TATA CONSULTANCY SERVICES
Experience certainty.

ATG Coding Standards

05/11/2012

Nandini S.Prasad

Retail/ATG

nandini.sprasad@tcs.com

Table of Content

1. ATG Coding Standards

This document will provide a heads-up on the ATG coding standards for beginners in ATG application development. The standards have been classified under frequently worked areas by developers.

1.1 Generic Standards

- Every module inside an application should have the following folder structure:
  - src – contains .java files of the application
  - config – contains .properties and .xml files
  - lib – contains .jar files
  - sql – contains .sql files
  - classes – auto-generated .class files post compilation
  - data – contains any documents relevant to the module
  - webservices - auto-generated files for a webservice call e.g. wsdl2java files
  - Each module should have its own build script i.e. build.xml when using ANT.

- Naming convention
  - Here is the classification of components based on their functionality (Suffix of component name):
  - Tools – Component that accesses the database using Repository API or RQL e.g. OrderTools.
  - DomainBridge – Component that sets the values retrieved from database into a bean object.
  - Bean – Class that has a list of properties with corresponding getter-setter.
  - FormHandler and Droplet – As the name suggests.
  - Impl – Component that implements an interface.
  - Constants – Class or Interface that contains constant's name-value pair.
  - Submitter – A scheduler that processes a request and sends it to a 3rd party.
  - Receiver - A scheduler that accesses a 3rd party to obtain a response and process it.
  - Service – Generic suffix for any component that performs a certain action like launch a webservice call or invokes other components to finish a task.
  - Use initial capital letter for Component (.java and .properties files) name.
  - Use camel case for - method name, variable name and property name (in .properties file)

- Never access a lower scoped component from higher scoped one e.g. global scoped component should not access request or session scoped component, session scoped should not access a request scoped component.

- Defining scope of classes, methods and variables:
  - Do not use default scope when your application has multiple modules instead go for public.
  - Reusable methods/classes should always have public scope.
  - Helper methods and member variables called only inside a class should have private scope.
  - Variables defined inside an interface should be 'public final'.

- For easy understandability, while creating a new component make sure that it's class file and properties file have the same file name and path.

- When loading a component A, nucleus load all other components required by component A, so avoid redefining already existing components.
  - e.g. Any component that uses OOTB Tools class, need not add corresponding repository name as its property as Tools class will already have repository property and its getter-setter in it. This can be reused. OrderTools.properties already has OrderRepository property in it.

- Configurable variables should be added in the .properties file of a component. Prevent mentioning such variables as class member variables or constants.

- Customization:
  - When extending an OOTB component, in order to avoid any confusion, the java class file name can be prefixed with the business client name instead of using the OOTB java file name whereas properties file name need not change.
  - e.g. /atg/userprofiling/ProfileTools.java can be extended as /xyz/userprofiling/XyzProfileTools.java.
  - Create a new /atg/userprofiling/ProfileTools.properties in the code to add new properties and map $class=xyz.userprofiling.XyzProfileTools.

- Transactions on orders, profiles, etc can be enclosed in a TransactionDemarcation (begin and end) block. TransactionDemarcation.end() has to be called in the finally block. Call transaction rollback in the catch block for TransactionDemarcationException.

- Use try-catch block where ever necessary. Never leave an exception unhandled. A message can be added to every exception that is thrown as it can make debugging easier.

- Ordering of jar files in MANIFEST.MF and .classpath should be given importance. When a class file is mentioned in multiple jars, MANIFEST.MF gives priority to the last mentioned jar and .classpath gives priority to the first mentioned jar.

- Javadoc comments:
  - Use javadoc class level and method level comments which will be helpful when creating reusable components.
  - When a method returns a collection like list, map, etc the comment should have details on what type of object is in the collection.
  - Make sure that no personal details like employee number or name is added as part of comments.

- Before committing code into subversion make sure that:
  - Java files should be free of compilation errors.
  - All jar files used by java files (about to be committed) must be added to the lib folder and committed.
  - When committing a jar file, the application's .classpath file should also be update with location of the jar file in the application and committed along with it.
  - Repository definition file need to be free of deployment errors or warnings so build and deploy the ear before committing it.

- Group the imported libraries in the class file header in the order – java, javax, atg, extended component structure.

1.2 Dynamo Logging

- Use ResourceBundle to hold - user messages (like form exceptions), log debug and log error messages. Each module can have a separate ResourceBundle for easy maintenance.

- A component can extend GenericService to avail the dynamo logging services.

- Always check if the respective log type flag is set before calling a LogEvent e.g.
  - if(isLoggingDebug)
    - logDebug(...)

- Do not set logDebug=true for any component in the code. However it can be turned on through DynAdmin during development and debugging. Write as many logDebug as necessary to make defect tracking easier.

- In the exception handling catch block, call logError with parameters message and throwable object.

1.3 Repository

- A new repository component and repository definition file need not necessarily have the same name. e.g. ProfileAdaptorRepository.properties is repository component and UserProfile.xml is repository definition file.

- While extending the OOTB repository, create a new repository definition file (.xml) in the same file path as the OOTB definition file. In this case the repository component (.properties) file need not be recreated unless any property value has to be changed.

- When extending an OOTB repository definition file, make sure to mention xml-combine attribute to specify whether the xml file in various config paths should be appended, replaced, etc. e.g. <gsa-template xml-combine=”append”>

- By default simple caching mode is used. 'item cache size' and 'query cache size' can be specified to improve performance.

- When using locked cache mode, configure ClientLockManager and relevant properties in the repository component.

- When writing a service that alters (add, update, delete) the items in a repository (like product catalog, inventory repository), cache flush needs to be done by calling 'invalidate cache' method in the code. In generic cases, cache timeout or cache expire attributes can be set in the repository definition file.

- OOTB table names should not be reused while creating new tables. New tables created can have the client's name prefixed to it for easier classification.
    - e.g. XYZ Client specific user properties can be created in xyz_user table but dps_user should not be reused or altered.

- In sql scripts, make sure to use foreign key references from primary table when creating multi tables and auxiliary tables.

- All sql scripts can be placed in the sql folder of each module for future reference.

1.4 Scheduler

- When an application has multiple application servers, the Schedulable component should extend SingletonSchedulableService as the latter will ensure that only a single instance of the schedulable component runs on any one server at the scheduled time. This will come in handy during server failover.

- Component that extends SingletonSchedulableService should be configured to have ClientLockManager, LockTimeout and LockName properties.

- Append the scheduler name to initialServices property inside initial.properties for that module so that these schedulers get scheduled in the Scheduler during server startup.

1.5 Pipeline

- When extending an OOTB pipeline, ‘xml-combine’ has to be defined in the pipeline definition file to mention whether a replace or append is required.

- When a new processor is added as the last pipeline link, STOP_CHAIN_EXECUTION has to be returned by its class file and the return value of the existing last pipeline link has to be altered accordingly.

1.6 FormHandler

- FormHandler component should extend GenericFormHandler and can only be request or session scoped. Every handle method should throw ServletException and IOException.
- Handle method should not explicitly return a boolean value, instead use checkFormRedirect method to redirect to the next page.
- Minimum of one XSuccessURL and XErrorURL needs to be set as configurable property.

1.7 Droplet

- Droplet component should extend DynamoServlet and should have one service method that throws ServletException and IOException.

1.8 Webservice Integration

- Maintain a separate sub-module structure for each third party integration inside a common parent module folder like /integration/Cybersource and /integration/BingMaps.
- Webservice related jar files can be placed in the module / sub-modules' 'lib' folder and has to be appended to the sub-modules' Manifest.MF file property named 'ATG-Class-Path'.
- WSDL files if any can be saved in the 'data' folder of the sub-module for reference.
- During development, the WSDL can be converted and the generated class/java files can be placed inside the sub-module's webservices sub folder.
- Maintain separate Constant, Tools and DomainBridge files specific to the third party integration inside each sub-module.

Thank You

About Tata Consultancy Services (TCS)

Tata Consultancy Services is an IT services, consulting and business solutions organization that delivers real results to global business, ensuring a level of certainty no other firm can match. TCS offers a consulting-led, integrated portfolio of IT and IT-enabled infrastructure, engineering and assurance services. This is delivered through its unique Global Network Delivery Model™, recognized as the benchmark of excellence in software development. A part of the Tata Group, India's largest industrial conglomerate, TCS has a global footprint and is listed on the National Stock Exchange and Bombay Stock Exchange in India.

For more information, visit us at **www.tcs.com**.

IT Services
Business Solutions
Outsourcing