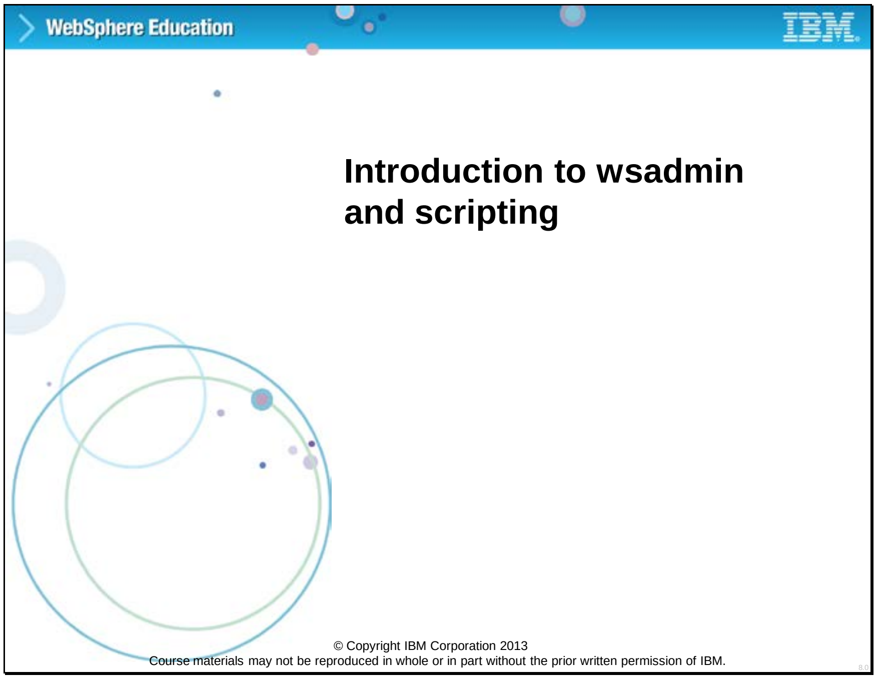
Slide 1



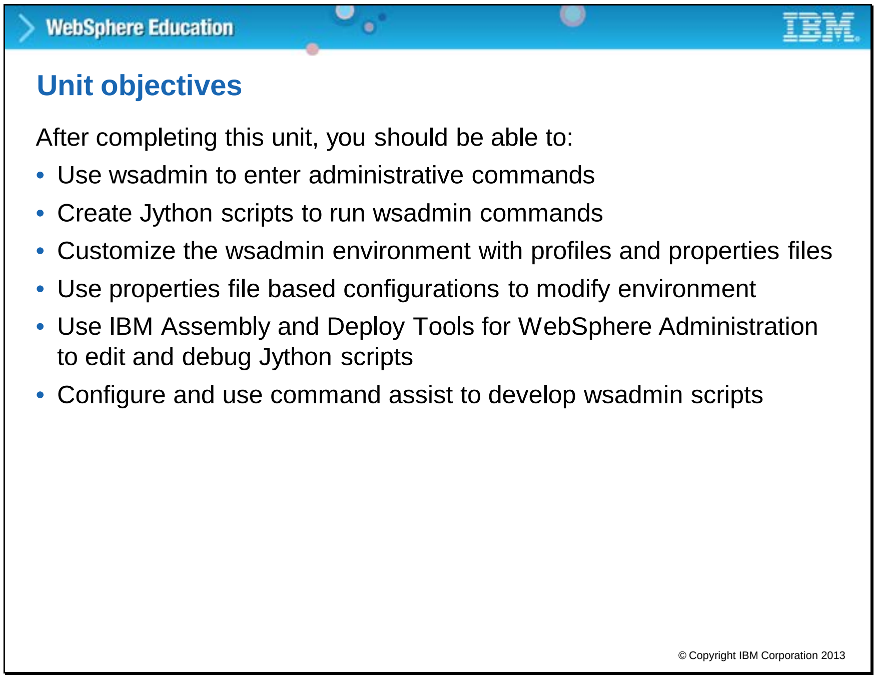
WebSphere Education

Introduction to wsadmin and scripting

Unit 12: Introduction to wsadmin and scripting

This unit covers the use of the wsadmin tool and scripting capabilities.

Slide 2


WebSphere Education
IBM

Unit objectives

After completing this unit, you should be able to:

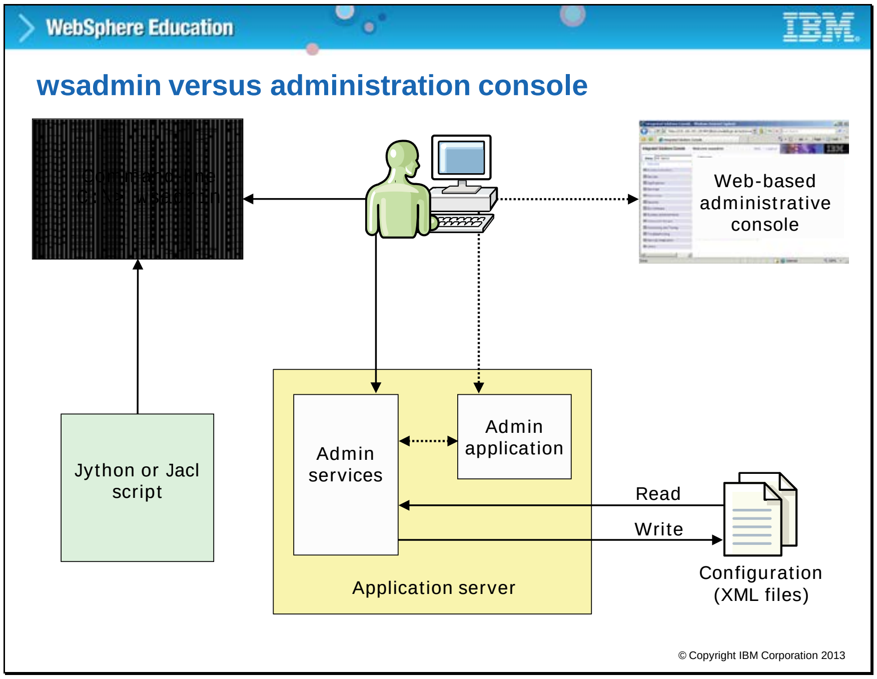
- Use wsadmin to enter administrative commands
- Create Jython scripts to run wsadmin commands
- Customize the wsadmin environment with profiles and properties files
- Use properties file based configurations to modify environment
- Use IBM Assembly and Deploy Tools for WebSphere Administration to edit and debug Jython scripts
- Configure and use command assist to develop wsadmin scripts

Title: Unit objectives

After completing this unit, you should be able to:

- Use wsadmin to enter administrative commands
- Create Jython scripts to run wsadmin commands
- Customize the wsadmin environment with profiles and property files
- Use property file based configurations to modify environment
- Use IBM Assembly and Deploy Tools for WebSphere Administration to edit and debug Jython scripts
- Configure and use command assist to develop wsadmin scripts

Slide 3



WebSphere Education

wsadmin versus administration console

Command-line
C:\> wsadmin

Web-based administrative console

Jython or Jacl script

Admin services

Admin application

Application server

Read

Write

Configuration (XML files)

Title: wsadmin versus administrative console

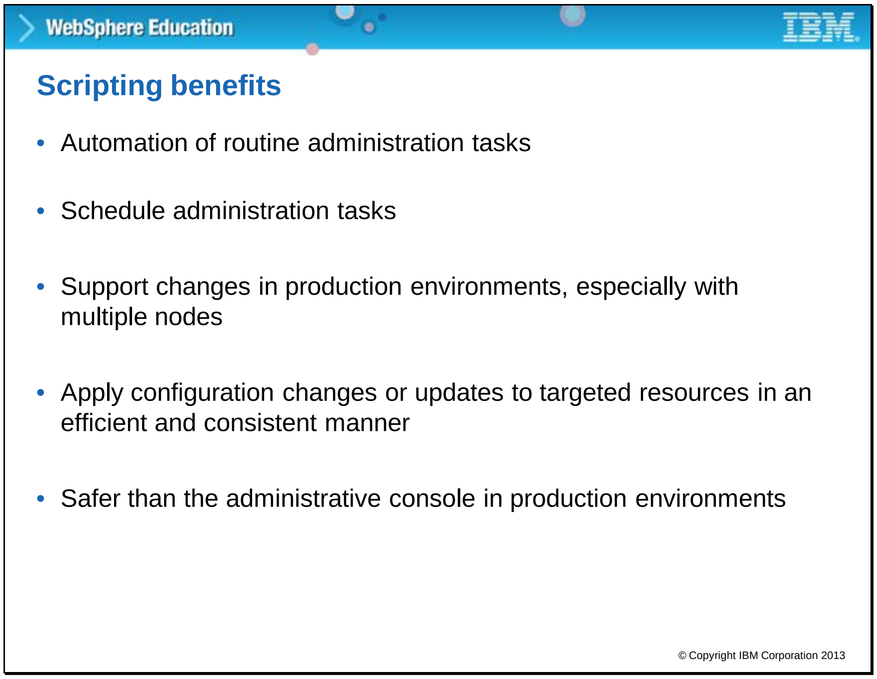
You can use the administrative console or wsadmin tool to manage an application server as well as the configuration, application deployment, and server runtime operations.

The administrative console is a graphical interface that allows you to manage your applications and do system administration tasks for your WebSphere Application Server environment. The administrative console runs in your web browser.

The wsadmin tool is a command-line client that runs Jython or Jacl scripts that allow you to manage your applications and do system administration tasks for your WebSphere Application Server environment.

The administrative console or wsadmin tool can access (read) and modify (write) a set of XML configuration files that are used to describe the application server environment.

Slide 4



WebSphere Education

Scripting benefits

- Automation of routine administration tasks
- Schedule administration tasks
- Support changes in production environments, especially with multiple nodes
- Apply configuration changes or updates to targeted resources in an efficient and consistent manner
- Safer than the administrative console in production environments

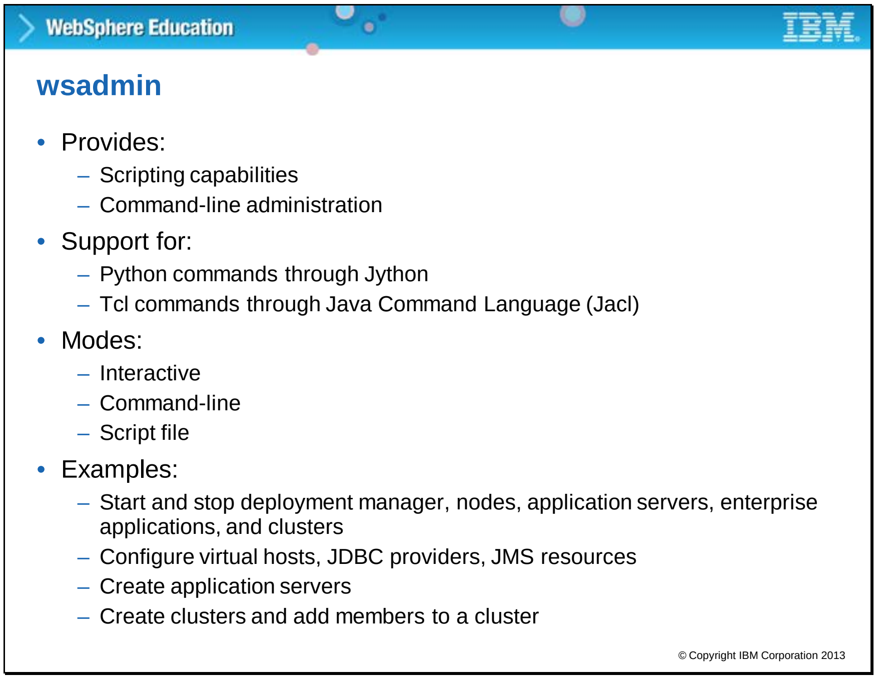
Title: Scripting benefits

There are several benefits of using scripting to manage an application server such as:

- Automation of routine administration tasks without relying on an operator to run the administrative console.
- Scheduling administration tasks to run at times when it might be inconvenient to have an operator to run the administrative console.
- Supporting changes in production environments, especially with multiple nodes.
- Applying configuration changes and updates to all targeted resources in an efficient and consistent manner.

Using wsadmin is safer than using the administrative console in production environments.

Slide 5



WebSphere Education

IBM

wsadmin

- Provides:
  - Scripting capabilities
  - Command-line administration
- Support for:
  - Python commands through Jython
  - Tcl commands through Java Command Language (Jacl)
- Modes:
  - Interactive
  - Command-line
  - Script file
- Examples:
  - Start and stop deployment manager, nodes, application servers, enterprise applications, and clusters
  - Configure virtual hosts, JDBC providers, JMS resources
  - Create application servers
  - Create clusters and add members to a cluster

© Copyright IBM Corporation 2013

Title: wsadmin

The WebSphere administrative scripting program (wsadmin) is a powerful, non-graphical command interpreter environment that enables you to run administrative operations in a scripting language. The wsadmin tool is intended for production environments and unattended operations. The wsadmin tool provides both scripting capabilities and command-line administration.

The wsadmin tool uses the Bean Scripting Framework (BSF), which supports various scripting languages, to configure and control your WebSphere Application Server installation. The wsadmin tool provides support for:

- Python commands through Jython
- Tcl commands through Java Command Language Jacl

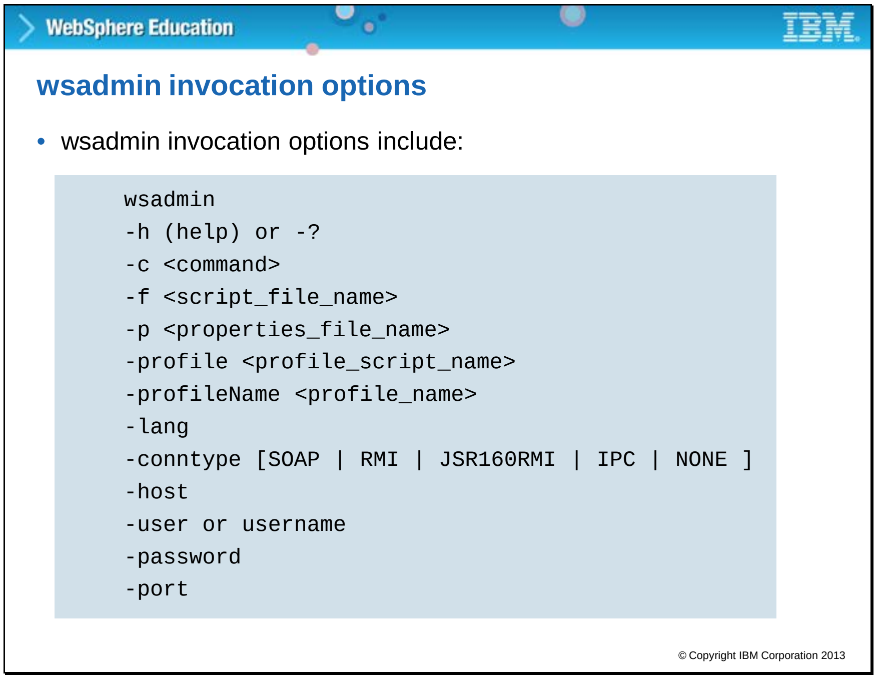
The wsadmin launcher makes administrative objects available through language-specific interfaces. Scripts use these objects for application management, configuration, operational control, and communication with MBeans running in WebSphere server processes. Three modes are used to start wsadmin:

- Interactive
- Command line
- Script file

You can use the wsadmin tool to do the same tasks that you can do by using the administrative console. You can use the wsadmin tool to manage a WebSphere Application Server V8.5 environment. Examples of tasks include:

- Start and stop deployment manager, nodes, application servers, applications, and clusters.
- Configure virtual hosts, JDBC providers, and JMS resources.
- Create application servers.
- Create clusters and add members to a cluster.

Slide 6



WebSphere Education
IBM

wsadmin invocation options

- wsadmin invocation options include:

```
wsadmin
-h (help) or -?
-c <command>
-f <script_file_name>
-p <properties_file_name>
-profile <profile_script_name>
-profileName <profile_name>
-lang
-conntype [SOAP | RMI | JSR160RMI | IPC | NONE ]
-host
-user or username
-password
-port
```

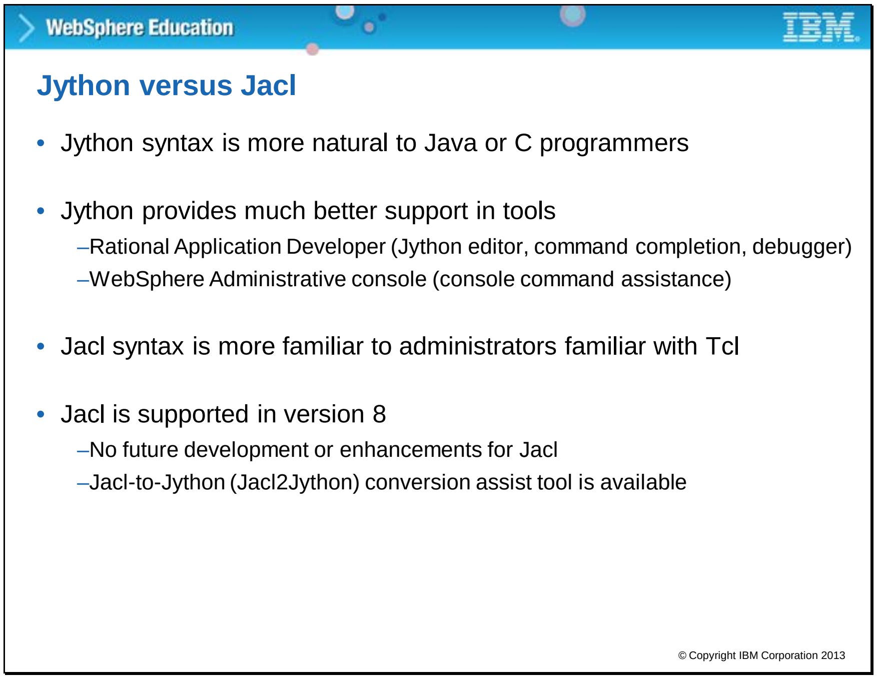
Title: wsadmin invocation options

There are several wsadmin invocation options available.

-h, -help, -?	Provides syntax help
-c <command>	Specifies to run a single command
-f <script_file_name>	Specifies a script to run

For a complete description of all the options, see the information center.

Slide 7



WebSphere Education

IBM

Jython versus Jacl

- Jython syntax is more natural to Java or C programmers
- Jython provides much better support in tools
  - –Rational Application Developer (Jython editor, command completion, debugger)
  - –WebSphere Administrative console (console command assistance)
- Jacl syntax is more familiar to administrators familiar with Tcl
- Jacl is supported in version 8
  - –No future development or enhancements for Jacl
  - –Jacl-to-Jython (Jacl2Jython) conversion assist tool is available

Title: Jython versus Jacl

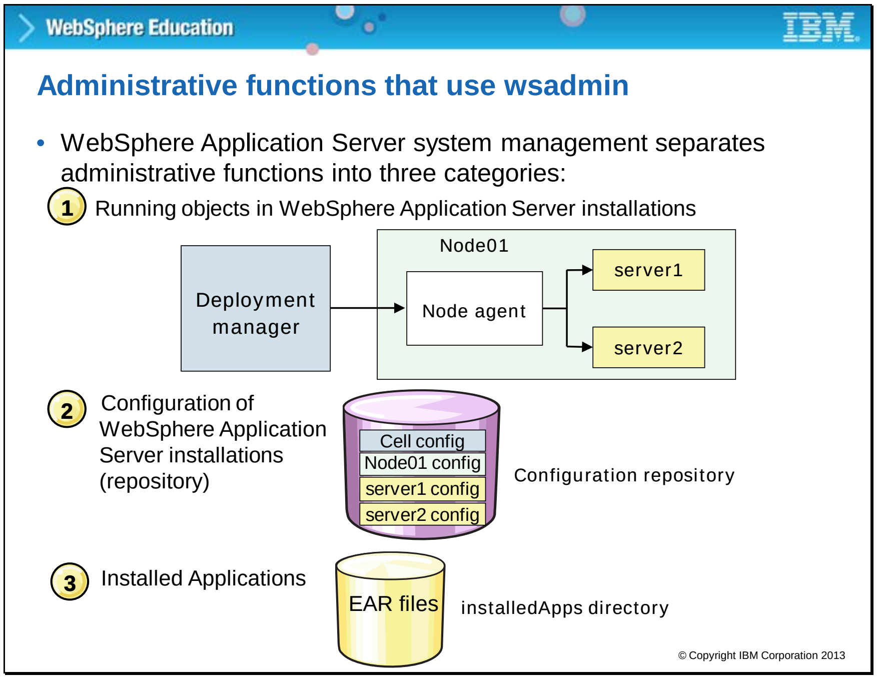
The wsadmin tool supports both Jython and Jacl scripting languages.

Jython is an alternative implementation of Python and is written entirely in Java. Jython syntax might seem more natural to Java or C programmers. Future investment and strategic direction is going to be focused on the Jython language. Jython has better tool support.

Jacl is an alternative implementation of Tcl and is written entirely in Java code. Jacl might seem more familiar to administrators familiar with Tcl. The Jacl language is stabilized in WebSphere Application Server V7.

Jacl-to-Jython (Jacl2Jython) is a conversion utility that converts Jacl syntax wsadmin scripts into equivalent Jython syntax wsadmin scripts.

Slide 8



WebSphere Education

IBM

Administrative functions that use wsadmin

- WebSphere Application Server system management separates administrative functions into three categories:

1 Running objects in WebSphere Application Server installations

2 Configuration of WebSphere Application Server installations (repository)

3 Installed Applications

© Copyright IBM Corporation 2013

Title: Administrative functions that use wsadmin

WebSphere Application Server system management separates administration functions into three categories:

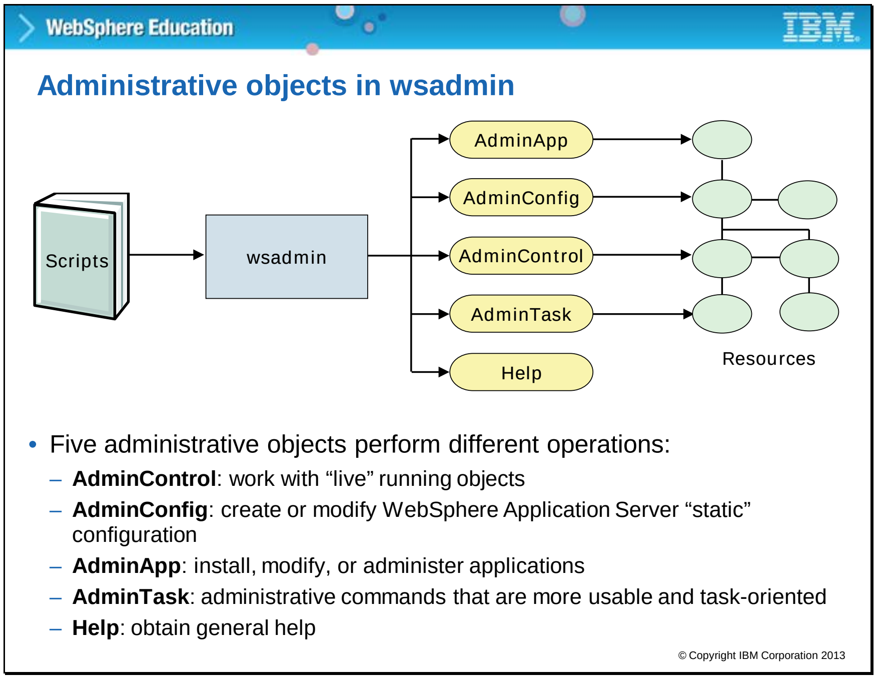
- Running objects in WebSphere Application Server installations
- Configuration of WebSphere Application Server installations (repository)
- Managing installed applications

Management Beans running in the WebSphere Application Server manage the runtime environment.

Configuration data is stored in several different XML files, which the server run time reads when it starts, and responds to the component settings stored there. The configuration data includes the settings for the run time, such as Java virtual machine (JVM) options, thread pool sizes, container settings, and port numbers the server uses. Other configuration files define the Java EE resources to which the server connects to obtain data for the application logic.

Administrative functions also manage installed application ear files that are typically stored in the installedApps directory.

Slide 9



Administrative objects in wsadmin

- Five administrative objects perform different operations:
  - **AdminControl**: work with “live” running objects
  - **AdminConfig**: create or modify WebSphere Application Server “static” configuration
  - **AdminApp**: install, modify, or administer applications
  - **AdminTask**: administrative commands that are more usable and task-oriented
  - **Help**: obtain general help

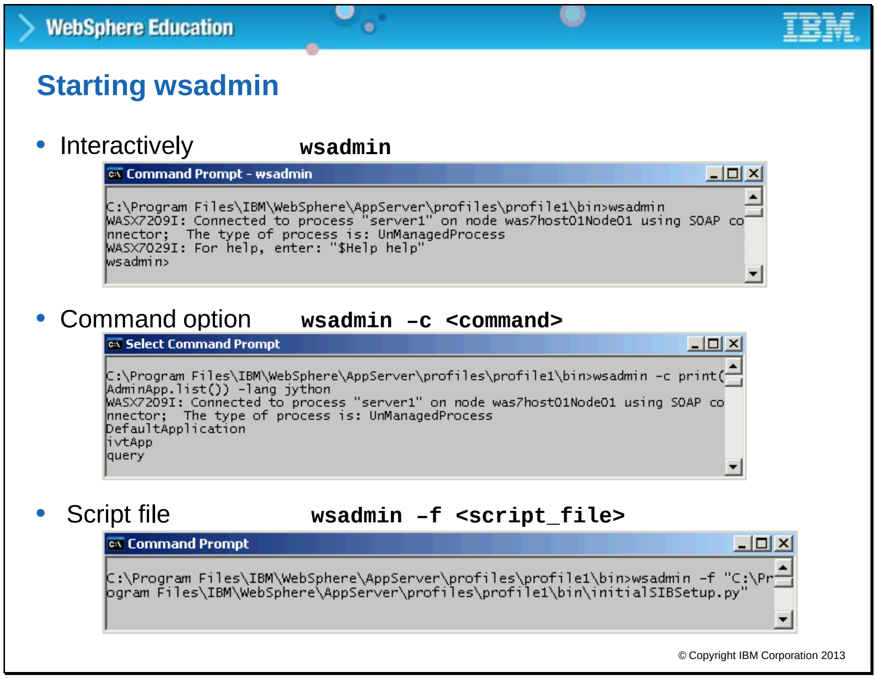
Title: Administrative objects in wsadmin

The wsadmin tool acts as an interface to Java objects by using scripts to access resources. The tool uses the same interface (through JMX) as the administrative console to make configuration changes and to control servers.

Five objects are available when you use scripts:

- Use **AdminControl** to run operational commands. This object allows you to work with “live” running objects, run traces, and make data type conversions.
- Use **AdminConfig** to run configurational commands to create or modify WebSphere Application Server ”static” configurational elements.
- Use **AdminApp** to install, modify, or administer applications.
- Use **AdminTask** to run administrative commands that are easier to use and more task-oriented.
- Use **Help** to obtain general help.

Slide 10



Starting wsadmin

- Interactively **`wsadmin`**

```
C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin>wsadmin
WASX7209I: Connected to process "server1" on node was7host01Node01 using SOAP co
nnector;  The type of process is: UnManagedProcess
WASX7029I: For help, enter: "$Help help"
wsadmin>
```

- Command option **`wsadmin –c <command>`**

```
C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin>wsadmin -c print(
AdminApp.list()) -lang jython
WASX7209I: Connected to process "server1" on node was7host01Node01 using SOAP co
nnector;  The type of process is: UnManagedProcess
DefaultApplication
ivtApp
query
```

- Script file **`wsadmin –f <script_file>`**

```
C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin>wsadmin -f "C:\Pr
ogram Files\IBM\WebSphere\AppServer\profiles\profile1\bin\initialSIBSetup.py"
```

Title: Starting wsadmin

The wsadmin tool can be started in three ways:

- **Interactively by using wsadmin**

  You can run wsadmin with options such as -f and -c, or without an option. The wsadmin tool starts and provides an interactive shell with a wsadmin prompt. The screen capture demonstrates how to use wsadmin in the interactive mode.

- **Using the -c <command> option**

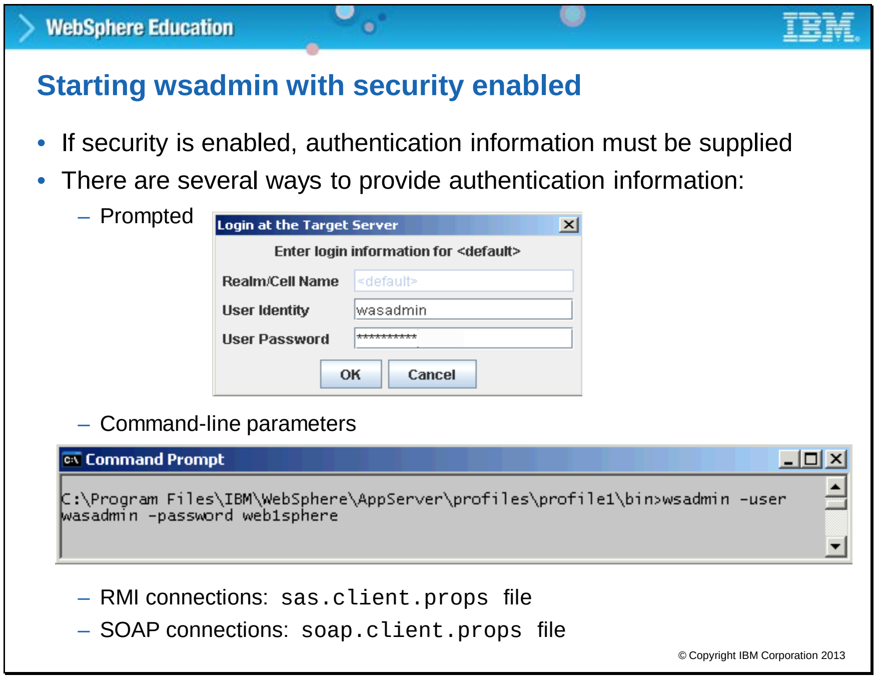
  Run the wsadmin tool with the noninteractive -c option. The screen capture demonstrates how to use the –c option to run a single command.

- **Using the -f <script_file> option**

  Run the wsadmin tool with the noninteractive **-f** option and place the commands that you want to run a script file. The screen capture demonstrates how to use to run a Jython script.

  When running **wsadmin** with the **-f** option, you do not need to specify the script language with the **-lang** command-line option. The wsadmin command recognizes the script language by looking at the extension of the file name (.py for Jython and .jacl for Jacl).

Slide 11



Starting wsadmin with security enabled

- If security is enabled, authentication information must be supplied
- There are several ways to provide authentication information:
  - Prompted

Login at the Target Server
Enter login information for <default>
OK
Cancel

Realm/Cell Name	<default>
User Identity	wasadmin
User Password	***********

  - Command-line parameters

```
C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin>wsadmin -user
wasadmin -password web1sphere
```

  - RMI connections: `sas.client.props` file
  - SOAP connections: `soap.client.props` file

Title: Starting wsadmin with security enabled

WebSphere Application Server administrative security is enabled by default.

If security is enabled, authentication data must be supplied. There are several ways to provide authentication data.

If security is enabled and you do not provide credentials when you start wsadmin, you are prompted to provide them. You can profile authentication data by using command-line parameters.

-user or **-username** can be used interchangeably to provide a user ID

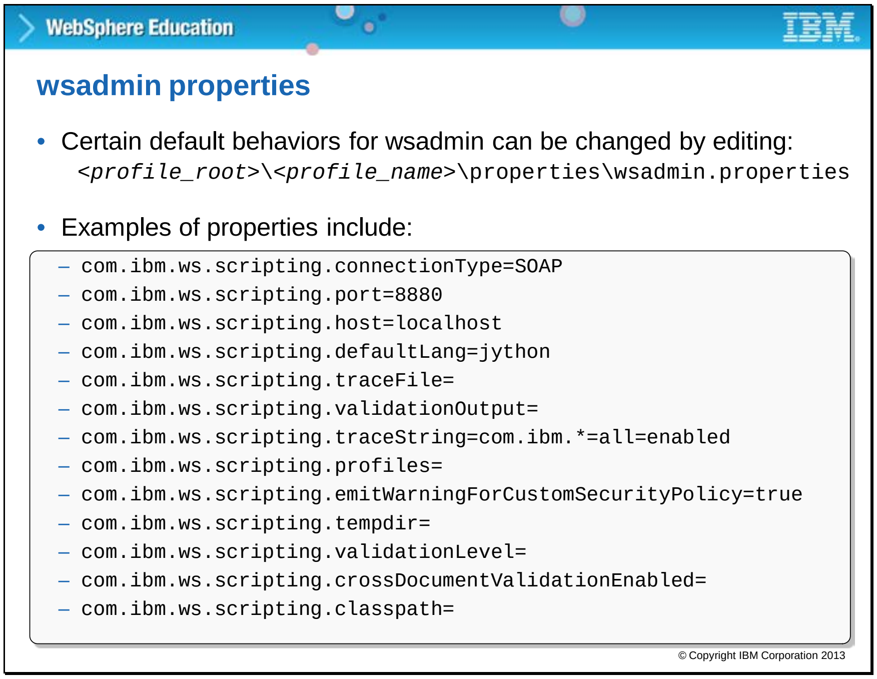
-password is used to provide a password

The screen capture demonstrates the use of command-line parameters to pass authentication credentials.

For RMI connections, authentication credentials can be provided in the **sas.client.props** file. This file is in the profile properties directory.

For SOAP connections, authentication credentials can be provided in the **soap.client.props** file. This file is in the profile properties directory.

Slide 12



WebSphere Education

wsadmin properties

- Certain default behaviors for wsadmin can be changed by editing:
  `<profile_root>\<profile_name>\properties\wsadmin.properties`
- Examples of properties include:

```
– com.ibm.ws.scripting.connectionType=SOAP
– com.ibm.ws.scripting.port=8880
– com.ibm.ws.scripting.host=localhost
– com.ibm.ws.scripting.defaultLang=jython
– com.ibm.ws.scripting.traceFile=
– com.ibm.ws.scripting.validationOutput=
– com.ibm.ws.scripting.traceString=com.ibm.*=all=enabled
– com.ibm.ws.scripting.profiles=
– com.ibm.ws.scripting.emitWarningForCustomSecurityPolicy=true
– com.ibm.ws.scripting.tempdir=
– com.ibm.ws.scripting.validationLevel=
– com.ibm.ws.scripting.crossDocumentValidationEnabled=
– com.ibm.ws.scripting.classpath=
```

Title: wsadmin properties

Scripting administration uses several Java properties files. Properties files can be used to control your system configurations. Before any properties file is specified on the command line, three levels of default properties files are loaded. These properties files include:

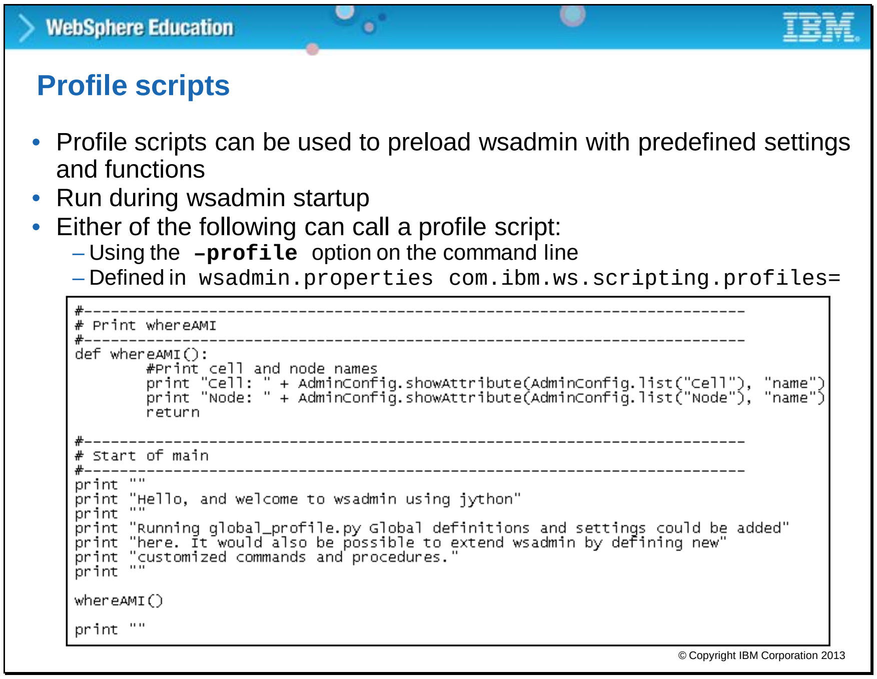
1. **Installation default file** represents an installation default that is in the profile properties directory for each application server profile and is called wsadmin.properties.
2. **User default file** represents a user default and is in the Java user.home property. This properties file is also called from the wsadmin.properties file.
3. **Properties file** is a properties file that is pointed to by the WSADMIN_PROPERTIES environment variable. This environment variable is defined in the environment where the wsadmin tool starts.

If one or more of these properties files are present, they are interpreted before any properties file that is present on the command line. The three levels of properties files load in the order that they are specified. The properties file that is loaded last overrides the ones that are loaded earlier.

Certain default behaviors for wsadmin can be changed by editing the wsadmin.properties file.

For complete details, see the information center.

Slide 13



WebSphere Education

Profile scripts

- Profile scripts can be used to preload wsadmin with predefined settings and functions
- Run during wsadmin startup
- Either of the following can call a profile script:
  - Using the **`-profile`** option on the command line
  - Defined in `wsadmin.properties com.ibm.ws.scripting.profiles=`

```
#---------------------------------------------------------------------------
# Print whereAMI
#---------------------------------------------------------------------------
def whereAMI():
        #Print cell and node names
        print "Cell: " + AdminConfig.showAttribute(AdminConfig.list("Cell"), "name")
        print "Node: " + AdminConfig.showAttribute(AdminConfig.list("Node"), "name")
        return

#---------------------------------------------------------------------------
# Start of main
#---------------------------------------------------------------------------
print ""
print "Hello, and welcome to wsadmin using jython"
print ""
print "Running global_profile.py Global definitions and settings could be added"
print "here. It would also be possible to extend wsadmin by defining new"
print "customized commands and procedures."
print ""

whereAMI()

print ""
```

Title: Profile scripts

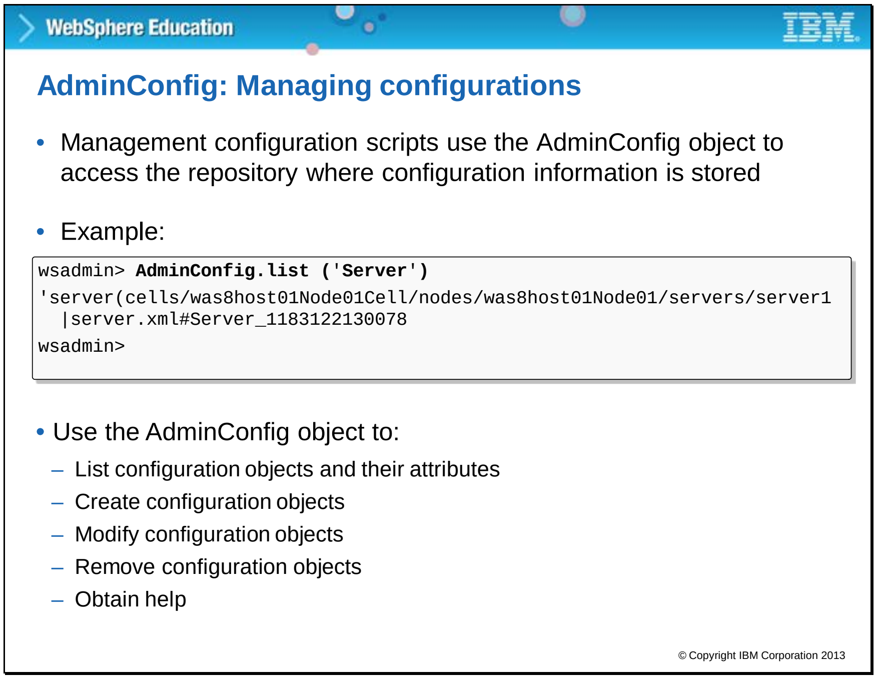
Profile scripts can be used to preload wsadmin with predefined settings and functions. Profile scripts are run during wsadmin startup. They can be called in one of two ways:

- Use the **-profile** option on the command line.
- Define the profile script in wsadmin.properties by using the **com.ibm.ws.scripting.profiles= property**.

The profile script runs before other commands, or scripts. If you specify **-c**, then the profile script runs before it starts this command. If you specify **-f**, then the profile script runs before it runs the script. In interactive mode, you can use the profile script to run any standard initialization that you want.

You can specify multiple **-profile** options on the command line and they start in the order that you supply them. The example on this slide shows a Jython script named **global_profile.py.**

Slide 14



WebSphere Education

AdminConfig: Managing configurations

- Management configuration scripts use the AdminConfig object to access the repository where configuration information is stored
- Example:

```
wsadmin> AdminConfig.list ('Server')
'server(cells/was8host01Node01Cell/nodes/was8host01Node01/servers/server1
  |server.xml#Server_1183122130078
wsadmin>
```

- Use the AdminConfig object to:
  - List configuration objects and their attributes
  - Create configuration objects
  - Modify configuration objects
  - Remove configuration objects
  - Obtain help

© Copyright IBM Corporation 2013

Title: AdminConfig: Managing configurations

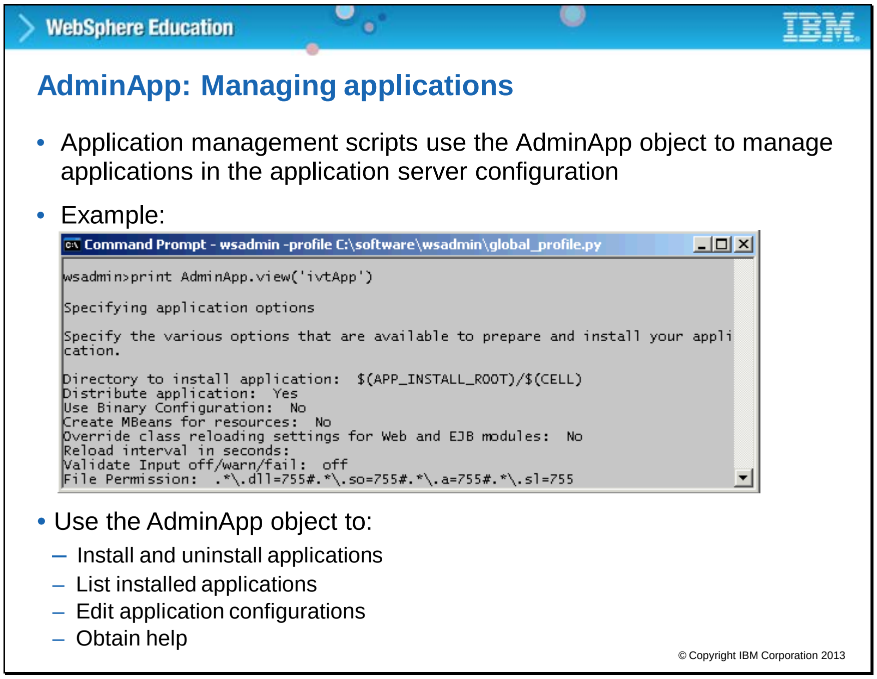
Management configuration scripts use the **AdminConfig** object to access the repository where configuration information is stored.

The screen capture shows an example of how to use the AdminConfig object to information about a server.

You can use the AdminConfig object to:

- List configuration objects and their attributes
- Create configuration objects
- Modify configuration objects
- Remove configuration objects
- Obtain help

Slide 15



AdminApp: Managing applications

- Application management scripts use the AdminApp object to manage applications in the application server configuration
- Example:

Command Prompt - wsadmin -profile C:\software\wsadmin\global_profile.py

```
wsadmin>print AdminApp.view('ivtApp')

Specifying application options

Specify the various options that are available to prepare and install your appli
cation.

Directory to install application:  $(APP_INSTALL_ROOT)/$(CELL)
Distribute application:  Yes
Use Binary Configuration:  No
Create MBeans for resources:  No
Override class reloading settings for Web and EJB modules:  No
Reload interval in seconds:
Validate Input off/warn/fail:  off
File Permission:  .*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755
```

- Use the AdminApp object to:
  - Install and uninstall applications
  - List installed applications
  - Edit application configurations
  - Obtain help

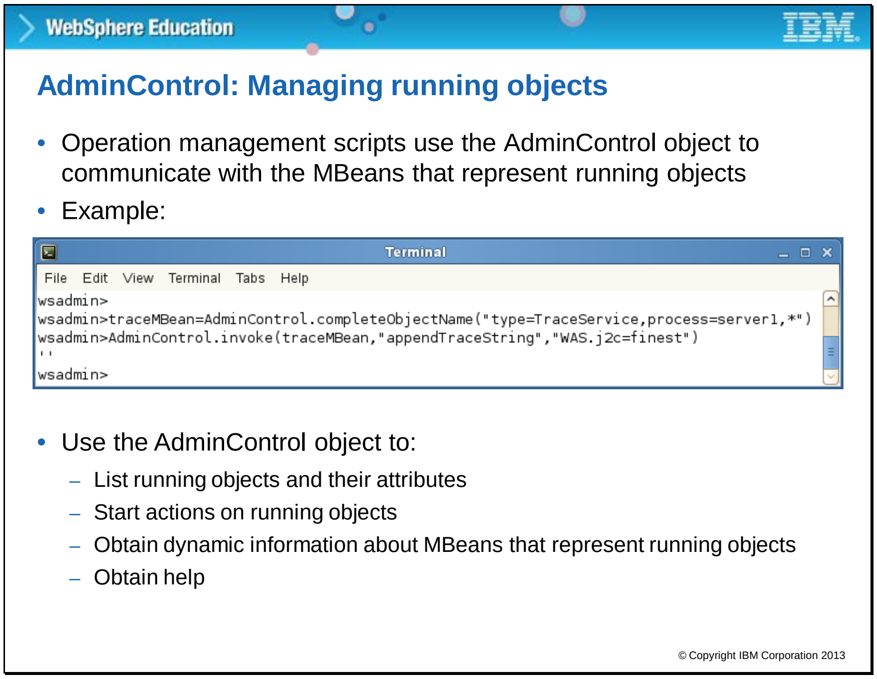
Title: AdminApp: Managing applications

Application management scripts use the **AdminApp** object to manage applications in the application server configuration.

Running the command produces output specific to the application. The Jython command output is easier to read when using the print option. You can use the **AdminApp** object to:

- Install and uninstall applications
- List installed applications
- Edit application configurations
- Obtain help

Slide 16



AdminControl: Managing running objects

- Operation management scripts use the AdminControl object to communicate with the MBeans that represent running objects
- Example:

```
wsadmin>
wsadmin>traceMBean=AdminControl.completeObjectName("type=TraceService,process=server1,*")
wsadmin>AdminControl.invoke(traceMBean,"appendTraceString","WAS.j2c=finest")
''
wsadmin>
```

- Use the AdminControl object to:
  - List running objects and their attributes
  - Start actions on running objects
  - Obtain dynamic information about MBeans that represent running objects
  - Obtain help

Title: AdminControl: Managing running objects

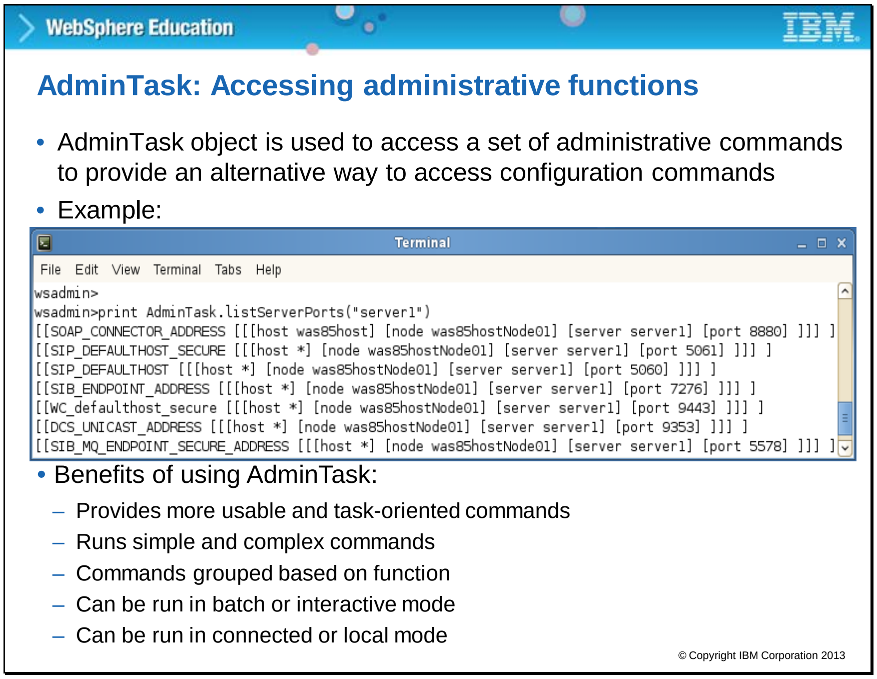
Operation management scripts use the **AdminControl** object to communicate with the MBeans that represent running objects.

You can use the **AdminControl** object to:

- List running objects and their attributes
- Start actions on running objects
- Obtain dynamic information about MBeans that represent running objects
- Obtain help

The number and type of MBeans available to the scripting client depends on the server to which the client is connected. If the client is connected to a deployment manager, then all the MBeans running in the deployment manager are visible. All the MBeans running in the node agents that are connected to this deployment manager, and all the MBeans running in the application servers on those nodes, are visible.

Slide 17



AdminTask: Accessing administrative functions

- AdminTask object is used to access a set of administrative commands to provide an alternative way to access configuration commands
- Example:

```
wsadmin>
wsadmin>print AdminTask.listServerPorts("server1")
[[SOAP_CONNECTOR_ADDRESS [[[host was85host] [node was85hostNode01] [server server1] [port 8880] ]]] ]
[[SIP_DEFAULTHOST_SECURE [[[host *] [node was85hostNode01] [server server1] [port 5061] ]]] ]
[[SIP_DEFAULTHOST [[[host *] [node was85hostNode01] [server server1] [port 5060] ]]] ]
[[SIB_ENDPOINT_ADDRESS [[[host *] [node was85hostNode01] [server server1] [port 7276] ]]] ]
[[WC_defaulthost_secure [[[host *] [node was85hostNode01] [server server1] [port 9443] ]]] ]
[[DCS_UNICAST_ADDRESS [[[host *] [node was85hostNode01] [server server1] [port 9353] ]]] ]
[[SIB_MQ_ENDPOINT_SECURE_ADDRESS [[[host *] [node was85hostNode01] [server server1] [port 5578] ]]] ]
```

- Benefits of using AdminTask:
  - Provides more usable and task-oriented commands
  - Runs simple and complex commands
  - Commands grouped based on function
  - Can be run in batch or interactive mode
  - Can be run in connected or local mode

Title: AdminTask: Accessing administrative functions

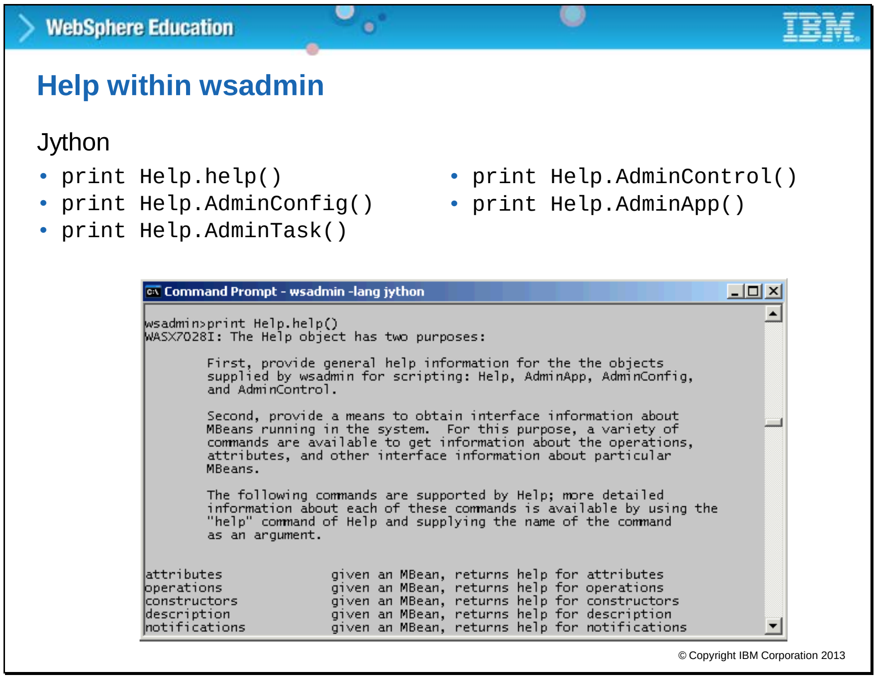
The AdminTask object is used to access a set of administrative commands that provide an alternative and easier way to access configuration commands. AdminTask commands enable you to run a single command to do administrative actions that might otherwise require multiple commands.

The benefits of using AdminTask include:

- Provides commands that are more task-oriented and easier to use
- Runs simple and complex commands
- Commands are grouped based on function
- Can be run in batch or interactive mode
- Can be run in connected or local mode

The administrative commands are discovered dynamically when you start a scripting client. The set of available administrative commands depends on the edition of WebSphere Application Server you install. You can use the AdminTask object commands to access these commands.

Slide 18



Help within wsadmin

Jython

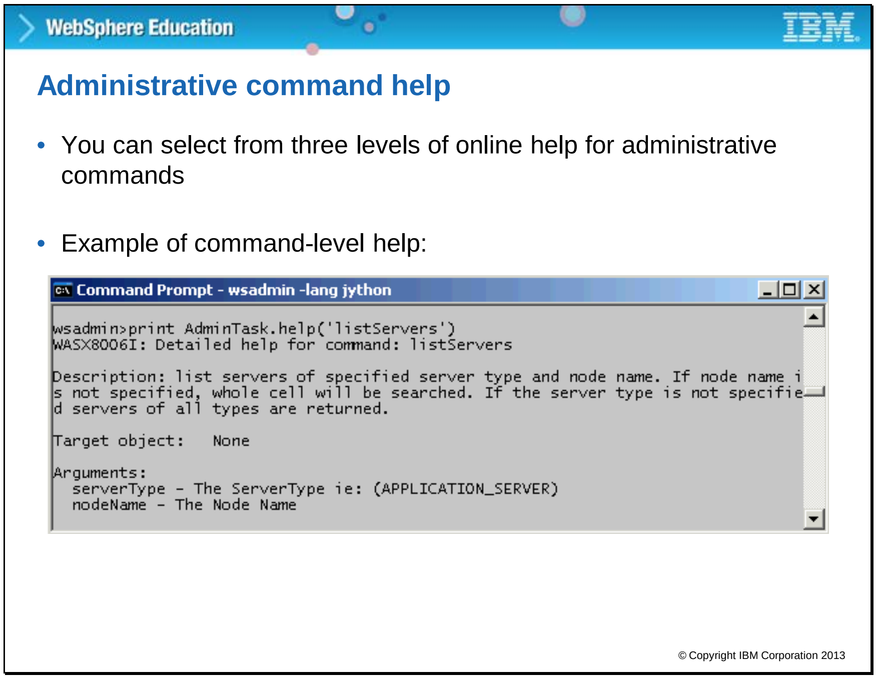
- `print Help.help()`
- `print Help.AdminConfig()`
- `print Help.AdminTask()`
- `print Help.AdminControl()`
- `print Help.AdminApp()`

```
wsadmin>print Help.help()
WASX7028I: The Help object has two purposes:

        First, provide general help information for the the objects
        supplied by wsadmin for scripting: Help, AdminApp, AdminConfig,
        and AdminControl.

        Second, provide a means to obtain interface information about
        MBeans running in the system.  For this purpose, a variety of
        commands are available to get information about the operations,
        attributes, and other interface information about particular
        MBeans.

        The following commands are supported by Help; more detailed
        information about each of these commands is available by using the
        "help" command of Help and supplying the name of the command
        as an argument.


attributes                given an MBean, returns help for attributes
operations                given an MBean, returns help for operations
constructors              given an MBean, returns help for constructors
description               given an MBean, returns help for description
notifications             given an MBean, returns help for notifications
```

Title: Help within wsadmin

You can find general help and dynamic online information about the currently running MBeans with the wsadmin tool. Use the **Help** object as an aid in writing and running scripts with the **AdminControl** object.

Slide 19



Administrative command help

- You can select from three levels of online help for administrative commands
- Example of command-level help:

```
wsadmin>print AdminTask.help('listServers')
WASX8006I: Detailed help for command: listServers

Description: list servers of specified server type and node name. If node name i
s not specified, whole cell will be searched. If the server type is not specifie
d servers of all types are returned.

Target object:   None

Arguments:
  serverType - The ServerType ie: (APPLICATION_SERVER)
  nodeName - The Node Name
```

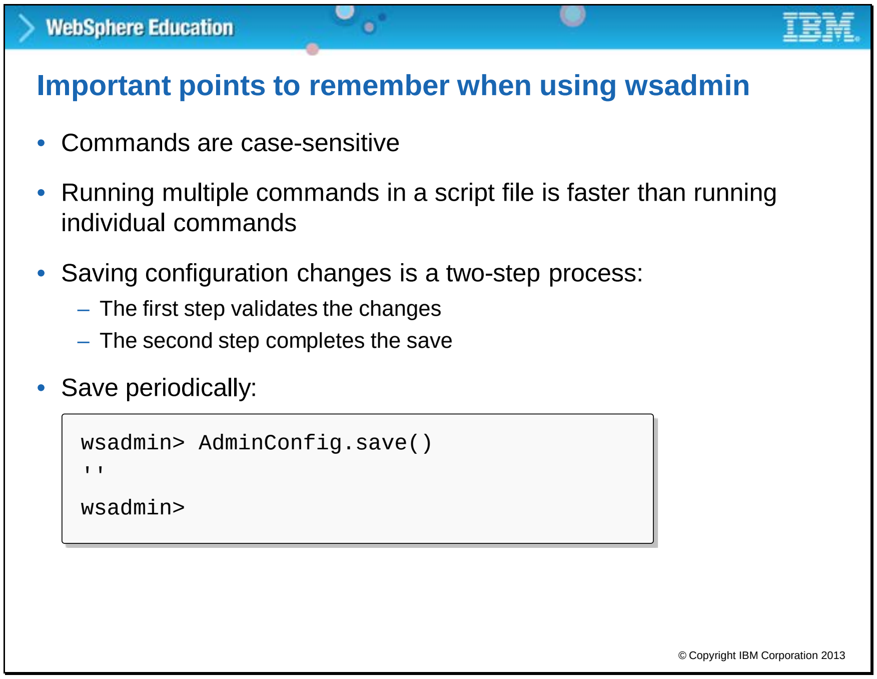
Title: Administrative command help

You can select from three levels of online help for administrative commands.

- Top-level help provides general information for the AdminTask object and associated commands.
- Second-level help provides information about all of the available administrative commands and command groups.
- Third-level help provides specific help on a command group, a command, or a step.

Command group-specific help provides descriptions for the command group that you specify and the commands that belong to the associated group. Command-specific help provides a description for the specified command and associated parameters and steps.
Step-specific help provides a description for the specified step and the associated parameters. For command and step-specific help, required parameters are marked with an asterisk (*) in the help output.

Slide 20



WebSphere Education
IBM

Important points to remember when using wsadmin

- Commands are case-sensitive
- Running multiple commands in a script file is faster than running individual commands
- Saving configuration changes is a two-step process:
  - The first step validates the changes
  - The second step completes the save
- Save periodically:

```
wsadmin> AdminConfig.save()
''
wsadmin>
```

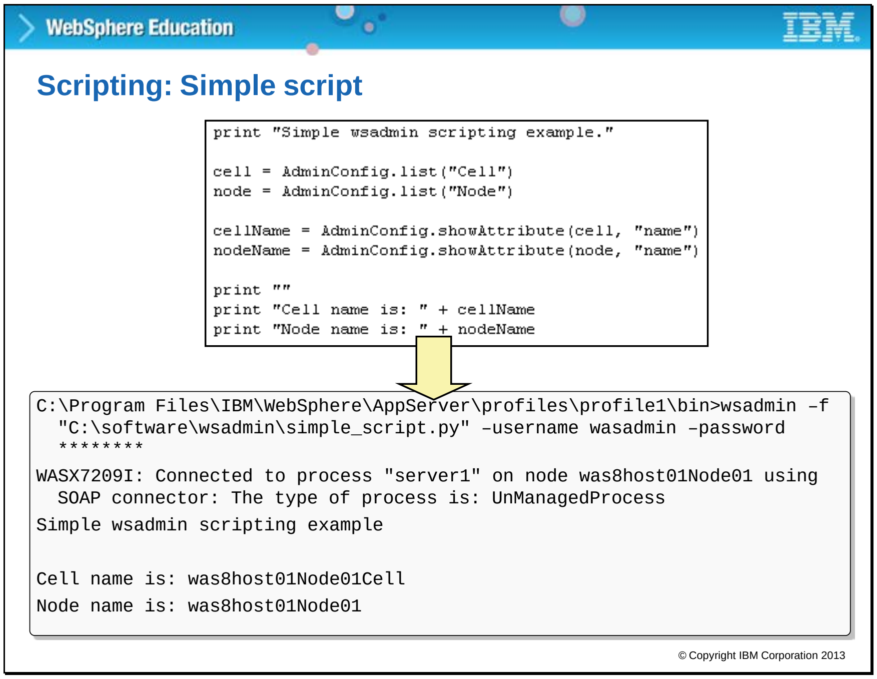
© Copyright IBM Corporation 2013

Title: Important points to remember when using wsadmin

When using wsadmin, remember the following important points:

- Commands are case-sensitive.
- Running multiple commands in a script file is faster than running individual commands. **wsadmin -f "script_file_name"** is faster than individual commands with **wsadmin –c.**
- Saving configuration changes is a two-step process:
  - The first step validates the changes.
  - The second step saves the changes.
- Run the save command periodically in the script file or in the interactive mode to persist configuration updates to the repository.

Slide 21



Scripting: Simple script

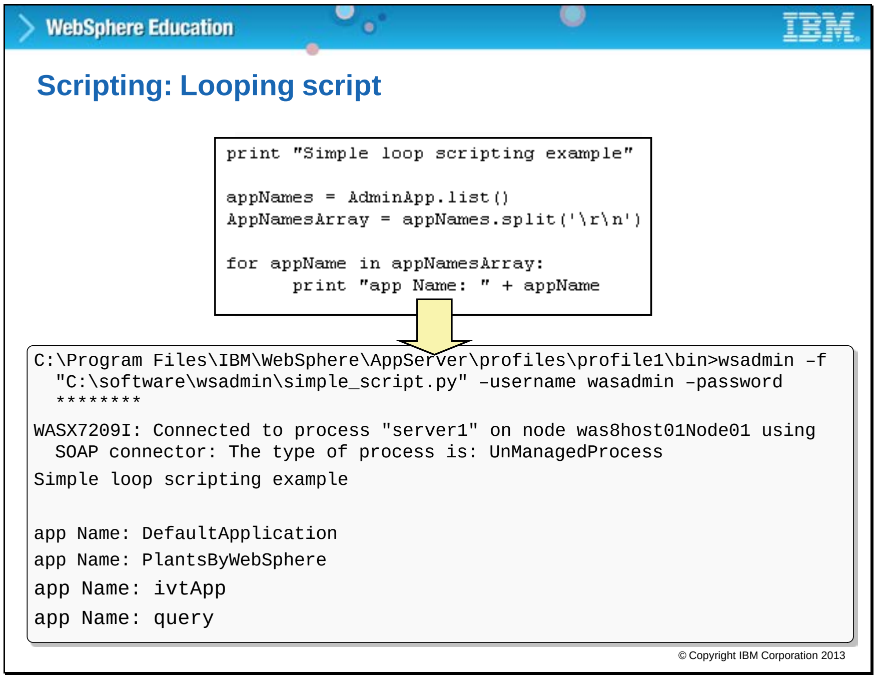
```
print "Simple wsadmin scripting example."

cell = AdminConfig.list("Cell")
node = AdminConfig.list("Node")

cellName = AdminConfig.showAttribute(cell, "name")
nodeName = AdminConfig.showAttribute(node, "name")

print ""
print "Cell name is: " + cellName
print "Node name is: " + nodeName
```

```
C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin>wsadmin –f
  "C:\software\wsadmin\simple_script.py" –username wasadmin –password
  ********
WASX7209I: Connected to process "server1" on node was8host01Node01 using
  SOAP connector: The type of process is: UnManagedProcess
Simple wsadmin scripting example


Cell name is: was8host01Node01Cell
Node name is: was8host01Node01
```

Title: Scripting: simple script

This example shows a simple script and the output from that script.

Slide 22



Scripting: Looping script

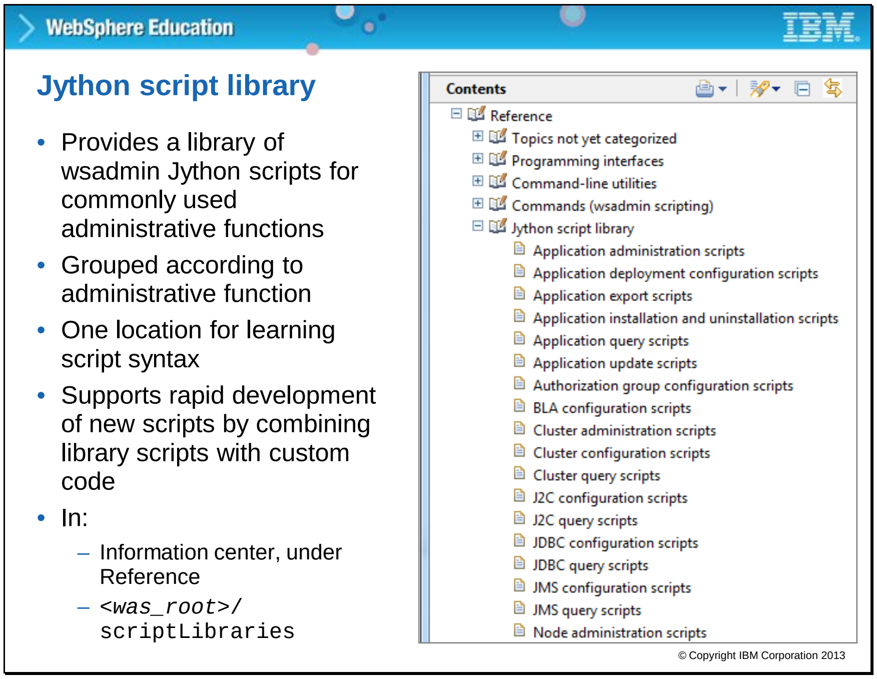
```
print "Simple loop scripting example"

appNames = AdminApp.list()
AppNamesArray = appNames.split('\r\n')

for appName in appNamesArray:
      print "app Name: " + appName
```

```
C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin>wsadmin –f
  "C:\software\wsadmin\simple_script.py" –username wasadmin –password
  ********
WASX7209I: Connected to process "server1" on node was8host01Node01 using
  SOAP connector: The type of process is: UnManagedProcess
Simple loop scripting example

app Name: DefaultApplication
app Name: PlantsByWebSphere
app Name: ivtApp
app Name: query
```

Title: Scripting: Looping script

This example shows a looping script and the output from that script.

Slide 23



WebSphere Education

Jython script library

- Provides a library of wsadmin Jython scripts for commonly used administrative functions
- Grouped according to administrative function
- One location for learning script syntax
- Supports rapid development of new scripts by combining library scripts with custom code
- In:
  - Information center, under Reference
  - `<was_root>/ scriptLibraries`

Contents

- ⊟ Reference
  - ⊞ Topics not yet categorized
  - ⊞ Programming interfaces
  - ⊞ Command-line utilities
  - ⊞ Commands (wsadmin scripting)
  - ⊟ Jython script library
    - Application administration scripts
    - Application deployment configuration scripts
    - Application export scripts
    - Application installation and uninstallation scripts
    - Application query scripts
    - Application update scripts
    - Authorization group configuration scripts
    - BLA configuration scripts
    - Cluster administration scripts
    - Cluster configuration scripts
    - Cluster query scripts
    - J2C configuration scripts
    - J2C query scripts
    - JDBC configuration scripts
    - JDBC query scripts
    - JMS configuration scripts
    - JMS query scripts
    - Node administration scripts

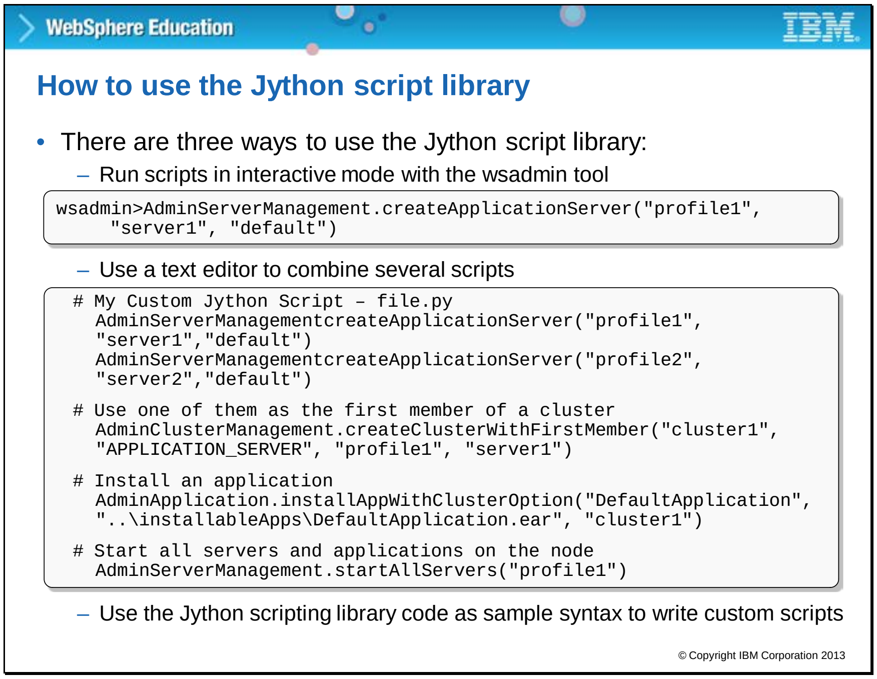
© Copyright IBM Corporation 2013

Title: Jython script library

The Jython script library provides a set of procedures to automate the most common application server administration functions. For example, you can use the script library to easily configure servers, applications, mail settings, resources, nodes, business-level applications, clusters, authorization groups, and more. You can run each script procedure individually, or combine several procedures to quickly develop new scripts.

The Jython script library is in **<*was_root*>/scriptLibraries**. Information regarding the script library is available in the information center under the Reference topic.

Slide 24



WebSphere Education

How to use the Jython script library

- There are three ways to use the Jython script library:
  - Run scripts in interactive mode with the wsadmin tool

```
wsadmin>AdminServerManagement.createApplicationServer("profile1",
     "server1", "default")
```

  - Use a text editor to combine several scripts

```
# My Custom Jython Script – file.py
  AdminServerManagementcreateApplicationServer("profile1",
  "server1","default")
  AdminServerManagementcreateApplicationServer("profile2",
  "server2","default")

# Use one of them as the first member of a cluster
  AdminClusterManagement.createClusterWithFirstMember("cluster1",
  "APPLICATION_SERVER", "profile1", "server1")

# Install an application
  AdminApplication.installAppWithClusterOption("DefaultApplication",
  "..\installableApps\DefaultApplication.ear", "cluster1")

# Start all servers and applications on the node
  AdminServerManagement.startAllServers("profile1")
```

  - Use the Jython scripting library code as sample syntax to write custom scripts

© Copyright IBM Corporation 2013

Title: How to use the Jython script library

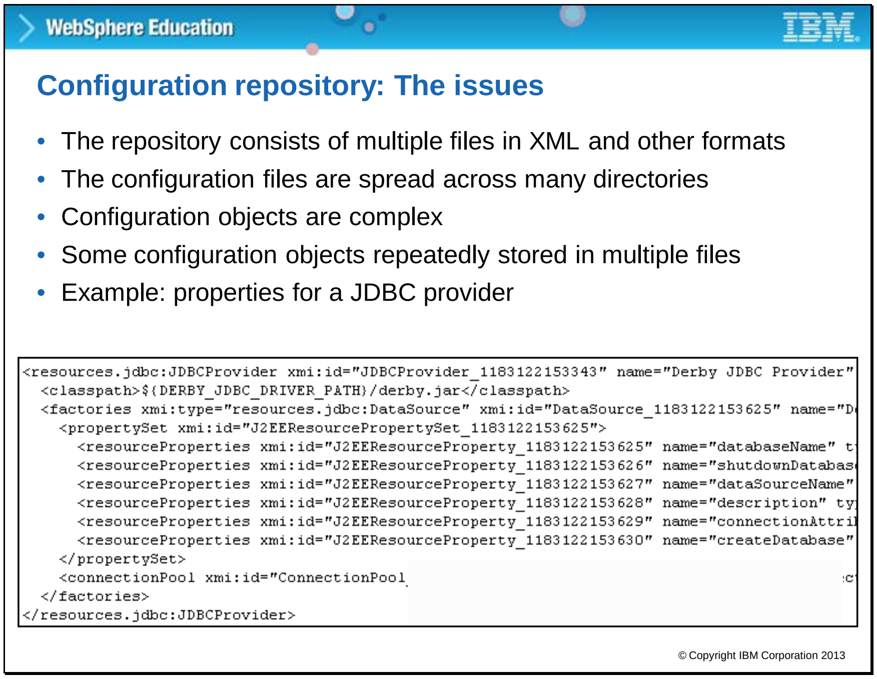
The scripting library provides a set of procedures to automate the most common application server administration functions. Each script in the script library demonstrates good examples and practices for writing wsadmin scripts.

The script library code is in the <*was_root*>/scriptLibraries directory. Within this directory, the scripts are organized into subdirectories according to function and version. For example, the <*was_root*>/scriptLibraries/application/V85 subdirectory contains procedures that run application management tasks that are applicable to version 8.5 and later of the product.

There are several ways to use the Jython script library.

- You can run scripts from the Jython script library in interactive mode with the wsadmin tool.
- You can start the wsadmin tool and run individual scripts that are included in the script library.
- You can use a text editor to combine several scripts from the Jython script library.
- You can use the Jython scripting library code as sample syntax to write custom scripts.

Slide 25



WebSphere Education

Configuration repository: The issues

- The repository consists of multiple files in XML and other formats
- The configuration files are spread across many directories
- Configuration objects are complex
- Some configuration objects repeatedly stored in multiple files
- Example: properties for a JDBC provider

```
<resources.jdbc:JDBCProvider xmi:id="JDBCProvider_1183122153343" name="Derby JDBC Provider"
  <classpath>${DERBY_JDBC_DRIVER_PATH}/derby.jar</classpath>
  <factories xmi:type="resources.jdbc:DataSource" xmi:id="DataSource_1183122153625" name="D
    <propertySet xmi:id="J2EEResourcePropertySet_1183122153625">
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153625" name="databaseName" t
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153626" name="shutdownDatabas
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153627" name="dataSourceName"
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153628" name="description" ty
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153629" name="connectionAttri
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153630" name="createDatabase"
    </propertySet>
    <connectionPool xmi:id="ConnectionPool_                                              c
  </factories>
</resources.jdbc:JDBCProvider>
```

Title: Configuration repository: The issues

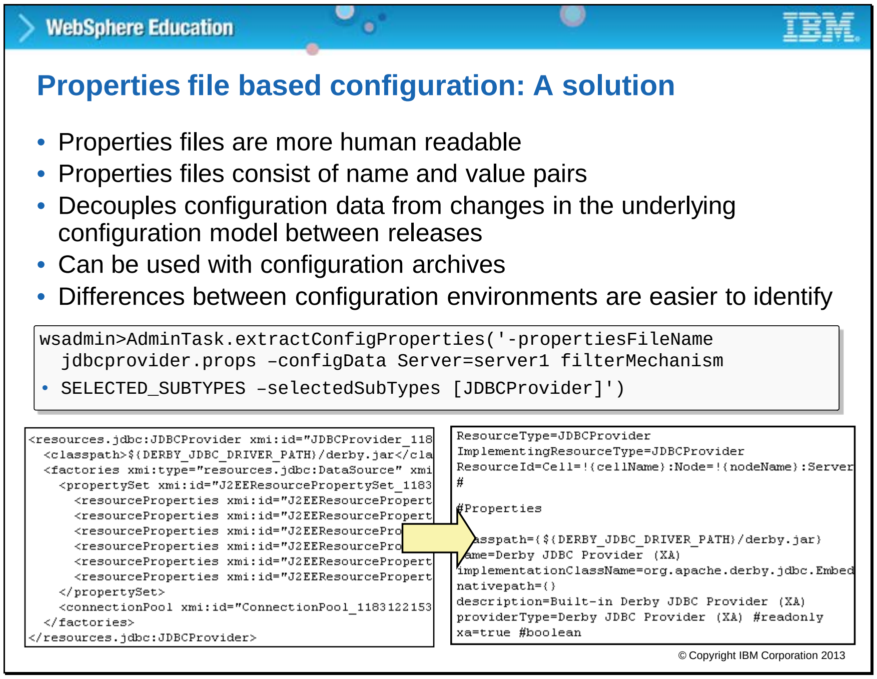
In previous releases of WebSphere Application Server, administrators used wsadmin, the administrative console, and Java APIs to query and modify configuration objects. With the WebSphere Application Server configuration repository, users are confronted with several issues:

- The repository consists of multiple files in XML and other formats.
- The configuration files are spread across many directories.
- Some files contain complex objects that are associated with the WebSphere Common Configuration Model.
- Some configuration objects are repeatedly stored in multiple files.

Properties-based file configuration is introduced as a tool to help users deal more easily with these issues. A new set of wsadmin commands is available that can extract and apply properties files to configuration objects.

The example on this slide shows JDBCProvider object content from the configuration XML file named resources.xml.

Slide 26


WebSphere Education
IBM

Properties file based configuration: A solution

- Properties files are more human readable
- Properties files consist of name and value pairs
- Decouples configuration data from changes in the underlying configuration model between releases
- Can be used with configuration archives
- Differences between configuration environments are easier to identify

```
wsadmin>AdminTask.extractConfigProperties('-propertiesFileName
  jdbcprovider.props –configData Server=server1 filterMechanism
• SELECTED_SUBTYPES –selectedSubTypes [JDBCProvider]')
```

```
<resources.jdbc:JDBCProvider xmi:id="JDBCProvider_118
  <classpath>${DERBY_JDBC_DRIVER_PATH}/derby.jar</cla
  <factories xmi:type="resources.jdbc:DataSource" xmi
    <propertySet xmi:id="J2EEResourcePropertySet_1183
      <resourceProperties xmi:id="J2EEResourcePropert
      <resourceProperties xmi:id="J2EEResourcePropert
      <resourceProperties xmi:id="J2EEResourcePro
      <resourceProperties xmi:id="J2EEResourcePro
      <resourceProperties xmi:id="J2EEResourcePropert
      <resourceProperties xmi:id="J2EEResourcePropert
    </propertySet>
    <connectionPool xmi:id="ConnectionPool_1183122153
  </factories>
</resources.jdbc:JDBCProvider>
```

```
ResourceType=JDBCProvider
ImplementingResourceType=JDBCProvider
ResourceId=Cell=!{cellName}:Node=!{nodeName}:Server
#

#Properties

asspath={${DERBY_JDBC_DRIVER_PATH}/derby.jar}
ame=Derby JDBC Provider (XA)
implementationClassName=org.apache.derby.jdbc.Embed
nativepath={}
description=Built-in Derby JDBC Provider (XA)
providerType=Derby JDBC Provider (XA) #readonly
xa=true #boolean
```

Title: Properties file based configuration: A solution

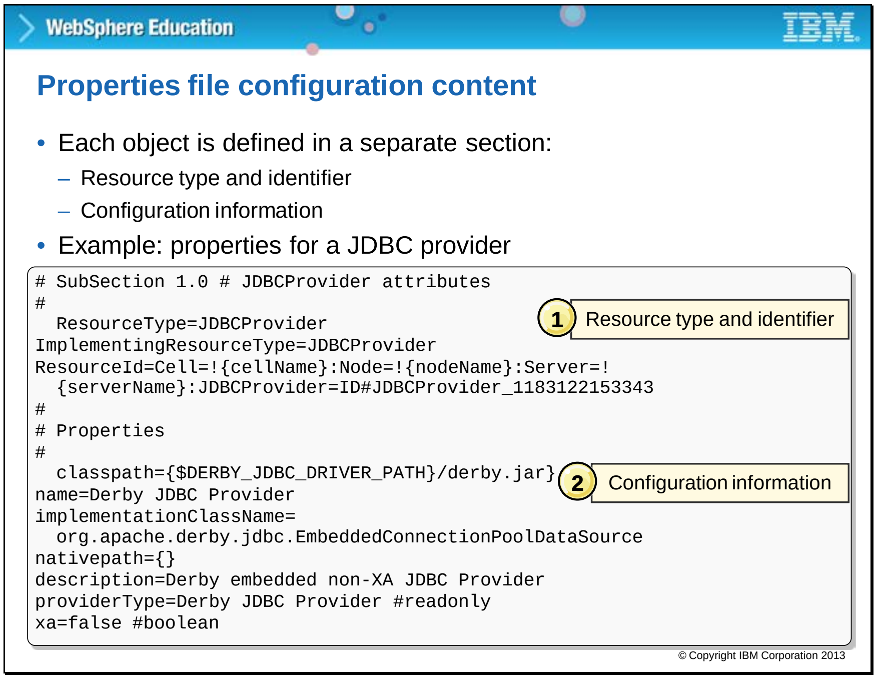
Using the **PropertiesBasedConfiguration** command group for the **AdminTask** object, you can extract the configuration attributes and values from your environment to properties files. You can use this feature for various purposes, such as:

- To modify your existing configuration in one location, instead of configuring multiple administrative console panels or running many commands.
- To improve the application development lifecycle.

WebSphere Application Server derives configuration information from the configuration repository, not from configuration properties files. To update the configuration repository to reflect the information in a configuration properties file, you must use wsadmin commands to apply the properties files to the configuration.

The graphic depicts the use of the **wsadmin extractConfigProperties** command to create a properties file with content based on information that is contained in the configuration XML file.

Slide 27



WebSphere Education
IBM

Properties file configuration content

- Each object is defined in a separate section:
  - Resource type and identifier
  - Configuration information
- Example: properties for a JDBC provider

```
# SubSection 1.0 # JDBCProvider attributes
#
  ResourceType=JDBCProvider
ImplementingResourceType=JDBCProvider
ResourceId=Cell=!{cellName}:Node=!{nodeName}:Server=!
  {serverName}:JDBCProvider=ID#JDBCProvider_1183122153343
#
# Properties
#
  classpath={$DERBY_JDBC_DRIVER_PATH}/derby.jar}
name=Derby JDBC Provider
implementationClassName=
  org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource
nativepath={}
description=Derby embedded non-XA JDBC Provider
providerType=Derby JDBC Provider #readonly
xa=false #boolean
```

1 Resource type and identifier

2 Configuration information

© Copyright IBM Corporation 2013

Title: Properties file configuration content

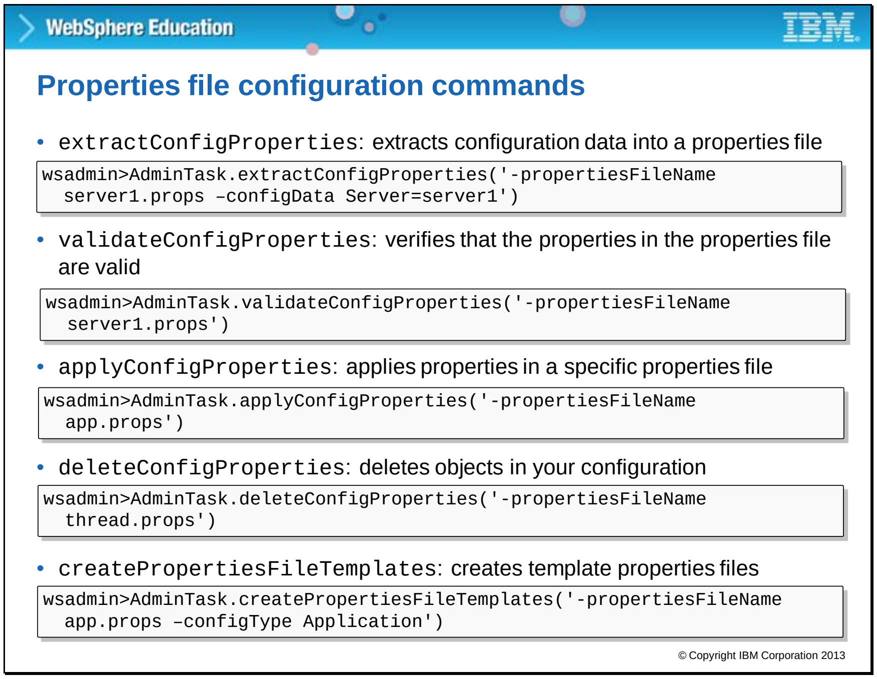
Configuration properties files contain a series of name-value pairs. Each configuration object is defined in two separate sections.

The first section of the example defines a resource type and a resource identifier. The identifier is often in a format that includes the cell, node, and server names, and ends with a string that contains the resource type and a large number.

In the example, the resource type is JDBC provider.

In the lower section, configuration information is specified by using name-value pairs.

Slide 28



WebSphere Education
IBM

Properties file configuration commands

- `extractConfigProperties`: extracts configuration data into a properties file

```
wsadmin>AdminTask.extractConfigProperties('-propertiesFileName
  server1.props –configData Server=server1')
```

- `validateConfigProperties`: verifies that the properties in the properties file are valid

```
wsadmin>AdminTask.validateConfigProperties('-propertiesFileName
  server1.props')
```

- `applyConfigProperties`: applies properties in a specific properties file

```
wsadmin>AdminTask.applyConfigProperties('-propertiesFileName
  app.props')
```

- `deleteConfigProperties`: deletes objects in your configuration

```
wsadmin>AdminTask.deleteConfigProperties('-propertiesFileName
  thread.props')
```

- `createPropertiesFileTemplates`: creates template properties files

```
wsadmin>AdminTask.createPropertiesFileTemplates('-propertiesFileName
  app.props –configType Application')
```

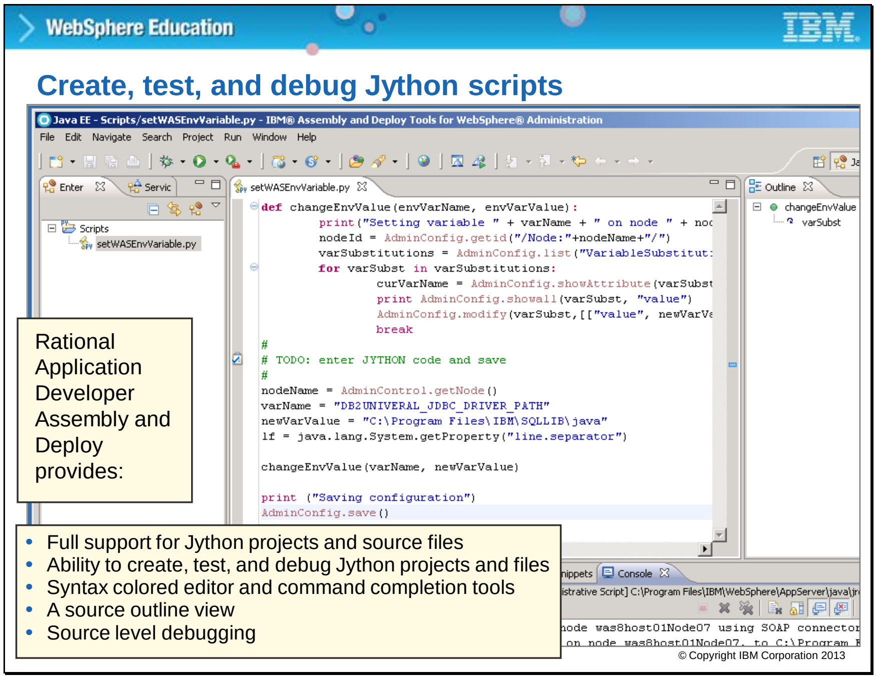
Title: Properties file configuration commands

Properties file-based configuration uses the five commands that are shown here.

- The **extractConfigProperties** command extracts configuration data in the form of a properties file.
- The **validateConfigProperties** command verifies that the properties in the properties file are valid and can be successfully applied to the new configuration.
- The **applyConfigProperties** command applies properties in a specific properties file to the configuration.
- The **deleteConfigProperties** command deletes properties in your configuration as designated in a properties file.
- Use the **createPropertiesFileTemplates** command to create template properties files to create or delete specific object types.

For more information, see the information center.

Slide 29


WebSphere Education
IBM
Java EE - Scripts/setWASEnvVariable.py - IBM® Assembly and Deploy Tools for WebSphere® Administration
File Edit Navigate Search Project Run Window Help
Enter
Servic
Scripts
setWASEnvVariable.py
Outline
changeEnvValue
varSubst
Snippets
Console

Create, test, and debug Jython scripts

```
def changeEnvValue(envVarName, envVarValue):
        print("Setting variable " + varName + " on node " + noc
        nodeId = AdminConfig.getid("/Node:"+nodeName+"/")
        varSubstitutions = AdminConfig.list("VariableSubstitut:
        for varSubst in varSubstitutions:
                curVarName = AdminConfig.showAttribute(varSubst
                print AdminConfig.showall(varSubst, "value")
                AdminConfig.modify(varSubst,[["value", newVarVa
                break
#
# TODO: enter JYTHON code and save
#
nodeName = AdminControl.getNode()
varName = "DB2UNIVERAL_JDBC_DRIVER_PATH"
newVarValue = "C:\Program Files\IBM\SQLLIB\java"
lf = java.lang.System.getProperty("line.separator")

changeEnvValue(varName, newVarValue)

print ("Saving configuration")
AdminConfig.save()
```

```
istrative Script] C:\Program Files\IBM\WebSphere\AppServer\java\jr
node was8host01Node07 using SOAP connector
 on node was8host01Node07, to C:\Program F
```

Rational Application Developer Assembly and Deploy provides:

- Full support for Jython projects and source files
- Ability to create, test, and debug Jython projects and files
- Syntax colored editor and command completion tools
- A source outline view
- Source level debugging

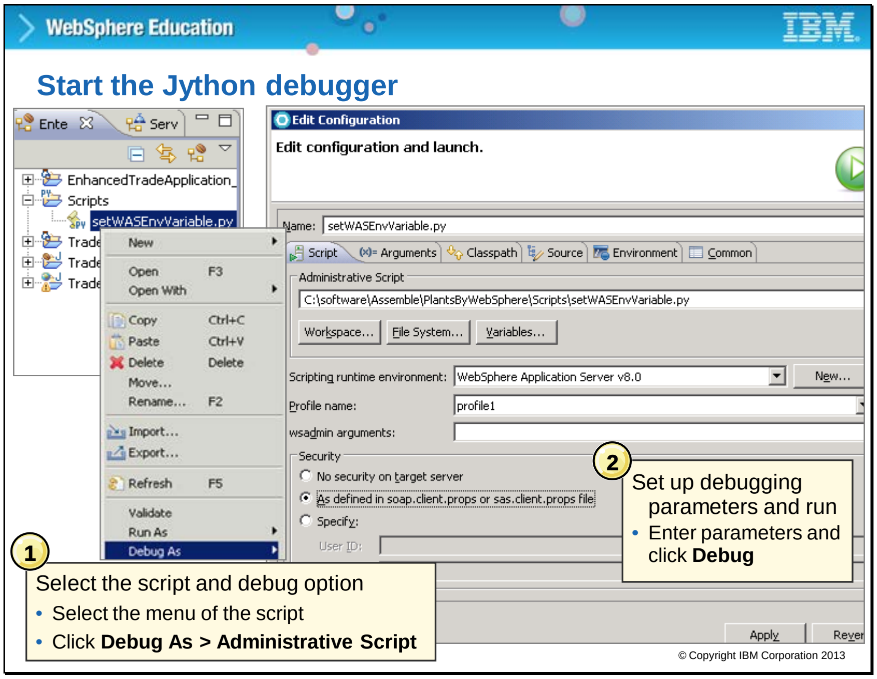
© Copyright IBM Corporation 2013

Title: Create, test, and debug Jython scripts

The IBM Assembly and Deploy Tool enables you to create, test, and debug Jython scripts. Jython projects and scripts are treated as first class objects. The Jython editor provides the same facilities as all other specialized editors, including:

- Full support for Jython projects and source files
- Ability to create, test, and debug Jython projects and files
- Syntax colored editor, automated formatting, and command completion tools
- A source outline view
- Source level debugging

Slide 30



Start the Jython debugger

Edit Configuration

Edit configuration and launch.

Name: setWASEnvVariable.py

Administrative Script: C:\software\Assemble\PlantsByWebSphere\Scripts\setWASEnvVariable.py

Scripting runtime environment: WebSphere Application Server v8.0

Profile name: profile1

wsadmin arguments:

Security

○ No security on target server

● As defined in soap.client.props or sas.client.props file

○ Specify:

User ID:

1 Select the script and debug option

- Select the menu of the script
- Click **Debug As > Administrative Script**

2 Set up debugging parameters and run

- Enter parameters and click **Debug**

© Copyright IBM Corporation 2013

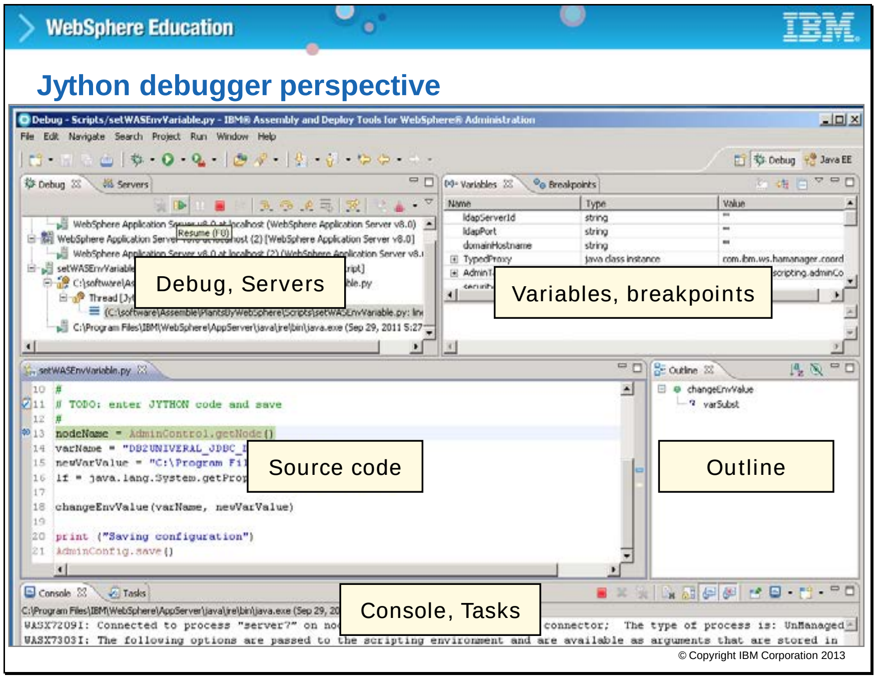
Title: Start the Jython debugger

To start the Jython debugger:

- Select the script and debug option.
- Highlight the script and right-click it to open the menu.
- Click **Debug As > Administrative script** to open the debugger runtime and security parameters window.
- Set up debugging parameters and start the debugger.
- Enter debugging parameters as appropriate.

Debugging a script for the first time requires setup of runtime and security parameters that are defined under the Script tab. Click **Debug** when parameter input is complete to start the Debug perspective.

Slide 31



Jython debugger perspective

```
#
# TODO: enter JYTHON code and save
#
nodeName = AdminControl.getNode()
varName = "DB2UNIVERAL_JDBC_I
newVarValue = "C:\Program Fil
lf = java.lang.System.getProp

changeEnvValue(varName, newVarValue)

print ("Saving configuration")
AdminConfig.save()
```

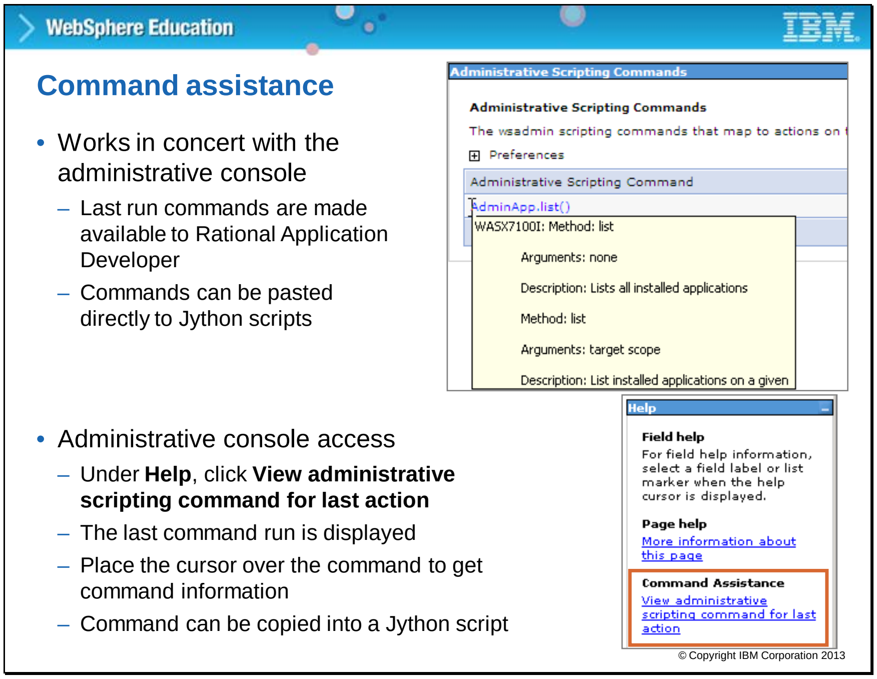
```
WASX7209I: Connected to process "server7" on no                      connector;  The type of process is: UnManaged
WASX7303I: The following options are passed to the scripting environment and are available as arguments that are stored in
```

Title: Jython debugger perspective

The Jython debugger perspective is a tool that can help you track the behavior and state of your Jython scripts and easily pinpoint logic errors. Using the debugger, you can pause the running script to examine the working code. You can identify problems such as the source of the bug and contributing errors. The debugger perspective is divided into views:

- The **Debug** view (upper left pane)
- The upper right pane has multiple tabs, including:
  - The **breakpoints** view, which lists all the breakpoints that set in the workbench projects. You can double-click a breakpoint to show its location in the editor.
  - From the **Expressions** view, you can inspect data from a scrapbook page, a stack frame of a suspended thread, and other places.
  - The **Variables** view shows information about the variables in the currently selected stack frame.
- The **Source** view (middle left pane) shows the result of evaluating an expression in the context of the current stack frame.
- The **Outline** view (middle right pane) shows an outline of the structure of the currently active Jython script in the editor area.
- The **Console** view (lower pane) shows messages that result from the execution of the Jython script.

Slide 32



Command assistance

- Works in concert with the administrative console
  - Last run commands are made available to Rational Application Developer
  - Commands can be pasted directly to Jython scripts
- Administrative console access
  - Under **Help**, click **View administrative scripting command for last action**
  - The last command run is displayed
  - Place the cursor over the command to get command information
  - Command can be copied into a Jython script

Administrative Scripting Commands

Administrative Scripting Commands

The wsadmin scripting commands that map to actions on t

⊞ Preferences

Administrative Scripting Command
AdminApp.list()

```
WASX7100I: Method: list

        Arguments: none

        Description: Lists all installed applications

        Method: list

        Arguments: target scope

        Description: List installed applications on a given
```

Help

Field help

For field help information, select a field label or list marker when the help cursor is displayed.

Page help

More information about this page

Command Assistance

View administrative scripting command for last action

© Copyright IBM Corporation 2013

Title: Command assistance

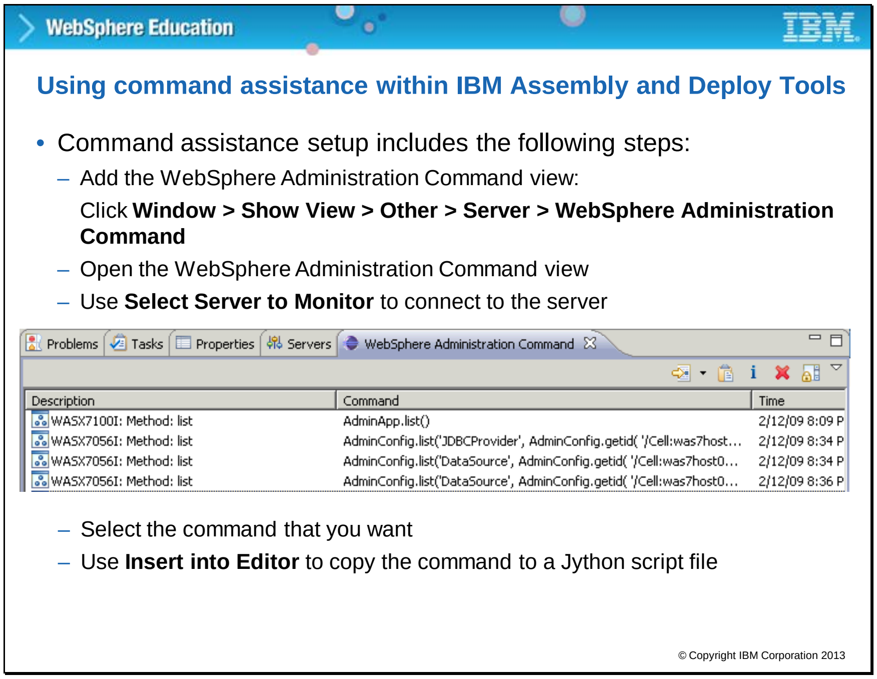
Using command assistance, you can see wsadmin scripting commands that correspond to actions in the administrative console. Seeing these commands might help you develop the commands necessary to administer WebSphere Application Server from the wsadmin utility. You can view wsadmin scripting commands in the Jython language for the last action that runs in the administrative console.

When you run server operations in the administrative console, the administrative command assistance tool captures and shows the wsadmin commands issued.

You can transfer the output from the administrative command view directly to a text editor, such as the Jython editor, enabling you to develop Jython scripts that are based on actual console actions.

Under **Help**, click **View administrative scripting command for last action.**

Slide 33



Using command assistance within IBM Assembly and Deploy Tools

- Command assistance setup includes the following steps:
  - Add the WebSphere Administration Command view:

    Click **Window > Show View > Other > Server > WebSphere Administration Command**
  - Open the WebSphere Administration Command view
  - Use **Select Server to Monitor** to connect to the server

Description	Command	Time
WASX7100I: Method: list	AdminApp.list()	2/12/09 8:09 P
WASX7056I: Method: list	AdminConfig.list('JDBCProvider', AdminConfig.getid('/Cell:was7host…	2/12/09 8:34 P
WASX7056I: Method: list	AdminConfig.list('DataSource', AdminConfig.getid('/Cell:was7host0…	2/12/09 8:34 P
WASX7056I: Method: list	AdminConfig.list('DataSource', AdminConfig.getid('/Cell:was7host0…	2/12/09 8:36 P

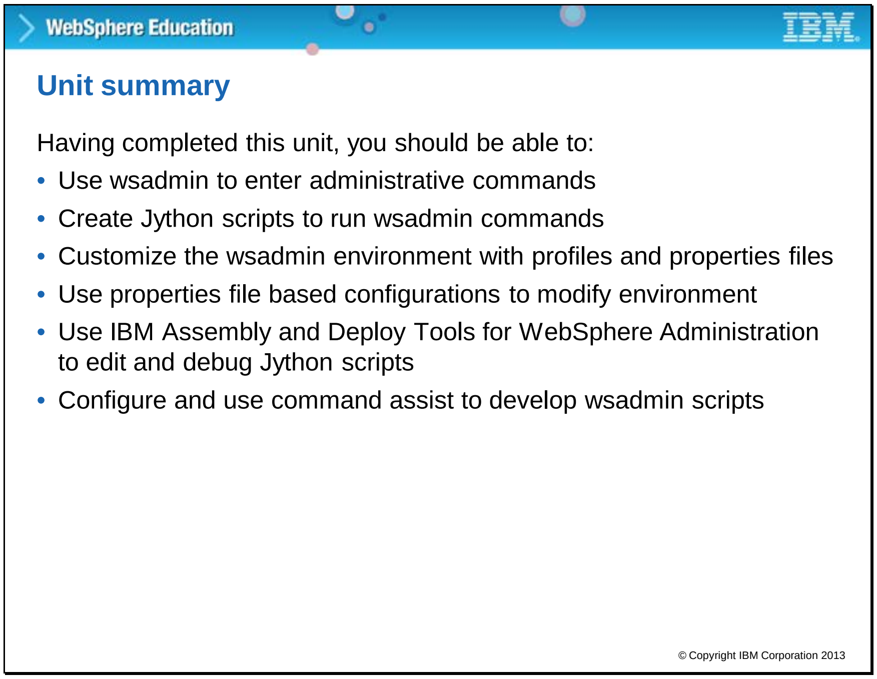
  - Select the command that you want
  - Use **Insert into Editor** to copy the command to a Jython script file

Title: Using command assistance within IADT

You can use command assistance from IADT and the administrative console. To enable command assistance from IADT, administrative console setup is required.

- Under **Help**, click **View administrative scripting command for last action**.
- Expand **Preferences** and enable both preference options:
  - **Log command assistance commands**
  - **Enable command assistance notifications**
- Click **Apply**.

Slide 34



WebSphere Education

Unit summary

Having completed this unit, you should be able to:

- Use wsadmin to enter administrative commands
- Create Jython scripts to run wsadmin commands
- Customize the wsadmin environment with profiles and properties files
- Use properties file based configurations to modify environment
- Use IBM Assembly and Deploy Tools for WebSphere Administration to edit and debug Jython scripts
- Configure and use command assist to develop wsadmin scripts

Title: Unit summary

Having completed this unit, you should be able to:

- Use wsadmin to enter administrative commands
- Create Jython scripts to run wsadmin commands
- Customize the wsadmin environment with profiles and property files
- Use property file based configurations to modify environment
- Use IBM Assembly and Deploy Tools for WebSphere Administration to edit and debug Jython scripts
- Configure and use command assist to develop wsadmin scripts