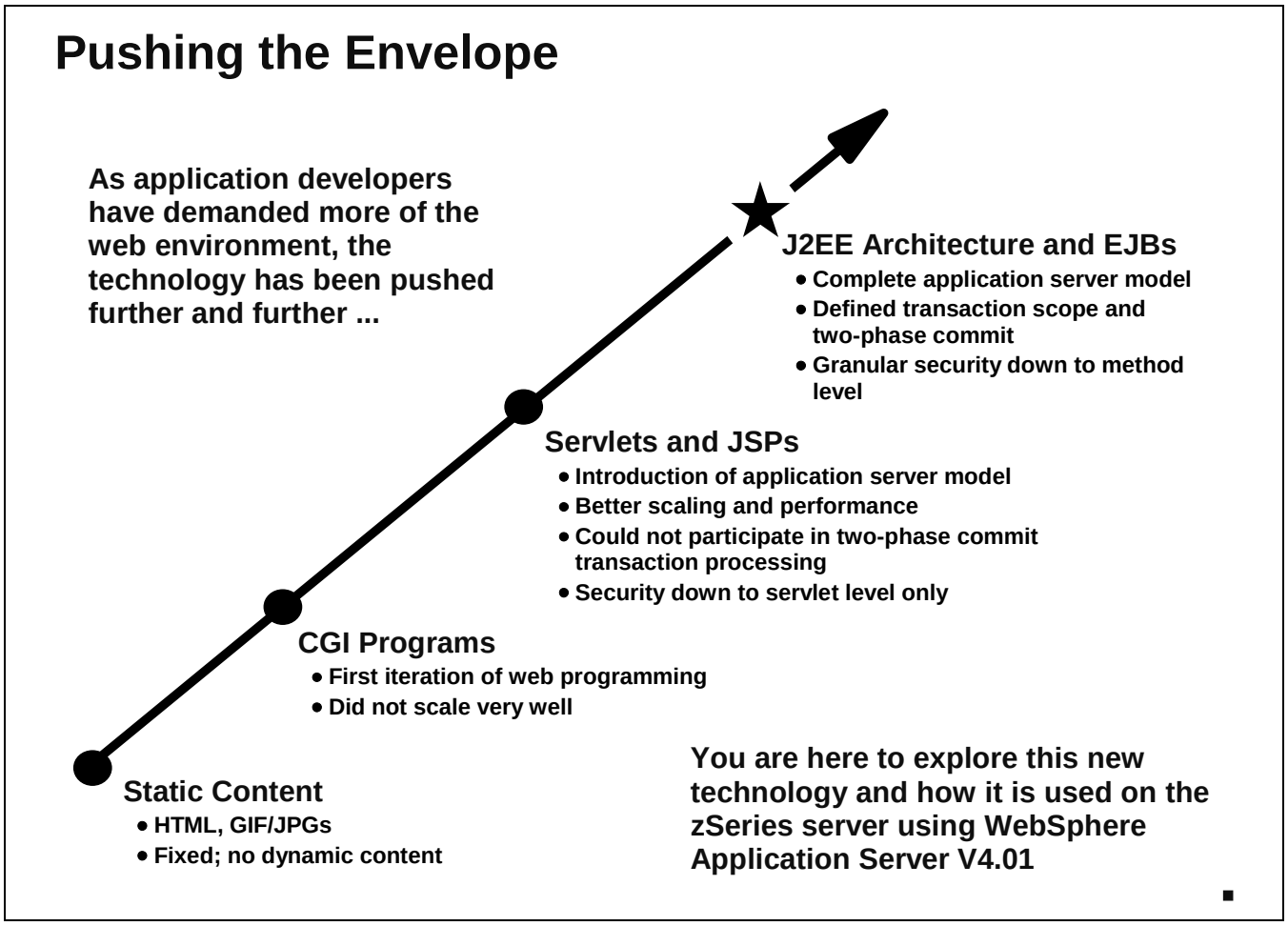
Web Enablement Using EJB J2EE Technology Workshop

(This page intentionally left blank)



In the beginning the web consisted of little more than static content: HTML pages with JPG/GIF images. It served the purpose of providing information to those on the web, but proved somewhat labor-intensive and limited. The desire existed to provide users of the website access to dynamically produced content.

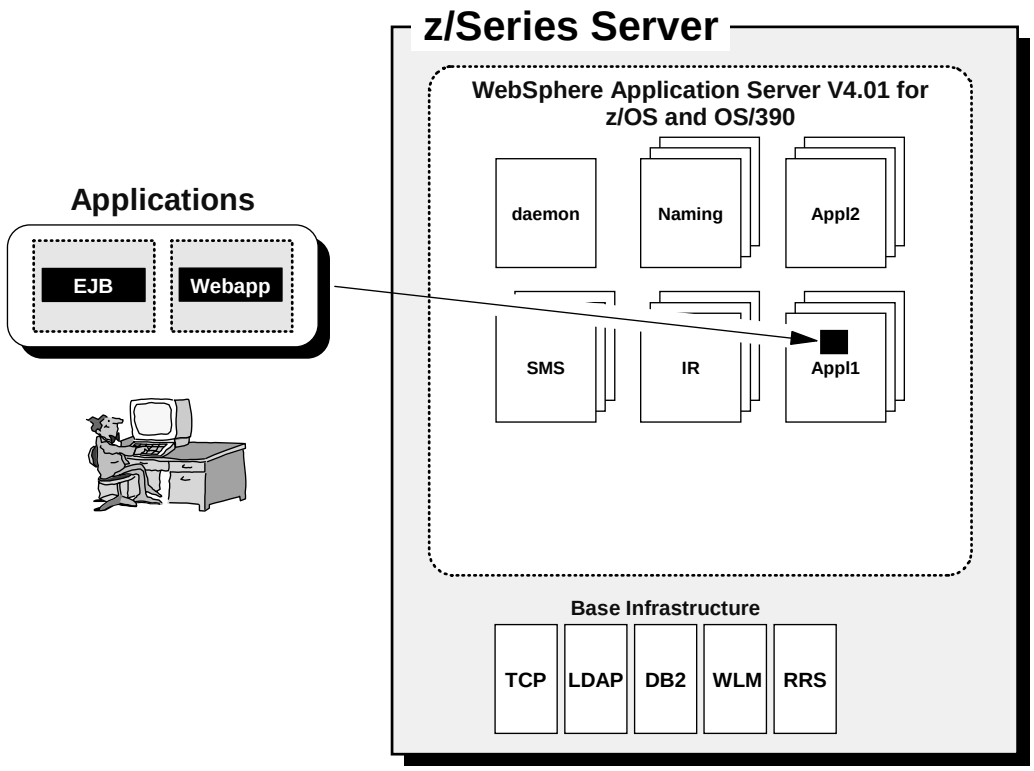
The next phase of this evolution was the introduction of CGI (Common Gateway Interface) programs, which provided dynamic content by allowing the web request to spawn a background program that would execute the request and provide dyamically produced HTML output in return. This worked well enough for relatively small usage, but failed to scale adequately.

Next came "Servlets," which were specially designed Java applications that ran within an "application server" program. IBM's application server is WebSphere. JSPs are special HTML documents with small Java programs embedded in them that provide dynamic content for inclusion in the HTML. This design provided much better scaling and performance.

But application developers, demanding more and more from the web, pushed the servlet architecture to its limit. One significant shortcoming of the servlet architecture is the fact that servlets can not participate in a two-phase commit process. Accessing a single backend data store is okay; multiple data stores is not, unless something else (such as CICS) manages the transaction. Further, the security model was defined only down to the servlet itself; the security on the individual methods of the servlet classes could not be defined with differing levels of security.

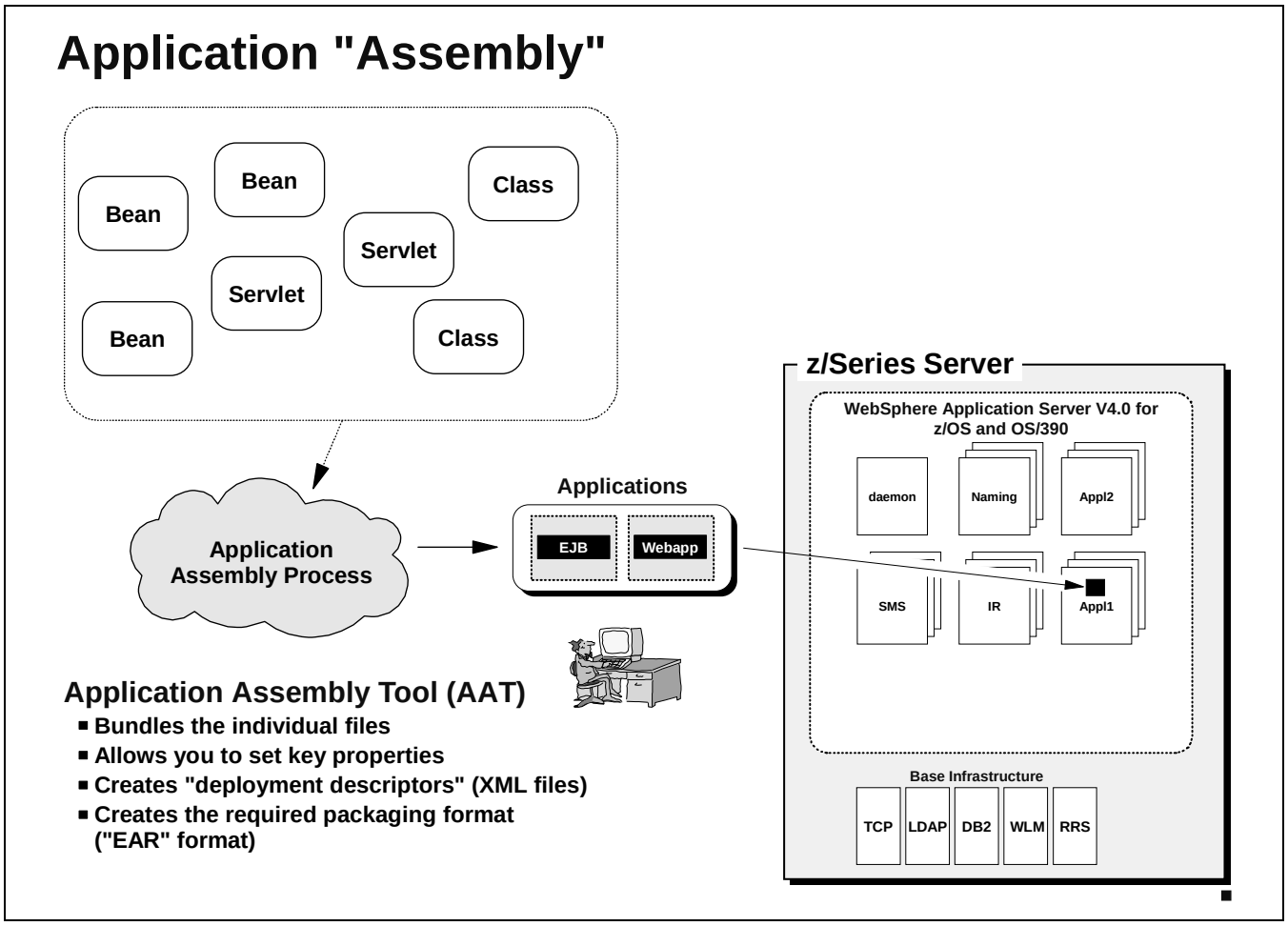
The next step is a full-fledged enterprise application model. And that model is the "J2EE" model (Java 2 Enterprise Edition) and "EJBs" (Enterprise Java Beans). This architecture provides the definition of transaction scope, two-phase commit, and a much more granular security model. In this course we'll cover this and how the J2EE Server itself is implemented on the IBM zSeries server.

100,000 Foot View



At a very high level what this class is all about is the WebSphere Application Server V4.01 product and the applications that run within this environment. The picture shown above is one you'll see over and over in this class because it represents the logical layout of the different components of the system.

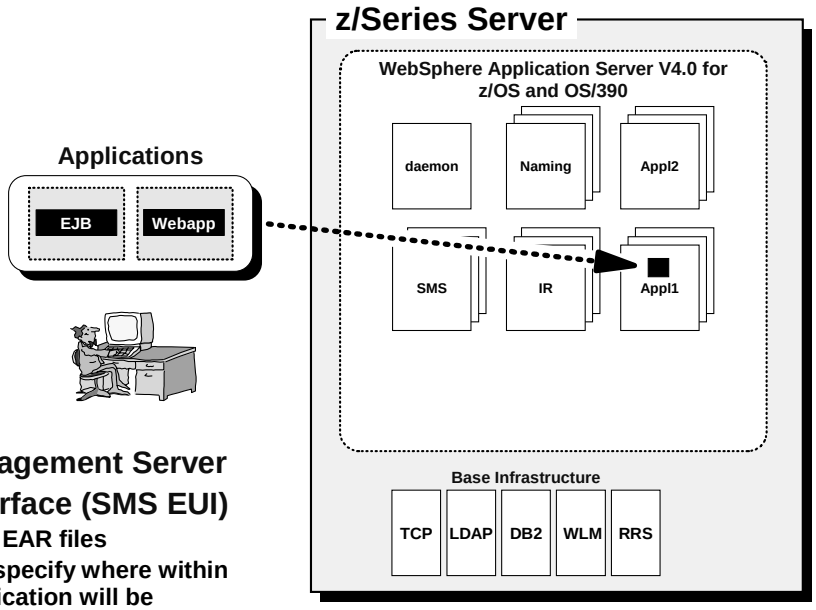
Underneath this picture is quite a bit of complexity. But to start with there are a few very basic concepts that will help frame this class and what it will deliver.



Applications that run in the WAS 4.01 server are different from traditional applications. They must be packaged in a specific, standard way before they can be introduced to the server environment. This act of packaging the application is known as "application assembly." There is a tool provided with WAS 4.0 to assist with this packaging: it is known as the Application Assembly Tool. Assembling an application involves taking all the various piece-parts of the application, setting properties related to the application, creating XML files known as "deployment descriptors" (which are XML files that describe what's in the application and how it is to be run), and finally putting it all into a specifically formatted file called an EAR file (Enterprise ARchive).

We'll pay particular attention to assembling applications in this class. It is one of the two primary functions involved with installing applications into the WAS 4.01 environment. (The other is deploying applications, which we'll talk about next).

Application "Deployment"



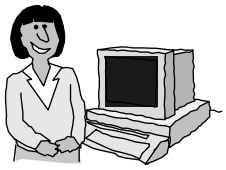
Systems Management Server End User Interface (SMS EUI)

- **Takes as input EAR files**
- **Allows you to specify where within WAS 4.01 application will be deployed**
- **Deployment properties set**

Once you have your application packaged into an EAR file, you are ready to deploy it. Deploying an application involves more than simply copying the file to the WAS system. Another tool is provided with the WAS product to assist with this: the Systems Management Server End User Interface. This is a workstation program that provides a graphical interface to the WAS 4.01 system. When you're ready to deploy your EAR file, you start what's called a "conversation" with the WAS system, you indicate where you wish the application to be installed, you set a few additional properties for the application and then you tell the SMS EUI tool to deploy the application. It then goes off and does a good many things behind the scenes as part of its deploying of the application.

This is the other major area of focus for this class.

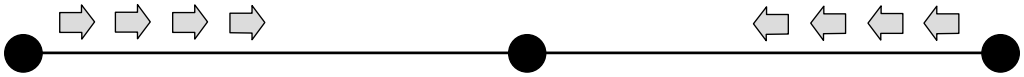
Who'll Do Those Roles?



Application Developer

Application Assembly and Deployment

Systems Programmer



- **The role is still evolving**
- **It will require some knowledge of both the application development as well as system programming worlds**
- **Different people may do "assembly" and "deployment"**
- **This class was built from the system programmer's perspective.**

■

These roles as "Application Assembler" and "Application Deployer" are relatively new. To perform those roles, one must know some things about the application as well as some things about the server environment. In that sense it sits half-way between the traditional roles of developer and systems programmer. Which of those two will fill that role? It's not certain at this point; the role is still evolving.

Some argue that the role of the application assembly fits closely with the role of the developer, while deployment is more closely aligned with systems programming. To that end, it is possible that different people will fill the role of application assembly and deployment. Many organizations prefer to keep such roles separate within the structure of the company. That's fine: as you'll soon learn, there's a fairly clean line between assembly and deployment, and different people (or groups) could do each.

We have structured this class more from the perspective of the systems programmer rather than the application developer. Much of the discussion that'll come in the next three days will focus on S/390 topics that will be more familiar to systems programmers than developers. We have less focus on pure Java programming issues and more on systems-related issues.

Agenda

- Introduction
- J2EE Architecture Overview
- WAS 4.01 System Architecture
  - Discovery Lab
- Installation and Configuration Process
  - ISPF Dialogs Lab
- Systems Management EUI Tool and Application Deployment
  - PolicyIVP Application Deployment
- CICS Connector
  - J2C Connector Lab
- JDBC Lab Intro
  - JDBC EJB Lab
- Configuring Web Applications
  - Webapp Lab
- Migration Issues

This is a three day class. The topics to be discussed are shown in order, and the accompanying labs are shown as well.

Introductions

- **Name**
- **Company**
- **Where's home?**
- **A little about your background**
- **What plans for WAS 4.01 do you have? (don't disclose secrets!)**

Now let's go around the room.

Administrivia

- **Start and Stop Times Each Day**
- **Breaks/Lunch**
- **Restrooms**
- **Phones and Analog Lines**
- **Smoking**
- **Emergency Procedures**
- **Messages**

Any questions?

End of Document