Cartilha do Desenvolvedor Java

Projeto de Software

Alphalinc Upgrade

Outubro/2014

TASC (the alpha supply chain)

1 Índice

2 Objetivo

Este documento tem como objetivo estabelecer as ações necessárias aos desenvolvedores em Java para adequação das funcionalidades e correção de problemas durante a validação do código convertido.

3 Histórico de Revisões

Data	*Versão*	*Autor*	*Descrição*
30/09/14	1.00	Alexandre van den Mosselaar	Criação do documento

4 Pontos de Intervenção

4.1 Referência de globais de índice (^...s)

Causa	As classes definidas como relacionais (códigos de mapeamento 0, 2 ou 4) não possuem globais de índice (somente as globais com mapeamento 90, 91 ou 92 irão possuir índices)
Consequência	A referência não irá encontrar nenhum registro e a aplicação não irá funcionar corretamente
Solução	Construir um resultset com acesso aos dados via SQL (o comando SQL deverá utilizar o índice correspondente no banco relacional)

4.2 Comandos SQL embutido (&SQL)

Causa	Os comandos SQL embutidos não são convertidos pelo conversor de código
Consequência	Os comandos SQL não serão executados no código convertido
Solução	Construir um resultset para cada comando SQL embutido

4.3 Comando SQL inválido em ResultSet

Causa	O banco de dados relacional gerou um erro com o comando SQL passado pelo resultset (os comandos gerados pelo resultset são ajustados conforme o banco, mas nem todas as opções são suportadas)
Consequência	O comando SQL não será executado (o Java irá gerar uma exception do erro SQL)
Solução	Procurar corrigir o problema no comando SQL (caso o problema seja corrigido mas seria importante ajustar o resultset para tratar este comando, repasse esta informação para o suporte técnico), caso o problema não possa ser corrigido no comando SQL, repasse o problema para o suporte técnico (conforme o caso pode ser necessário criar novas funções no banco de dados relacional para suportar o comando SQL).

4.4 Comando interpretado muito complexo (xecute)

Causa	O comando interpretado não foi suportado pelo xecute
Consequência	O comando não será executado (o Java irá gerar uma exception do problema no xecute)
Solução	Verificar se é possível simplificar o comando ou retirar a execução interpretada (caso o problema seja corrigido mas seria importante contemplar este comando no futuro, repasse esta informação para o suporte técnico), caso não seja possível, encaminhar o problema para o suporte técnico.

4.5 Acionamento de rotinas não convertidas

Causa	A rotina ainda não foi utilizada dentro da aplicação
Consequência	O método não será executado (o Java irá gerar uma exception de ausência de classe e/ou de rotina)
Solução	Obter o código convertido e acrescentar ao projeto

4.6 Acionamento de classes/objetos V2 não convertidos

Causa	A classe ainda não foi utilizada dentro da aplicação
Consequência	O método não será executado (o Java irá gerar uma exception de ausência de classe e/ou de rotina) ou poderão ocorrer erros de compilação da classe Java
Solução	Obter o código convertido e acrescentar ao projeto (no momento as classes persistentes da V2 ainda não estão sendo convertidas, neste caso verifique a possibilidade de comentar a chamada e ajustar futuramente, caso não seja viável, interrompa o processo e passe para outra funcionalidade convertida)

4.7 Acionamento de classes/objetos da biblioteca inexistentes

Causa	A classe ainda não foi utilizada dentro da aplicação
Consequência	O método não será executado (o Java irá gerar uma exception de ausência de classe e/ou de rotina) ou poderão ocorrer erros de compilação da classe Java
Solução	Verificar se este acionamento é realmente necessário, caso seja fundamental para o funcionamento informar a necessidade da

	classe/objeto ao suporte técnico

4.8 Referência de variáveis não inicializadas

Causa	A variável está sendo referenciada mas ainda não foi inicializada
Consequência	O comando não será executado (o Java irá gerar uma exception de variável indefinida)
Solução	Identificar no sistema o motivo desta variável não ter sido inicializada (caso necessário, utilizar a depuração no Caché para verificar o valor que esta variável deveria possuir e localizar o ponto em que é inicializada)

4.9 Funções não suportadas

Causa	A função sendo acionada não é suportada pelo framework
Consequência	O comando não será executado (o Java irá gerar uma exception de função não suportada) ou poderão ocorrer erros de compilação da classe Java
Solução	Verificar a necessidade desta função e se esta poderia ser suportada dentro do ambiente Java, caso seja inevitável, acionar o suporte técnico (as funções $text, $stack e $view não serão suportadas, assim como, diversas opções da $zutil)

5 Pontos de Atenção

5.1 Parâmetro por referência com redeclaração da variável dentro da rotina

Exemplo	do x1(.a) x1(p1) new a
Causa	Vide exemplo acima
Consequência	O parâmetro não irá possuir o valor correto dentro da rotina (ou método) podendo dar erro de variável indefinida
Solução	Alterar o nome da variável dentro da rotina (ou método) (pode-se somente alterar o nome da variável para o contexto: mVar a = m$.var("_a");)

5.2 Parâmetro por referência com utilização como uma variável em subrotina

Exemplo	do x1(.a) x1(p1) do x2 x2() set v2=p1
Causa	Vide exemplo acima
Consequência	O parâmetro não irá possuir o valor correto dentro da rotina (ou método) podendo dar erro de variável indefinida
Solução	Forçar a inicialização de uma variável com o nome do parâmetro (m$.var("p1").set(p1.get());)

5.3 Indireção com chamada de funções (@)

Causa	Utilização de indireção com referência de funções para determinar o valor
Consequência	Geração do valor incorreto da indireção
Solução	Ajustar o comando para que não seja necessária a execução da função via indireção, transformar o comando em um xecute ou, caso nenhuma destas soluções seja possível, informar o problema ao suporte técnico.

6 Registro de Anotação

Qualquer alteração nos códigos convertidos deve possuir uma anotação correspondente que identifica o tipo de alteração realizada, o autor e a data. Este registro é importante para facilitar a comunicação entre os desenvolvedores, esclarecer questões quanto às soluções adotadas e possibilitar a atualização do código convertido preservando as customizações realizadas.

No início do método:

@<anotação>(date="dd/mm/aaaa", author="<nome>", description="<descrição>")

No código, acima do ponto alterado (caso necessário demarque o início e término do código afetado):

// <anotação>(date="dd/mm/aaaa", author="<nome>", description="<descrição>")

6.1 CUSTOM

Customização permanente para solução de problema, adequação do código convertido, otimização de desempenho ou outra implementação realizada em caráter definitivo.

6.2 REVIEW

Implementação paliativa para funcionamento da aplicação, o código foi desenvolvido em caráter temporário e necessita ser revisado futuramente.

6.3 TODO

Ponto pendente de implementação ou de adequação de alguma parte do código, o sistema pode ou não funcionar temporariamente sem esta implementação..

6.4 REMOVE

Implementação paliativa para teste ou contorno de problema da aplicação, deve obrigatoriamente ser removida (não revisada) no futuro.