Rotina Java

Java é uma linguagem muito poderosa, robusta e possui excelentes ferramentas de desenvolvimento de software. Para definir ações do tipo rotina Java, você precisa ter os conhecimentos básicos sobre a plataforma Java e contar com um ambiente de desenvolvimento, como por exemplo, o Eclipse.

Uma ação do tipo Rotina Java, nada mais é que uma classe java que deve ser implementada e disponibilizada em um JAR (do inglês, Java Archive ou Arquivo java, comumente denominado ".jar"), em que, será inserido no Sankhya-Om através do Cadastro de Módulos Java. Esse cadastro foi criado para incluir classes personalizadas dentro do Sankhya-Om, tornando possível usar essas classes em conjunto com as do próprio sistema em pontos específicos. Na funcionalidade aqui discutida, cada ação será representada por uma classe e um arquivo .jar, que pode conter várias classes, assim, podemos ter dezenas de ações em um único JAR. Além disso, cada módulo poderá possuir vários JARs, possibilitando, por exemplo, o uso de bibliotecas/utilitários de terceiros.

A seguir, exibiremos como as classes escritas em Java estarão no Sankhya-Om.

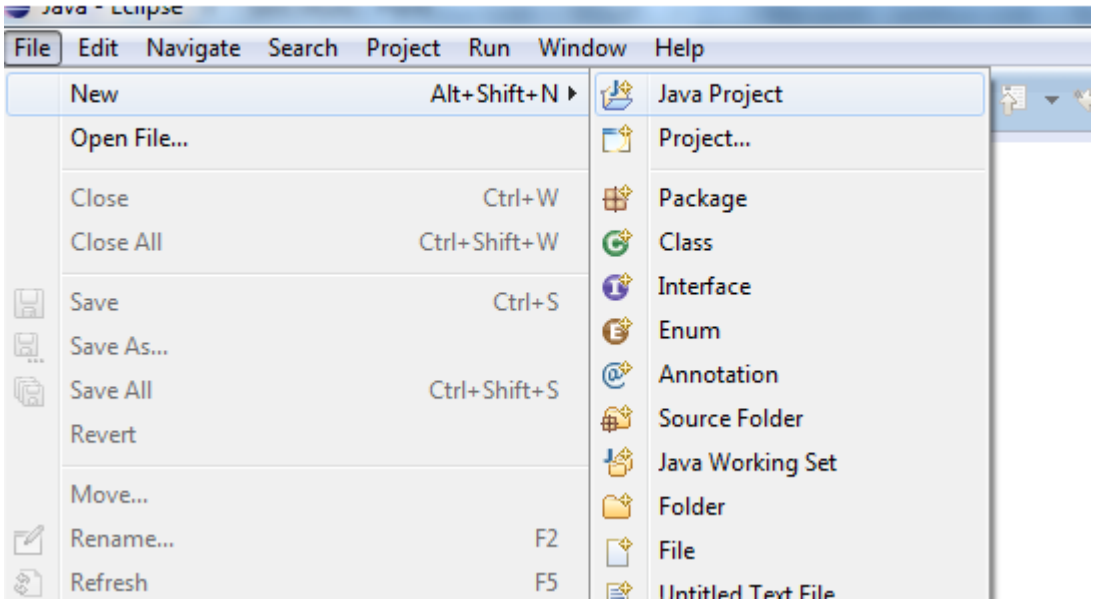
Uma classe só é considerada uma ação se implementar a interface AcaoRotinaJava. Essa interface é bastante simples e só exige que o método doAction seja implementado. Esse é o método que será executado quando você disparar uma ação.

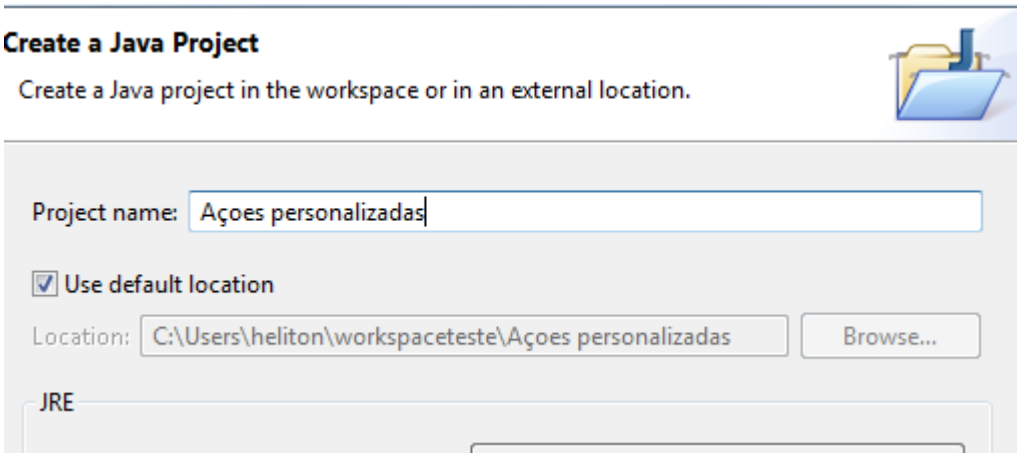
Vamos refazer em Java, o exemplo **"Gerar financeiro"** aplicado à ação JavaScript:

> ### Observação
>
> Aqui, trataremos apenas do Eclipse, mas você pode utilizar Netbeans ou qualquer outra IDE. Dependendo da versão do Eclipse, as telas podem ter aparência ligeiramente diferente, porém, essa mudança não afetará as ações a serem realizadas.

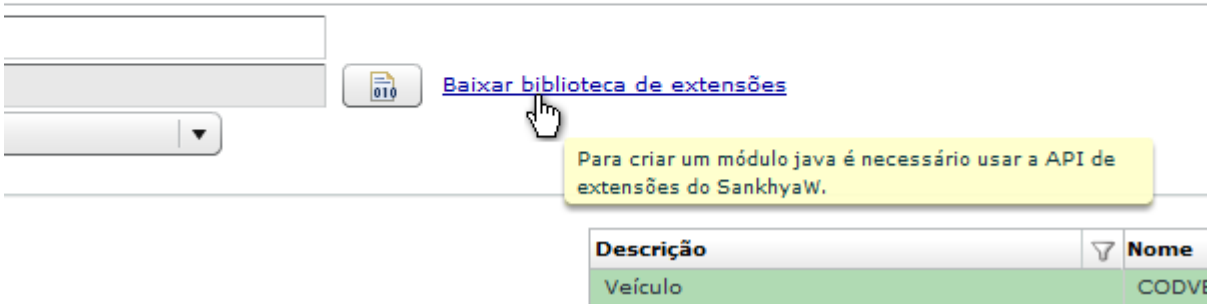
Obtenha uma versão do Eclipse pelo site http://www.eclipse.org/downloads, sendo que, você deve optar pela versão "Eclipse IDE for Java EE Developers", pois caso precise "debugar" sua ação, essa versão permite rodar um servidor de aplicações diretamente dentro do Eclipse. Se ainda não estiver familiarizado com a ferramenta, existe uma extensa documentação, incluindo tutoriais no mesmo site disponibilizado acima.

Passo 2 – Crie um projeto Java através de “File->New->Java Project”. No passo seguinte informe um nome para o projeto e clique em finish.





Em seguida, baixe a biblioteca de extensão do Sankhya-Om, clicando no botão **"Baixar biblioteca de extensões"** e salve o arquivo em um local conhecido, por exemplo: **"C:\Bibliotecas Java"**.

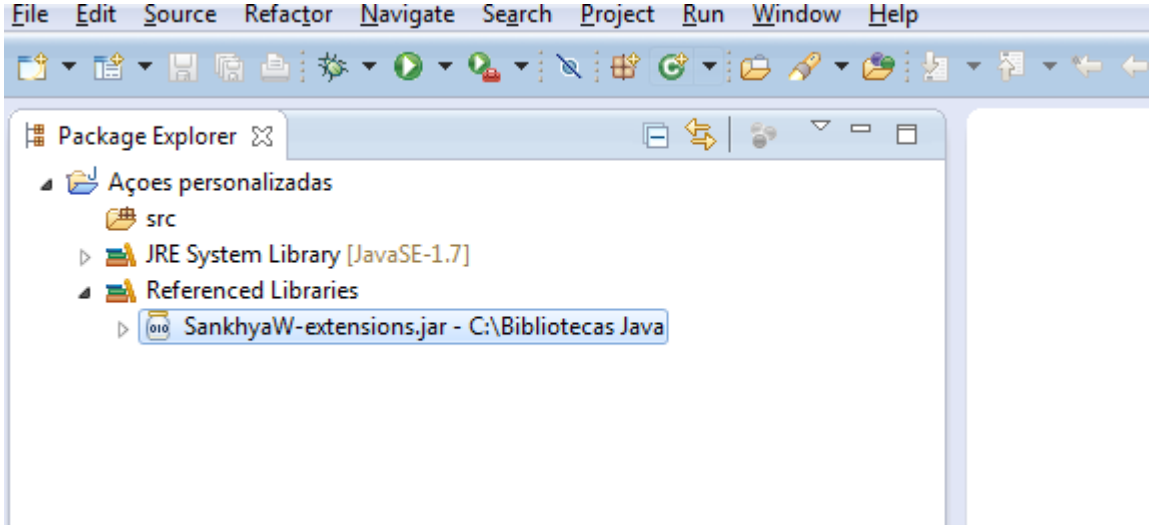


Nota: Este botão só fica disponível quando o tipo de ação escolhido for **"Rotina Java"**.

Adicione a biblioteca ao projeto:

Clique com o botão direito sobre o projeto e escolha a opção **"Build Path"** e em seguida selecione a opção **"Add External Archives"**.

Selecione o arquivo que acabamos de baixar, e finalize. O resultado deve ser algo parecido com:



Crie a classe que irá representar a ação da seguinte maneira:

Clique no menu "File > New > Class".

Defina o pacote em que a classe deverá ser criada.

> **! Atenção**
>
> você deve atentar-se ao último ponto, pois evita conflitos entre a sua classe e uma outra classe que coincidentemente tenha o mesmo nome. Existe uma convenção para a definição do pacote que inverte os fragmentos do domínio, sendo ele o endereço pelo qual acessamos as páginas na Web. Por exemplo, classes da Sankhya sempre estarão abaixo do pacote br.com.sankhya que é justamente o domínio da Sankhya invertido.

- Informe o nome da classe, pode ser algo como AcaoGerarFinanceiro.
- Clique no botão **Add**, do lado do campo Interfaces.
- Em **Choose interfaces** defina a interface AcaoRotinaJava para ser implementada e clique em **OK**.
- Clique em **Finish** e deve obter um resultado semelhante a:

AcaoGerarFinanceiro.java

```
1  package br.com.sankhya;
2
3  import br.com.sankhya.extensions.actionbutton.AcaoRotinaJava;
5
6  public class AcaoGerarFinanceiro implements AcaoRotinaJava {
7
8      @Override
9      public void doAction(ContextoAcao arg0) throws Exception {
10         // TODO Auto-generated method stub
11
12     }
13
14 }
15
```

Observe que o método doAction recebe um parâmetro do tipo ContextoAcao. Esse objeto é responsável por todas as funcionalidades que descrevemos anteriormente, como obter o valor de um parâmetro, agendar o envio de um email, criar uma nova linha, etc.

Em seguida, mude a implementação do método doAction assim:

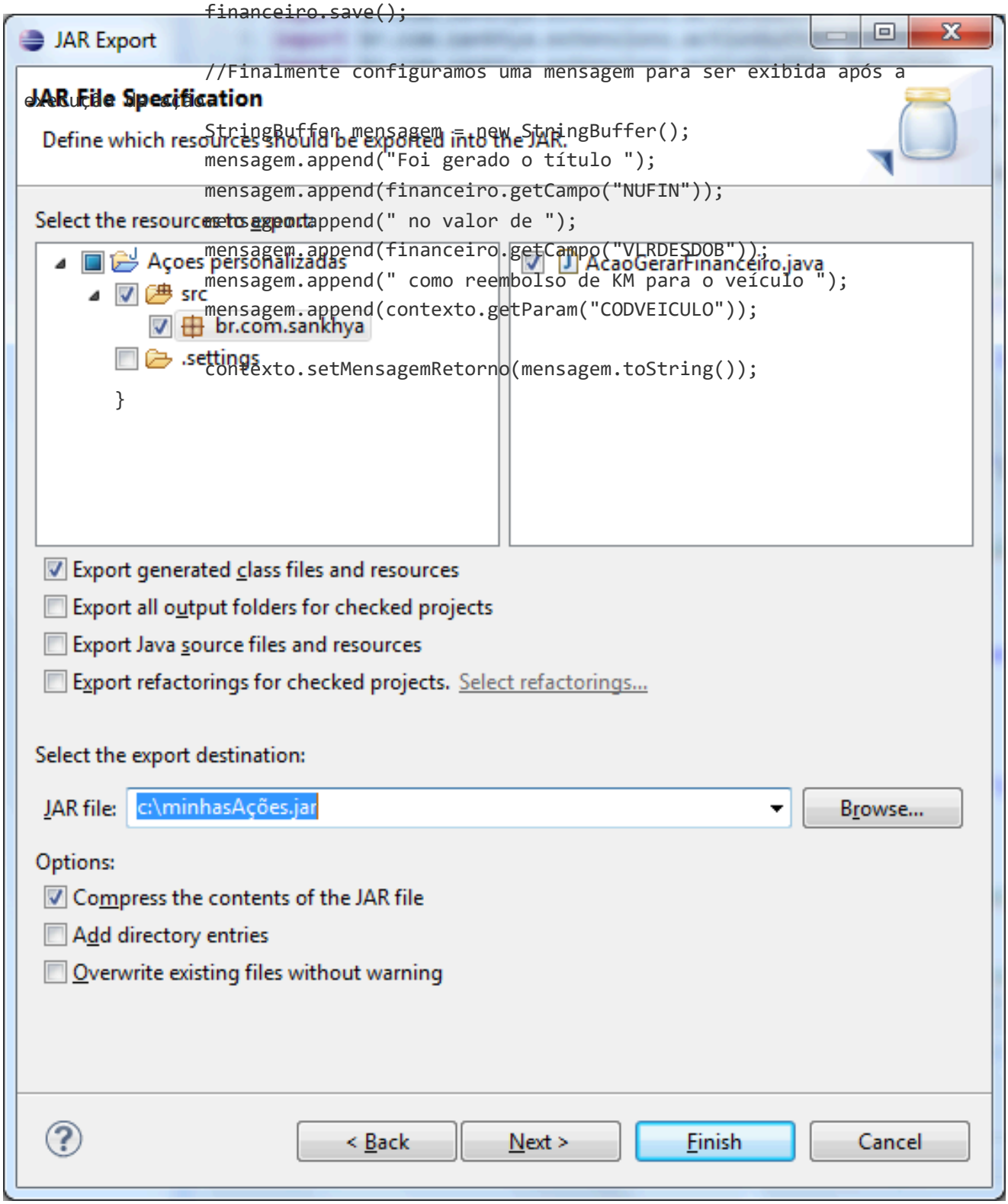
Java

```
public void doAction(ContextoAcao contexto) throws Exception {
                //Obtemos uma consulta para buscar os lançamentos
                QueryExecutor query = contexto.getQuery();

                //preparamos a execução da query, incluindo o parâmetro CODVEICULO.
                query.setParam("CODVEICULO", contexto.getParam("CODVEICULO"));

                query.nativeSelect("SELECT * FROM AD_TADCKM WHERE CODVEICULO =
{CODVEICULO}");

                double vlrDesdob = 0;
                while(query.next()){
                        double reembolso = query.getDouble("REEMBOLSO");

                        //Só permitimos gerar o título quando todos
                        //os lançamentos estiverem com reembolso calculado.
                        if(reembolso &gt; 0){
                                vlrDesdob += reembolso;
                        } else {
                                contexto.mostraErro("O reembolso do lançamento " +
query.getInt("SEQUENCIA") + " não foi calculado ainda.");
                        }
                }

                if(vlrDesdob == 0){
                    contexto.confirmar("Valor do título zerado", "O veículo
informado não possui lançamentos para reembolso, o título terá valor de
desdobramento igual a zero. Deseja continuar?", 1);
                }

                //por questões de desempenho é aconselhavel
                //fechar a consulta sempre que ela não for mais necessária.
                query.close();

                //Solicitamos a inclusão de uma linha no financeiro
                Registro financeiro = contexto.novaLinha("TGFFIN");

                //Informamos os campos desejados para incluir o financeiro
                financeiro.setCampo("VLRDESDOB", vlrDesdob);

                financeiro.setCampo("RECDESP", -1);
                financeiro.setCampo("CODEMP", 11);
                financeiro.setCampo("NUMNOTA", 0);
                financeiro.setCampo("DTNEG", "04/10/2012");
                financeiro.setCampo("CODPARC", 0);
```

```
financeiro.setCampo("CODNAT", 3050200);
financeiro.setCampo("CODBCO", 0);
financeiro.setCampo("CODTIPTIT", 2);
financeiro.setCampo("DTVENC", "04/10/2012");
financeiro.setCampo("HISTORICO", "REEMBOLSO DE KM PARA O VEÍCULO " +
contexto.getParam("CODVEICULO"));

//Quando o "save" do registro é acionado,
//a alteração é feita no Banco de dados.
//Portanto aqui estamos incluindo um registro na TGFFIN.
financeiro.save();

//Finalmente configuramos uma mensagem para ser exibida após a
execução da ação.
StringBuffer mensagem = new StringBuffer();
mensagem.append("Foi gerado o título ");
mensagem.append(financeiro.getCampo("NUFIN"));
mensagem.append(" no valor de ");
mensagem.append(financeiro.getCampo("VLRDESDOB"));
mensagem.append(" como reembolso de KM para o veículo ");
mensagem.append(contexto.getParam("CODVEICULO"));

contexto.setMensagemRetorno(mensagem.toString());
}
```

Para o próximo passo, exporte o projeto para um arquivo JAR:

- Clique em "**File** > **Export**".
- No diálogo "**Destino**" selecione a opção "**Java** > **JAR file**" na árvore e clique em "**Next**".
- No passo seguinte, selecione a sua classe como conteúdo do arquivo, defina o nome do arquivo e clique em Finish (os passos seguintes podem ser ignorados).



Em seguida, acesse a tela Configurações > Avançado > Módulo Java e crie um módulo.

> **Importante**
>
> Para evitar conflitos entre os módulos, fique muito atento ao campo Identificador, pois o objetivo dele é identificar unicamente o seu módulo para efeito de exportação/Importação de módulos. Esse campo será usado como identificador de recursos. É extremamente aconselhável usar o mesmo padrão de domínio reverso mencionado para a criação de pacotes.

Na aba **"Arquivo Módulo (Jar)"**, Ao clicar no botão + **"Cadastrar Arquivo Módulo (Jar)[F8]"**, adicione o JAR que você criou.

Na aba de Ações, altere a ação **"Gerar financeiro"**selecionando no campo **"Tipo"**, a opção **"Rotina Java"**, selecione o módulo e clique no botão classe_java.jpg **"Seleção da classe que será executada nesta ação"** ao lado do campo **"Classe Java"**, e escolha a ação que criamos.

Como tirar dúvidas?

Para tirar dúvidas e compartilhar informações, use a sala Botões de Ação da comunidade Sankhya Developer.

Updated 1 day ago

Próxima página

Rotina Lançador →

Rotina Banco de dados →

Rotina Javascript →