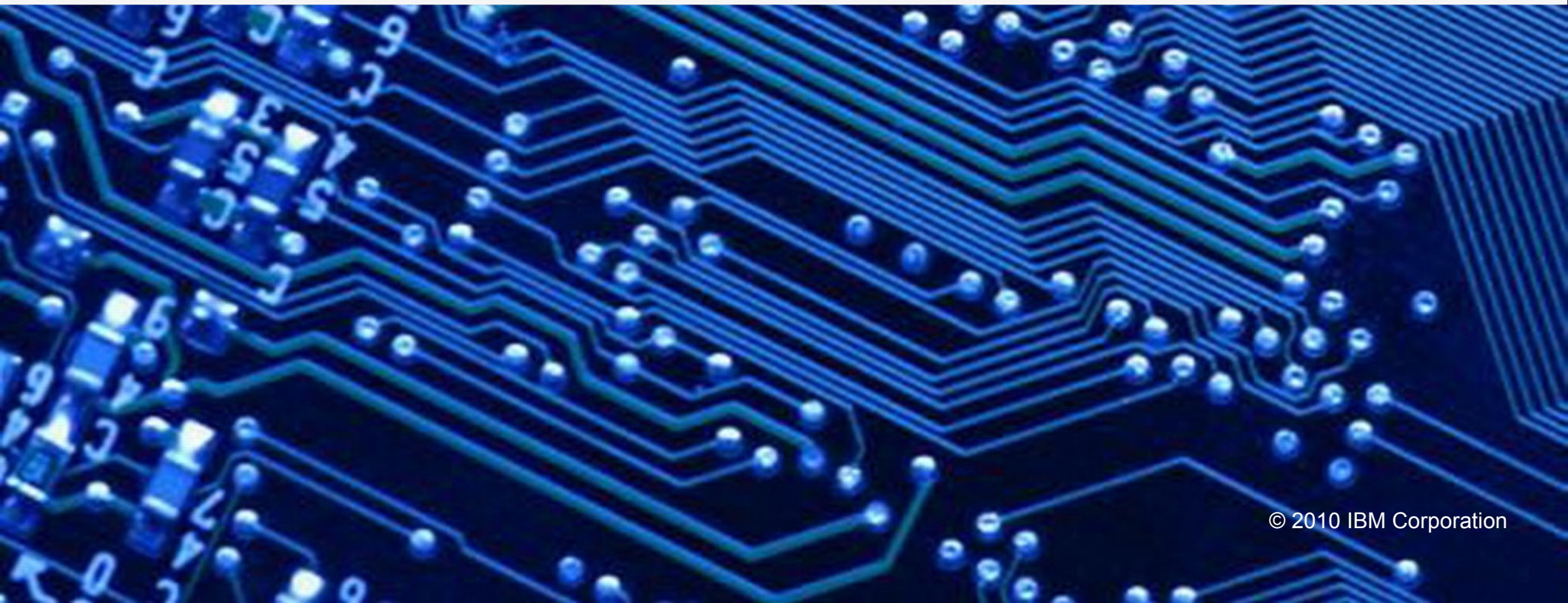
WebSphere Application Server for z/OS

I'm Not a Dummy But ...



Other Sessions

Room	Day	Time	Title	Speaker
312	Monday	12:15	Lab	Stephen
203	Monday	4:30	WebSphere: What's New?	Follis
203	Wednesday	9:30	WebSphere 101	Houde / Stephen
201	Wednesday	1:30	Introduction to IBM Support Assistant (ISA)	Hutchinson
200	Wednesday	3:00	WebSphere Process Manager and Business Process Manager Configuration	Hutchinson
200	Wednesday	4:30	OSGi/JPA/Batch Feature Packs	Follis / Bagwell
203	Wednesday	6:00	WebSphere for z/OS: I'm no longer a dummy but..	Bagwell
310	Thursday	8:00	WOLA Application Designs	Bagwell
310	Thursday	9:30	Security Architecture: How Does WebSphere Play?	O'Donnell
310	Thursday	11:00	WAS on z/OS High Availability Considerations	Bagwell
200	Thursday	12:15	Staged Application Development in a WebSphere ND Cluster	Loos
310	Thursday	1:30	WAS on z/OS and WLM Interactions	Follis

Agenda ...

Objectives and Agenda

Objective:

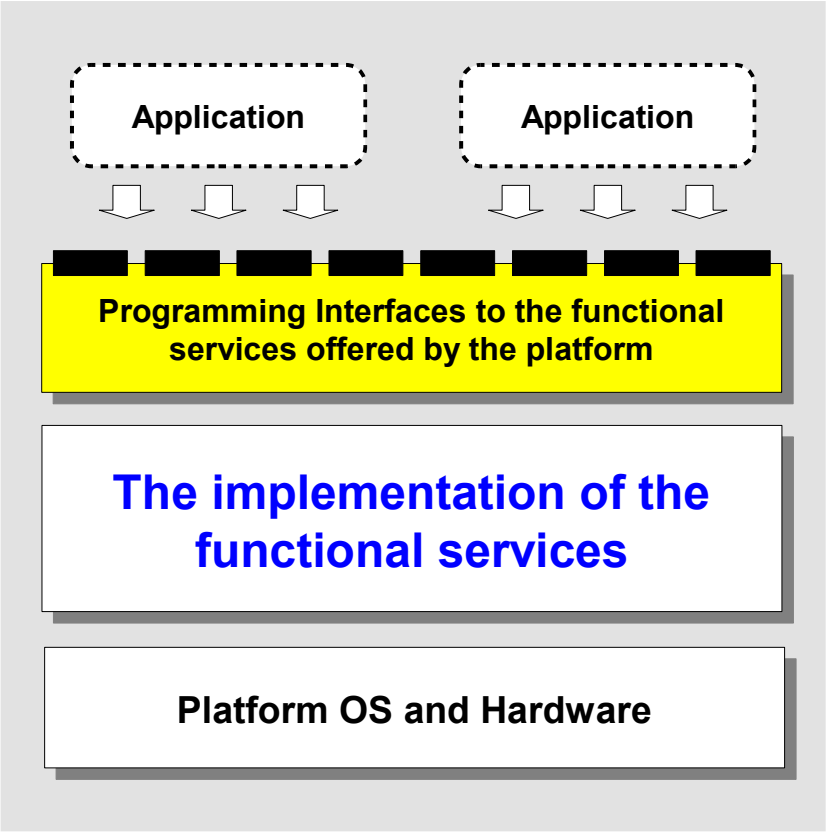
To extend your understanding of WAS z/OS to include things you might not have otherwise considered

Agenda:

- **Baseline level-set ... a few key points**
- **Exploring the application space**
- **Exploring the "zDiff" items**

An "Application Server"

The whole point of an "application server" is to provide functional services so the developers who write the business logic do not have to:



- Security
- Transaction
- Persistence
- Servlet
- JSP
- EJB
- Web Services
- etc.

Not a new concept

- CICS is an application server
- IMS is an application server
- The iPhone® is an application server!

WAS is all about open standards

- Java and related specifications
- InfoCenter search: `rovr_spec`

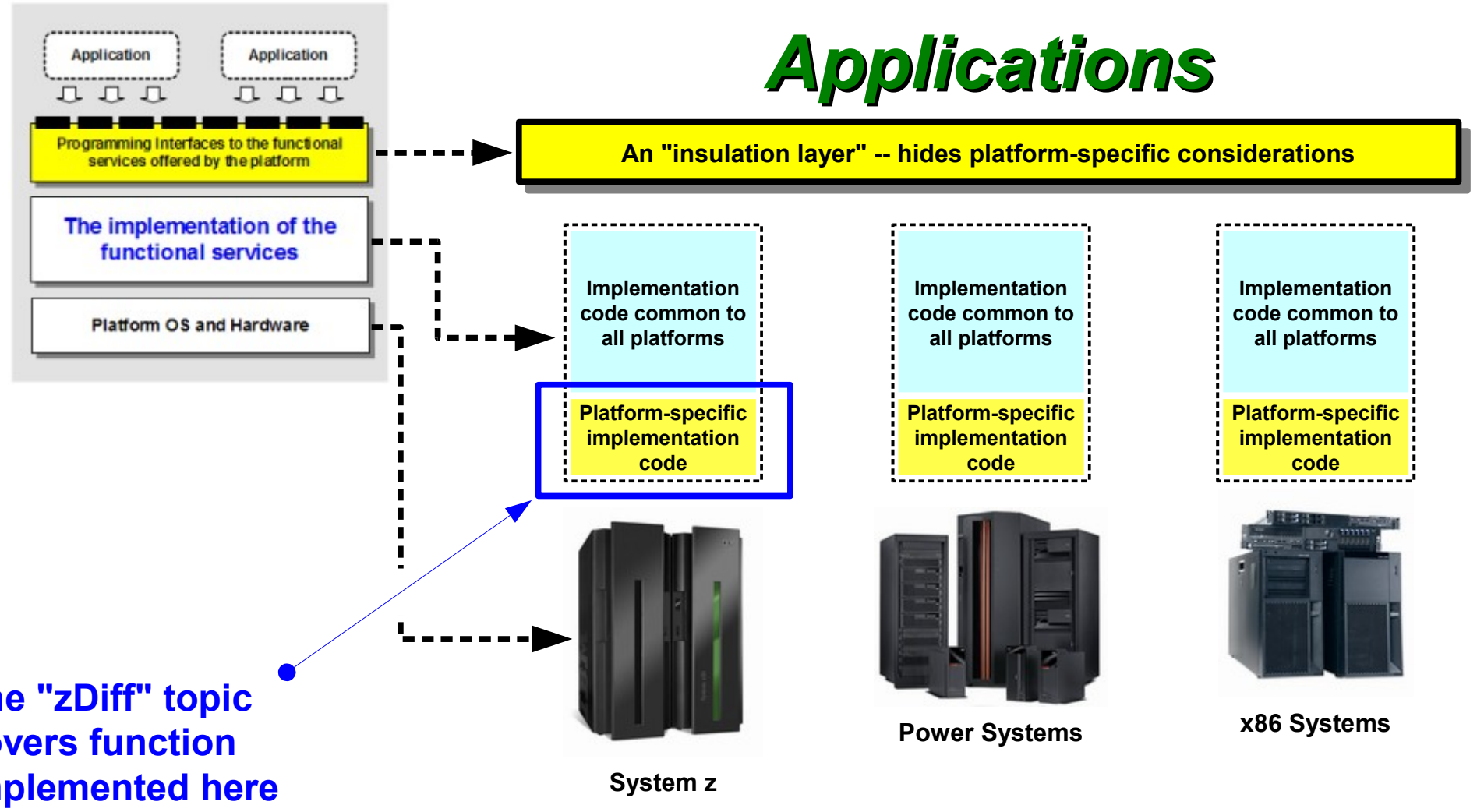
Keep this in mind as we go through this material

It's all about providing support functionality to the application

The application represents the business function, which means that's the end people *really* care about

"WAS is WAS" ... *Above the Spec Line*

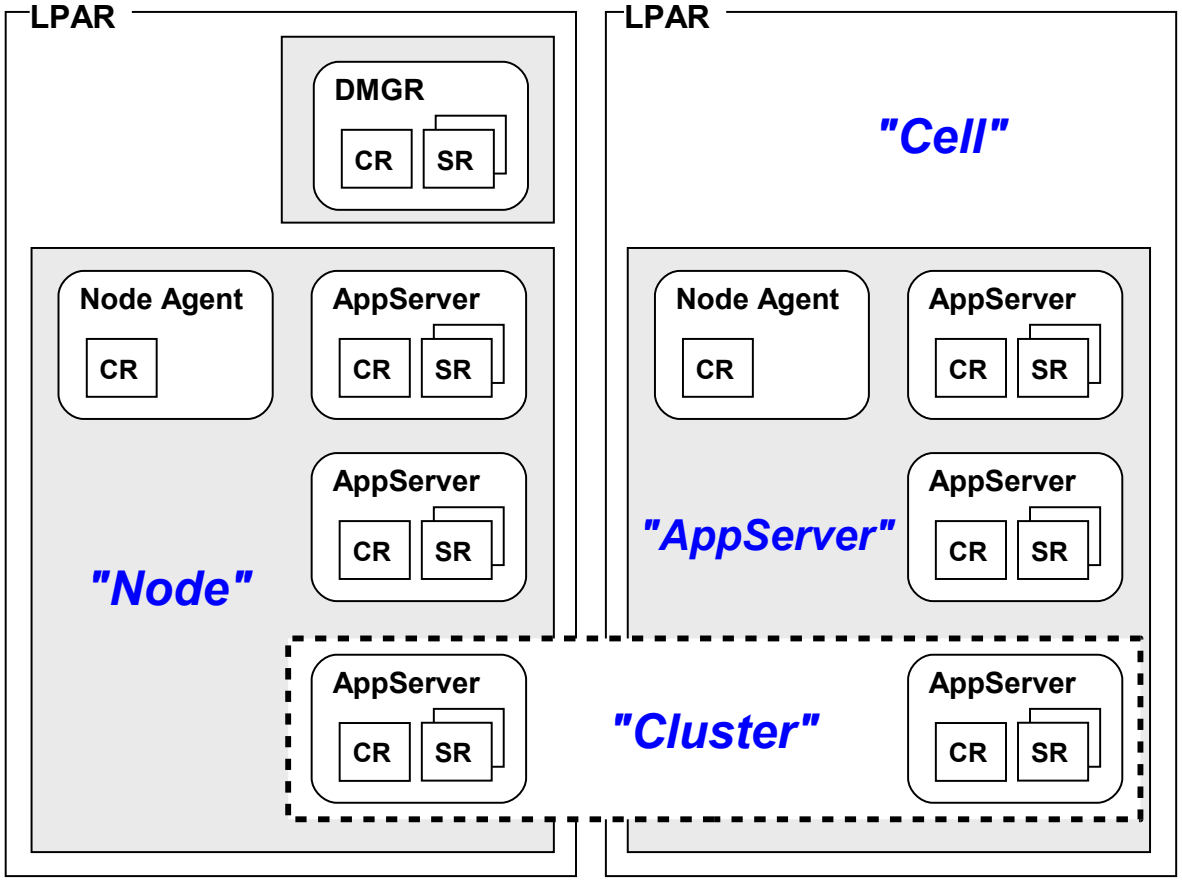
This is a key point -- the specification support provides the common interface point for applications. But the implementation may have *platform-specific exploitation:*



Servers, Node, Cell ...

Our Common Picture of a "Cell"

This is what we show to represent a two-LPAR WAS z/OS cell. This provides the backdrop to the rest of this presentation:



AppServer
Where the application runs

Cluster
A logical grouping of AppServers

Node
A logical grouping of AppServers on an LPAR

Cell
A logical grouping of nodes that represents the administrative domain.

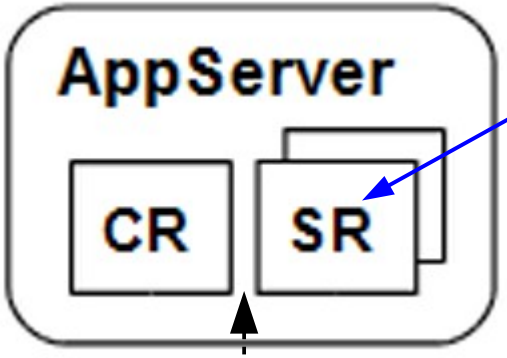
Our starting point ... we're assuming you're familiar enough with these key concepts

Applications ...

WebSphere Applications

The Central Point of it All

Applications are the reason we talk about the Application *Server*. There are several types of applications we're going to cover:



Applications run in the AppServer's servant region

We'll explore:

- Web Applications
- EJB Applications
- Batch Applications

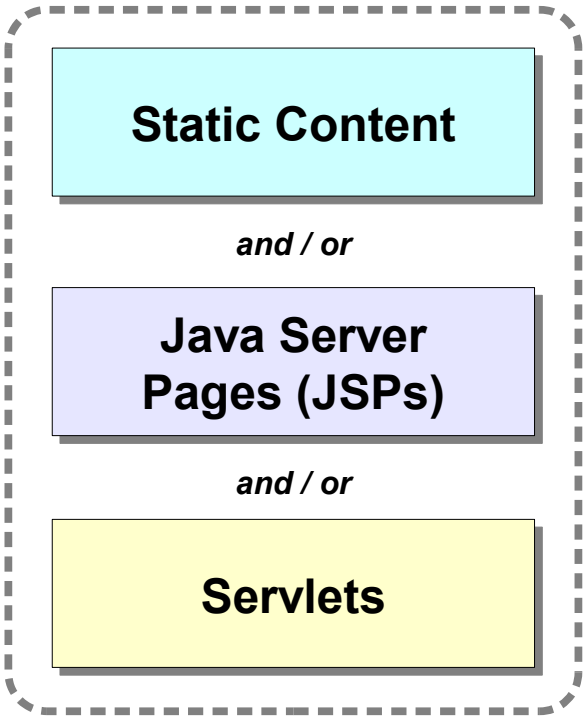
We'll look at:

- What they are
- How they work
- How they're packaged
- Understanding the deployment process

Go to Dave Follis' session Thursday at 1:30 for a good peek at what *really* goes on between the CR and the SR

Web Applications

The easiest to understand and perhaps the most common type deployed



Content that is fixed and does not change. It is simply served down to the browser:

- JPG/GIF picture files
- HTML screen formatting

These provide browser displays that may be different based on what the user does at the browser

This is Java code that provides more logical things, such as helping the user navigate the application functions, and interacting with other things behind the servlet

This is the essential core of a web application. There are things used above and beyond these. Those are scripting and framework technologies like JavaScript, PHP, JavaServer Faces, Struts, AJAX ...

Scripting and Frameworks

There are many extensions to the basic web application components ... *a sample:*

JavaScript	Most commonly a component of the HTML that is fed down to the browser. Browsers have the ability to "run" the JavaScript to provide a richer user interface.
PHP	A scripting language that's embedded into the HTML document and interpreted on the server by a PHP component that generates the HTML that goes to the user. Very common on the Internet, a little less common for enterprise applications.
JavaServer Faces	A framework of pre-built functions and make designing and building very rich user interaces easier for the developer.
Struts	Another framework, this extends the Servlet API to provide a set of functions to make rich user interface construction easier and more structured
AJAX	A group of technologies (HTML, cascaded style sheets, JavaScript) with the aim of going back to the server and page rendering related to the specific user action, not the whole page time after time.

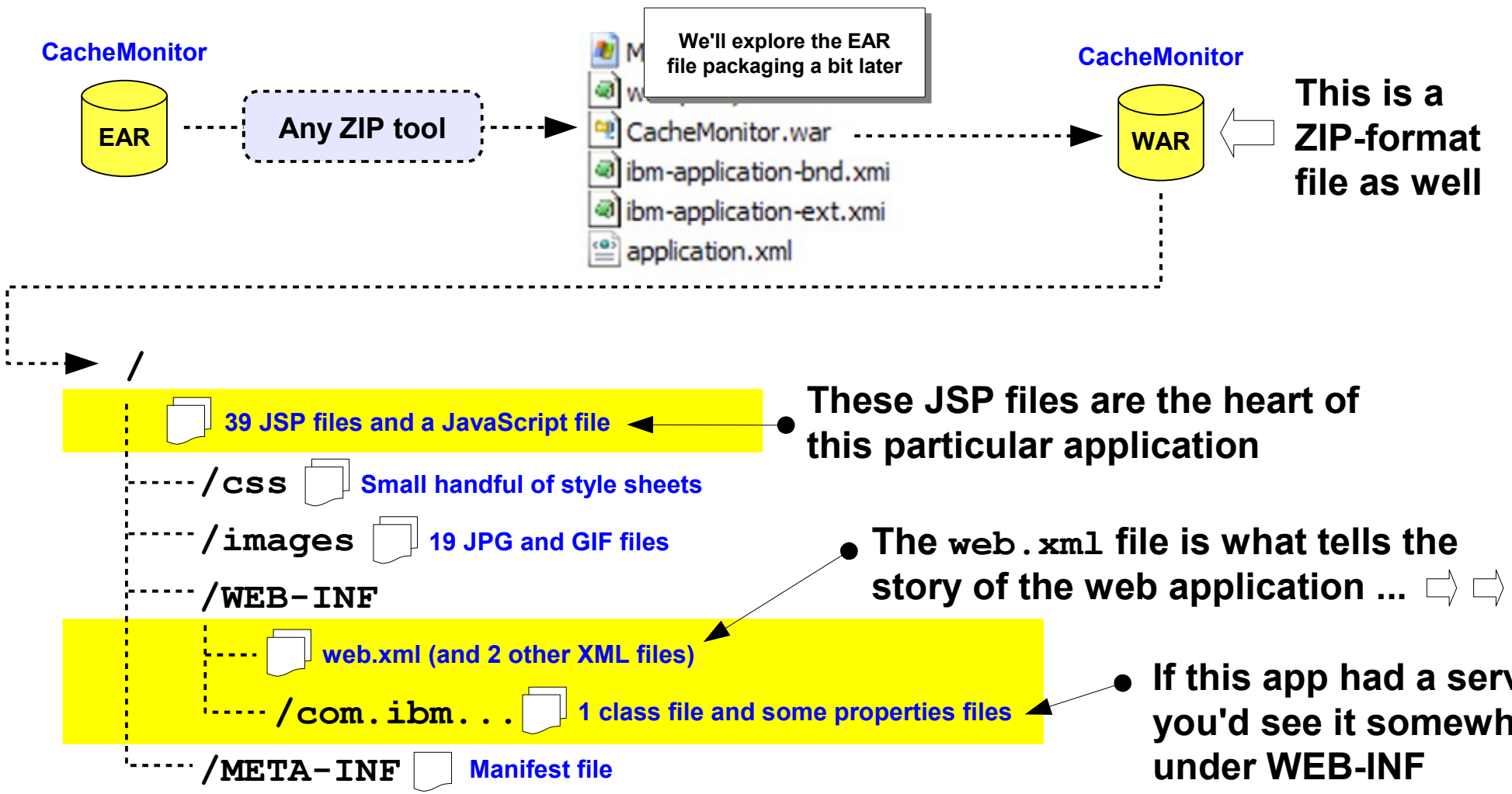
Two key points:

- **You may have to insure the WAS AppServer has access to supporting class files for some of these (JSF and Struts)**
- **These are all built on the core web application components and the packaging and deployment concepts are essentially the same**

The WAR file ...

The WAR File

"Web ARchive" file ... it is the packaging format for web applications. As an example, we'll show the CacheMonitor application that comes with WAS[1]:



1 /<mount_point>/DeploymentManager/installableApps/CacheMonitor.ear

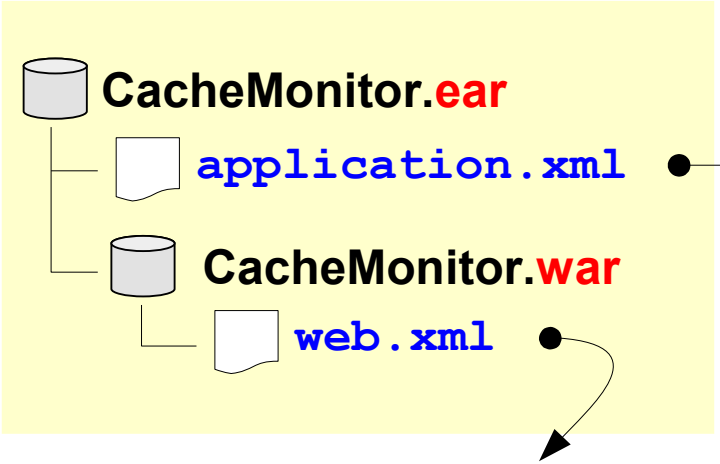
That CacheMonitor Application ...

Is packaged as an EAR (not just a WAR) and is invoked with the following URL. How do all those things tie together?

`http://<host:port>/cachemonitor/`

"Servlet Mapping"

"Context Root"



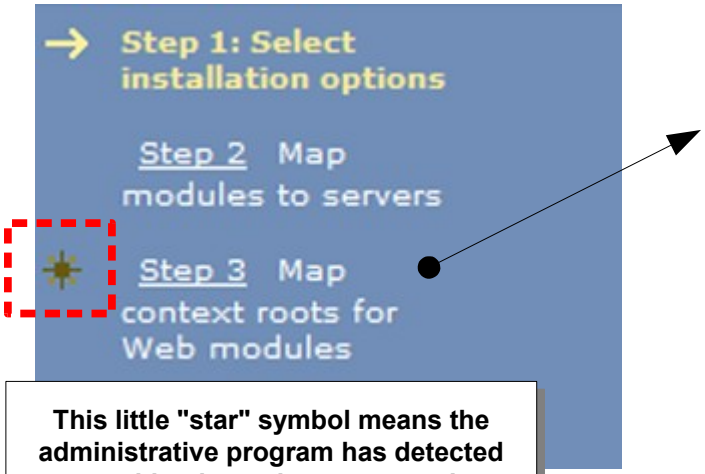

```
<module id="WebModule_1">
  <web>
    <web-uri>CacheMonitor.war</web-uri>
    <context-root>/cachemonitor</context-root>
  </web>
</module>
```

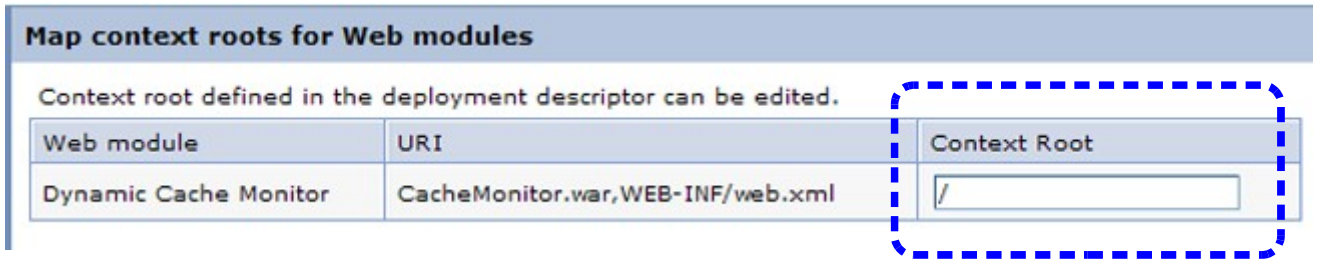
```
<web-resource-collection id=....>
  <web-resource-name>cachemonitor</web-resource-name>
    <url-pattern>/</url-pattern>
    <url-pattern>/banner.jsp</url-pattern>
    <url-pattern>/browser_detection.jsp</url-pattern>
      :
  </web-resource-name>
</web-resource-collection>
```

Deploying WAR without EAR ...

Deploying WAR without Surrounding EAR

It is possible and it sometimes does take place. What's the difference? No EAR so no `application.xml` ... which means context root must be specified on Admin Console:





Here is where you supply the context root for the web module found in the WAR file

Pause and think about what's going on here ... when you deploy an application, the administrative function reads these XML files to see what's there, what's not, and what needs your input.

If you ever wondered why some applications had only a few steps while others had many, this is why.

Those XML files are called "Deployment Descriptors"

EJB Applications

Enterprise Java Beans (EJBs) are intended for business logic. There are several flavors of EJBs we'll explore *briefly* here:

Session Bean

Perhaps the most common form. Session beans are often used to perform business functions and to access data using mechanisms such as JDBC, JCA or JMS

JDBC = relational access ; JCA = non-relational (ex: CICS) ; JMS = messaging

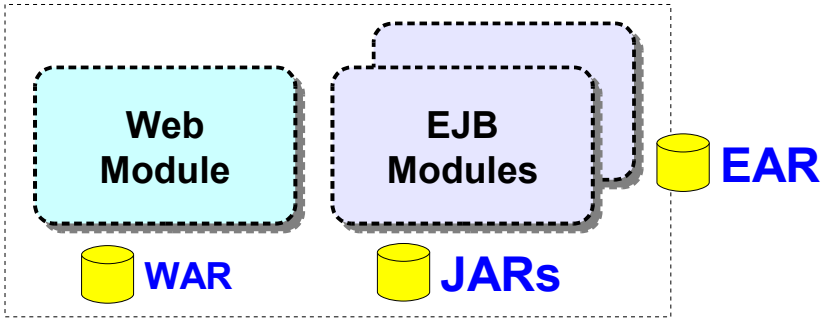
Message Driven Bean

These are EJBs that are configured to listen on a JMS queue and take action when a message arrives.

Asynchronous Bean

Sometimes an application wants to do a process that might take some extended period of time. An asynch bean allows that to take place -- the caller invokes the asynch bean and goes back to other processing. The asynch bean does its thing, and when ready signals the caller that the work is done.

Very common to see a deployable application consist of a web module and one to several EJB modules:

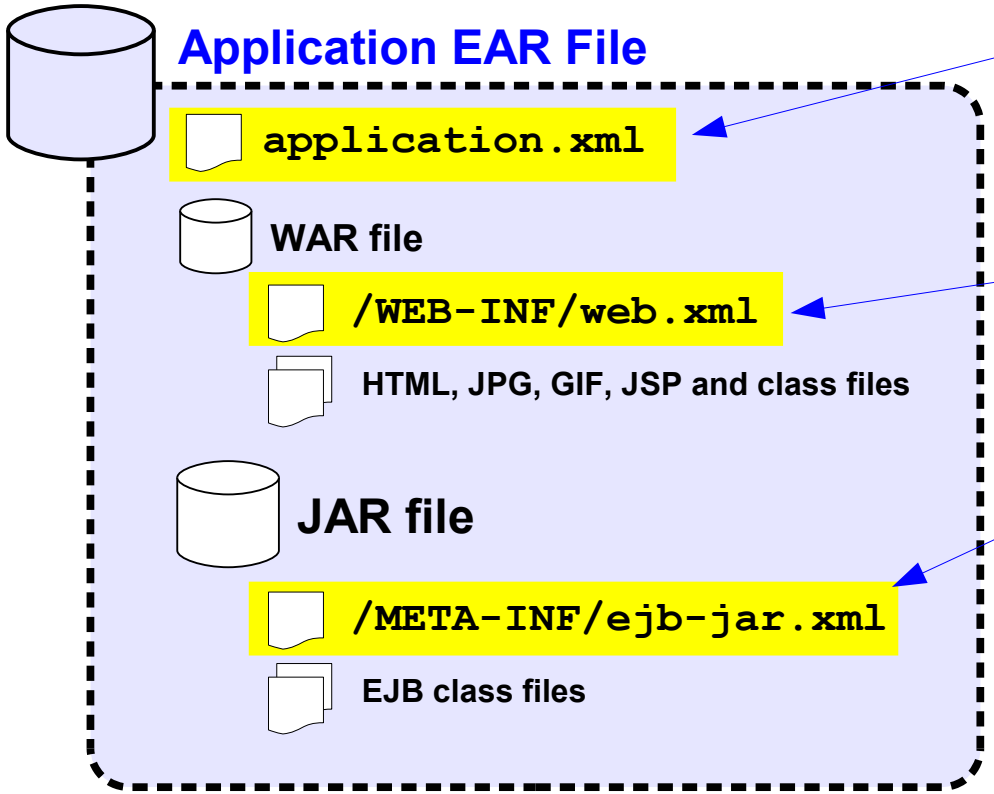


Let's start tying this together ...

JAR files ...

JAR Files and Their Deployment Descriptors

JAR files are used to hold EJB modules. The deployment descriptor in the JAR is the `ejb-jar.xml` file. Within the context of the EAR file it looks like this:



Overall information about the application:

- Application name
- Context root for web modules
- Inventory of modules, both web and EJB

Information about the web module:

- Servlet mappings
- Servlet class file names
- Security information

Information about the EJB module:

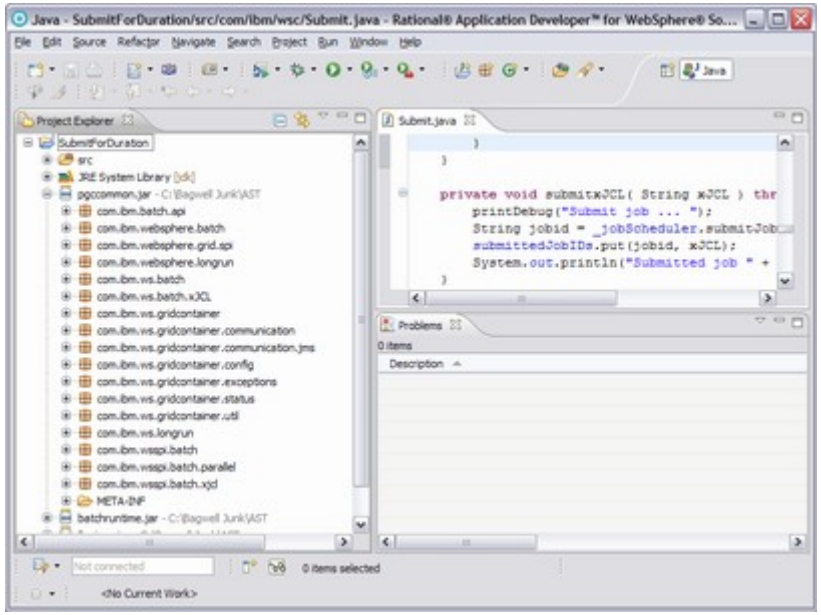
- EJB class file names
- Any references to data resources:
- Any references to other EJBs:

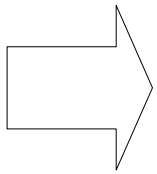
```
<resource-ref ...>
  <res-ref-name>jdbc/xyz</res-ref-name>
</resource-ref>

<ejb-local-ref ...>
  <ejb-ref-name>ejb/abcdef</ejb-ref-name>
</ejb-local-ref>
```

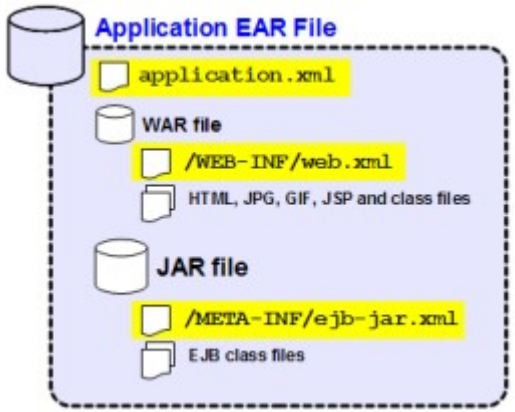
IDE - Integrated Development Environment

These applications are developed using tools designed to produce the proper format and construction. You will not be asked to create any of those XML files

Example: IBM Rational Application Developer (RAD)



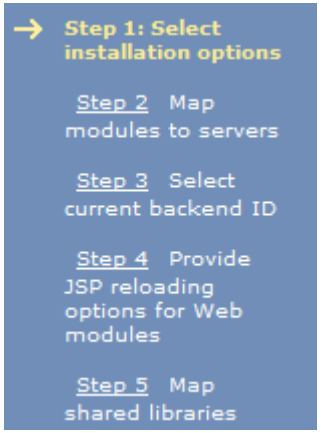




Admin Console behavior ...

Admin Console Behavior ...

What you see in the Admin Console is simply what it sees in the application's XML:



The number of steps is a function of the elements found in the deployment descriptors

ejb-jar.xml

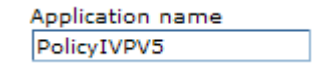
Provide JNDI names for beans

Map EJB references to beans

Map resource references to resources

application.xml

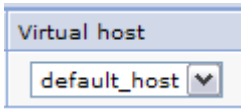
Application Name



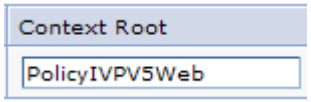
Map Modules to Servers

web.xml

Map virtual hosts for Web modules



Map context roots for Web modules

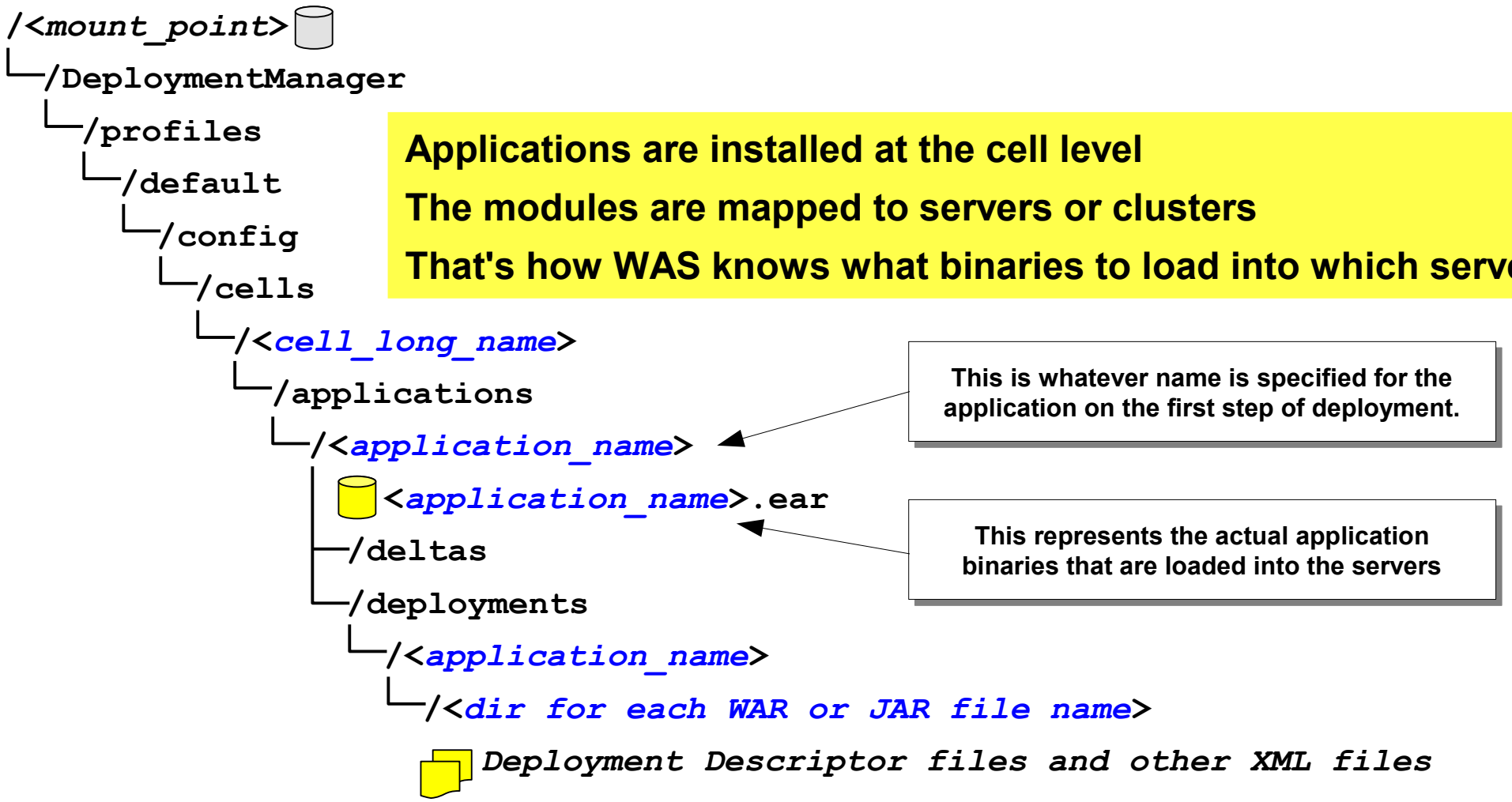


You can learn a lot about an application prior to deployment by cracking it open with a zip utility and browsing the contents and XML files.

Where files go ...

Where The Application Actually Goes ...

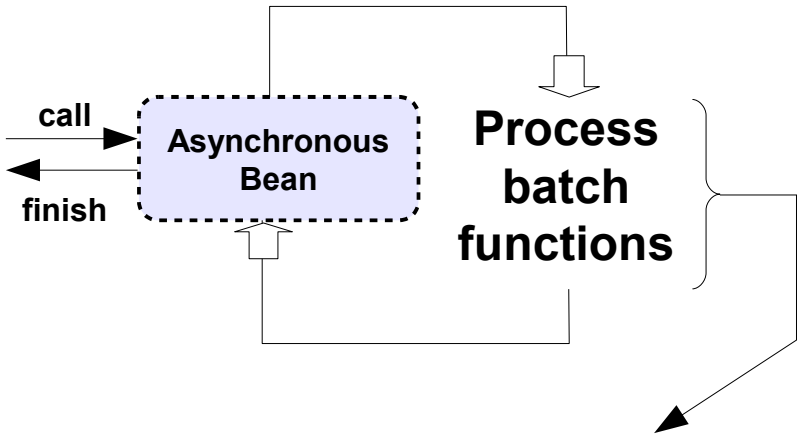
Here's a map of what happens when an application gets deployed:



You'll also find a reference to applications mapped to servers in the `serverindex.xml` file, which is under the `/nodes` directory in each node.

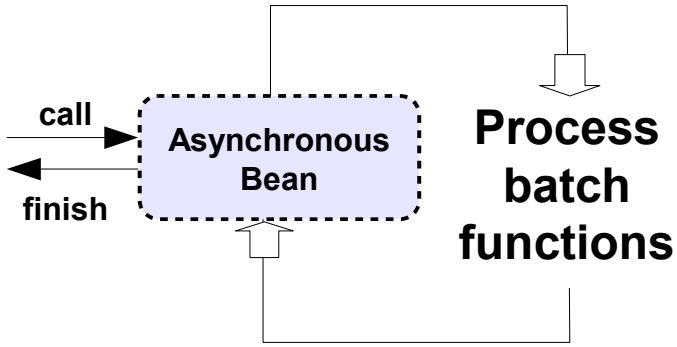
Running Batch in WAS

In order to run a batch job inside of WAS[1], you'd have to run it as an asynchronous bean[2]. The question is -- how much custom support code do you want to write?



If this is a relatively simple serial execution then this is doable.

But when you try to introduce more advanced functions like conditional multi-step processing, checkpoint processing, job pause and restart, a job console into the JVM ... then you start writing more and more *custom support code.*



IBM-supplied batch framework and integration with WAS runtime to supply those features.

This is what the Feature Pack for Modern Batch and WebSphere Compute Grid does for you -- it supplies the middleware functionality so you can focus on your batch requirements.

Advantages of Java batch in WAS over JZOS:

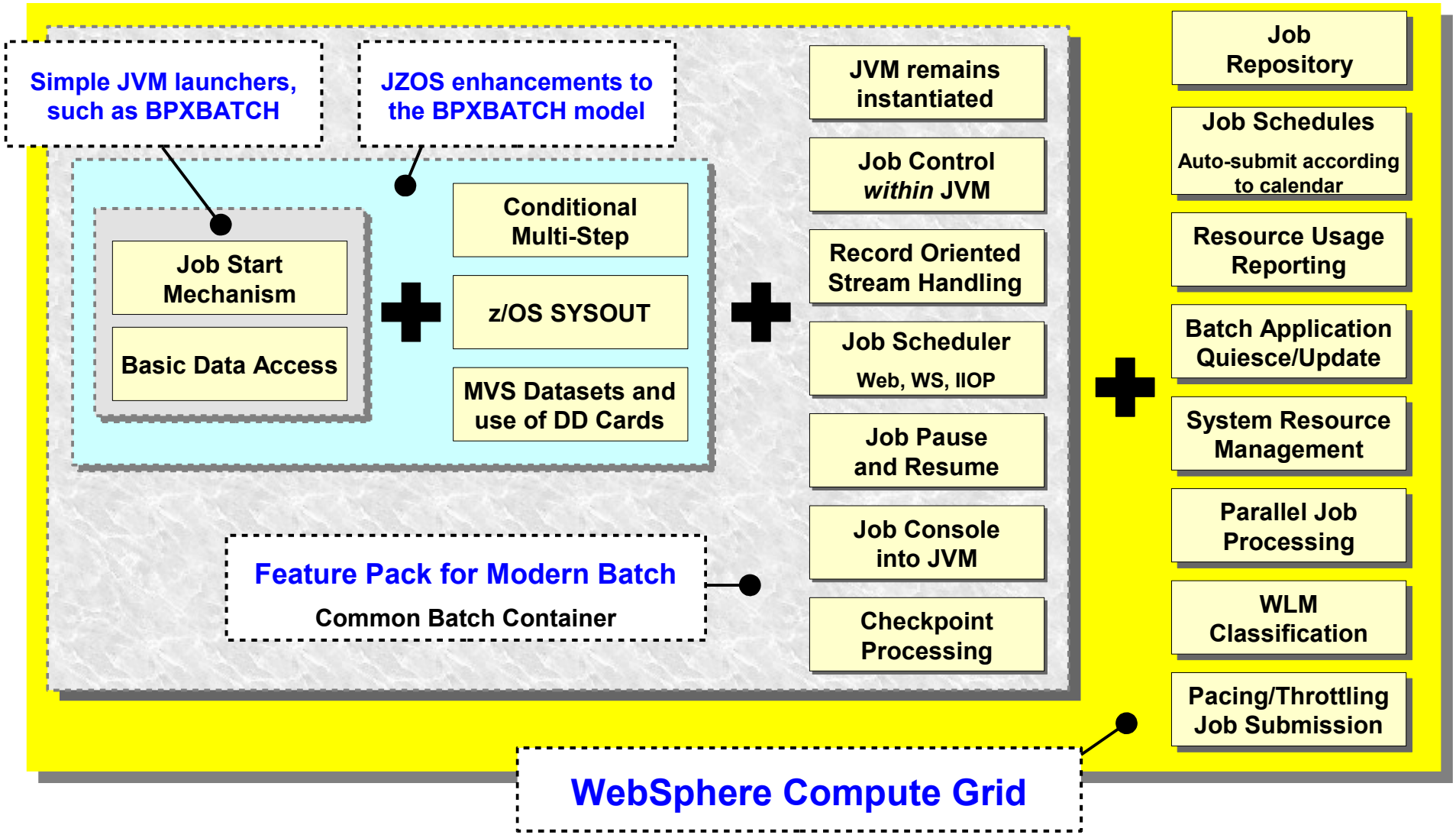
- **Eliminates multiple JVM launch / tear-downs**
- **Provides a job scheduler / dispatcher function**
- **Integrate batch with OLTP within managed framework**
- **Job console into JVM to manage status and act upon submitted jobs**
- **Other functional advantages**

[1] As opposed to BPXBATCH or JZOS
[2] For a long-running batch job it's the best way to avoid timeouts

Batch offerings ...

Beyond BPXBATCH or JZOS

If Java batch applications are part of the mix you'll find that even JZOS takes you only so far. There's the Feature Pack for Modern Batch or Compute Grid to consider:

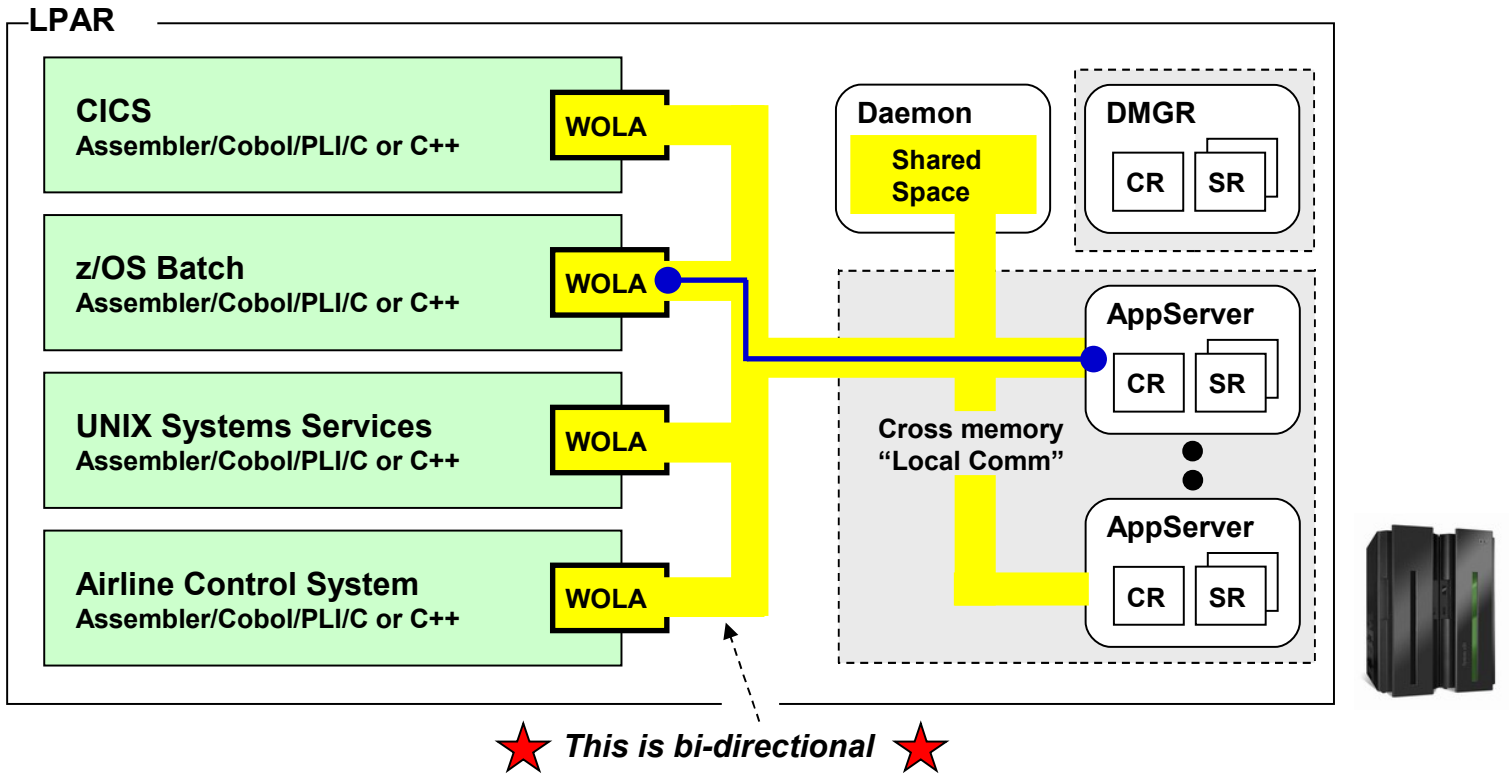


zDiff ...

zDiff Items

WebSphere Optimized Local Adapters

A means of connecting external addresses spaces to the high-speed "local comm" cross memory structure inside the WAS z/OS cell.

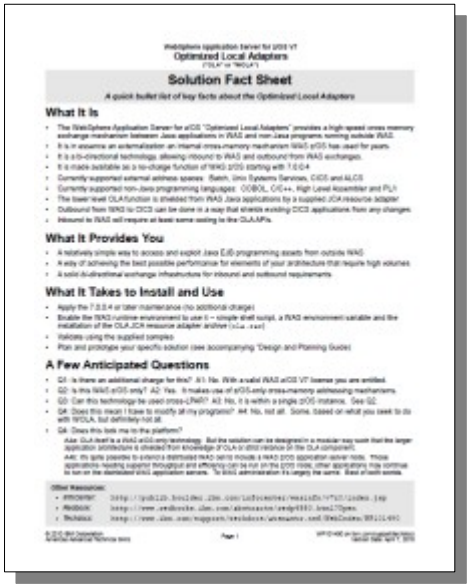


"WOLA Application Design Considerations" tomorrow at 8:00 AM

Techdocs ...

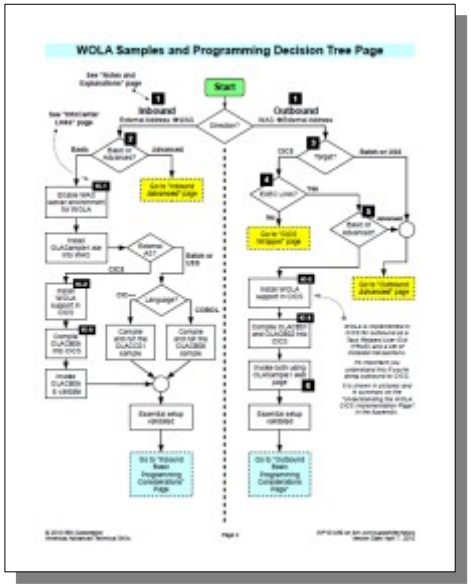
Techdocs

We've turned WP101490 as "WOLA Central" for our documentation:



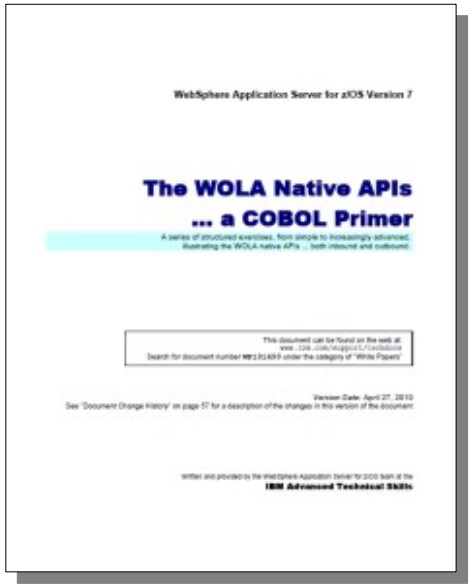
One page "Fact Sheet"

Easy to read one-pager that spells out the essentials of WOLA



"Planning and Design Guide"

Provides a decision flowchart to help you drill down to the APIs and features of WOLA best for your project



Native APIs with COBOL Primer

Step-by-step exercises to introduce you to the native APIs using COBOL.

Plus other white papers and presentation material

`http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101490`

The rest ...

FRCA, Thread Hang, SMF 120.9

A brief overview of each ... just in case you weren't aware of these V7 features:

FRCA

FRCA = "Fast Response Caching Accelerator"
FRCA is a feature of z/OS TCP. It is a low-level caching mechanism
WAS z/OS V7 extended the WAS Dynacache to allow FRCA as an "external cache provider."
If app pushes content to Dynacache, WAS z/OS will push to FRCA
Objective is to save CPU cycles and increases throughput
All the Dynacache facilities apply: invalidate, purge, timeout, etc.

Thread Hang Recovery

Before: when a request timeout occurs WAS marks the thread hung and EC3 abends the servant region. That's the only way JVM spec allows for clearing hung Java threads.
Thread Hang Recovery feature provides a series of mechanisms that first try to shake the thread loose, then it provides several delaying mechanisms.
Objective is to avoid EC3 if at all possible.

SMF 120.9

Before: WAS z/OS SMF records spread across several different subtypes; the kind of information customers needed wasn't provided; overhead to cut records was high.
Now: new SMF 120.9 record architected from the ground up to provide meaningful information
Overhead is very low, so cost of turning on to get useful information reduced

`http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101342`