January 14, 2018

Programing Best Practices

DXC.technology

Course Objectives

- At the end of the course, you will have acquired sufficient knowledge to:
  - ✓ Make your code more readable
  - ✓ Easy to maintain
  - ✓ Clearly identify between good and bad code
  - ✓ Refactor bad code to good code
  - ✓ Better understanding of error handling

Agenda

- Java Code Convention
- Java Code in JSP
- CSS Code Convention
- JS Code Convention
- Clean Code
- Best Practices

Assessment Disciplines

- Pass score: 70%

Java Code Convention

File & Names

- File Suffixes:

File Type	Suffix
Java Source	.java
Java Bytecode	.class

- Standard naming conventions:

  http://www.oracle.com/technetwork/java/codeconventions-150003.pdf

File & Names

- Avoid using keyword as identifiers

abstract	boolean	break	byte	case	catch
char	class	const	continue	default	do
double	else	extends	final	finally	float
for	goto	if	implements	import	instanceof
int	interface	long	native	new	package
private	protected	public	return	short	static
strictfp	super	switch	synchronized	this	throw
throws	transient	try	void	volatile	while
assert	enum				

File & Names
(also apply for JS Code Convention)

Identifier Type	Examples
Package	`package com.dxc.life.agents.procedures`
Classes/Interfaces	`class PremiumCalculator;` `interface Runable;`

Methods	`calculatePremiumValue();`
	`checkAgeProspect();`
	`getBackground();`

Variables	`int i;`
	`char *cp;`
	`float myExampleWidth;`

Constants	`int MIN_WIDTH = 4;`
	`int MAX_WIDTH = 999;`
	`int GET_THE_CPU = 1;`

Program file and document

- Place each class in separate file.

WRONG

- Class1.java

```
public class Class1{
}
class Class2{}
class Class3{}
```

- ClassDemo2.java

```
public class Class1{
}
public class Class2{}
public class Class3{}
```

CORRECT

- Class1.java

```
Public class Class1{
}
```

- Class2.java

```
class Class2{
}
```

- Class3.java

```
class Class3{
}
```

Program file and document (Cont.)

- Beginning Comments

```
/**
 * Screen variables for S4122
 * @version 1.0 generated on 26/12/08 09:03
 * @author Quipoz
 */
```

- Package and Import Statements

```
package com.dxc.tutorial.programs;
import com.dxc.tutorial.records;
```

- Indentation
- Line Length

Program file and document (Cont.)

```
package com.dxc.demo;

import java.io.File;

import net.sourceforge.tess4j.ITesseract;
import net.sourceforge.tess4j.Tesseract;
import net.sourceforge.tess4j.TesseractException;

public class TessaractExample {
public static void main(String[] args) {
File imageFile = new File("D:\\Picture\\cobolScreen\\cobolSmartScreenShort2_ja.png");
ITesseract instance = new Tesseract();
instance.setDatapath("D:\\Softwares\\Tess4J");
instance.setLanguage("jpn");
try {
String result = instance.doOCR(imageFile);
System.out.println(result);
} catch (TesseractException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
```

Program file and document (Cont.)

```java
package com.dxc.demo;

import java.io.File;

import net.sourceforge.tess4j.ITesseract;
import net.sourceforge.tess4j.Tesseract;
import net.sourceforge.tess4j.TesseractException;

public class TessaractExample {
    public static void main(String[] args) {
        File imageFile = new File("D:\\Picture\\cobolScreen\\cobolSmartScreenShort2_ja.png");
        ITesseract instance = new Tesseract();
        instance.setDatapath("D:\\Softwares\\Tess4J");
        instance.setLanguage("jpn");
        try {
            String result = instance.doOCR(imageFile);
            System.out.println(result);
        } catch (TesseractException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Structure
(also apply for JS Code Convention)

- Comments
  - ✓ Block

```
/**
 * @param defectStatusList
 * @return
 */
private List<DefectDate> getDefectDateList(List<DefectStatus>   defectStatusList) {
        ...
}
```

Structure (also apply for JS Code Convention)

- Comments (Cont.)

✓ Single-Line

```
/* Get the collating character */
Connection conn = null;
PreparedStatement aps = null;
ResultSet rs = null;
```

✓ Trailing

```
if (a == 2) {
    return TRUE;          /* special case */
} else {
    return isPrime(a);    /* works only for
                             odd a */
}
```

✓ End-Of-Line

```
if (foo > 1) {
    // Do a double-flip.
    ...
}
else {
    return false;    // Explain why here.
}
//if (bar > 1) {
//
//    // Do a triple-flip.
//    ...
//}
//else {
//    return false;
//}
```

Structure (also apply for JS Code Convention)

- Comments (Cont.)

  ✓Documentation

```
/**
 * The Example class provides ...
 * @author  Zara Ali
 * @version 1.0
 * @since   2014-03-31
 */
public class Example {
...
}
```

Structure (also apply for JS Code Convention)

- Do-while Statements

```
do {
        statements;
} while (condition);
```

```
public class Examples {
    public static void main(String[] args) {
        int i = 1;
        do {
            System.out.println(i);
            i++;} while (i <= 10);
    }
}
```

```
public class Examples {
    public static void main(String[] args) {
        int i = 1;
        do {
            System.out.println(i);
            i++;
        } while (i <= 10);
    }
}
```

Structure (also apply for JS Code Convention)

- Switch Statements

```
switch (condition) {
case ABC:
     statements;
     /* falls through */
 case DEF:
     statements;
      break;
 default:
     statements;
      break;
```

Structure (also apply for JS Code Convention)

- Switch Statements (Cont.)

```
public static void main(String[] args) {
    char grade = 'C';

    switch (grade) {
    case 'A':
        System.out.println("Excellent!");
    case 'B':
    case 'C':
        System.out.println("Well done");
        break;
    case 'D':
        System.out.println("You passed");
        break;
    case 'F':
        System.out.println("Better try again");
        break;
    default:
        System.out.println("Invalid grade");
    }
    System.out.println("Your grade is " + grade);
}
```

```
public static void main(String[] args) {
    char grade = 'C';

    switch (grade) {
    case 'A':
        System.out.println("Excellent!");
    case 'B':
        /* falls through */
    case 'C':
        System.out.println("Well done");
        break;
    case 'D':
        System.out.println("You passed");
        break;
    case 'F':
        System.out.println("Better try again");
        break;
    default:
        System.out.println("Invalid grade");
        break;
    }
    System.out.println("Your grade is " + grade);
}
```

Structure (also apply for JS Code Convention)

- Try catch Statements

```
try {
        statements;
 } catch (ExceptionClass e) {
        statements;
 }
```

```
try {
    p = Runtime.getRuntime().exec("cmd /c C:\\Users\\Vnguyen86\\Desktop\\" + "training.WS");
}
catch (IOException e) {System.out.println("RunCobalSmart: " + e.getMessage());}
```

```
try {
    p = Runtime.getRuntime().exec("cmd /c C:\\Users\\Vnguyen86\\Desktop\\" + "training.WS");
} catch (IOException e) {
    System.out.println("RunCobalSmart: " + e.getMessage());
}
```

Structure (also apply for JS Code Convention)

- White Space

  ✓Blank lines

```
public AutoItKit() {
    String jacobDllVersionToUse;
    if (System.getProperty("sun.arch.data.model").contains("32")) {
        jacobDllVersionToUse = "jacob-1.18-x86.dll";
    } else {
        jacobDllVersionToUse = "jacob-1.18-x64.dll";
    }
    File file = new File("lib", jacobDllVersionToUse);
    System.setProperty(LibraryLoader.JACOB_DLL_PATH, file.getAbsolutePath());
    //init AutoItX
    autoItX = new AutoItX();
}
public void sleep(int timeInSecond) {
    autoItX.sleep(timeInSecond);
}
public void inputText(String text, String controlTitle, String controlID) {
    autoItX.ControlSetText(controlTitle, "", controlID, text);
}
public void clickControl(String controlTitle, String controlID) {
    autoItX.controlClick(controlTitle, "", controlID);
}
```

```
public AutoItKit() {
    String jacobDllVersionToUse;

    if (System.getProperty("sun.arch.data.model").contains("32")) {
        jacobDllVersionToUse = "jacob-1.18-x86.dll";
    } else {
        jacobDllVersionToUse = "jacob-1.18-x64.dll";
    }
    File file = new File("lib", jacobDllVersionToUse);
    System.setProperty(LibraryLoader.JACOB_DLL_PATH, file.getAbsolutePath());

    //init AutoItX
    autoItX = new AutoItX();
}

public void sleep(int timeInSecond) {
    autoItX.sleep(timeInSecond);
}

public void inputText(String text, String controlTitle, String controlID) {
    autoItX.ControlSetText(controlTitle, "", controlID, text);
}

public void clickControl(String controlTitle, String controlID) {
    autoItX.controlClick(controlTitle, "", controlID);
}
```

Structure (also apply for JS Code Convention)

- While Space (continue)
  - ✓Blank Spaces

```
if(true){
    a+=c+d;
    a=(a+b)/(c*d);
    n++;
    System.out.println("size is "+foo+"\n");
}

for(int i=0;i<10;i++){
    //statement
}
```

```
if (true) {
    a += c + d;
    a = (a + b) / (c * d);
    n++;
    System.out.println("size is " + foo + "\n");
}

for (int i = 0; i < 10; i++) {
    //statement
}
```

JAVA Codes in JSP

JAVA Codes in JSP

- **AVOID** writing Java code (declarations, scriptlets and expressions) in your JSP pages:
  - ✓ Harder to debug
  - ✓ Harder to maintain, read & understand, especially for page authors who may not be Java experts
  - ✓ JSP pages are primarily intended for presentation logic

JAVA Codes in JSP

```
<%
    // NOT RECOMMENDED TO BE DONE AS A SCRIPTLET!

    Connection conn = null;
    try {
        // Get connection
        InitialContext ctx = new InitialContext();
        DataSource ds = (DataSource)ctx.lookup("customerDS");
        conn = ds.getConnection();

        // Get customer names
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT name FROM customer");

        // Display names
        while ( rs.next() ) {
            out.println( rs.getString("name") + "<br>");
        }
    } catch (SQLException e) {
        out.println("Could not retrieve customer names:" + e);
    } finally {
        if ( conn != null )
            conn.close();
    }
%>
```

JAVA Codes in JSP

```
<!-- CORRECT -->
<myTags:dataSource
    name="customerDS"
    table="customer"
    columns="name"
    var="result" />
<c:forEach var="row" items="${result.rows}">
    <c:out value="${row.name}" />
    <br />
</c:forEach>
```

JAVA Codes in JSP

```
<table>
    <%  for ( int i=0; i<customers.length; i++ ) { %>
        <tr>
            <td><%= customers[i].getLastName() %></td>
            <td><%= customers[i].getFirstName() %></td>
        </tr>
    <%  } %>
</table>
```

JAVA Codes in JSP

```
<table>
    <c:forEach var="customer" items="${customers}">
        <tr>
            <td><c:out value="${customer.lastName}"/></td>
            <td><c:out value="${customer.firstName}"/></td>
        </tr>
    </c:forEach>
</table>
```

JSP Implicit Objects

```
<web-app>
…
<context-param>
<param-name>parameter1</param-name>
<param-value>ValueOfParameter1</param-value>
</context-param>
</web-app>
```

- Not use:

```
getServletConfig().getServletContext().getInitParameter("parameter1")
```

- Should use:

```
application.getInitParameter("parameter1")
```

```
<c:out value="${initParam['parameter1']}" />
```

Use of JavaScript Technology

- Include JavaScript file to JSP.
  - ✓ This improves the chance for the JavaScript code to be reused, maintains the consistent behavior of the JavaScript code across multiple JSP pages, and reduces the complexity of JSP page

```
<script language=javascript src="/js/main.js">
```

Use of JavaScript Technology

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Implementing css and javascript</title>
<link rel="stylesheet" href="../css/style.css" type="text/css"></link>
</head>

<form enctype="multipart/form-data" onSubmit="return validate(this)" method="post" action="Welcome.html">
<table border = "0">
    <tr align="left" valign="top">
        <td>User Name:</td>
        <td><input type="text" name ="user" class="inputbox"/></td>
    </tr>
    <tr align="left" valign="top">
        <td>Password:</td>
        <td><input type="password" name ="pass" class="inputbox"/></td>
    </tr>
    <tr align="left" valign="top">
        <td></td>
        <td><input type="submit" name="submit" value="submit" class="submitButton"/></td>
    </tr>
</table>
</form>
<script type="text/JavaScript">
    function validate(theForm){
        if(theForm.user.value.length==0){
            alert("UserId can't be blank");
            theForm.user.focus();
            return false;
        }else if(theForm.pass.value.length==0){
            alert("Password can't be blank");
            theForm.pass.focus();
            return false;
        }else if(theForm.pass.value.length<6){
            alert("Password length can't be less than 6 char");
            theForm.pass.focus();
            return false;
        }
    }
</script>
```

Use of JavaScript Technology

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Implementing css and javascript</title>
<link rel="stylesheet" href="../css/style.css" type="text/css"></link>
</head>

<form enctype="multipart/form-data" onSubmit="return validate(this)" method="post" action="Welcome.html">
<table border = "0">
    <tr align="left" valign="top">
        <td>User Name:</td>
        <td><input type="text" name ="user" class="inputbox"/></td>
    </tr>
    <tr align="left" valign="top">
        <td>Password:</td>
        <td><input type="password" name ="pass" class="inputbox"/></td>
    </tr>
    <tr align="left" valign="top">
        <td></td>
        <td><input type="submit" name="submit" value="submit" class="submitButton"/></td>
    </tr>
</table>
</form>
```

Use of JavaScript Technology

```
<script type="text/JavaScript">
    function validate(theForm){
        if(theForm.user.value.length==0){
            alert("UserId can't be blank");
            theForm.user.focus();
            return false;
        }else if(theForm.pass.value.length==0){
            alert("Password can't be blank");
            theForm.pass.focus();
            return false;
        }else if(theForm.pass.value.length<6){
            alert("Password length can't be less than 6 char");
            theForm.pass.focus();
            return false;
        }
    }
</script>
```

Use of JavaScript Technology

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Implementing css and javascript</title>

<link rel="stylesheet" href="../css/style.css" type="text/css"></link>
<script language="JavaScript" type="text/JavaScript" src="../script/validate.js"></script>
</head>

<form enctype="multipart/form-data" onSubmit="return validate(this)" method="post" action="Welcome.html">
```

CSS Code Conventions

How to include CSS

- External

```
<html>
    <head>
        <!-- GOOD WAY -->
        <link
rel="stylesheet"
href="my_theme.css"
type="text/css">
</head>
……
</html>
```

- Internal

```
<html>
    <body>
        <!- BAD WAY !!! -->
        <span
style="color:red;">Inline
CSS is forbidden!<span>
    </body>
</html>
```

- Embedded

```
<html>
    <head>
        <!- BAD WAY !!! -->
        <style
type="text/css">
          <!--
          .myclass
          {color: red;}
          -->
        </style>
</head>
……
</html>
```

Separating concerns

- For HTML tags
  - ✓ *reset.css*
  - ✓ *default-theme-html.css*
  - ✓ *blue-theme-html.css*
- For useful classes
  - ✓ *default-theme-classes-page-layout.css*
  - ✓ *default-theme-classes-nice-forms.css*
  - ✓ *default-theme-ids-dynamic-content.css*

Multiple Selectors

```
h1,
h2,
h3,
table th,
p.myclass,
.yourclass,
#thisone
{color: #00FFA0;}
```

Properties

- Multi-values properties

```
Body {font-family: helvetica, sans-serif;}
```

- Shorthand properties

```
h1 {border-width: 1px;
    margin: 5px;
    padding: 0px;}
h1 {margin: 0px 5px 10px 5px;} /* BAD !!! */
h1{margin-top: 0px;
    margin-right: 5px;
    margin-bottom: 10px;
    margin-left:5px;}              /* GOOD */
h1 {margin: 0px; margin-top: 5px; margin-left: 10px;} /* BAD !!!   */
h1{margin: 0px;
    margin-top: 5px; margin-left: 10px;} /* GOOD */
```

Naming convention

- Compose names using hyphen “-” separator

```
.sales-order /* GOOD */
.salesOrder, /* BAD !!! */
.SalesOrder, /* BAD !!! */
.sales_order /* BAD !!! */
```

- Meaning full English description

```
.first-name, /* GOOD */
.last-name, /* GOOD */
.middle-initial/* GOOD */
```

- Rather than the shorter versions

```
.f-name, /* BAD !!! */
.last-n, /* BAD !!! */
.m-I     /* BAD !!! */
```

Naming convention

- Choose semantic names based on functionality, not on appearance or position

  For example, use functional names like:

```
.budget-debit{color: #FF0000;}
.budget-credit{color: #00FF00;} /* GOOD */
```

- Do not use appearance names like:

```
.red {color: #FF0000;} /* BAD !!! */

.perishable-product {font-weight: normal;} /* GOOD */
#selected-perishable-product {font-weight: bold;} /* GOOD */

.perishable-product {font-weight: normal;} /* BAD !!! */
#perishable-product {font-weight: bold;}
```

Colors, Unit, URLs, Grouping, Resetting html elements

- Colors
  - ✓ Hexadecimal format and uppercase: #00FFA0
  - ✓ Except for Black, White and Transparent, which are written to plain text in lower case: white.
- Unit:
  - ✓ Prefer measurement units which are re-sizeable by browsers: “%” or “em”.
  - ✓ For precision positioning (relative image positions) use: “px”. It is forbidden to use “pt”, use “px” instead.
- URLs:

  All url are written surrounded by quotes ‘

```
table th{background: url('img/table-header.png') repeat-x;}
```

- Resetting html elements

```
html{height: 100%; margin-bottom: 1px; /* force vertical scrollbar */}
```

JS Code Conventions

Function Declarations

- There should be no space between the name of a function.

```
function outer(c, d) {
        var e = c * d;
        function inner(a, b) {
            return (e * a) + b;
        }
        return inner(0, 1);
    }
```

- There should be one space between the word function and the (

```
div.onclick = function (e) {
      return false;
  };
```

Clean Code

Using check list

- ✓ Follow coding convention
- ✓ Follow design (code design, unit test)
- ✓ Follow Software Development (SonarQuebe code quality)
- ✓ Follow best practice

Check List Code Convention

Checklist Item	Category
Use Intention-Revealing Names	Meaningful Names
Pick one word per concept	Meaningful Names
Use Solution/Problem Domain Names	Meaningful Names
Classes should be small!	Classes
Functions should be small!	Functions

Check List Code Convention

Checklist Item	Category
Do one Thing	Functions
Don't Repeat Yourself (Avoid Duplication)	Functions
Explain yourself in code	Comments
Make sure the code formatting is applied	Formatting
Use Exceptions rather than Return codes	Exceptions
Don't return Null	Exceptions

Best Practices

General

- Use Strings carefully

```
//Slower Instantiation
  String bad = new String("Yet another string object");
//Faster Instantiation
  String good = "Yet another string object"
```

- Dilemma between Array and ArrayList

```
public static void main(String args[]) {
    int[] myArray = new int[6];
    myArray[7] = 10; // ArraysOutOfBoundException
    // Declaration of ArrayList. Add and Remove of elements is easy.
    ArrayList<Integer> myArrayList = new ArrayList<>();
    myArrayList.add(1);
    myArrayList.add(2);
    myArrayList.add(3);
    myArrayList.add(4);
    myArrayList.add(5);
    myArrayList.remove(0);
    for (int i = 0; i < myArrayList.size(); i++) {
        System.out.println("Element: " + myArrayList.get(i));
    }
    // Multi-dimensional Array
    int[][][] multiArray = new int[3][3][3];

}
```

General

- Avoiding Memory leaks by simple tricks
  - ✓ Always release database/stream/http connections when querying/performing is complete
  - ✓ Try to use Finally block as often possible

- Choice between Float and Double

Data type	Bytes used	Significant figures (decimal)
Float	4	7
Double	8	15

General

- FileOutputStream Vs. FileWriter

```
File foutput = new File(file_location_string);
FileOutputStream fos = new FileOutputStream(foutput);
BufferedWriter output = new BufferedWriter(new OutputStreamWriter(fos));
output.write("Buffered Content");

FileWriter fstream = new FileWriter(file_location_string);
BufferedWriter output = new BufferedWriter(fstream);
output.write("Buffered Content");
```

- Log in Java
  - ✓ Logging level (DEBUG, INFO, WARN and ERROR)
  - ✓ Performance affect

```
if(logger.isDebugEnabled()){
    logger.debug("java logging level is DEBUG Enabled");
}
```

SQL Injection

- Define POST variables

```
uname = request.POST['username’]
passwd = request.POST['password’]
```

- SQL query vulnerable to SQLi

```
sql = “SELECT id FROM users WHERE username=’” + uname + “’AND password=’” + passwd + “’”
```

- Execute the SQL statement

```
database.execute(sql)
```

- Secure Usage

```
PreparedStatement stmt = connection.prepareStatement("SELECT * FROM users WHERE username
=? AND password =?");
stmt.setString(1, uname);
stmt.setString(2, password);
ResultSet rs = stmt.executeQuery();
```

REST service

- Use nouns but no verbs

Resource	GET read	POST create	PUT update	DELETE
/cars	Returns a list of cars	Create a new car	Bulk update of cars	Delete all cars
/cars/711	Returns a specific car	Method not allowed (405)	Updates a specific car	Deletes a specific car

- GET method and query parameters should not alter the state

```
GET /users/711?activate
GET /users/711/activate
```

- Use plural nouns

```
/cars instead of /car
/users instead of /user
/products instead of /product
/settings instead of /setting
```

REST service

- Use sub-resources for relations

```
GET /cars/711/drivers/ Returns a list of drivers for car 711
GET /cars/711/drivers/4 Returns driver #4 for car 711
```

- Use HTTP headers for serialization formats

  Content-Type defines the request format.

  Accept defines a list of acceptable response formats.

- Use HATEOAS

```
{
  "id": 711,
  "seats": 5,
  "drivers": [
    {
      "id": "23",
      "name": "Stefan Jauker",
      "links": [
        {
          "rel": "self",
          "href": "/api/v1/drivers/23"
        }
      ]
    }
  ]
```

REST service

- Provide filtering, sorting, field selection and paging for collections

```
GET /cars?color=red Returns a list of red cars
GET /cars?seats<=2 Returns a list of cars with a maximum of 2 seats

GET /cars?sort=-manufactorer,+model

GET /cars?fields=manufacturer,model,id,color

GET /cars?offset=10&limit=5
```

- Version your API

```
/blog/api/v1
```

- Handle Errors with HTTP status codes

Error Handling

- Type of exceptions
  - ✓ Checked exceptions
  - ✓ Unchecked exceptions
  - ✓ Errors

```
class Main {
    public static void main(String[] args) throws IOException {
        FileReader file = new FileReader("C:\\test\\a.txt");
        BufferedReader fileInput = new BufferedReader(file);

        // Print first 3 lines of file "C:\test\a.txt"
        for (int counter = 0; counter < 3; counter++)
            System.out.println(fileInput.readLine());

        fileInput.close();
    }
}
```

Error Handling

- User defined custom exceptions

```
}catch (NoSuchMethodException e) {
      throw new DaoObjectNotFoundException("Couldn't find dao with id "
                  + id, e);
}
```

- Never swallow the exception in catch block

```
catch (NoSuchMethodException e) {
    return null;
}
```

- Declare the specific checked exceptions that your method can throw

```
public void foo() throws SpecificException1, SpecificException2
{

} //Correct way
```

Error Handling

- Always correctly wrap the exceptions in custom exceptions so that stack trace is not lost

```
catch (NoSuchMethodException e) {
       throw new MyServiceException("Some information: " +
              e.getMessage());  //Incorrect way
}
catch (NoSuchMethodException e) {
       throw new MyServiceException("Some information: " , e);  //Correct way
}
```

- Do not catch the Exception class rather catch specific sub classes

```
try {
   someMethod();
} catch (Exception e) {
   LOGGER.error("method has failed", e);
}
```

Error Handling

- Always catch only those exceptions that you can actually handle

```
catch (NoSuchMethodException e) {
   throw e; //Avoid this as it doesn't help anything
}
```

- Never use exceptions for flow control in your program

Q&A

Revision History

Date	Version	Description	Updated by	Reviewed and Approved By
9 Jan 2018	1.0	First draft	Vy Nguyen	
12 Jan 2018	1.1	Modified Trainer Name to Trainer Correct text referring CSC to DXC Add examples Remove XML convention section Slide 2: remove duplicated slide	Vy Nguyen	
15 Jan 2018	1.2	Highlight keywords in red Slide10,11,12: add Cont. in title Slide 12: remove text “Example” Slide 13,14: change color for comment and code statement Slide 18,19: zoom out example images	Vy Nguyen	
17 Jan 2018	1.3	Slide 4: remove the first two bullets. Slide 30: split into 2 slides	Vy Nguyen	Quang Tran
19 Apr 2018	1.4	Slide 15: add more comment	Vy Nguyen	Quang Tran