Basic Code Review Checklist

Applicability	EBX Add-ons	QMS Level	3
Purpose	To list the basic requirements when review code	**Page Status**	PUBLISHED
Standards & Compliance	ISO 27001: A.14.2.1	**Publish Date**	09 Sep 2020
Owner/Author Email	Minh Tran Quang	**Version**	1.0
Classification	For internal use only - TIBCO Confidential		
Page Editing	This page limits editing; suggestions and changes are welcome and can be made via the page comments or contact the owner(s) of this page.		

1.Explain yourself in code

When you have to comment then you have written code that doesn't communicate its purpose well.

DON'T

```
    //for EBXGO to set UserServiceContext
    JAXRSContextInitializerUtils.initWithoutCheck(
        this.httpRequest,
        this.dataSpace,
        this.dataSet,
        this.tablePath,
        this.sessionContext,
        this.selectedRecordIds);
```

DO

```
JAXRSContextInitializerUtils.initContextForEBXGO(
    this.httpRequest,
    this.dataSpace,
    this.dataSet,
    this.tablePath,
    this.sessionContext,
    this.selectedRecordIds);
```

2. Only comment when code is implicit

Comment for obvious things actually makes code hard to follow

DON'T

```
public boolean hasSaveGraphPermission(String owner, Session session)
{
    UserReference userReference = session.getUserReference();
    // same user
    if (userReference.format().equals(owner))
    {
        return true;
    }

    // user in role
    if (PermissionUtils.isRole(owner)
        && PermissionUtils.userInRole(session, userReference, (Role) Profile.parse(owner)))
    {
        return true;
    }

    //user is in admin role
    return session.getDirectory().isUserInRole(userReference, Role.ADMINISTRATOR);
}
```

DO

Those comments should be removed, so obvious.

3. Naming

- ClassName: should be noun, CamelCase, ActionPermission, SchemaExtension
- Method name: should be verb, details what method does (not how it's done), i.e.: getPermission, hideAllNodes…..
- Variable name: actionPermission, schemaExtension, users (dont: usersList/usersSet…)
- Constant: SERVICE, TIME_TO_LIVE...
- Note: a boolean field should not contain "is" as a prefix, the method does
    - DO: boolean hidden, boolean success;
    - DO: public boolean isHidden()
- No encoding: no type/scope information.
    - DON'T: DirectionEnum, ShapeTypeEnum
    - DO: Direction, Shape

4. Name's length depends on scope

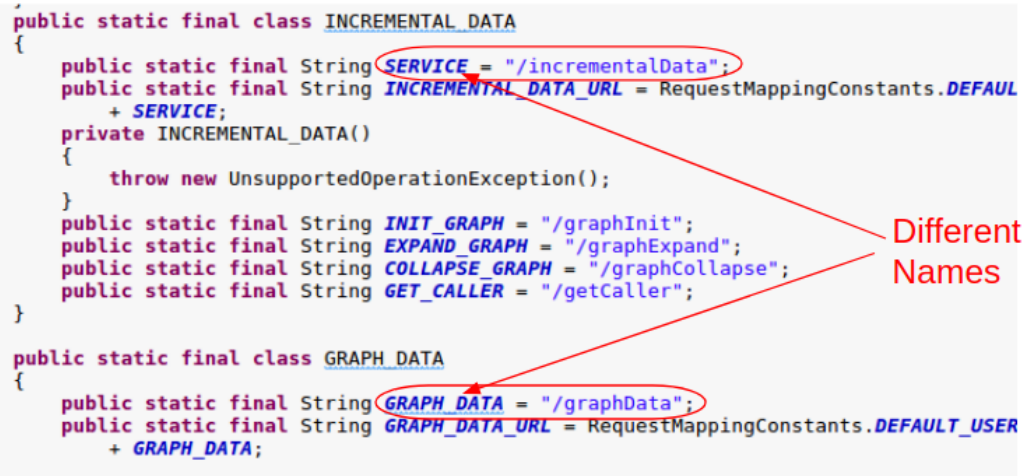
Use long name for long scope

Fields > Parameters > Locals > Loop variables

Long	Short

5. Same words for the same concepts

Multiple developers will contribute to the project, it will be very difficult to read code if each invents another word for the same concept.



6. Use “Final”

- When Class not intended to be extended
- When Method not intended to be overridden
- When Variable not intended to be reassigned
- When you are not sure if later you will need to extend/override/reassign or not, use Final and remove it later when you need to extend/override /reassign

7. Must declare access modifiers

Minimize accessibility when possible: prefer private over protected over public

8. Declare static string within class if it never changes

9. Avoid unused objects, methods, variables

10. Avoid String concat, use StringBuilder

DON'T

```
String result = "";
for (int i=0;i<10;i++){
    result += "some text";
}
```

DO

```
StringBuilder result = new StringBuilder();
for (int i=0;i<10;i++){
    result.append("some text");
}
```

11. DRY: Don’t repeat yourself

```
        LinksToDocumentation.USER_GUIDE.USER_UPLOAD_ASSETS_AND_ATTACH_TO_FIELD);
    this.damDataset = DAMConfigurationAccess.getInstance()
        .getDAMConfigurationDataset(this.contentController.getRepository());
    String dacCode = AddonWebComponentUtils
        .getSpecificParameterValue(this.contentController.getSession(), DAMConstants.DAC_CODE
    String selectedDriveId = this.request.getParameter(DAMConstants.DRIVE_ID).trim();
    String webName = AddonWebComponentUtils
        .getSpecificParameterValue(this.contentController.getSession(), DAMConstants.WEBNAME)
    if (AddonStringUtils.isEmpty(webName))
    {
        webName = DAMConstants.EMPTY_STRING;
    }

    String uploadAssetContext = AddonWebComponentUtils.getSpecificParameterValue(
        this.contentController.getSession(),
        DAMConstants.UPLOAD_ASSET_IN_CONTEXT);
    DAMCommonUtils damCommonUtils = DAMCommonUtils.getInstance();
    UploadAssetContextEnum uploadAssetContextEnum = UploadAssetContextEnum
        .parse(uploadAssetContext);
    this.uploadAssetContextKey = new UploadAssetContextKey(
        this.contentController.getCurrentDataset().getAdaptationName().getStringName(),
        this.contentController.getCurrentHome().getKey().format(),
        dacCode,
        selectedDriveId,
```

Duplicated (there are 2 more usages of this call in the method)

→ final Session currentSession = this.contentController.getSession()

Convention

```
1120        for (RecordGroupingExecutionResult record : entryByUUID.getValue())
1121        {
1122            String executionUUID = record.getExecutionUUID();
1123            if (!AddonStringUtils.isEmpty(executionUUIDForOtherGroup)
1124                && !executionUUIDForOtherGroup.equals(executionUUID))
1125            {
1126                continue;
1127            }
1128            if (AddonStringUtils.isEmpty(executionUUIDForOtherGroup))
1129            {
1130                executionUUIDForOtherGroup = executionUUID;
1131            }
```

```
1727                String executionUUID = record.getExecutionUUID()
1728                if (!AddonStringUtils.isEmpty(executionUUIDForOt
1729                    && !executionUUIDForOtherGroup.equals(execut
1730                {
1731                    continue;
1732                }
1733                if (AddonStringUtils.isEmpty(executionUUIDForOth
1734                {
1735                    executionUUIDForOtherGroup = executionUUID;
1736                }
```

.....

```
1152        // sort by group Name
1153        List<Entry<String, Integer>> groupNameEntries = new ArrayList(
1154            numberOfRecordsByGroupName.entrySet());
1155        Collections.sort(groupNameEntries, new Comparator<Map.Entry<String, Integer>>()
1156        {
1157            public int compare(Entry<String, Integer> groupName1, Entry<String, Integer> groupName2)
1158            {
1159                if (groupName1.getValue().compareTo(groupName2.getValue()) == 0)
1160                {
1161                    return groupName1.getKey().compareTo(groupName2.getKey());
1162                }
1163                return highestOrderValue ? groupName2.getValue().compareTo(groupName1.getValue())
1164                    : groupName1.getValue().compareTo(groupName2.getValue());
1165            }
```

.....

```
1757        // sort by group Name
1758        List<Entry<String, Integer>> groupNameEntries = new ArrayList(
1759            numberOfRecordsByGroupName.entrySet());
1760        Collections.sort(groupNameEntries, new Comparator<Map.Entry<String, Integer>>()
1761        {
1762            public int compare(
1763                Entry<String, Integer> groupName1,
1764                Entry<String, Integer> groupName2)
1765            {
1766                if (groupName1.getValue().compareTo(groupName2.getValue()) == 0)
1767                {
1768                    return groupName1.getKey().compareTo(groupName2.getKey());
1769                }
1770                return highestOrderValue
1771                    ? groupName2.getValue().compareTo(groupName1.getValue())
1772                    : groupName1.getValue().compareTo(groupName2.getValue());
1773            }
```

Block of code 1120 → 1165 is copied to block 1727 → 1773 (even the comments)

→ Very BAD
→ Should extract to method, util to process

These kinds of duplication can be detected by Sonar.

12. Don't ignore exception, must log meaningful message

DON'T

```
public void handleDeleteAdaptation(String primaryKey)
{
    this.lock.lock();
    try
    {
        if (this.data.containsKey(primaryKey))
            this.data.remove(primaryKey);

        if (this.indexedList.contains(primaryKey))
            this.deleteRevertIndex(primaryKey);

        this.updateCacheOfIncomingTables(primaryKey, this.getRelatedTableNotification()
    }
    finally
    {
        this.lock.unlock();
    }
}
```

The purpose of “try” here is to release the lock. However, if there’s an exception inside the “try” block then we will never know what happened and why.

13. Avoid returning null

DON'T

```
public List<String> getNodeChildren(String nodeKey)
{
    if (!this.containNode(nodeKey))
    {
        return null;
    }

    return PropertyConverter
        .cast(this.getNode(nodeKey).getProperty(NodeModelDTOPropertyKey.CHILDREN
}
```

DO: return empty List, use Null Object pattern…

14. If a method may return null the caller must check null
15. Method should do one thing
16. Minimize scope of local variables
17. Declare variable near where it’s used
18. Refer to Objects by their interface

DON'T

```
public void updateVisibleTableList(Set<IncrementalNodeDisplayBean> ne
{
    for (IncrementalNodeDisplayBean node : newNodes)
    {
        IncrementalNodeImp<NodeDTOPropertyKey> incrementalNode = (Inc
            .getNode(node.getDisplayNodeKey());
        Table currentTable = this.getTableOfNode(incrementalNode);
        this.updateDisplayTableList(currentTable, locale);
        node.setTable(this.tableInfos.get(currentTable));
    }
}
```

DO: Use interface Node instead

19. Early exit return asap, or continue asap

    This helps remove arrow functions, makes code cleaner for readers:

    DON'T

```
if (condition)
{
    // do
    // some
    // complex
    // computation
    // and
    // computation
    // here
}
else
{
    // simple task/or return something fails here
}
```

    DO

```
if (!condition)
{
    // simple task/return something fails here
}

// do
// some
// complex
// computation
// and
// computation
// here
```

20. Classes those have sub-classes and no initialization must be abstract
21. Avoid complex methods (refer to SonarQube report)

    A method with maximum complexity around 17 is allowed.

22. equals() and hashCode()

    When override equals(), must also override hashCode and cache the hashCode for better performance

23. Small param list (< 7)
24. Remove auto generated comments and annotations
25. No commented out code

Just remove it

Others points to check when reviewing

- Can a computation be moved out of loop?
- Can loops operating on the same data be combined into one
- Are objects casted between methods
- Encapsulate complex conditions to meaningful statement

DON'T

```
String userName = userReference.format();
boolean checkAdmin = session.getDirectory().isUserInRole(userReference, Role.ADMINISTRATO
if ((checkAdmin && (shareProfileList.contains(Profile.ADMINISTRATOR.format())
    || owner.equals(Profile.ADMINISTRATOR.format())))
    || owner.equals(Profile.EVERYONE.format())
    || (owner.equals(userName) || shareProfileList.contains(userName)
        || shareProfileList.contains(Profile.EVERYONE.format())))
{
    return true;
}
```

DO

```
boolean isAdmin = ......;
boolean isEveryone = ......;

if (isAdmin || isEveryone)
{
    //do stuff
}
```

Sign-off History

Action	Name	Date
Prepared by	Thi Viet Phuong Luu	21 Feb 2020
Approved	Minh Tran Quang	09 Sep 2020

Revision History

Version	Date	Authors	Description
1.0	09 Sep 2020	Minh Tran Quang	Initial version