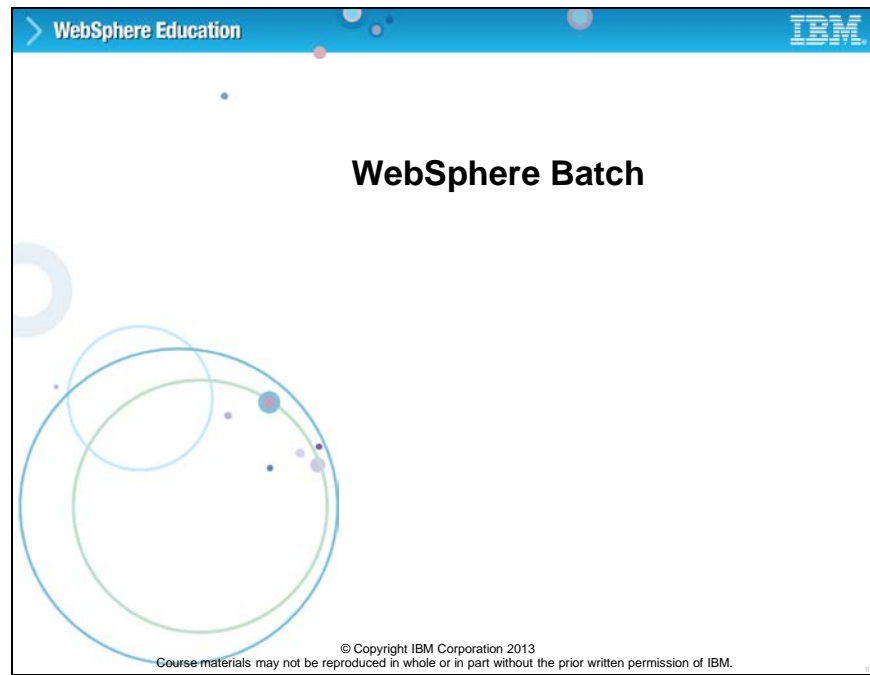


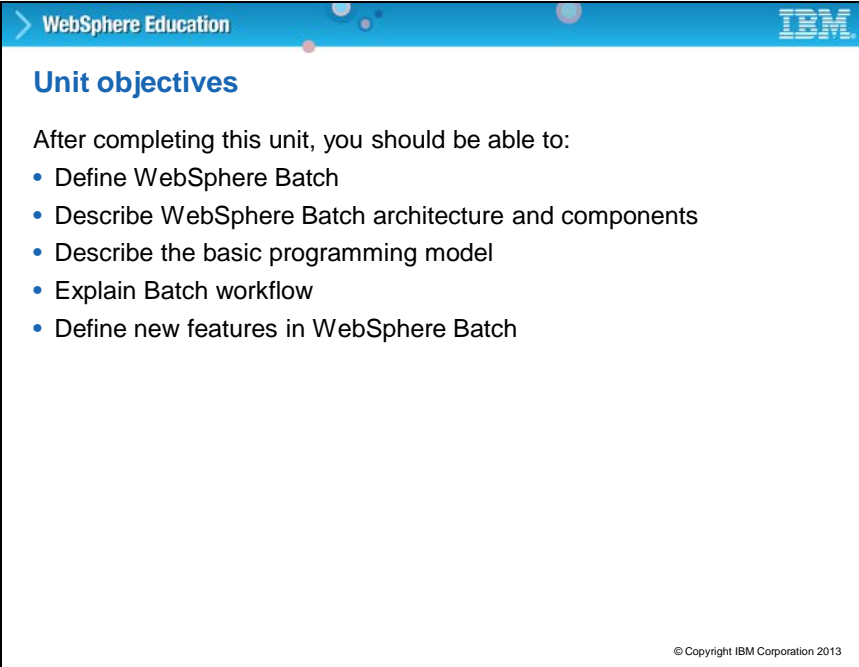
Slide 1




WebSphere Batch. In this unit, you learn about the WebSphere Batch features in WebSphere Application Server V8.5.

.

Slide 2



The slide is titled 'WebSphere Education' in the top left corner and features the IBM logo in the top right corner. The main heading is 'Unit objectives' in blue. Below it, a paragraph states 'After completing this unit, you should be able to:' followed by a bulleted list of five objectives. The bottom right corner contains a small copyright notice.

WebSphere Education 

Unit objectives

After completing this unit, you should be able to:

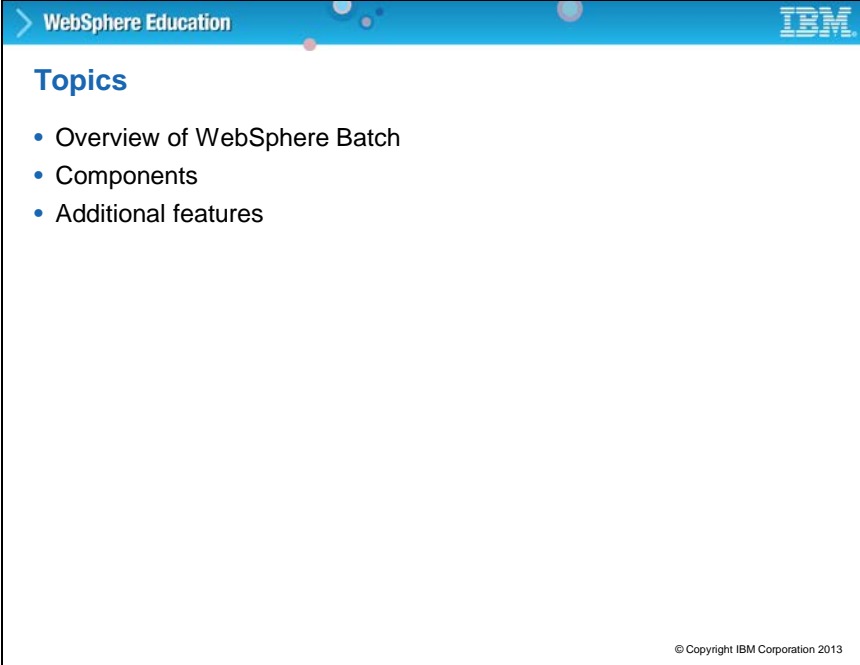
- Define WebSphere Batch
- Describe WebSphere Batch architecture and components
- Describe the basic programming model
- Explain Batch workflow
- Define new features in WebSphere Batch

© Copyright IBM Corporation 2013

After completing this unit, you should be able to:

- Define WebSphere Batch
- Describe WebSphere Batch architecture and components
- Describe the basic programming model
- Explain Batch workflow
- Define new features in WebSphere Batch

Slide 3



The slide is titled "WebSphere Education" in the top left corner and features the IBM logo in the top right corner. The main content area is titled "Topics" and contains a bulleted list of three items: "Overview of WebSphere Batch", "Components", and "Additional features". The slide has a blue header bar and a white background for the main content area.

WebSphere Education

IBM

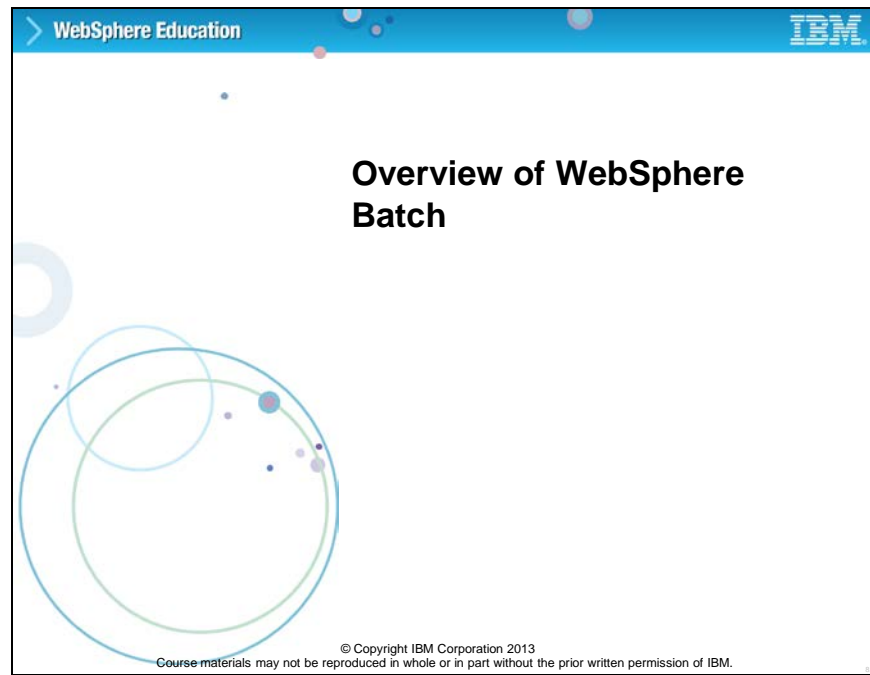
Topics

- Overview of WebSphere Batch
- Components
- Additional features

© Copyright IBM Corporation 2013

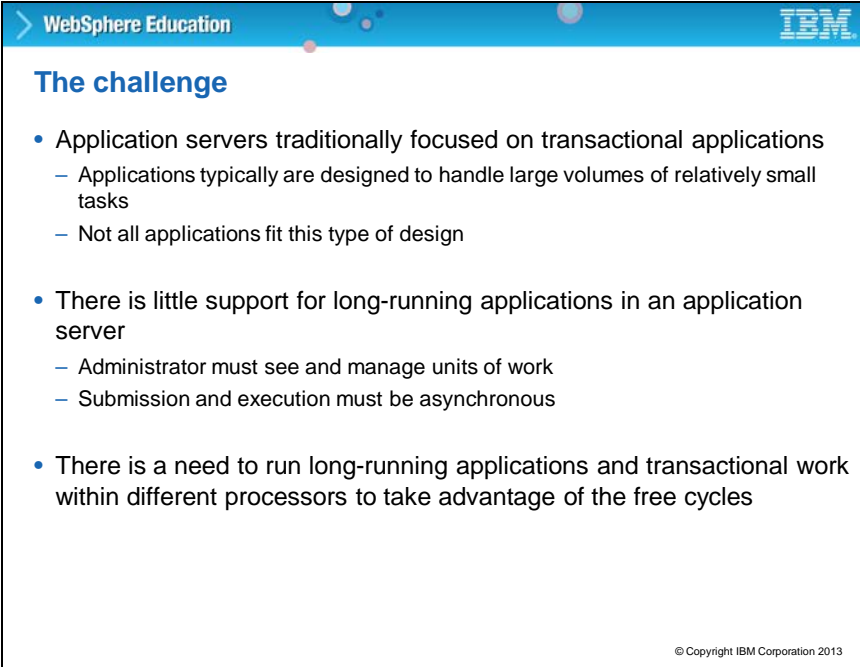
This unit is divided into three topics.

Slide 4



Topic: Overview of WebSphere Batch. In this topic, you get an overview of WebSphere Batch.

Slide 5



The slide is titled "WebSphere Education" in the top left corner and features the IBM logo in the top right corner. The main heading is "The challenge". Below this, there is a bulleted list of three points. The first point is "Application servers traditionally focused on transactional applications", which has two sub-points: "Applications typically are designed to handle large volumes of relatively small tasks" and "Not all applications fit this type of design". The second point is "There is little support for long-running applications in an application server", with sub-points: "Administrator must see and manage units of work" and "Submission and execution must be asynchronous". The third point is "There is a need to run long-running applications and transactional work within different processors to take advantage of the free cycles". At the bottom right of the slide, there is a small copyright notice: "© Copyright IBM Corporation 2013".

WebSphere Education

IBM

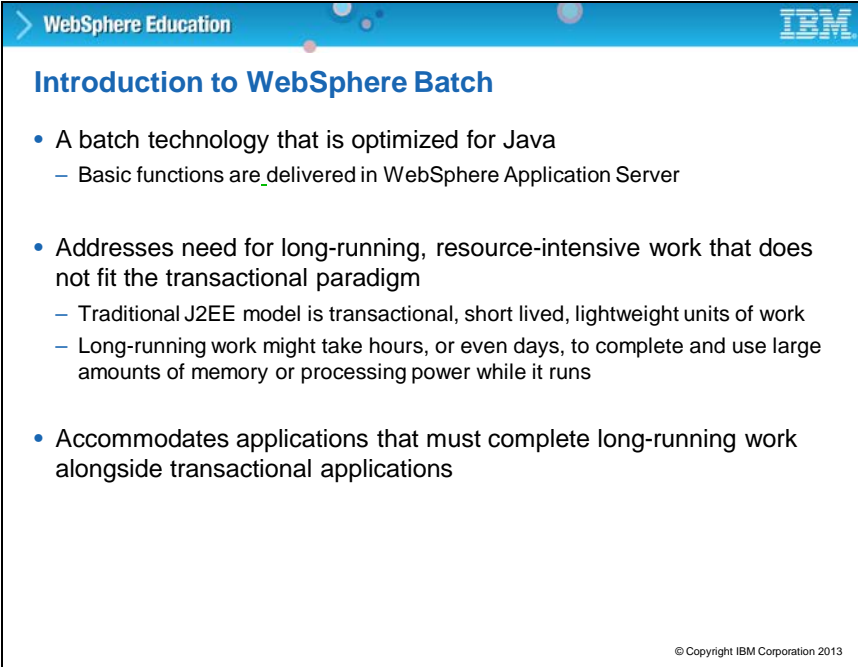
The challenge

- Application servers traditionally focused on transactional applications
 - Applications typically are designed to handle large volumes of relatively small tasks
 - Not all applications fit this type of design
- There is little support for long-running applications in an application server
 - Administrator must see and manage units of work
 - Submission and execution must be asynchronous
- There is a need to run long-running applications and transactional work within different processors to take advantage of the free cycles

© Copyright IBM Corporation 2013

WebSphere Application Server and Java Platform, Enterprise Edition servers in general classically focused on light-weight transactional work. Typically, an individual request can be handled in a few seconds of processor time and relatively small amounts of memory. However, other styles of long-running applications require more resources and different types of support from the runtime environment. WebSphere Batch provides support within WebSphere Application Server for long-running applications. Within an enterprise environment, it is usually preferable to run long-running and transactional work on separate processors, running them within the same processor can negatively affect performance for the application. Long-running work might take hours or even days to complete and use large amounts of memory or processing power while it runs. WebSphere Batch provides the capability to deploy different types of applications within your environment, and can balance the work that is based on policy information.

Slide 6



The slide is titled "Introduction to WebSphere Batch" and is part of a "WebSphere Education" presentation. It features a blue header with the IBM logo. The content is organized into a bulleted list describing the capabilities of WebSphere Batch technology.

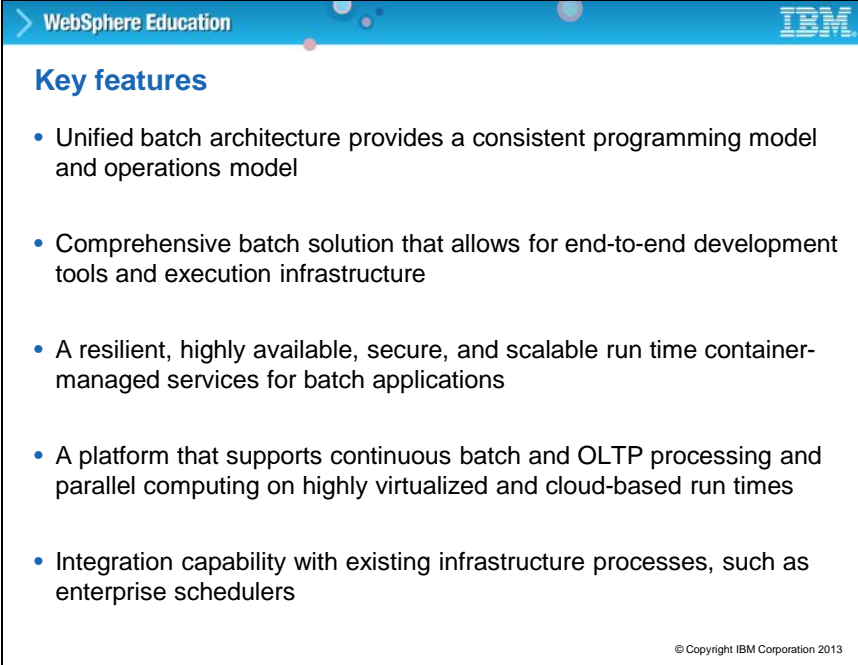
- A batch technology that is optimized for Java
 - Basic functions are delivered in WebSphere Application Server
- Addresses need for long-running, resource-intensive work that does not fit the transactional paradigm
 - Traditional J2EE model is transactional, short lived, lightweight units of work
 - Long-running work might take hours, or even days, to complete and use large amounts of memory or processing power while it runs
- Accommodates applications that must complete long-running work alongside transactional applications

© Copyright IBM Corporation 2013

Batch applications are designed to run long and complex transaction processing that typically runs computationally intensive work. This type of processing requires more resources than traditional online transactional processing (OLTP) systems. Batch applications that are run as background jobs described by a job control language, and use a processing model that is based on submit, work, and result actions. The execution of batch processes can take hours and the tasks are typically transactional, involving multi-step processes.

The Batch feature for IBM WebSphere Application Server V8.5 is also available in the WebSphere Application Server V7.0 Feature Pack for Modern Batch and in IBM WebSphere Application Server V8.0.

Slide 7

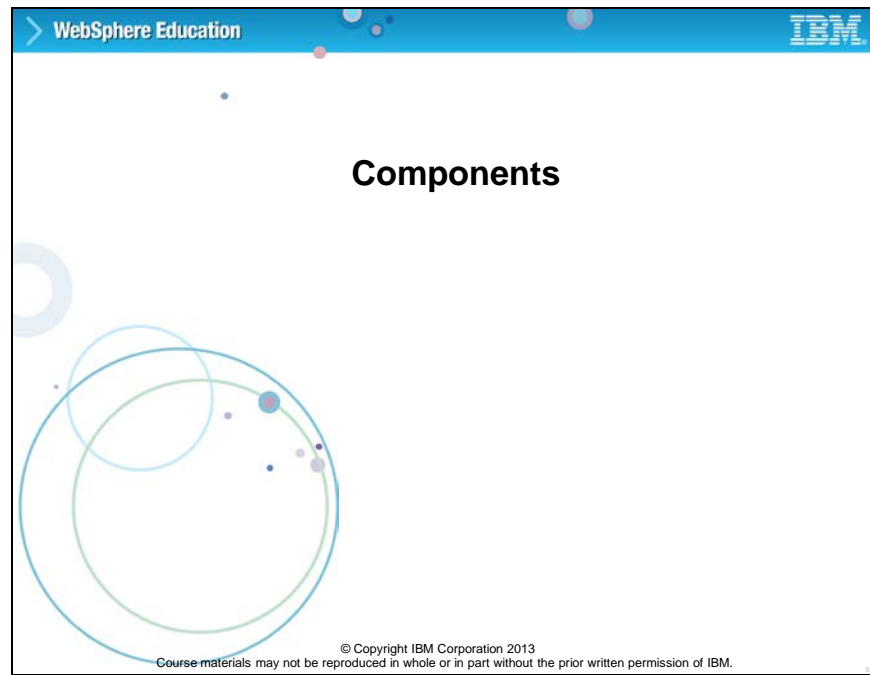


The slide is titled 'WebSphere Education' in the top left corner and features the IBM logo in the top right corner. The main heading is 'Key features' in blue. Below it is a bulleted list of five features. At the bottom right, there is a small copyright notice: '© Copyright IBM Corporation 2013'.

- Unified batch architecture provides a consistent programming model and operations model
- Comprehensive batch solution that allows for end-to-end development tools and execution infrastructure
- A resilient, highly available, secure, and scalable run time container-managed services for batch applications
- A platform that supports continuous batch and OLTP processing and parallel computing on highly virtualized and cloud-based run times
- Integration capability with existing infrastructure processes, such as enterprise schedulers

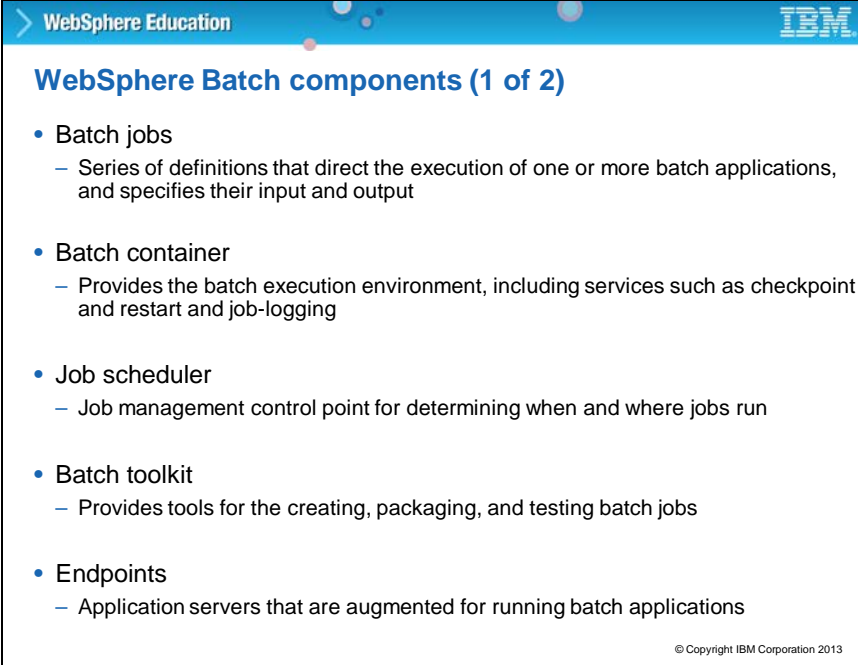
The WebSphere Batch feature for IBM WebSphere Application Server provides a robust Java batch programming model that enables the integration of online and batch processing within an architected framework across multiple platforms. WebSphere Batch provides a comprehensive execution environment for efficient Java batch processing. The environment includes new key aspects such as the ability to run batch jobs in parallel by running Java batch inside WebSphere Application Server, support for COBOL, and a unified batch architecture for the enterprise. Using XML job control language (xJCL), WebSphere Batch provides consistent programming and operational models. WebSphere Batch uses a batch technology that is optimized for Java and supports long-running applications that ensure agility, scalability, and cost efficiency for enterprises.

Slide 8



Topic: Components. In this topic, you learn about the various batch components.

Slide 9



The slide is titled "WebSphere Batch components (1 of 2)" and lists five components with their descriptions:

- **Batch jobs**
 - Series of definitions that direct the execution of one or more batch applications, and specifies their input and output
- **Batch container**
 - Provides the batch execution environment, including services such as checkpoint and restart and job-logging
- **Job scheduler**
 - Job management control point for determining when and where jobs run
- **Batch toolkit**
 - Provides tools for the creating, packaging, and testing batch jobs
- **Endpoints**
 - Application servers that are augmented for running batch applications

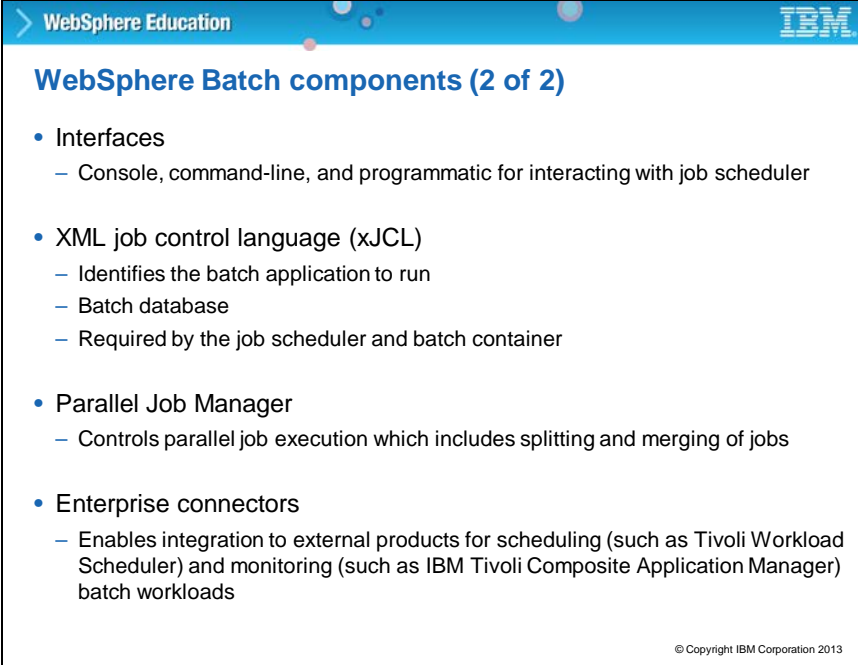
© Copyright IBM Corporation 2013

A batch job consists of a series of definitions that direct the execution of one or more batch applications, and specifies their input and output. A batch job can be composed of one or more batch steps. All steps in a job are processed sequentially to accomplish specific business functions. Batch job workloads are run in a batch container in WebSphere Application Server environments. This batch container is the main engine responsible for the execution of batch applications. It runs batch jobs under the control of an asynchronous bean, which can be thought of as a container-managed thread. The batch container ultimately processes job definitions and carries out the lifecycle of jobs.

The job scheduler is the batch component that is hosted in an application server or in an application server cluster. It provides all job management functions, such as submit, cancel, and restart. It also maintains a history of all jobs, including jobs that are waiting to run, jobs that are running, and the jobs that already ran.

The batch toolkit that is supplied with WebSphere Application Server includes tools to facilitate batch application development. It combines batch development tools into a ready-to-use environment and includes simple command-line utilities that deal with packaging applications and other tasks.

The grid endpoints are application servers that are augmented to provide a special runtime environment that batch applications need. This runtime environment is a product-provided Java EE application, the batch execution environment. The system automatically deploys this application when a batch application is installed, and it serves as an interface between the job scheduler and batch applications. It provides the runtime environment for both compute-intensive and transactional batch applications.



The slide is titled "WebSphere Batch components (2 of 2)" and is part of a presentation from WebSphere Education, as indicated by the header. It lists four main components of WebSphere Batch:

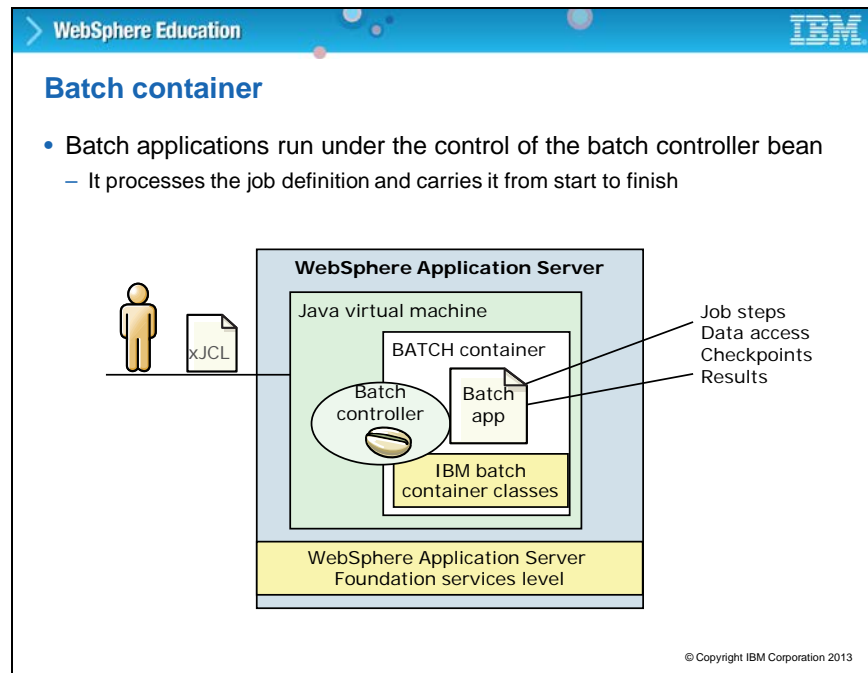
- **Interfaces**
 - Console, command-line, and programmatic for interacting with job scheduler
- **XML job control language (xJCL)**
 - Identifies the batch application to run
 - Batch database
 - Required by the job scheduler and batch container
- **Parallel Job Manager**
 - Controls parallel job execution which includes splitting and merging of jobs
- **Enterprise connectors**
 - Enables integration to external products for scheduling (such as Tivoli Workload Scheduler) and monitoring (such as IBM Tivoli Composite Application Manager) batch workloads

© Copyright IBM Corporation 2013

The job scheduler exposes three API types to access its management functions. You can use the job management console, command-line, and APIs that are available as either web services or EJBs.

Jobs are described by using an XML dialect called XML Job Control Language (xJCL). This dialect has constructs for expressing all of the information that is needed for both compute-intensive and batch jobs, although some elements of xJCL are only applicable to compute-intensive or batch jobs. The job description identifies which application to run and its input and output. The xJCL definition of a job is not part of the batch application. This definition is constructed separately and submitted to the job scheduler to run. The job scheduler uses information in the xJCL to determine where and when the job runs.

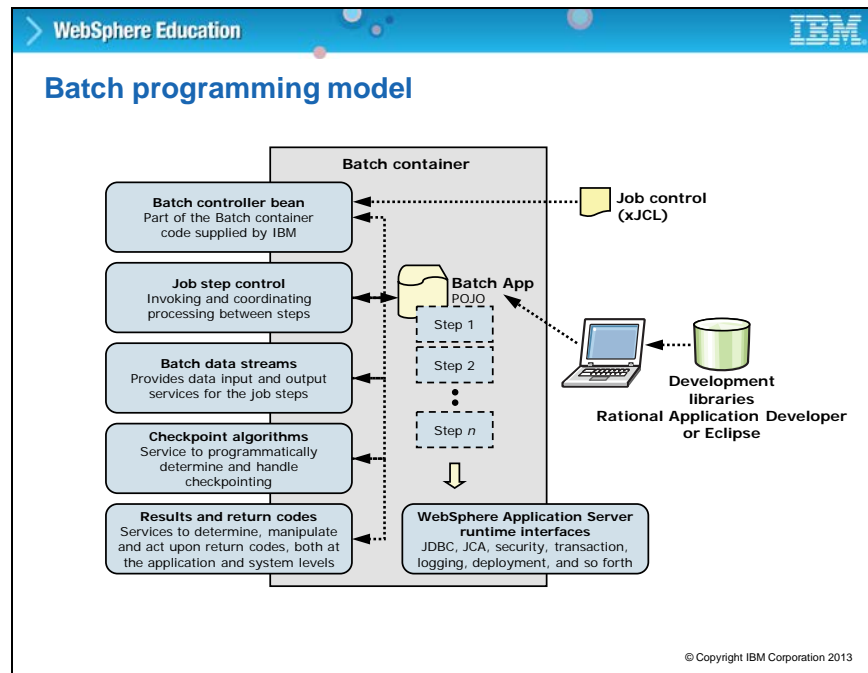
The batch container stores checkpoint information for transactional batch applications. The parallel job manager and enterprise connectors are described in more detail later in this unit.



The batch container is the heart of the batch application support that is provided in WebSphere Application Server. It runs a batch job under the control of an asynchronous bean, which can be thought of as a container-managed thread. The batch container ultimately processes a job definition and carries out the lifecycle of a job.

The batch container provides these services:

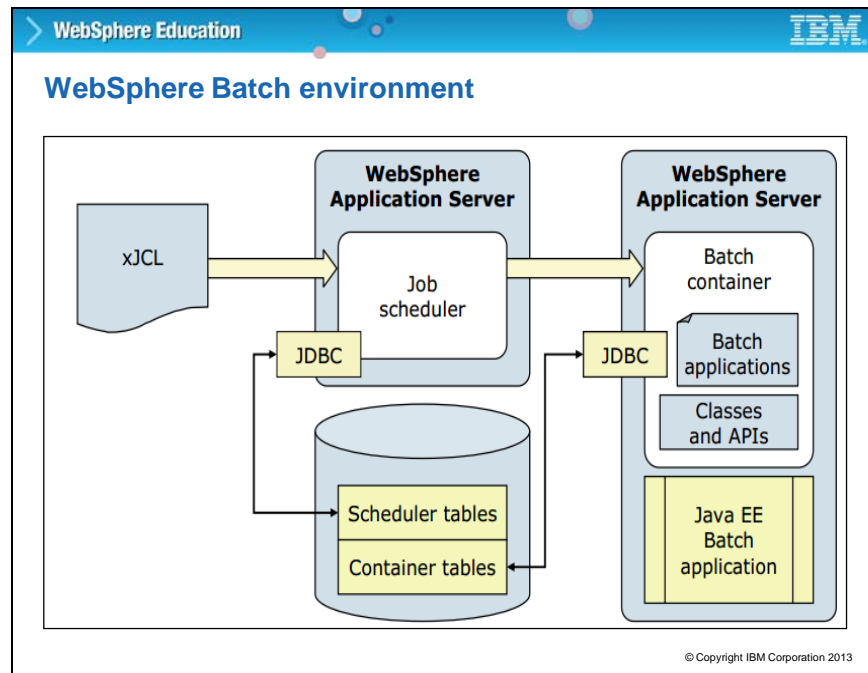
- Check pointing, which involves resuming batch work from a selected position.
- Result processing, which involves intercepting and processing step and job return codes.
- Batch data stream management, which involves reading, positioning, and repositioning data streams to files, relational databases, z/OS data sets, and many other different types of input and output resources.



The batch programming model consists of four principal interfaces, two of which are essential to building a batch application, and two that are optional and intended for advanced scenarios. The first essential item is the batch job step, which defines the interaction between the batch container and the batch application. The other essential item is the batch data stream. The batch data stream abstracts a particular input source or output destination for a batch application and defines the interaction between the batch container and a concrete BatchDataStream implementation.

An optional checkpoint policy algorithm defines the interaction between the batch container and a custom checkpoint policy implementation. A checkpoint policy is used to determine when the batch container will checkpoint a running batch job to enable a restart to occur after a planned or unplanned interruption. WebSphere Application Server includes two ready-to-use checkpoint policies.

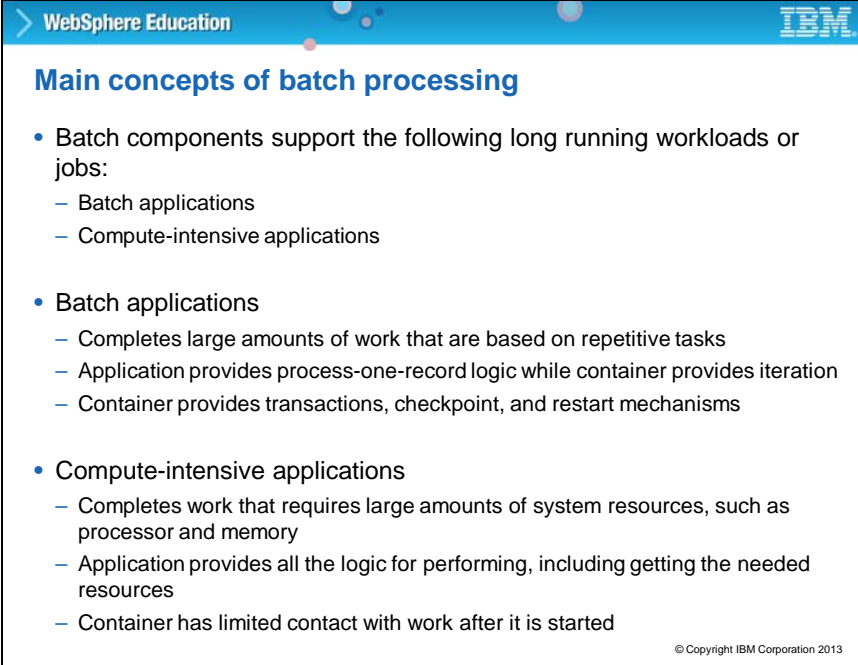
An optional results algorithm defines the interaction between the batch container and a custom results algorithm. The purpose of the results algorithm is to provide the overall return code for a job. The algorithm has visibility to the return codes from each of the job steps. WebSphere Application Server includes one ready-to-use results algorithm.



A typical batch environment consists of a job scheduler, batch container, batch applications, jobs, interfaces for management functions, and database tables. A diagram of a typical batch environment is displayed on the slide.

The job scheduler provides the job management functions such as submit, cancel, restart, and others. The job scheduler accepts job submissions and determines where to run them. As part of managing jobs, the job scheduler stores job information in an external database. The job database can be any relational database that WebSphere Application Server supports. If the job scheduler is clustered, the database must be a network database, such as DB2. Configurations for the job scheduler include the selection of the deployment target, data source JNDI name, database schema name, and endpoint job log location that is configured for the job scheduler.

Jobs are described by using job control language called XML Job Control Language (xJCL), which identifies the batch application to run and the inputs and outputs. The Batch container provides the execution environment for batch jobs. There can be multiple Batch containers in a WebSphere cell. Batch applications are regular WebSphere Java Enterprise Edition (Java EE) applications that are deployed as enterprise archive (EAR) files.



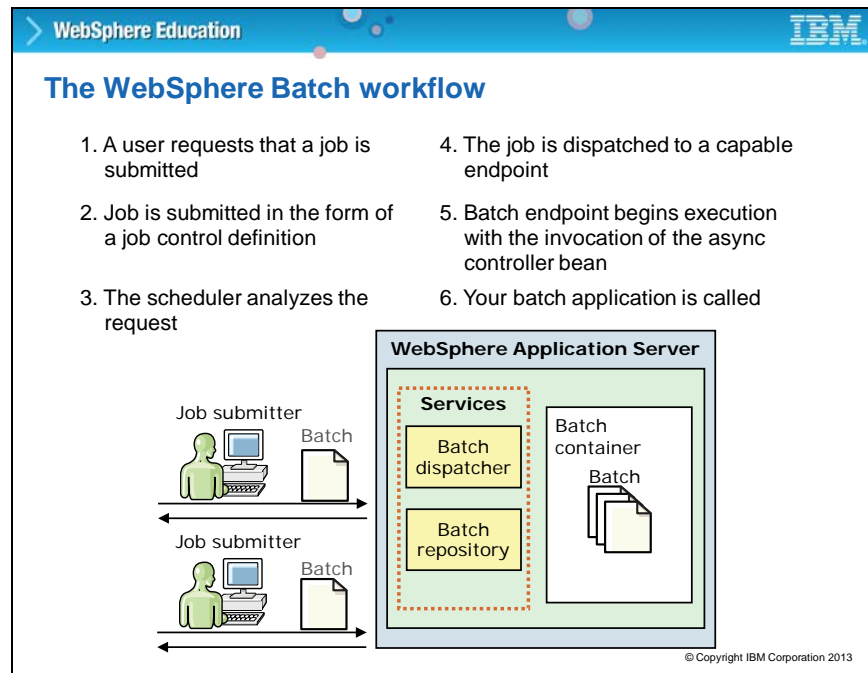
The slide is titled "Main concepts of batch processing" and is part of a "WebSphere Education" presentation, as indicated by the header. It features a blue header bar with the "WebSphere Education" text and the IBM logo. The main content is a bulleted list of concepts. The first bullet point is "Batch components support the following long running workloads or jobs:", which includes two sub-points: "Batch applications" and "Compute-intensive applications". The second bullet point is "Batch applications", which includes three sub-points: "Completes large amounts of work that are based on repetitive tasks", "Application provides process-one-record logic while container provides iteration", and "Container provides transactions, checkpoint, and restart mechanisms". The third bullet point is "Compute-intensive applications", which includes three sub-points: "Completes work that requires large amounts of system resources, such as processor and memory", "Application provides all the logic for performing, including getting the needed resources", and "Container has limited contact with work after it is started". A small copyright notice "© Copyright IBM Corporation 2013" is located at the bottom right of the slide.

- Batch components support the following long running workloads or jobs:
 - Batch applications
 - Compute-intensive applications
- Batch applications
 - Completes large amounts of work that are based on repetitive tasks
 - Application provides process-one-record logic while container provides iteration
 - Container provides transactions, checkpoint, and restart mechanisms
- Compute-intensive applications
 - Completes work that requires large amounts of system resources, such as processor and memory
 - Application provides all the logic for performing, including getting the needed resources
 - Container has limited contact with work after it is started

© Copyright IBM Corporation 2013

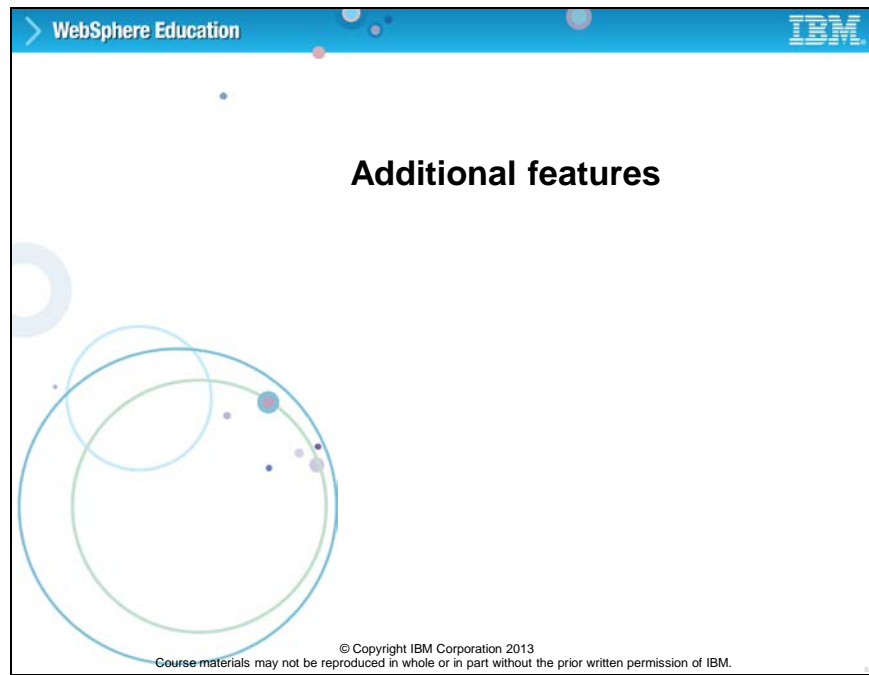
WebSphere runs batch applications that are written in Java and implement a WebSphere batch programming model. They are packed as EAR files and are deployed to the batch container hosted in an application server or cluster. Batch applications are run non-interactively in the background. Batch applications implement one of two programming models: transactional batch and compute-intensive applications. Transactional batch applications are applications that handle large amounts of work that is based on repetitive tasks, such as processing many records. The container provides transactions, checkpoint, and restart mechanisms.

Compute-intensive applications handle work that requires large amounts of system resources, in particular processing and memory. The application is responsible for providing all the logic for doing the necessary work. The container has limited contact with work after it starts.

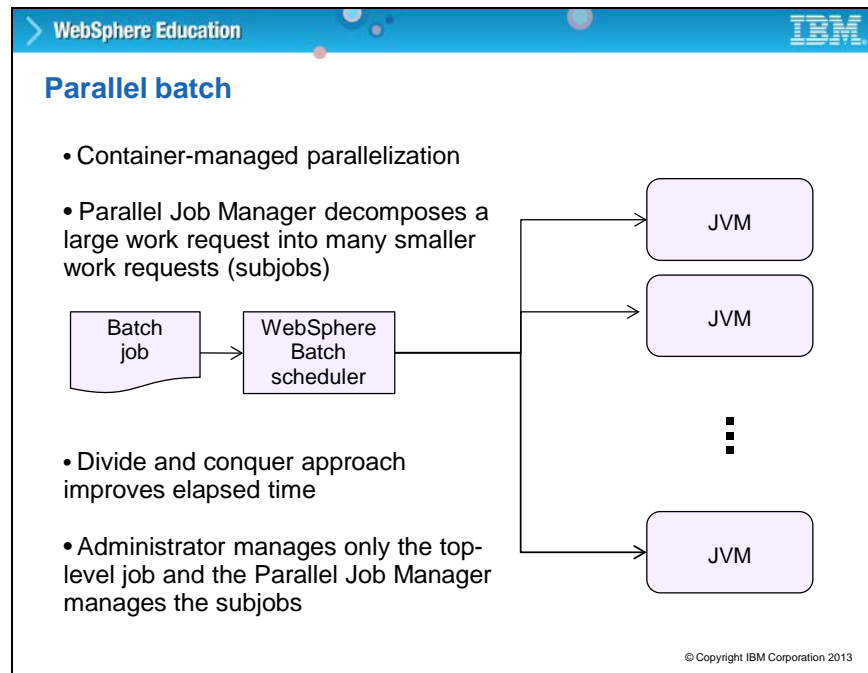


Batch jobs are submitted to the system by using the Job Management Console or programmatically by way of Enterprise JavaBeans (EJB), Java Message Service (JMS), or web services. Each job is submitted in the form of an XML Job Control Language (xJCL) document. The Job Dispatcher then selects the best endpoint application server for job execution that is based on several different metrics. The endpoint application server sets up the jobs in the Batch Container and runs the batch steps that are based on the definitions in the xJCL. While the job is running, the Job Dispatcher aggregates job logs and provides lifecycle management functions.

Slide 16



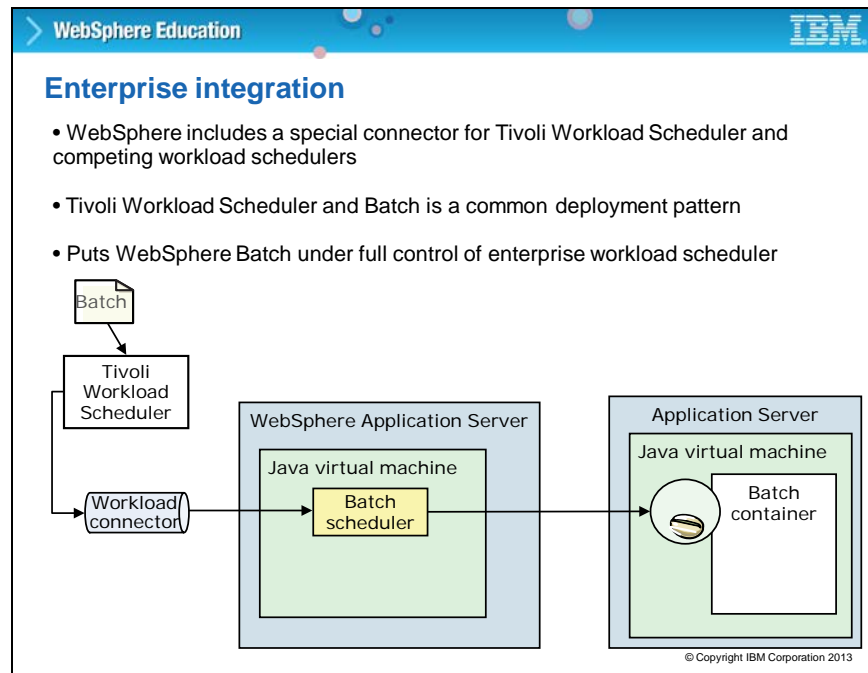
Topic: Other features. The Feature Pack for Modern Batch, available with WebSphere Application Server V7, began as a good starting point by providing consistent programming models and tools. However, it did not contain advanced batch capabilities that are required in enterprise settings. WebSphere Application Server V8.5 fills these gaps by offering new features that are introduced in this topic.



Parallel processing of the batch load across the enterprise infrastructure is a new feature in WebSphere Application Server V8.5 Batch. An enterprise application server environment consists of multiple servers that are working together to provide a high performance and highly available infrastructure. Simple batch processing runs on a single server, so it is unable to optimally use the available application server infrastructure. WebSphere Batch provides a parallel job manager that enables container-managed parallel processing, thus enabling the batch job to be divided across the various servers for efficiency. Running the job as one job from a batch perspective provides operational control. Splitting the task internally across the servers ensures optimal utilization of the hardware resources and also reduces the time that is taken to complete the task.

The Parallel Job Manager provides the support for building transactional batch applications as a job, and then divides the job into subordinate jobs. The subordinate jobs can run independently and in parallel. The Parallel Job Manager is used to submit and manage the transactional batch jobs.

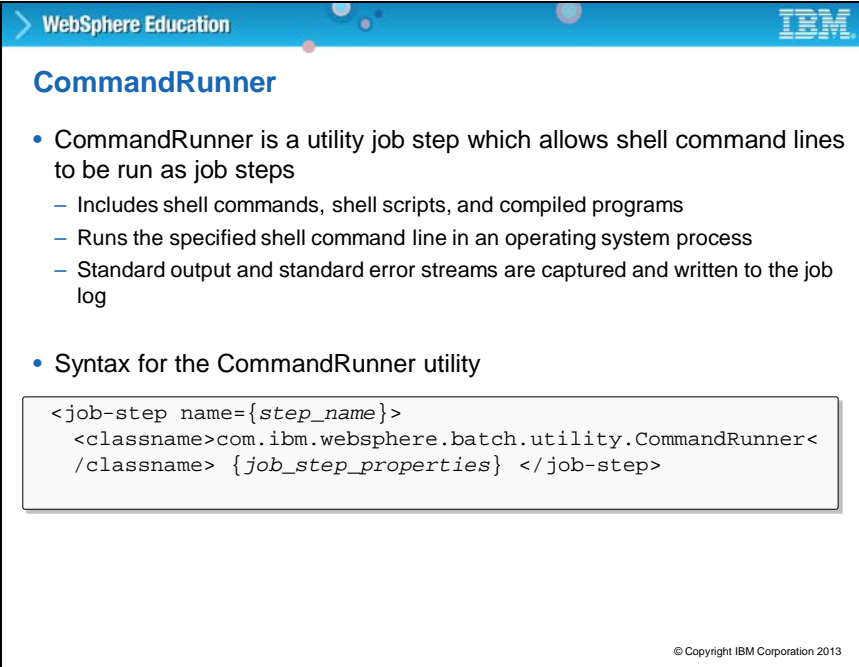
WebSphere Batch allows the running of parallel jobs and their subjobs in the same JVM if needed. This capability is especially useful when subjobs are typically short in duration and the overhead of distributing the subjobs across servers is much larger compared to the actual work to complete.




WebSphere Application Server V8.5 Batch provides an integration capability with many external workload schedulers, such as Tivoli Workload Scheduler. Since an external scheduler does not know how to directly manage batch jobs, a proxy model is used. The proxy model uses a regular JCL job to submit and monitor the batch job. The JCL job step calls a special program that batch provides, named WSGRID. WSGRID enables Tivoli Workload Scheduler and similar products to dispatch and monitor batch activities.

Tivoli Workload Scheduler helps you establish an enterprise workload automation backbone by driving composite workloads according to business policies. It provides automation capabilities to control the processing of a production workload, including batch and online services. Tivoli Workload Scheduler functions as an automatic driver for composite workloads by maximizing the velocity of workloads, optimizing IT resource usage, and resolving workload dependencies. It extends the scope for integrated application and systems management by driving workloads on multiple, heterogeneous platforms and ERP systems.

Slide 19



The slide is titled 'WebSphere Education' and 'CommandRunner'. It contains two main bullet points. The first describes CommandRunner as a utility job step for running shell commands, listing its features: including shell commands/scripts/programs, running in an OS process, and capturing output/error streams. The second bullet point is 'Syntax for the CommandRunner utility', followed by a code block showing the XML structure for a job step.

WebSphere Education 

CommandRunner

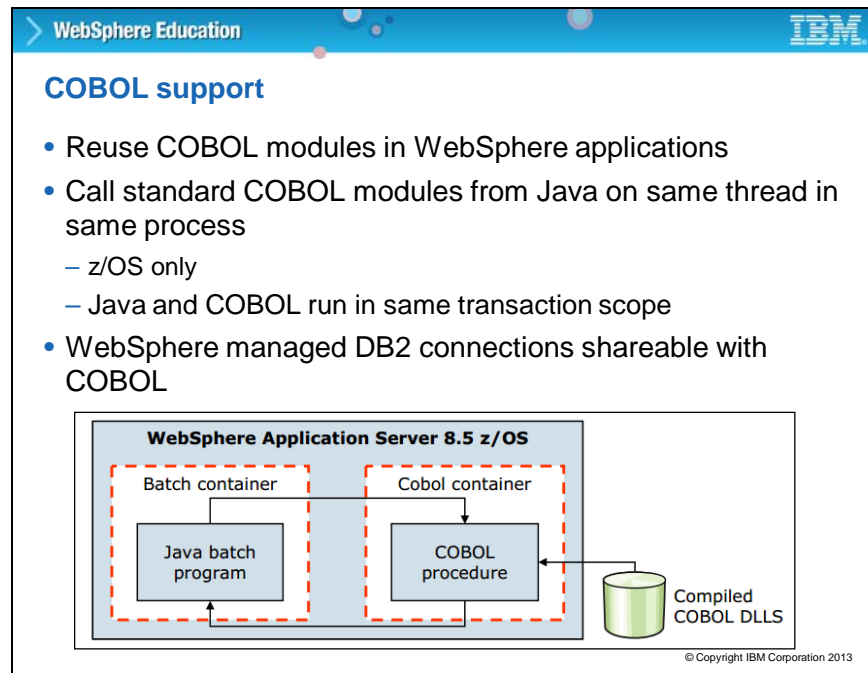
- CommandRunner is a utility job step which allows shell command lines to be run as job steps
 - Includes shell commands, shell scripts, and compiled programs
 - Runs the specified shell command line in an operating system process
 - Standard output and standard error streams are captured and written to the job log
- Syntax for the CommandRunner utility

```
<job-step name={step_name}>  
  <classname>com.ibm.websphere.batch.utility.CommandRunner<  
    /classname> {job_step_properties} </job-step>
```

© Copyright IBM Corporation 2013

The CommandRunner utility job step allows shell command lines, including scripts and compiled programs to be run as job steps. The CommandRunner utility adds all job step properties, after substitution, to the process variable pool for the process in which the specified command-line runs.

Syntax for the CommandRunner utility is listed on the slide.



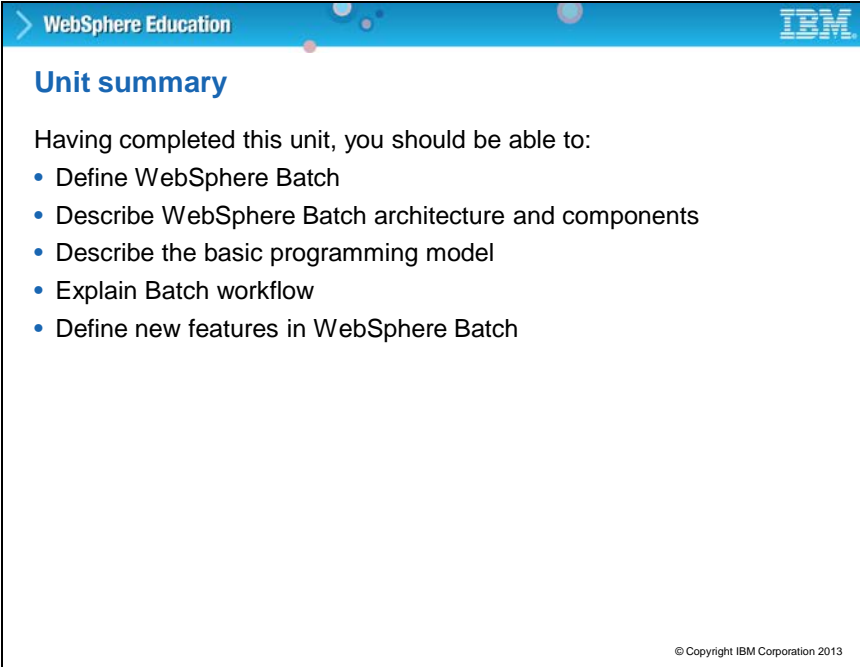
COBOL is the prevalent language to run batch-style workloads in the past, and even today there is a large existing base of COBOL code. On the z/OS platform, Java batch adds support to easily call into existing COBOL assets to run modern batch workloads.

COBOL support allows the usage of COBOL modules in WebSphere applications. With WebSphere Application Server V8.5, COBOL support includes the following key features:

- In z/OS, you can call standard COBOL modules from Java on the same thread in same process.
- Java and COBOL run in same transaction scope.
- WebSphere managed DB2 connections are shareable with COBOL.
- You can use COBOL working storage isolation per job step or per remote call.
- IBM Rational Application Developer tools are available for Java call stub generation.

The new COBOL container allows COBOL modules to be loaded into the WebSphere Application Server for z/OS address space and called directly. It provides the means of direct integration of COBOL resources into WebSphere Java processing. The container itself is implemented as a handful of DLLs and JAR files.

Slide 21



The slide is titled "Unit summary" and is part of a "WebSphere Education" presentation, as indicated by the header. It lists five learning objectives for the unit. The IBM logo is in the top right corner, and a copyright notice for IBM Corporation 2013 is in the bottom right corner.

WebSphere Education

IBM

Unit summary

Having completed this unit, you should be able to:

- Define WebSphere Batch
- Describe WebSphere Batch architecture and components
- Describe the basic programming model
- Explain Batch workflow
- Define new features in WebSphere Batch

© Copyright IBM Corporation 2013

You completed this unit.

Having completed this unit, you should be able to:

- Define WebSphere Batch
- Describe WebSphere Batch architecture and components
- Describe the basic programming model
- Explain Batch workflow
- Define new features in WebSphere Batch