
ORACLE®

Using Nucleus Components

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Oracle Training Materials – Usage Agreement

Use of this Site (“Site”) or Materials constitutes agreement with the following terms and conditions:

1. Oracle Corporation (“Oracle”) is pleased to allow its business partner (“Partner”) to download and copy the information, documents, and the online training courses (collectively, “Materials") found on this Site. The use of the Materials is restricted to the non-commercial, internal training of the Partner’s employees only. The Materials may not be used for training, promotion, or sales to customers or other partners or third parties.

2. All the Materials are trademarks of Oracle and are proprietary information of Oracle. Partner or other third party at no time has any right to resell, redistribute or create derivative works from the Materials.

3. Oracle disclaims any warranties or representations as to the accuracy or completeness of any Materials. Materials are provided "as is" without warranty of any kind, either express or implied, including without limitation warranties of merchantability, fitness for a particular purpose, and non-infringement.

4. Under no circumstances shall Oracle or the Oracle Authorized Delivery Partner be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this Site of Materials. As a condition of use of the Materials, Partner agrees to indemnify Oracle from and against any and all actions, claims, losses, damages, liabilities and expenses (including reasonable attorneys' fees) arising out of Partner’s use of the Materials.

5. Reference materials including but not limited to those identified in the Boot Camp manifest can not be redistributed in any format without Oracle written consent.

Agenda

- Understanding Component Scope
- Understanding CONFIGPATH layering
- Using the Nucleus Component

Learning Objectives

At the end of this lesson you should be able to:

- Use component scope appropriately
- Register Nucleus components using the CONFIGPATH system variable
- Define attribute values using property file layering
- Understand Nucleus interfaces
- Create Nucleus components by extending the GenericService class
- Log application behavior correctly using ATG Logging

Section 1:
Component Scope

Component Scopes

An application component can be set to one of the following scopes

- **global** : Component shared among all site visitors
- **session** : Separate instances of the component are provided to each site visitor
- **request** : Separate instances of the component are provided in each active request
- **window** : Separate instances of the component are provided in each browser window.

Specifying Component Scopes

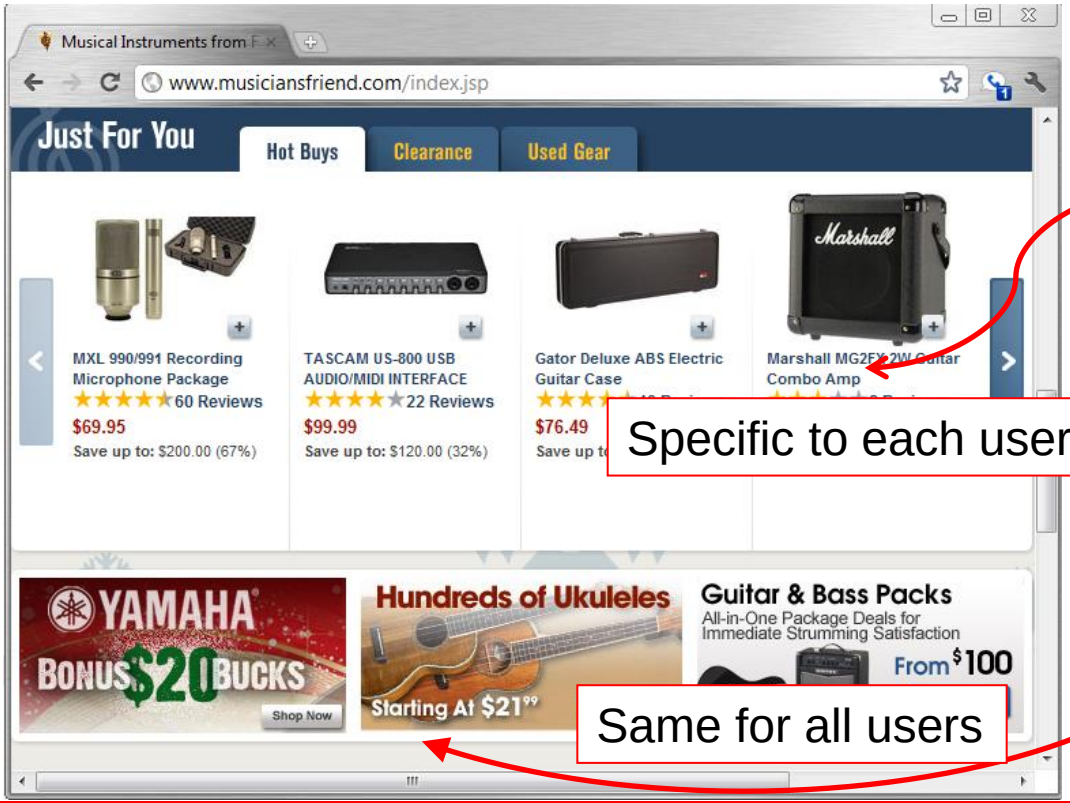
- Component scope is specified on the $scope property as follows:

```
$class=com.myproject.Person
$scope=session
name=Bill
age=28
```

- If the scope variable is not explicitly set, it is by default set to global.
- Component properties should always refer to other components whose scope is equal to or broader than its own.
- For example:
  - request scoped component can point to global, session and request scoped component
  - session scoped component can point to session and global

Global Scope

- For global scoped components, there is only one instance for the entire application and it can be accessed across multiple sessions.
- The state of the component is presented the same for all users.



Component:
Recommendations
Scope: session

Specific to each users

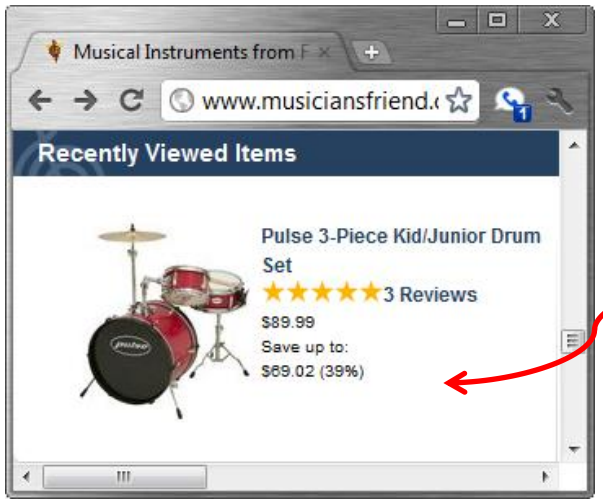
Component:
GlobalPromotions
Scope: global

Same for all users

Session Scope

- For session scoped components, an instance is created for each user session.
- At any given point, there will be multiple instances of the same nucleus component with its own session values.
- The ATG platform uses cookies or rewritten URLs to identify requests from a given browser as belonging to the same session.
- Session gets invalidated after a period of inactivity as specified by the application server.
- All the session scoped components are removed from memory after the session expires.
- A User Profile is a good example of a session scoped component.
- Another example would be a component that is used to track the recently viewed products by a user.

Session Scoped Component

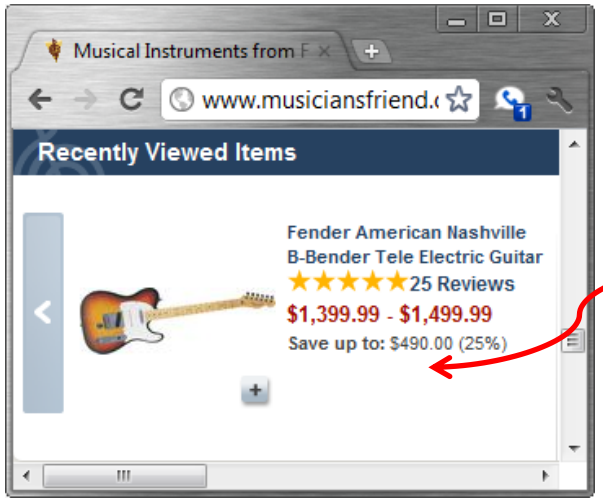


User 1

Instance:
RecentlyViewed
Scope: session
Product: Drum Set
count: 2

Takes the default for count and add product

Component:
RecentlyViewed
scope: session
count:2



User 2

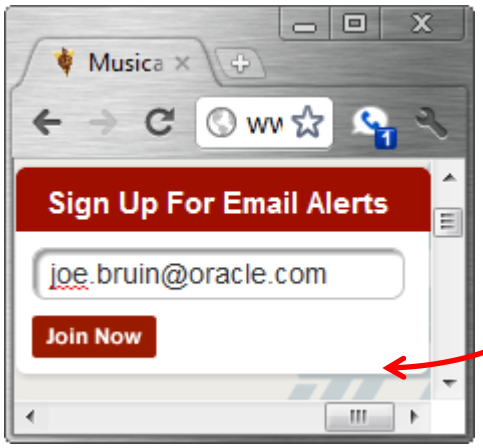
Instance:
RecentlyViewed
Scope: session
Product: Guitar
count: 3

Count changed based on user preference

Request Scope

- In a request scoped component, an instance is created for each request from the user's session.
- If the same session sends two simultaneous requests, they will each get a different instance.
- Each instance has no effect on the other.
- Request scoped components are most commonly used for forms.
- Each request of the form will submit their own values without affecting other form submissions.
- A local redirect from within the code will preserve the objects in the request.
- Request scoped instances are marked for garbage collection after the request is serviced.

Request Scoped Component

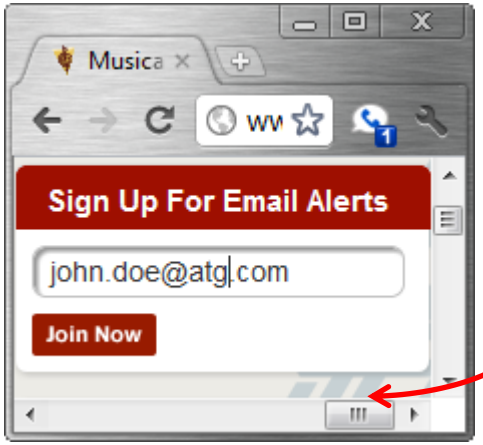


User 1

Instance:
AlertSignUpForm
Scope: request
email: joe.bruin
Sender: EmailSender

Leverages the same Global Email Sender

Component:
AlertSignupForm
scope: request
Sender:EmailSender



User 1: Different Request

Instance:
AlertSignUpForm
Scope: request
email: john.doe
Sender: EmailSender

Setting Properties of Session and Request Scoped Components

- At any given time, there may be multiple instances of session and request scoped components.
- To save on memory, ATG does not always create any new instances, but uses pointers to existing instances.
- This means that changing the contents of a mutable object property affects other instances.
- Always replace the object of a mutable property rather than changing its existing object value.
- Strings and other primitive data types can safely be modified.

Example: Scope

- Global: Email Sender, Store Configuration
- Session: User Profile, Recently Viewed
- Request: Forms

- Please specify scopes for components that
  - Keep track of how long a user has been on a site?
  - Keep track of how many users are on the site?
  - Look up an item in a database?
  - Handle form input from the user?

Section 1
Check Your Understanding

What are the different scopes that a component can have?

a. Request, session, global, and window.
b. Request, session, global
c. Request, session, singleton and prototype
d. Request, session, globalSession
e. Prototype, session, globalSession

Answer: a

Section 1
Check Your Understanding

If a component records users activities when logged in, what scope would it normally have?

a. Request scoped as activities are on a request basis
b. Session scoped as it is specific to a user session.
c. Global scoped to improve on performance
d. Singleton scoped as only one component needs to record activity
e. GlobalSession since there can be only one per session

Answer: b

Section 1
Check Your Understanding

What components can a session scoped component refer to?

a. Can only refer to session
b. Can only refer to session and global scoped components.
c. Can only refer to session and request
d. Can refer to global, session, request and window
e. Can refer to session and singleton

Answer: b

Section 1 
Check Your Understanding

What scope would a typical form handler have?

a. Singleton
b. Prototype
c. Request
d. Session
e. Global

Answer: c

Section 1

Check Your Understanding

How many instances does a session scoped component have during the server run time?

a. There will be as many instances as there are active sessions in the server.

b. There is exactly one instance. The component tracks values for sesion in a map

c. There are configured MIN_INSTANCES pool of components. As required, these instances are handed over to the requesting service threads

d. There is exactly one instance in the primary server that is servicing the request and one instance is the session reconvery instance attached to that instance.

Answer: a

Summary : Component Scope

- ATG component scopes include global, session, request, and window.
- Global scope component is shared by all users of the application.
- In session scoped components, each user has its own nucleus component.
- In a request scoped component, each request gets a different copy of the component.
- A web application uses several components with various scopes to achieve the business requirements.

Section 2:

CONFIG PATH and Property File Layering

Specifying CONFIGPATH

- CONFIGPATH is where ATG looks for .properties files.
- The CONFIGPATH is specified in the module's manifest.
- The MANIFEST.MF file is located at

```
ModuleName/META-INF/MANIFEST.MF
```

- A typical MANIFEST.MF file is:

```
ATG-Product: My Project Inc.
Manifest-Version: 1.0
ATG-Required: DAS
ATG-Config-Path: config/config.jar
```

- ATG will merge the config paths based on the hierarchy in which the components are loaded during the server startup. The result is:

```
<atg_dir>\Dynamo10.0.x\DAS\config\config.jar;
<atg_dir>\Dynamo10.0.x\module-name\config.jar;
DYNAMO_HOME\localconfig
```

CONFIGPATH Variable (1)

- ATG looks for **`.properties`** files in the directories specified by the CONFIGPATH.
- In a developer written module, the CONFIGPATH points to module/config/config.jar.
- The jar file is created from a directory called config in the source code.
- Components path is equal to the relative path in the config directory.
- For example, if a property file is located at

  /MyModule/config/ship/OneDay.properties

  The component path is /ship/OneDay
- Proper naming convention must be followed while creating and naming components.

CONFIGPATH Variable (2)

- CONFIGPATH contains multiple folders from various modules and ATG modules.
- ATG will examine all directories and jar files specified by the CONFIGPATH.
- The ATG nucleus reads the entire CONFIGPATH from left-to-right and every matching properties file is loaded.
- If the same fully qualified .properties file is specified in two different config paths, they are implicitly merged with the later (left-to-right) values taking precedence.
- If the same property value is defined in two files, the last value is used, when read from left-to-right.
- The resulting instance will get all the merged values used in the initialization.

Property File Layering

- If two configuration property files exist for the same component name in two different modules, the ATG nucleus will merge the values.
- The properties defined in the file that appears later in the CONFIG PATH overrides the property value that appears earlier.
- For instance, config path is

  DAS/config/config.jar;MyModule/config/config.jar

- The properties defined in the file

  /atg/commerce/ShoppingCart.properties

  is in both the DAS Module and a customized module
- ATG will merge the two ShoppingCart.properties.
- Property file layering can be used to override the default ATG behavior.

Property File Layering (Example)

- Example CONFIGPATH:

```
<atg10dir>\DAS\config\config.jar;
<atg10dir>\MyModule\config;
DYNAMO_HOME\localconfig
```

…\DAS\config.jar\atg\commerce\ShoppingCart.propertieses

```
$class=com.atg.ShoppingCart
persistOrders=true
AnonymousOrders=true
…
```

…\MyModule\config.jar\atg\commerce\ShoppingCart.properties

```
$class=com.mysite.MyCart
persistsOrders=false
maxOrders=10
```

Component:
/atg/Commerce/**ShoppingCart**

```
$class=com.mysite.MyCart
persistOrders=false
AnonymousOrders=true
maxOrders=10
```

CONFIGPATH Layer (Override Property)

- Example:

  CONFIGPATH:

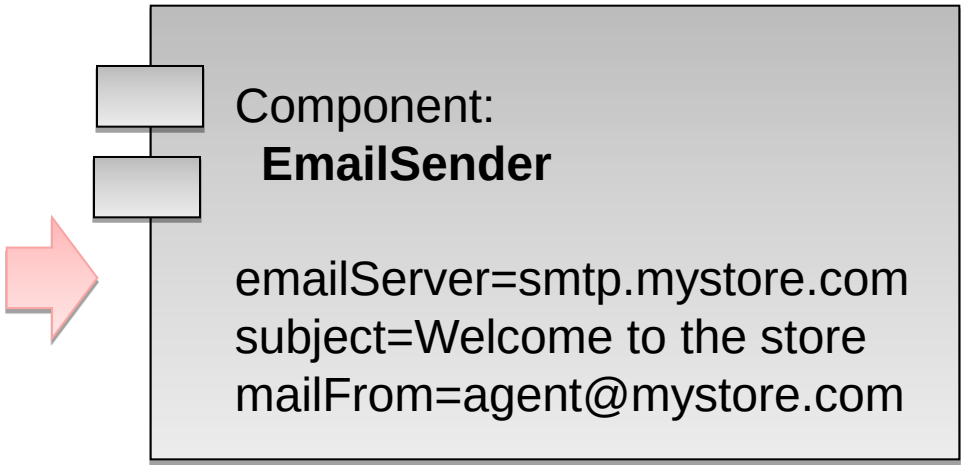
  ```
  <atg10dir>\MyMod\Store\config\config.jar;
  <atg10dir>\FW\Store\config;
  DYNAMO_HOME\localconfig
  ```

…\Store\config.jar\email\EmailSender.properties

emailServer=localhost
subject=Welcome to the store

…\Intgr\config.jar\email\EmailSender.properties

emailServer=smtp.mystore.com
mailFrom=agent@mystore.com



CONFIGPATH Layer (Combine Property)

- Example:

  CONFIGPATH:

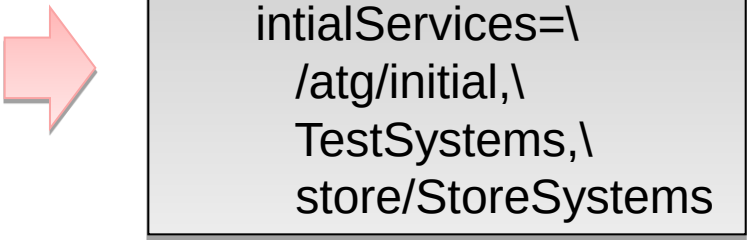
  ```
  <atg10dir>\MyMod\Store\config\config.jar;
  <atg10dir>\FW\Store\config;
  DYNAMO_HOME\localconfig
  ```

…\Store\config.jar\atg\dynamo\Initial.properties

```
initialServices=/atg/initial,\
                TestSystems
```

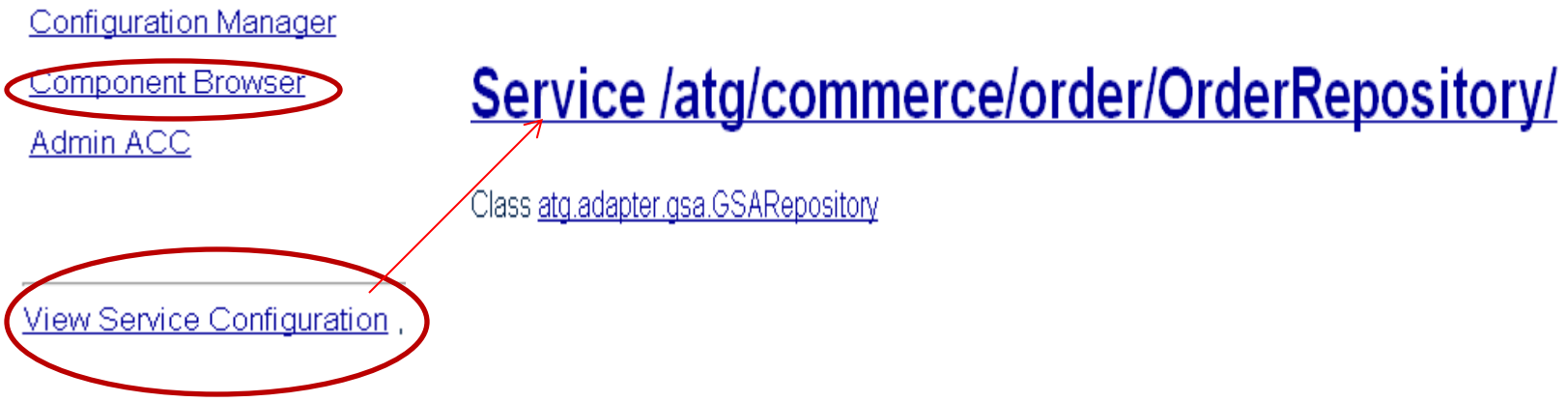
…\Intgr\config.jar\atg\dynamo\Initial.properties

```
intialServices+=\
                store/StoreSystems
```



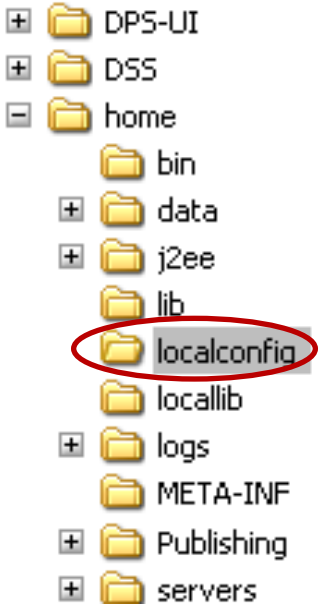
Tracing Component Property Settings

- Use the ATG Dynamo Server Admin Component Browser to determine how a given component is configured:
  - Navigate to the target component.
  - Click View Service Configuration link to view a hierarchical listing of the properties files for that component.



Local Config

- Local config contains custom properties settings, and has the highest priority in the configuration path. Settings in localconfig are preserved across product upgrades; changes to base configuration properties can be set here.
- Local config =<atg10dir>/home/localconfig.
- Note that Local config setting are not in the source and hence present a maintainability problem.
- Settings in Local config should only be used for development, testing, and emergency production fixes and not in a production process or deployment.



Component Instantiation

- The ATG nucleus manages the life cycle of the component.
- Components are created as needed when they are referenced by other components or used.
- When Nucleus creates a new instance of a component, it:
  - Loads the property files in the CONFIGPATH folders
  - Creates a live component instance, setting values from properties file(s)
- Once instantiated, the values of the component can be changed by system interactions.
- Depending on the scope of the component, the component may be reinitialized at the end of the request, session, or never reinitialized.

Configured vs. Live Property Values

- Component property values are initialized from properties configuration files when the component is instantiated.
- Live values can be changed programmatically or from other system interactions without changing the properties file.
- For example, if the User Profile Component is initialized as ‘not logged in’, it may change to ‘logged in’ when the user performs the login action.
- Since user profile is session scoped, this value will remain as long as the session remain active or the user logs off.

Section 2

Check Your Understanding

Where does ATG load Nucleus configuration property files?

a. From CONFIGPATH specified in ATG Database and set using BCC
b. From CONFIGPATH in the System Environment variable
c. From CONFIGPATH specified in the MANIFEST.MF module file
d. From CONFIGPATH in ATG_HOME\DAS\config\config.jar
e. From Local Config ATG_HOME\localconfig only

Answer: c

Section 2
Check Your Understanding

Custom properties settings and test or emergency fix settings are typically put in ATG Local Config folder. Where is the ATG Local Config folder located?

a. ATG_HOME\..\DAS\localconfig
b. ATG_HOME\localconfig
c. ATG_HOME
d. ATG_HOME\<<APP MODULE>>\localconfig
e. JBOSS_HOME\servers\<<ServerName>>\deploy\ATG\localconfig

Answer: b

Section 2

Check Your Understanding

Which order does ATG read CONFIG PATH entries?

a. From left to right. Later ones override the earlier ones.
b. From right to left. Property once set cannot be overridden
c. In specific order. Property conflicts cause startup warning and indeterminate behavior, if different
d. One Config Path as configured. Local Config overrides Module config.

Answer: a

Section 2  Check Your Understanding

How does ATG deal with two property files that refer to the same ATG component?

a. This causes errors
b. Smartly merging the two. Later property file values (config path order) overrides earlier property file values.
c. Smartly merging the two. Earlier property file values (config path order) overrides later property file values if in conflict.
d. First property file (config path order) is the active one. The others are ignored.
e. Later property file (config path order) is the active one. The earlier ones are ignored.

Answer: b

Section 2

Check Your Understanding

ATG OOTB component defines a list of currencies that the site can support. Can you change this list for your site?

a. Change the component property in BCC
b. Change the component property in DynAdmin and Save it.
c. Define the component again and add new currencies in the attribute value
d. Request a patch from ATG support
e. Change the config in ATG_HOME\..\DAS\config.jar where the orginal component attributes are defined.

Answer: c

Summary : Nucleus Component

- The CONFIGPATH is used to specify the location to look for in the Nucleus property files.
- Paths are read from Left to right with the later property definition taking precedence over the earlier property definition.
- If more than one component configuration property file is found, it is implicitly merged.
- This feature can be leveraged to override default or add new behavior in an ATG web application.

Section 3:
Using Nucleus Components

Using Nucleus Components

- Any JavaBean component can be a Nucleus component.
- However, most components extend and implement nucleus interfaces, to achieve ATG specific functionality
- To gain name binding, the use of Name Binding interface class are used in the nucleus container.
- Service interfaces allow components to be notified and notify other components during its life cycle of events.
- Logging interfaces allow the component to log activity such as error, warning, or debug.
- GenericService implements the key interfaces and can be used as a simple mechanism to gain all of the above.

Nucleus Interfaces – Name Binding

- The following are Name Binding useful interfaces that a component can implement to achieve additional nucleus functionality
  - **NameContextBindingListener**: Allows for notification of binding to name context.
  - **NameContextElement**: Exposes name and nameContext as properties.
  - **NameContext**: Allows components to be containers of other components.
  - **NameContextBindingEventSource**: Allows to transmit a binding to name context event.

Nucleus Interfaces – Other

- The following are other useful interfaces that a component can implement to achieve additional nucleus functionality
  - **ServiceListener**: Allows the component to receive a notification that the properties have been initialized.
  - **Service**: Allows the administrator to examine various component properties.
  - **ApplicationLogging**: Allows a component to log information.
  - **AdminableService**: Allows a component to provide an administration servlet to the admin console.

GenericService Overview

- Instead of implementing all the previously specified nucleus interfaces, you can extend the GenericService class which implements most of the key Nucleus interfaces.
- Generic Service Class:
  - Supplies method like doStartService and doStopService for the Component life-cycle management.
  - Exposes logging service that can be centrally managed and set.
  - Exposes servlet components that can be managed by the Dynamo Server Admin.
  - Implements getAbsoluteName and resolveName to help in resolving the nucleus component by name.

Component Life-Cycle

- The Component life-cycle has two stages - start and stop.
- The GenericService has a starting and ending in its component life-cycle
  - The ***doStartService*** method will be called after Nucleus creates the component
  - The ***doStopService*** method will be called when the service stops
- Your component can override the doStartService and/or doStopService to start or initialize anything needed for the nucleus component.
- Typical activities in start method includes opening ports, creating threads, and opening files. The stop method performs the closing tasks.

Resolve Component By Name

- GenericService implements methods for resolving names in the Nucleus namespace:

```
String profileName = "/atg/userprofiling/Profile"
Profile profile = resolveName(profileName);
```

```
String pName = "/atg/userprofiling/Profile"
ComponentName name = ComponentName.getComponentName(pName);
Profile profile = resolveName(name );
```

- Resolve name tries to resolve the name relative to component's parent if absolute name is not provided.
- `resolveName("Pricer")` get the component Pricer that is a sibling to the component making this call.

How to Create a New Component

- Creating a component involves two steps:
  - Define a class by extending the Generic Service class.
  - Create a component properties file.
- The class should be placed in a location that is accessible to the nucleus.
  - This is typically in a jar archive which is present in the CLASSPATH.
- The property should be placed in a location that is accessible in the nucleus:
  - This is typically in a file structure or a jar file that is present in the CONFIGPATH of the module.
  - The directory where the property file is located defines the fully qualified hierarchical name of the component.

Define Java Class for the Component

- A Java Class must have a constructor that takes no arguments
- Get, Set, and Action methods are created as needed.

```
Public class PersonService extends GenericeService {
  private SMTPEmailSender emailSender;

  public PersonService(){super();}

  public SMTPEmailSender getEmailSender(){
     return emailSender
  }
  public void setEmailSender(SMTPEmailSender emailSender){
      this. emailSender = emailSender;
  }

  public void sendEmailWhenPersonChanged() {
    getEmailSender().sendEmail(XXX);
  }
}
```

Java CLASSPATH

- The CLASSPATH tells Java Virtual Machine where to search for programs.
- .class files found earlier in CLASSPATH supersede files found later.
- CLASSPATH can either be a jar archive or a root folder.

Note: Never place .class files in the configuration path directory path. Doing so will cause errors.

Define Component Properties File

- The Component's properties file looks like :

```
$description=service of person domain.
$class=Person
$scope=global
emailSender=/atg/dynamo/service/SMTPEmail
```

- If no scope is defined, global is assumed.
- Comments are pre-pended with #.
- Place each properties file in the folder structure that defines the logical hierarchy of its component.
- A properties file must be deployed to a location in the CONFIGPATH of the module.

Logging

- Components need to log information about their status.
- In ATG, Components can fire a logEvents event.
- This broadcasts to listeners that can handle the event.
- ATG has listener services that send the events to files, screen, databases, etc.
- ATG defines four standard logging levels
  - Error: Represents faults that indicate serious problems.
  - Warning: Faults that indicate that there may be a problem.
  - Info: An event that occurred during normal operation and is expected.
  - Debug: Used for debugging purposes and considered too verbose for normal use.

Using Logging

- ATG provides convenient methods to check logging level configured for the component and to log a message.
  - isLoggingError() and logError()
  - isLoggingWarning() and logWarning()
  - isLoggingInfo() and logInfo()
  - isLoggingDebug() and logDebug()
- To avoid overhead, you should always check the logging level before logging.

```
// Log a debug message
if (isLoggingDebug()) {
  logDebug("Price was calculated to be zero");
}
```

GLOBAL.properties (1)

- GLOBAL.properties file is used to set the same property in all nucleus components.
- The property settings in a GLOBAL.properties file apply to all components in the file's configuration directory and subdirectories.
- As an example:

localconfig/GLOBAL.properties

```
loggingError=true
loggingWarning=true
loggingInfo=true
loggingDebug=true
```

- The above turns all logging to true for all components in ATG.

GLOBAL.properties(2)

- A component's own property settings have precedence over global property settings.

localconfig/GLOBAL.properties

```
loggingError=true
loggingWarning=true
loggingInfo=false
loggingDebug=false
```

Person.properties will override Global.properties value.

/MyModule/service/Person.properties

```
loggingError=true
loggingWarning=true
loggingInfo=true
loggingDebug=true
```

- The Person nucleus component will have all debug messages enabled.

GLOBAL.properties(3)

- Combining Global and Component Settings:

```
Dynamo10.x/DAS/config/GLOBAL.properties
affectedCities=Detroit,Boston,Los Angeles
```

Person.properties will combined with Global.properties

```
/localconfig/service/Person.properties
affectedCities+=Chicago
```

- The Person nucleus component will have affectedCities set to Detroit, Boston, Los Angeles, and Chicago.

Starting the Nucleus Component

- ATG will start the Nucleus component when it is first used or started by the Initial services.
- Also, at application start, ATG reads a file named Nucleus.properties that contains the name of the first component to create.
- The standard Nucleus.properties looks like this:

Nucleus.properties

```
$class=atg.nucleus.Nucleus
initialServiceName=/Initial
```

- You may change the Initial.properties to load your services at startup.

Initial Service

- ATG Nucleus is configured to start a component called Initial.
- The standard Initial.properties looks as follows:

Initial.properties

```
$class=atg.nucleus.InitialService
initialServices=\
         /atg/Initial,\
         VMSystem,\
         /atg/dynamo/StartServers
```

- You may add a component to Initial.properties to have it start at the startup of the application.
- Components that are not global scoped, do not need to be started this way. Instead, they are created and started when first used.

Component Types

- ATG provides the following types of Component:
  - Service component
  - Servlet bean
  - Form handler
  - Droplet
  - Schedule
  - Domain Object

ATG Profile Component

- The Profile component is central to all personalization on the ATG platform.
- It's component name is : /atg/userprofiling/Profile
- The Profile component is a live representation of a user's information.
- At the start of a user's session, the Profile component is unpopulated.

ATG Order Related Components

- ATG provides helpful order related components to process and handle orders.
- The Order Component will store all order information in the ATG platform.
- It is tightly coupled with the ATG OrderRepository.
- Order is a session scope component, so every user's session has its own order component.

Section 3
Check Your Understanding

What is the easiest and recommended way to get naming, logging, and other additional functionality for your Nucleus components?

a. Implement AdminableServer and Applucation Logging interfaces
b. Extend GenericService
c. Extend NucleusComponent
d. Extend any of the relevant DAS class files and override appropriate functions
e. Implement ATGNucleusInterfaces

Answer: b

Section 3 
Check Your Understanding

Which Nucleus Interface allows components to be containers of other components

a. NameContextBindingListener
b. NameContextElement
c. NameContext
d. NameContextBindingEventSource
e. NameContextContainer
f. Generic Server

Answer: c

Section 3
Check Your Understanding

What two additional functionality do you get by extending GenericService?

a. Naming
b. Shopping Cart
c. Logging
d. Personalization
e. Repository Mapping

Answer: a, c

Section 3

Check Your Understanding

What is the best way to enable logging in all components in a config path?

a. Change ATG Generic Service implementation to enable logging
b. Change properties on Generic Service component
c. Create GLOBAL.properties and configure the logging properties there
d. Use Logging.properties and configure the logging properties there
e. Change log4j.properties and configure logging properties

Answer: c

Section 3 
Check Your Understanding

How to you log a debug message?

a. Call Log.debug()
b. Use isLoggingDebug() and if true, calls Log.debug()
c. Call logDebug()
d. Use isLoggingDebug() and if true, calls logDebug()
e. Use logDebug() and if true, call Log.debug()

Answer: d

Summary : Using Nucleus Component

- ATG Provides various interfaces to gain additional functionality such as naming, notification, administration servlet, and logging.
- GenericService class can be extended to gain all the features above. It implements most of the above interfaces.
- ATG provides convenient methods to resolve components from paths.
- Logging methods can be used to log application specific event messages.
- GLOBAL.properties provides a convenient mechanism to set properties in multiple components.

ORACLE IS THE INFORMATION COMPANY


ORACLE®