

**Cartilha do Desenvolvedor Java**

**Projeto de Software**

**Alphalinc Upgrade**

**Outubro/2014**



***TASC (the alpha supply chain)***

# Índice

[1 Índice 2](#_Toc400017474)

[2 Objetivo 3](#_Toc400017475)

[3 Histórico de Revisões 3](#_Toc400017476)

[4 Pontos de Intervenção 4](#_Toc400017477)

[4.1 Referência de globais de índice (^...s) 4](#_Toc400017478)

[4.2 Comandos SQL embutido (&SQL) 4](#_Toc400017479)

[4.3 Comando SQL inválido em ResultSet 4](#_Toc400017480)

[4.4 Comando interpretado muito complexo (xecute) 5](#_Toc400017481)

[4.5 Acionamento de rotinas não convertidas 5](#_Toc400017482)

[4.6 Acionamento de classes/objetos V2 não convertidos 5](#_Toc400017483)

[4.7 Acionamento de classes/objetos da biblioteca inexistentes 5](#_Toc400017484)

[4.8 Referência de variáveis não inicializadas 6](#_Toc400017485)

[4.9 Funções não suportadas 6](#_Toc400017486)

[4.10 Parâmetro por referência com redeclaração da variável dentro da rotina 6](#_Toc400017487)

[4.11 Parâmetro por referência com utilização como uma variável em subrotina 7](#_Toc400017488)

[4.12 Indireção com chamada de funções (@) 7](#_Toc400017489)

[5 Registro de Anotação 8](#_Toc400017490)

[5.1 CUSTOM 8](#_Toc400017491)

[5.2 REVIEW 8](#_Toc400017492)

[5.3 TODO 8](#_Toc400017493)

[5.4 REMOVE 8](#_Toc400017494)

# 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.

# Histórico de Revisões

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

# Pontos de Intervenção

# 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) |

**(\*) Este ponto não gera mensagem de erro**

# 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 |

**(\*) Este ponto não gera mensagem de erro**

# 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). |

# 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. |

# 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 |

# 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) |

# 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 |

# 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) |

# 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) |

# 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”);) |

**(\*) Este ponto não gera mensagem de erro**

# 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());) |

**(\*) Este ponto não gera mensagem de erro**

# 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. |

**(\*) Este ponto não gera mensagem de erro**

# 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 (acima da declaração do método):

@<anotação>(date=”dd/mm/aaaa”, author=”<nome>”, description=”<descrição>”, issue=”<identificação da demanda/número JIRA>”)

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>”, issue=”<identificação da demanda/número JIRA>”)

# CUSTOM

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

# REVIEW

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

# TODO

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

# REMOVE

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