**ATG Coding Rules**

ATG COE

(Version 1.0)

Contents

[Introduction 4](#_Toc391042899)

[Guidelines 4](#_Toc391042900)

[**1.** **RedirectInHandleMethods** - Never call sendLocalRedirect() or sendRedirect() in form handler handle\*() methods" 4](#_Toc391042901)

[**2.** **StringLiteralCheckFormRedirect** - Never pass hard-coded strings to checkFormRedirect 4](#_Toc391042902)

[**3.** **GetParameterInHandleMethods - Never call getParameter() in form handler handle\*() methods** 5](#_Toc391042903)

[**4.** **PipelineExceptionResourceBundle - Never pass a hard-coded string to PipelineResult.addError** 5](#_Toc391042904)

[**5.** **InternationalizedDropletException - Never pass a hard-coded string to the constructor of DropletExceptions** 5](#_Toc391042905)

[**6.** **GetMessageDropletException - Never call \*.getMessage() in the constructor of DropletExceptions** 6](#_Toc391042906)

[**7.** **GetMessageLogError**  - Never use \*.getMessage() in logError 6](#_Toc391042907)

[**8.** **GetMessagePipelineException - Never pass \*.getMessage() to PipelineResult.addError** 6](#_Toc391042908)

[**9.** **HardCodedFileExtensions - Avoid hard-coded file extensions** 7](#_Toc391042909)

[**10.** **HardCodedURLs - Avoid hard-coded URLs** 7](#_Toc391042910)

[**11.** **SystemCurrentTimeMillis - Avoid calling System.currentTimeMillis(). Use /atg/dynamo/service/CurrentDate instead** 7](#_Toc391042911)

[**12.** **StringLiteralServiceLocalParameter - Avoid passing a hard-coded string to serviceLocalParameter** 8](#_Toc391042912)

[**13.** **StringLiteralGetLocalParameter - Avoid passing a hard-coded string to getLocalParameter** 8](#_Toc391042913)

[**14.** **StringLiteralGetObjectParameter - Avoid passing a hard-coded string to getObjectParameter** 9](#_Toc391042914)

[**15.** **GetParameter - Avoid DynamoHttpServletRequest.getParameter(). Use DynamoHttpServletRequest.getLocalParameter() instead** 9](#_Toc391042915)

[**16.** **StringLiteralGetParameter - Avoid passing a hard-coded string to getParameter** 10](#_Toc391042916)

[**17.** **HardCodedPerformanceMonitorOperations - Avoid passing a hard-coded string to PerformanceMonitor.startOperation** 10](#_Toc391042917)

[**18.** **HardCodedRQLs - Never hard-code RQL queries** 11](#_Toc391042918)

[**19.** **ServiceLocalParameterInLogging - Never call serviceLocalParameter() inside of an if(isLogging\*())** 11](#_Toc391042919)

[**20.** **HardCodeRequestNucleusLookup - Avoid passing a hard-coded string to a resolveName() method call** 12](#_Toc391042920)

[**21.** **NucleusLookup - Avoid resolving Nucleus services statically (such as ServletUtil.getCurrentRequest().resolveName() or Nucleus.getGlobalNucleus().resolveName())** 12](#_Toc391042921)

[**22.** **HardCodingNucleusPaths - Avoid hard-coding Nucleus component paths** 12](#_Toc391042922)

[**23.** **GettersSettersForDropletsFormHandlers - Never reference droplets or form handlers from other droplets or form handlers** 12](#_Toc391042923)

[**24.** **StringLiteralSetParameter - Avoid passing a hard-coded string to setParameter() in form handlers or droplets. Use a constant instead** 13](#_Toc391042924)

[**25.** **CallTransactionManagerDirectly- Avoid interacting with the transaction manager directly** 13](#_Toc391042925)

[**26.** **StringLiteralRunProcess - Never pass a string to PipelineManager().runProcess** 14](#_Toc391042926)

[**27.** **GenerateFormExceptionMethod - Avoid defining and using the method generateFormException** 14](#_Toc391042927)

[**28.** **HardCodedJNDINames - Never hard-code JNDI names** 14](#_Toc391042928)

[**29.** **SecondSetPropertyValueString - Avoid having the second argument of setPropertyValue() be a string** 14](#_Toc391042929)

[**30.** **ReturnsLogging - Never 'return' from within if(isLogging\*()) statements** 15](#_Toc391042930)

[**31.** **ContinuesLogging - Never 'continue' from within if(isLogging\*()) statements** 15](#_Toc391042931)

[**32.** **BreaksLogging - Never 'break' from within if(isLogging\*()) statements** 16](#_Toc391042932)

[**33.** **PrintStackTraceInLogging - Never call printStackTrace() from within if(isLogging[Debug/Info/Trace])** 16](#_Toc391042933)

[**34.** **HardCodedSQL - Never hard-code SQL statements** 16](#_Toc391042934)

[**35.** **SystemOutPrintln - Avoid calling System.out.print** 17](#_Toc391042935)

[**36.** **HardCodedUrlParameters - Never hard-code URL parameters** 17](#_Toc391042936)

[**37.** **WrapLogDebug - Always wrap logDebug() with if(isLoggingDebug)** 17](#_Toc391042937)

[**38.** **WrapLogWarning - Always wrap logWarning() with if(isLoggingWarning** 17](#_Toc391042938)

[**39.** **WrapLogError - Always wrap logError() with if(isLoggingError())** 18](#_Toc391042939)

[**40.** **WrapLogInfo - Always wrap logInfo() with if(isLoggingInfo())** 18](#_Toc391042940)

[**41.** **WrapLogTrace - Always wrap logTrace() with if(isLoggingTrace())** 18](#_Toc391042941)

[**42.** **Finalizers - Avoid the user of finalizers** 19](#_Toc391042942)

[**43.** **ActionClassName - Try to name classes that extend PublishingAction \*Action** 19](#_Toc391042943)

[**44.** **FormHandlerClassName - Try to name classes that extend \*FormHandler \*FormHandler** 19](#_Toc391042944)

[**45.** **ServletClassName - Try to name classes that extend InsertableServletImpl \*Servlet** 20](#_Toc391042945)

[**46.** **DropletClassName - Try to name classes that extend DynamoServlet (Droplets) \*Droplet** 20](#_Toc391042946)

[**47.** **GetSessionAttribute - Do not use request.getSession().getAttribute()** 21](#_Toc391042947)

[**48.** **GetRequestAttribute - Do not use request.getAttribute()** 21](#_Toc391042948)

[**49.** **RQL - Avoid use of RQL. Try to use the QueryBuilder API instead** 21](#_Toc391042949)

[**50.** **ServletUtilCurrentUserProfile - Avoid use of ServletUtil.getCurrentUserProfile()** 22](#_Toc391042950)

[**51.** **OneDeclarationPerLine - Only declare one variable per line** 22](#_Toc391042951)

# **Introduction**

This document defines the standards/rules and best practices that needs to be followed while doing code development using ATG Platform. It also provides examples on how the rules should be followed.

This document is primarily intended for developers involved in ATG code development.

# **Guidelines**

## **RedirectInHandleMethods** - Never call sendLocalRedirect() or sendRedirect() in form handler handle\*() methods"

Example :

// bad

public boolean handle\*(DynamoHttpServletRequest pRequest, DynamoHttpServletResponse pResponse) throws ServletException, IOException

{

...

pResponse.sendLocalRedirect(url, pRequest);

pResponse.sendRedirect(url);

}

// good

public boolean handle\*(DynamoHttpServletRequest pRequest, DynamoHttpServletResponse pResponse) throws ServletException, IOException

{

...

return checkFormRedirect(getSuccessURL(), getErrorURL(), pRequest, pResponse);

}

## **StringLiteralCheckFormRedirect** - Never pass hard-coded strings to checkFormRedirect

Example

// bad

return checkFormRedirect("/successURL.jsp", "/errorURL.jsp", pRequest, pResponse);

// good

return checkFormRedirect(getSuccessURL(), getErrorURL(), pRequest, pResponse);

## **GetParameterInHandleMethods - Never call getParameter() in form handler handle\*() methods**

Example:

// bad

public boolean handle\*(DynamoHttpServletRequest pRequest, DynamoHttpServletResponse pResponse) throws ServletException, IOException

{

...

String firstName = pRequest.getParameter("firstName");

}

// good

public boolean handle\*(DynamoHttpServletRequest pRequest, DynamoHttpServletResponse pResponse) throws ServletException, IOException

{

...

// where getFirstName() is set using on the JSP page as a bean and retrieved using traditional ATG-style getter/setters

String firstName = getFirstName();

}

## **PipelineExceptionResourceBundle - Never pass a hard-coded string to PipelineResult.addError**

Example

// bad

pipelineResult.addError("message", "message");

// good

pipelineResult.addError(resourceBundle.getString("key"), resourceBundle.getString("key"));

## **InternationalizedDropletException - Never pass a hard-coded string to the constructor of DropletExceptions**

Example:

// bad

addFormException((new DropletException("message"));

// good

addFormException((new DropletException(resourceBundle.getString("key")));

## **GetMessageDropletException - Never call \*.getMessage() in the constructor of DropletExceptions**

Example

// bad

addFormException(new DropletException("message" + e.getMessage()));

addFormException(new DropletException(resourceBundle.getString("key") + e.getMessage()));

// good

addFormException(new DropletException(resourceBundle.getString("key")));

## **GetMessageLogError** - Never use \*.getMessage() in logError

Example:

// bad

logError(e.getMessage());

logError("message" + e.getMessage());

// good

logError("message", e);

## **GetMessagePipelineException - Never pass \*.getMessage() to PipelineResult.addError**

Example:

// bad

pipelineResult.addError("message" + e.getMessage(), "message" + e.getMessage());

pipelineResult.addError(resourceBundle.getString("key") + e.getMessage(), resourceBundle.getString("key") + e.getMessage());

// good

pipelineResult.addError(resourceBundle.getString("key"), resourceBundle.getString("key"));

## **HardCodedFileExtensions - Avoid hard-coded file extensions**

Example:

// bad

List allowableImageExtensions = new ArrayList();

allowableImageExtensions.add(".jpg");

allowableImageExtensions.add(".gif");

allowableImageExtensions.add(".png");

// good

List allowableImageExtensions = getAllowablefileExtensions();

## **HardCodedURLs - Avoid hard-coded URLs**

Example

// bad

final String successURL = "/context-path/home.jsp";

// good

final String successURL = getSuccessURL();

## **SystemCurrentTimeMillis - Avoid calling System.currentTimeMillis(). Use /atg/dynamo/service/CurrentDate instead**

Example:

// bad

long now = System.currentTimeMillis();

// good

// where getCurrentDate() is a reference to /atg/dynamo/service/CurrentDate

long now = getCurrentDate().getTime();

## **StringLiteralServiceLocalParameter - Avoid passing a hard-coded string to serviceLocalParameter**

Example:

// bad

public void service(DynamoHttpServletRequest request, DynamoHttpServletResponse response) throws ServletException, IOException

{

..

request.serviceLocalParameter("key", request, response);

}

// good

public void service(DynamoHttpServletRequest request, DynamoHttpServletResponse response) throws ServletException, IOException

{

...

request.serviceLocalParameter(CONSTANT, request, response);

## **StringLiteralGetLocalParameter - Avoid passing a hard-coded string to getLocalParameter**

Example:s

// bad

public void service(DynamoHttpServletRequest request, DynamoHttpServletResponse response) throws ServletException, IOException

{

...

String str = (String)pRequest.getLocalParameter("key");

}

// good

public void service(DynamoHttpServletRequest request, DynamoHttpServletResponse response) throws ServletException, IOException

{

...

String str = (String)pRequest.getLocalParameter(CONSTANT);

}

## **StringLiteralGetObjectParameter - Avoid passing a hard-coded string to getObjectParameter**

Example:

// bad

public void service(DynamoHttpServletRequest request, DynamoHttpServletResponse response) throws ServletException, IOException

{

...

String str = (String)pRequest.getObjectParameter("key");

}

// good

public void service(DynamoHttpServletRequest request, DynamoHttpServletResponse response) throws ServletException, IOException

{

...

String str = (String)pRequest.getObjectParameter(CONSTANT);

}

## **GetParameter - Avoid DynamoHttpServletRequest.getParameter(). Use DynamoHttpServletRequest.getLocalParameter() instead**

Example:

// bad

public class \*Droplet extends DynamoServlet

{

public void service(DynamoHttpServletRequest request, DynamoHttpServletResponse response) throws ServletException, IOException

{

...

String str = (String)pRequest.getParameter(KEY);

}

}

// good

public class \*Droplet extends DynamoServlet

{

public void service(DynamoHttpServletRequest request, DynamoHttpServletResponse response) throws ServletException, IOException

{

...

String str = (String)pRequest.getLocalParameter(KEY);

}

}

## **StringLiteralGetParameter - Avoid passing a hard-coded string to getParameter**

Example:

// bad

public void service(DynamoHttpServletRequest pRequest, DynamoHttpServletResponse pResponse) throws ServletException, IOException

{

...

String str = (String)pRequest.getParameter("key");

}

// good

public void service(DynamoHttpServletRequest pRequest, DynamoHttpServletResponse pResponse) throws ServletException, IOException

{

...

String str = (String)pRequest.getParameter(CONSTANT);

}

## **HardCodedPerformanceMonitorOperations - Avoid passing a hard-coded string to PerformanceMonitor.startOperation**

Example:

// bad

PerformanceMonitor.startOperation("key");

PerformanceMonitor.endOperation("key");

// good

final String KEY = "key";

...

PerformanceMonitor.startOperation(KEY);

PerformanceMonitor.endOperation(KEY);

## **HardCodedRQLs - Never hard-code RQL queries**

Example:

// bad

RqlStatement statement = RqlStatement.parseRqlStatement("active = true");

// good

RqlStatement statement = RqlStatement.parseRqlStatement(getActiveRqlQuery());

## **ServiceLocalParameterInLogging - Never call serviceLocalParameter() inside of an if(isLogging\*())**

Example :

// bad

if (isLoggingDebug())

{

...

request.serviceLocalParameter(EMPTY, request, response);

}

// good

if (isLoggingDebug())

{

...

}

request.serviceLocalParameter(EMPTY, request, response);

## **HardCodeRequestNucleusLookup - Avoid passing a hard-coded string to a resolveName() method call**

Example:

// bad

Component c = (Component)pRequest.resolveName("/com/path/to/Component");

// better (but still not great)

Component c = (Component)pRequest.resolveName(COMPONENT\_PATH);

## **NucleusLookup - Avoid resolving Nucleus services statically (such as ServletUtil.getCurrentRequest().resolveName() or Nucleus.getGlobalNucleus().resolveName())**

Example:

// bad

Component c = (Component)ServletUtil.getCurrentRequest().resolveName(COMPONENT\_PATH);

c.doSomething();

// good

getComponent().doSomething(); // where getComponent() has a corresponding setter method and the path to the component is set through a properties file

## **HardCodingNucleusPaths - Avoid hard-coding Nucleus component paths**

Example:

// bad

public static String COMPONENT\_PATH\_PROFILE = "/atg/userprofiling/Profile";

// good

getProfile();

## **GettersSettersForDropletsFormHandlers - Never reference droplets or form handlers from other droplets or form handlers**

Example:

// bad

private \*Droplet \*Droplet;

public \*Droplet get\*Droplet()

{

return \*Droplet;

}

public void set\*Droplet(\*Droplet \*Droplet)

{

this.\*Droplet = \*Droplet;

}

...

getAllScreenDisplayDroplet().doSomething();

// good

get\*Manager().doSomething();

## **StringLiteralSetParameter - Avoid passing a hard-coded string to setParameter() in form handlers or droplets. Use a constant instead**

Example:

// bad

prequest.setParameter("key", "value");

// good

prequest.setParameter(KEY, value);

## **CallTransactionManagerDirectly- Avoid interacting with the transaction manager directly**

Example:

// bad

getTransactionManager().rollback();

// good - example of proper ATG-style transaction demarcation

TransactionDemarcation td=new TransactionDemarcation();

boolean rollback=false;

td.begin(getTransactionManager(), TransactionDemarcation.REQUIRES\_NEW);

...

rollback=true;

...

td.end(rollback);

## **StringLiteralRunProcess - Never pass a string to PipelineManager().runProcess**

Example:

// bad

PipelineResult pResult = getPipelineManager().runProcess("chainId", params);

// good

PipelineResult pResult = getPipelineManager().runProcess(get\*ChainId(), params);

## **GenerateFormExceptionMethod - Avoid defining and using the method generateFormException**

Example:

// bad

public void generateFormException(String whatException, DynamoHttpServletRequest pRequest)

{

...

}

// good

addFormException(new DropletException(resourceBundle.getString("key")));

## **HardCodedJNDINames - Never hard-code JNDI names**

Example:

// bad

DataSource ds = (DataSource) ic.lookup("java:\*\_ds");

// good

DataSource ds = (DataSource)get\*DataSource();

## **SecondSetPropertyValueString - Avoid having the second argument of setPropertyValue() be a string**

Example:

// bad

setPropertyValue(PROPERTY\_NAME, "value");

// good

setPropertyValue(PROPERTY\_NAME, value);

## **ReturnsLogging - Never 'return' from within if(isLogging\*()) statements**

Example

// bad

if (isLogging\*())

{

...

return;

}

// good

if (isLogging\*())

{

...

}

## **ContinuesLogging - Never 'continue' from within if(isLogging\*()) statements**

Example:

// bad

if (isLogging\*())

{

...

continue;

}

// good

if (isLogging\*())

{

...

}

## **BreaksLogging - Never 'break' from within if(isLogging\*()) statements**

Example:

// bad

if (isLogging\*())

{

...

break;

}

// good

if (isLogging\*())

{

...

}

## **PrintStackTraceInLogging - Never call printStackTrace() from within if(isLogging[Debug/Info/Trace])**

Example

// bad

if (isLoggingDebug())

{

e.printStackTrace();

}

// good

if (isLoggingError())

{

logError("message", e);

}

## **HardCodedSQL - Never hard-code SQL statements**

// bad

String sql = "select \* from table ...";

// good

String sql = getSqlQuery();

## **SystemOutPrintln - Avoid calling System.out.print**

Example:

// bad

System.out.println("message");

// good

if (isLogging\*())

{

log\*("message");

}

## **HardCodedUrlParameters - Never hard-code URL parameters**

Example:

// bad

String url = "/page.jsp?param=value";

// good

String url = getUrl();

## **WrapLogDebug - Always wrap logDebug() with if(isLoggingDebug)**

Example:

// bad

logDebug("message");

// good

if (isLoggingDebug())

{

logDebug("message");

}

## **WrapLogWarning - Always wrap logWarning() with if(isLoggingWarning**

Example:

// bad

logWarning("message");

// good

if (isLoggingWarning())

{

logWarning("message");

}

## **WrapLogError - Always wrap logError() with if(isLoggingError())**

Example

// bad

logError("message", e);

// good

if (isLoggingError())

{

logError("message", e);

}

## **WrapLogInfo - Always wrap logInfo() with if(isLoggingInfo())**

Example:

// bad

logInfo("message");

// good

if (isLoggingInfo())

{

logInfo("message");

}

## **WrapLogTrace - Always wrap logTrace() with if(isLoggingTrace())**

Example:

// bad

logTrace("message");

// good

if (isLoggingTrace())

{

logTrace("message");

}

## **Finalizers - Avoid the user of finalizers**

// bad

public void finalize()

{

// do something

}

## **ActionClassName - Try to name classes that extend PublishingAction \*Action**

**Example**

// bad

public class \* extends PublishingAction

{

...

}

// good

public class \*Action extends PublishingAction

{

...

}

## **FormHandlerClassName - Try to name classes that extend \*FormHandler \*FormHandler**

Example:

// bad

public class \* extends PublishingAction

{

...

}

// good

public class \*FormHandler extends \*FormHandler

{

...

}

## **ServletClassName - Try to name classes that extend InsertableServletImpl \*Servlet**

Example:

// bad

public class \* extends InsertableServletImpl

{

...

}

// good

public class \*Servlet extends InsertableServletImpl

{

...

}

## **DropletClassName - Try to name classes that extend DynamoServlet (Droplets) \*Droplet**

Example:

// bad

public class \* extends DynamoServlet

{

...

}

// good

public class \*Droplet extends DynamoServlet

{

...

}

## **GetSessionAttribute - Do not use request.getSession().getAttribute()**

Example:

// bad

String attributeX = request.getSession().getAttribute("attributeX");

// good

String attributeX = getY().getAttributeX(); // where getY() resolves to a session-scoped Nucleus component

## **GetRequestAttribute - Do not use request.getAttribute()**

Example:s

// bad

String attributeX = request.getAttribute("attributeX");

// good

String attributeX = getY().getAttributeX(); // where getY() resolves to a request-scoped Nucleus component

## **RQL - Avoid use of RQL. Try to use the QueryBuilder API instead**

Example:

// bad

RqlStatement statement = RqlStatement.parseRqlStatement("property=?0", true);

RepositoryItem[] items = statement.executeQuery(getRepository().getView("view"), new String["value"] {});

// good

RepositoryView view = getImportMetaDataRepository().getView("view");

QueryBuilder qb = view.getQueryBuilder();

QueryExpression property = qb.createPropertyQueryExpression("property");

QueryExpression value = qb.createConstantQueryExpression("value");

Query query = qb.createComparisonQuery(property, value, QueryBuilder.EQUALS);

RepositoryItem[] items = view.executeQuery(query);

## **ServletUtilCurrentUserProfile - Avoid use of ServletUtil.getCurrentUserProfile()**

Example

// bad

RepositoryItem profile = ServletUtil.getCurrentUserProfile();

// good

Profile profile = getProfileServices().getCurrentProfile();

## **OneDeclarationPerLine - Only declare one variable per line**

Example:

// bad

String lastName,name;

// good

String name;

String lastName;