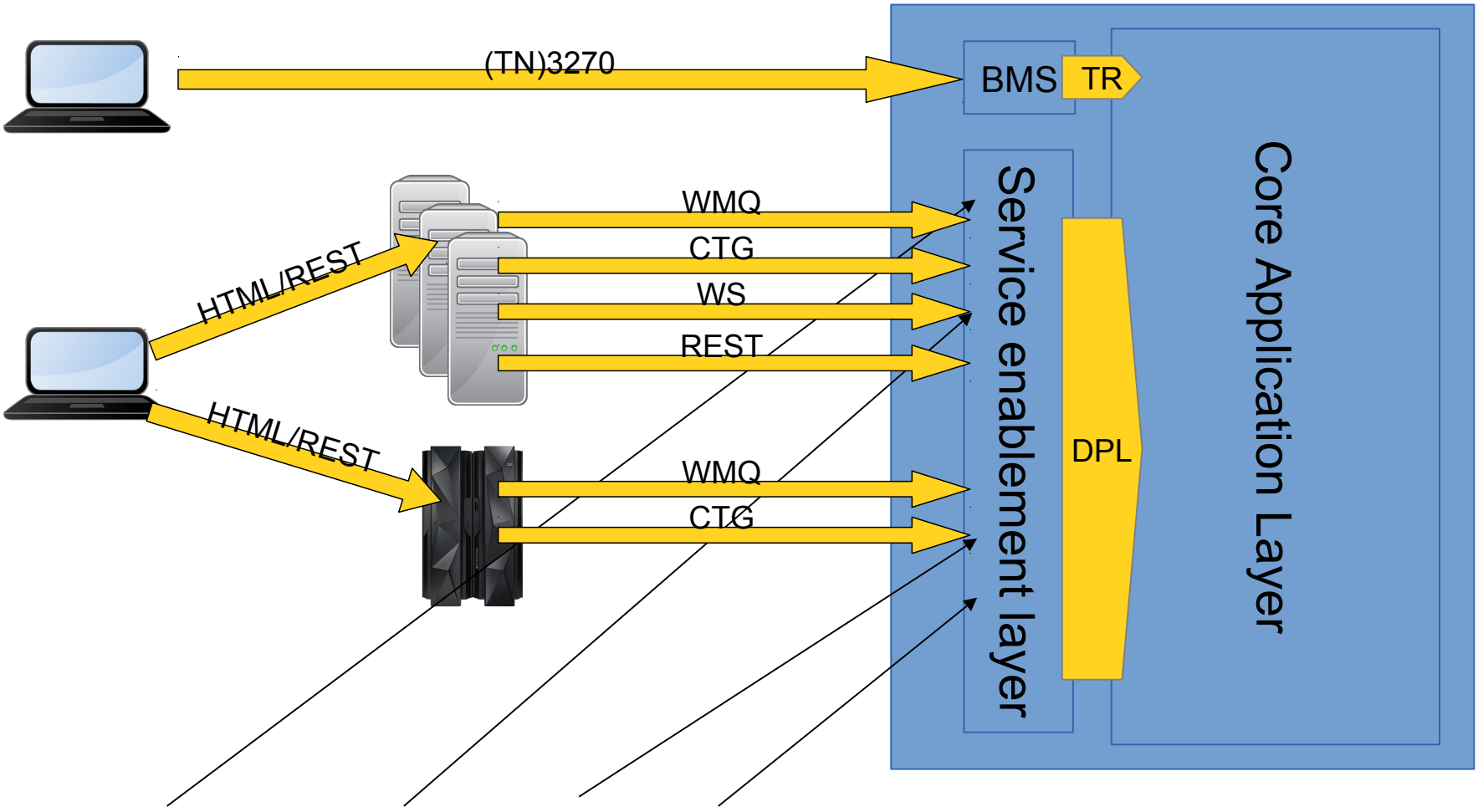
# Adding the Web container to the Connectivity landscape



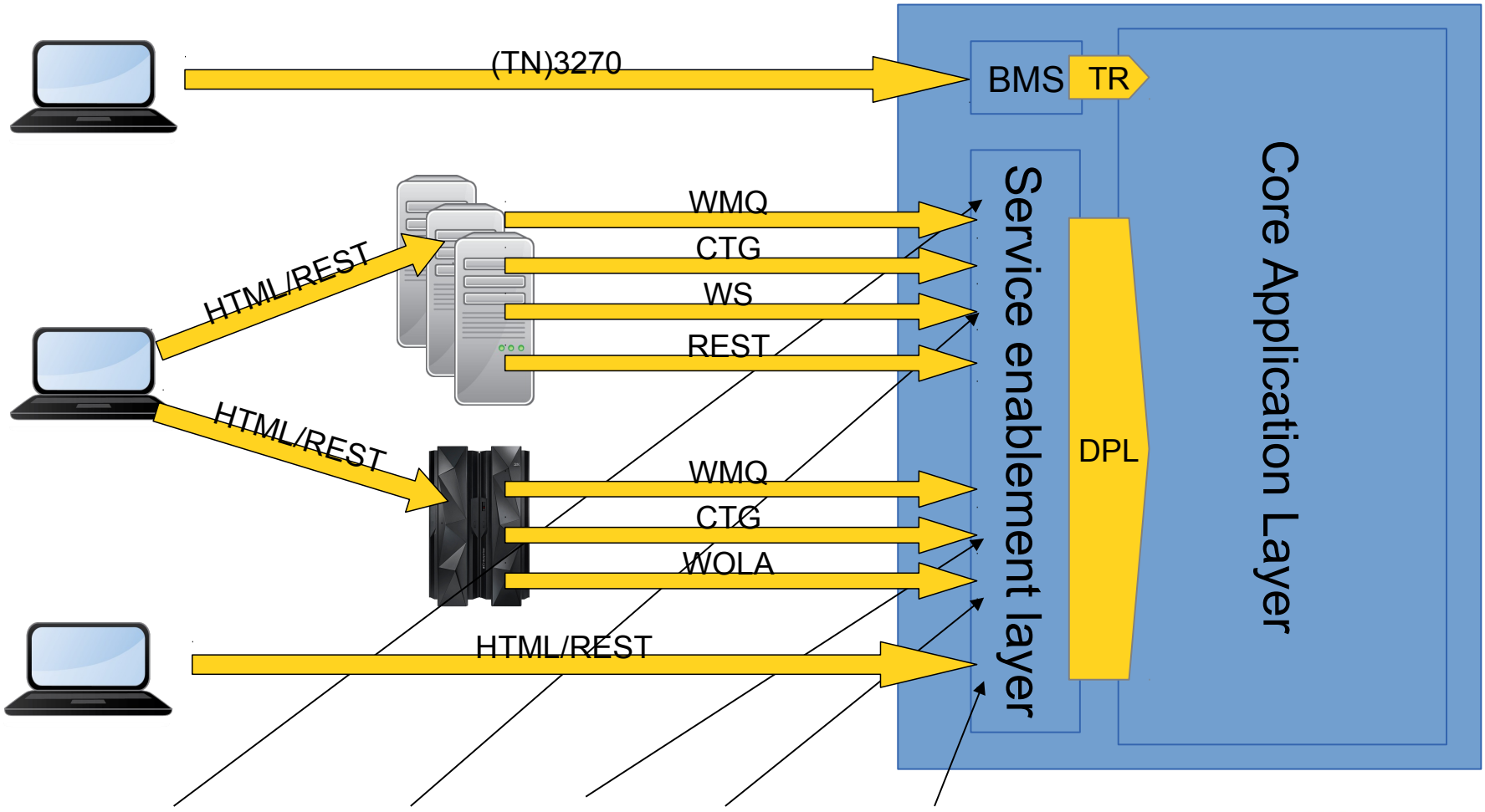
# Adding the Web container to the Connectivity landscape



# Adding the Web container to the Connectivity landscape



# Adding the Web container to the Connectivity landscape



# Part 4 – Summary and Future

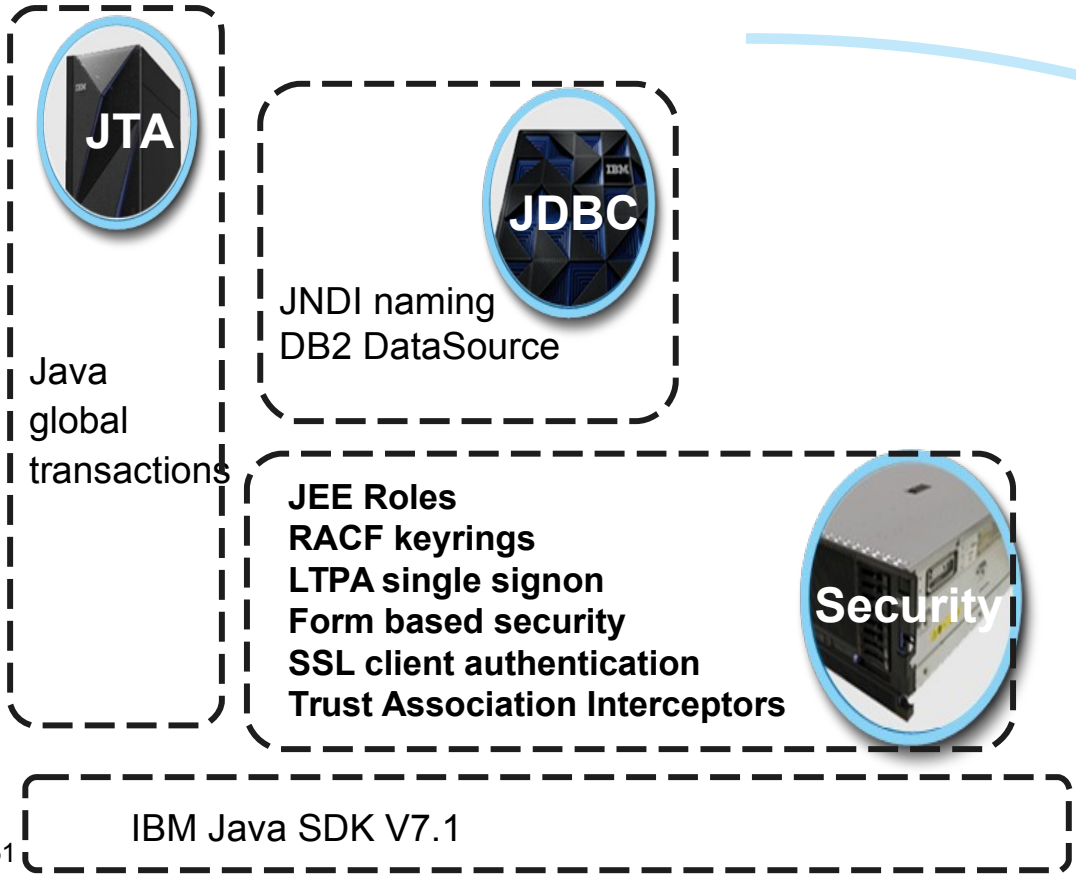
# Liberty Features (as of WAS 8.5.0)

- Bean validation
- Blueprint
- Java Database Connectivity (JDBC)
- Java Management Extensions (JMX)
- Java Persistence API (JPA)
- JavaServer Faces (JSF)
- JavaServer Pages (JSP)
- JAX-RS
- Secure Sockets Layer (SSL)
- Security, supported by either the basic user registry or a Lightweight Directory Access Protocol (LDAP) user registry
- Servlet
- Web application bundle (WAB)
- Web security
- zOS Security
- zOS Transactions

# Liberty Features ( for CICS TS V5.1 GA)

- Bean validation
- **Blueprint – via Service Stream**
- **Java Database Connectivity (JDBC)**
- Java Management Extensions (JMX)
- Java Persistence API (JPA)
- **JavaServer Faces (JSF)**
- **JavaServer Pages (JSP)**
- **JAX-RS, JSON – via Service Stream**
- **Secure Sockets Layer (SSL)**
- Security, supported by either the basic user registry or a Lightweight Directory Access Protocol (LDAP) user registry
- **Servlet**
- **Web application bundle (WAB) – via Service Stream**
- **Web security**

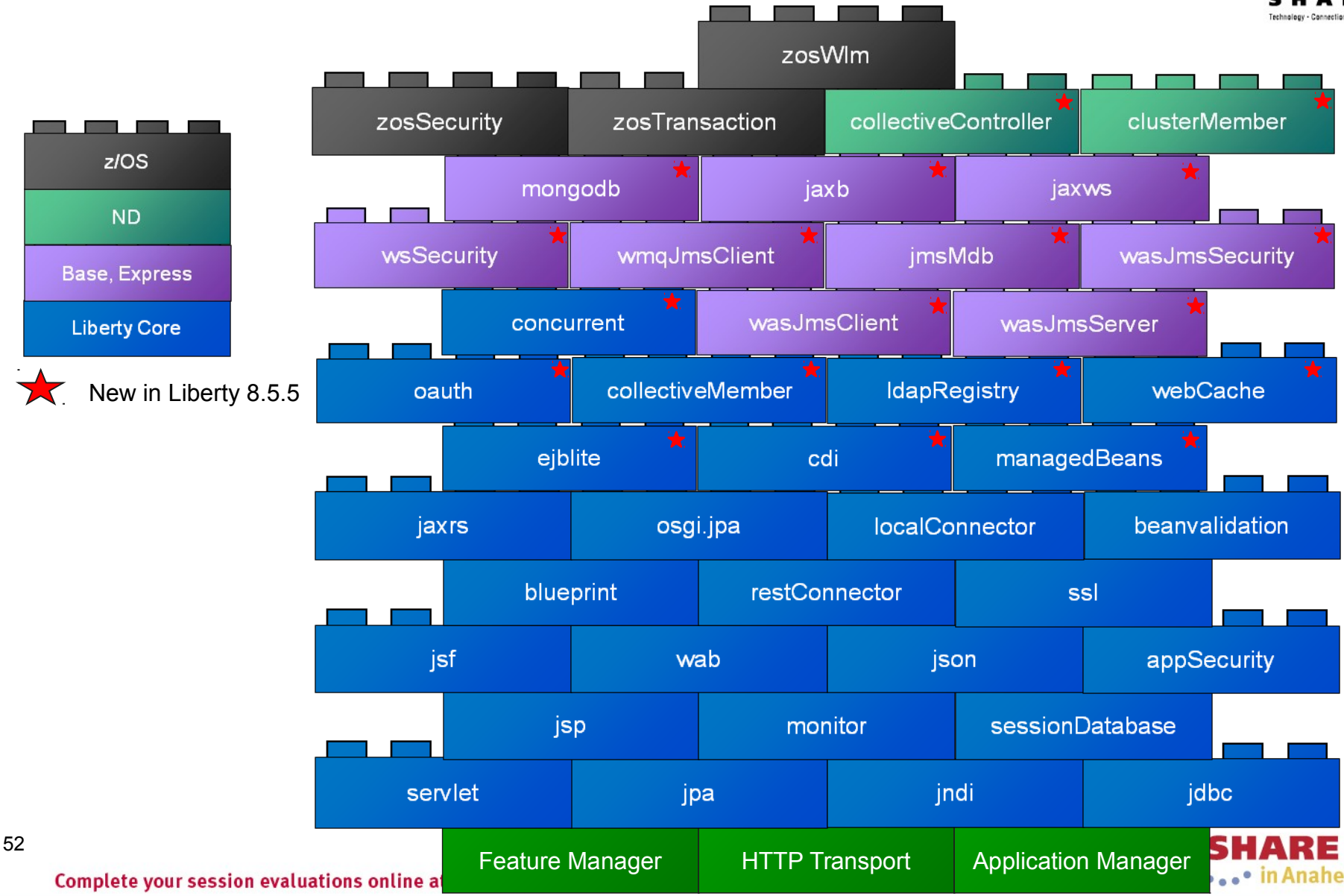
# CICS TS V5.2 beta – Liberty runtime extensions



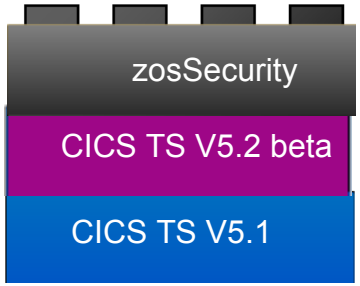

Liberty 8.5.5

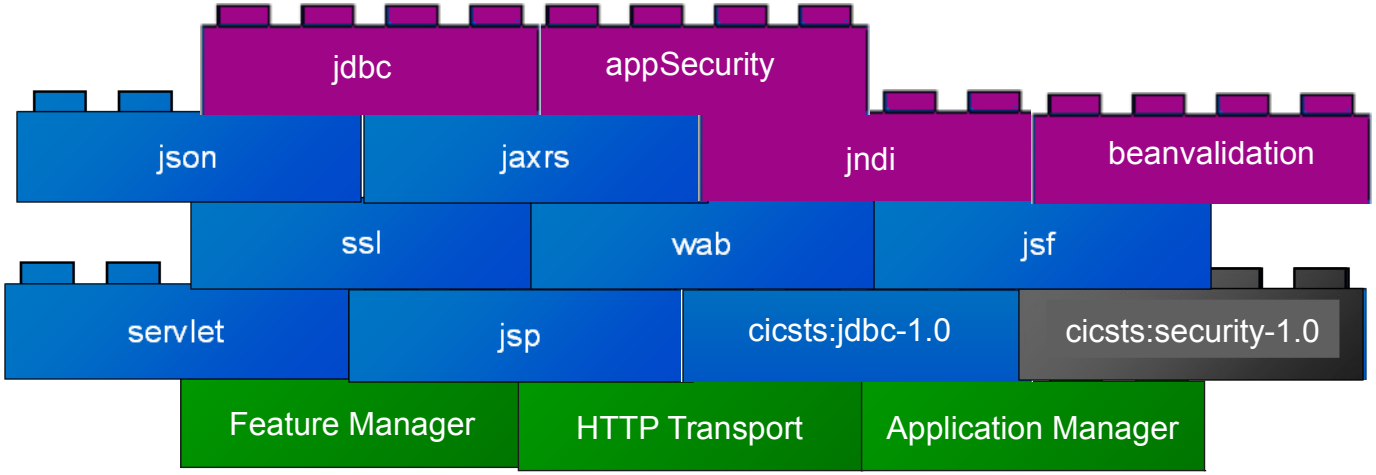
Integrated and optimized for CICS Web workloads

# WAS/Liberty feature set

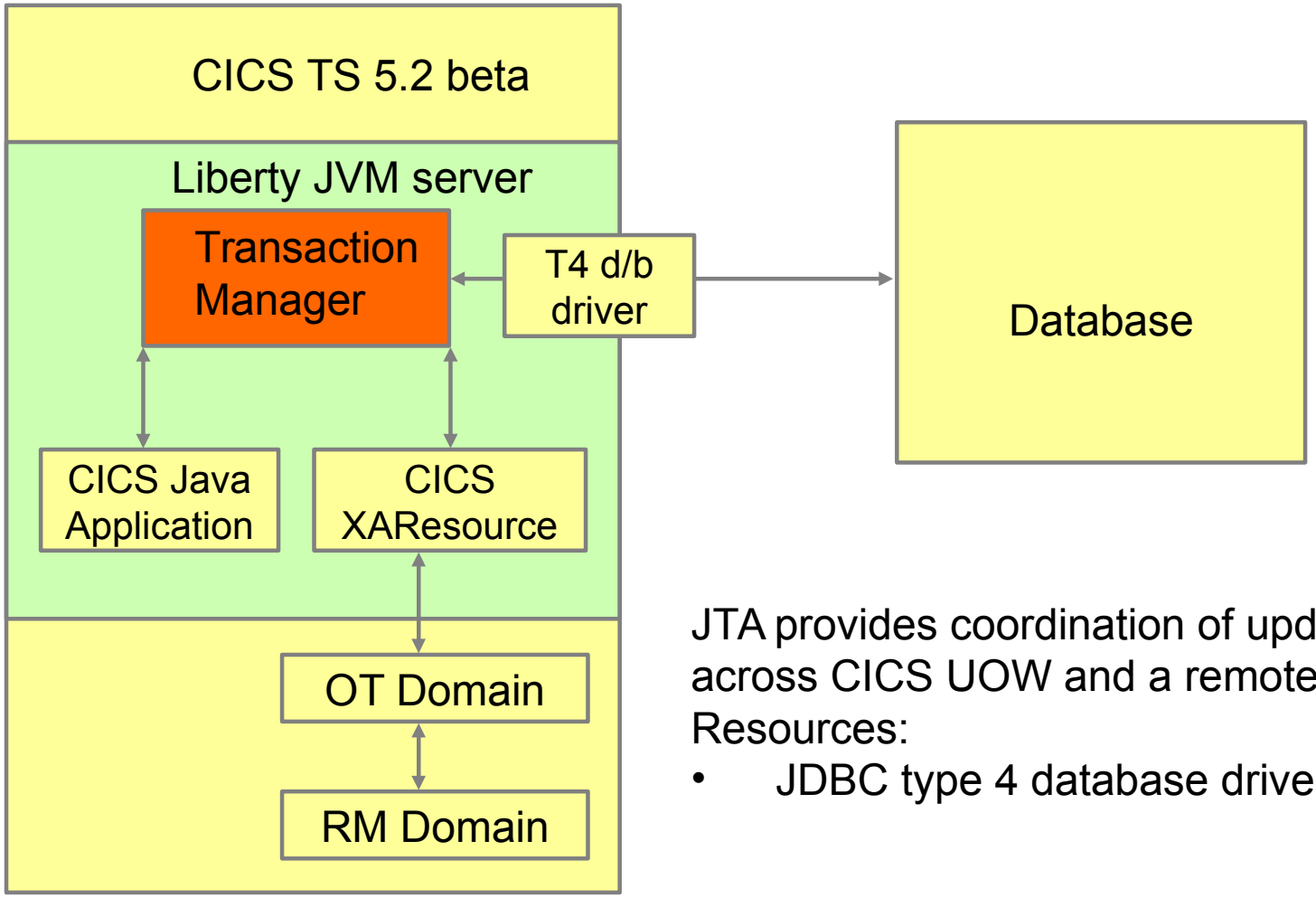


# CICS/Liberty feature set - CICS TS V5.2 Open Beta



# Java Transaction API (JTA)



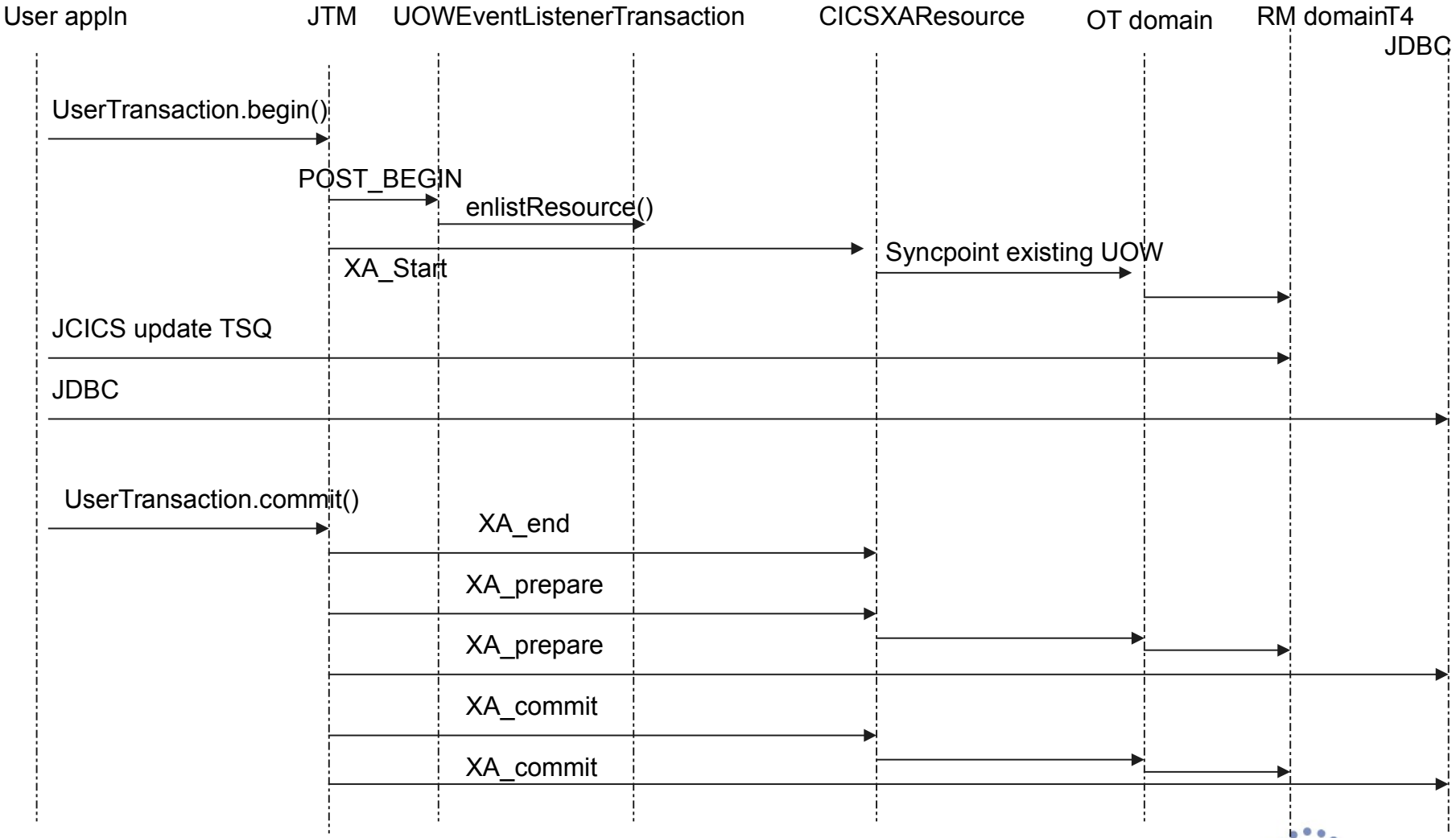

JTA provides coordination of updates across CICS UOW and a remote XA Resources:

- JDBC type 4 database driver

# JTA – UserTransaction

```
InitialContext ctx = new InitialContext();
UserTransaction tran =
   (UserTransaction)ctx.lookup("java:comp/UserTransaction");

DataSource ds = (DataSource)ctx.lookup("jdbc/SomeDB");
Connection con = ds.getConnection();

// Start the User Transaction
tran.begin();

// Perform updates to CICS resources via JCICS API
// Access DataSource

if (allOk) {
  // Commit updates on both systems
  tran.commit();
} else {
  // Backout updates on both systems
  tran.rollback();
}
```

# JTA commit() with CICS updates and additional RM



# CICS TS V5.2 – Liberty Security

- RACF keyrings for SSL
- AppSecurity
  - Authentication
    - Basic authentication
    - Form logon
    - TAI/ JAAS
    - SSL client authentication
    - Customer user registry
  - Authorisation
    - JEE roles
    - EJB roles
    - CICS Transaction/Resource security
    - SynctoOSthread – USS security

# Liberty JVM server security

- WLP Angel process used to access authorized services
  - SAF password authentication
  - SAF role authorization (EJBROLEs)

- All WLP application security options supported in CICS TS V5.2 Open Beta
  - LTPA/SSO provides optimized performance for authentication
  - TAI/JAAS modules can be used to customize authentication

# Liberty JVM server - URIMAP

- Liberty 'threads' run under default transaction 'CJSA'
- URIMAP extended for type 'JVMSERVER'
  - Allows setting of a User transaction other than CJSA
    - Integrates with CICS Transaction Security
    - Used for monitoring and audit purposes
    - Transaction class (TCLASS) support for scheduling
- CICS Task userid determined as follows:
  - Protected URI, use USERID from Http request (basic-auth header)
  - Unprotected URI, use USERID value from URIMAP
  - No URIMAP, or no USERID in URIMAP, use CICS DFLTUSER

# CICS Security with Liberty Profile

- V5.1 - Security credentials provided by basic-auth (http or https) or URIMAP.
- V5.2 Open Beta – Form logon, SSL client cert mapping, TAI

- CICS transaction security pre-requisites:
  - SEC=YES (triggers JVM server to add 'CICS' UserRegistry to Liberty)
  - Optional use of URIMAP to switch to 'user transaction'
  - <application-bnd> in server.xml must be present (CICS bundle deployed apps always get this added when SEC=YES)
  - <security-constraint> in Client application “web.xml” must be present ( Explorer SDK provides emamples/template)

# Application:
# Web.xml – Example <security-constraint>

```
<security-constraint>
    <display-name>examples.web_SecurityConstraint</display-name>
    <web-resource-collection>
        <web-resource-name>examples.web</web-resource-name>
        <description>Protection area for examples.web</description>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <description>All Authenticated users of examples.web</description>
        <role-name>cicsAllAuthenticated</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Must match application-bnd on Server

Best practice to force HTTPS

# Server.xml – Example <application-bnd>

```
<server>
    <!-- Include file for CICS bundle installed applications -->
    <application id="examples.web" name="examples.web" type="war"
                 location="${server.config.dir}/installedApps/examples.web.war">
        <application-bnd>
            <security-role name="cicsAllAuthenticated">
                <special-subject type="ALL_AUTHENTICATED_USERS"/>
            </security-role>
        </application-bnd>
    </application>
</server>
```

Must match Web.xml role

# Summary of Key Benefits

**Local. Lightweight. Fast.** Web Applications run locally in CICS with direct access to CICS data and resources. No adapters, no converters, same address space.

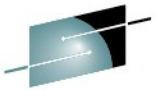
**Standard tools for developers.** Familiar, industry standard tools with Eclipse and Dynamic Web Projects. CICS Explorer SDK enhances the deployment experience.

**Portable.** Presentation logic in Servlets, business logic in OSGi bundles. Servlets are portable across runtimes. Bundles provide componentization.

**Modular design**. Architected in a modular way using OSGi, the server only enables and starts the features required by the applications and configuration. If you're not using a feature, it won't start in your server runtime

**Dynamic runtime**. Features can be added to the server dynamically, using the OSGi framework, while the server is running, with zero downtime and server restarts. Similarly server and application config can be updated without the need to restart.

**Eclipse based tools**. The eclipse tools for the Liberty Profile are small and very well integrated with the Liberty Profile environment




धन्यवाद
Hindi

多謝
Traditional Chinese

ขอบคุณ
Thai

Спасибо
Russian

Thank You
English

Bedankt
Nederlands

شكراً
Arabic

Merci
French

Obrigado
Brazilian Portuguese

Gracias!
Spanish

多谢
Simplified Chinese

Danke
German

நன்றி
Tamil

ありがとうございました
Japanese

감사합니다