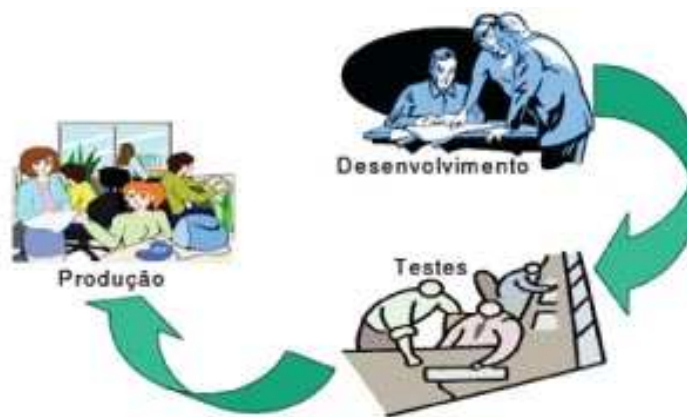


AMBIENTE DE DESENVOLVIMENTO

Qual é o ambiente de desenvolvimento disponibilizado para o FM? Qual é o editor? Temos CVS? Perdi o meu código, como recuperá-lo?



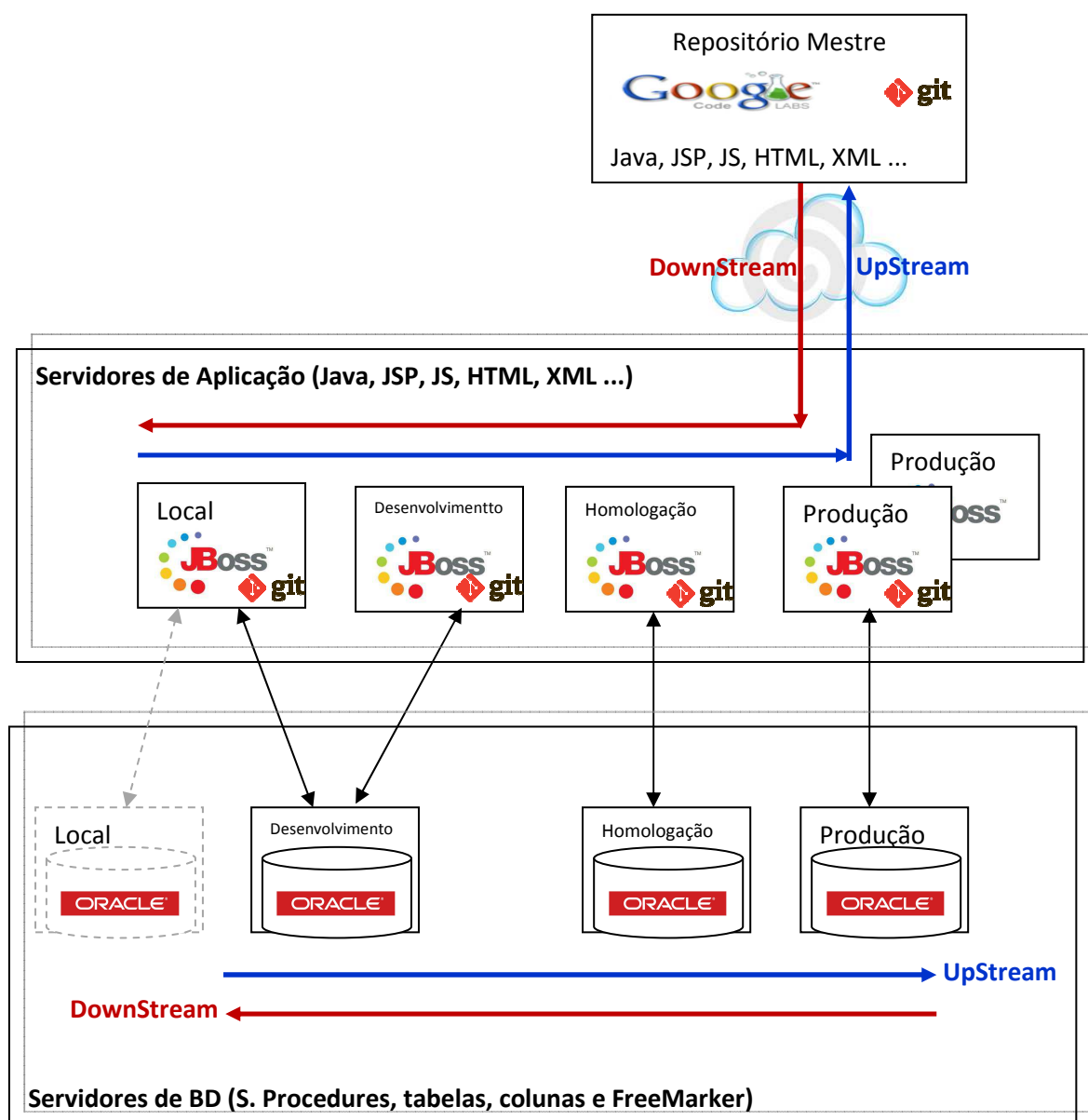
12.1 - O Ambiente de Desenvolvimento

Ambiente clássico:

- Local (Work Directory)
 - Desenvolvimento
 - Homologação e/ou Treinamento
 - Produção
- } Específica para cada programador.
- } Compartilhada por todos programadores.

O ambiente local é composto por uma IDE que possui, além de outros, de um editor de código que verifica a sintaxe em tempo de codificação (syntax highlighting), fornece dicas para comandos, fornece uma ferramenta para depurar (debug) o código e está integrado a uma ferramenta de controle de versão de código (CVS).

Visão geral para atualização dos servidores de aplicação e BD



AS IDEs atendem as atualizações nos servidores de aplicação, porém nos servidores de BD, temos que realizar tudo manualmente.

Existem duas correntes de atualizações:

- **Upstream:** deveria ser sempre a única e principal abordagem. Ela começa na máquina local do desenvolvedor e vai subindo (escalando) até chegar a produção e ao Repositório Mestre. Esta abordagem é sempre utilizada no desenvolvimento de produtos novos e melhorias, porém nem sempre respeitada nas correções, principalmente nas urgentes, visto que muitos programadores preferem atacar o problema direto na produção e, depois(????????), realizar o downstream.
- **Downstream:** é a mão oposta do upstream, como explicado anteriormente. É utilizada no caso de correções urgentes aplicadas diretamente no ambiente de produção (que devem ser realizadas nos ambientes inferiores) e no caso de se perder o controle dos códigos do ambiente de homologação e desenvolvimento, sendo que neste último caso faz-se um downstream forçado da produção para estes ambientes, e às vezes, criando alguns problemas se não tivermos um ambiente local (work directory).

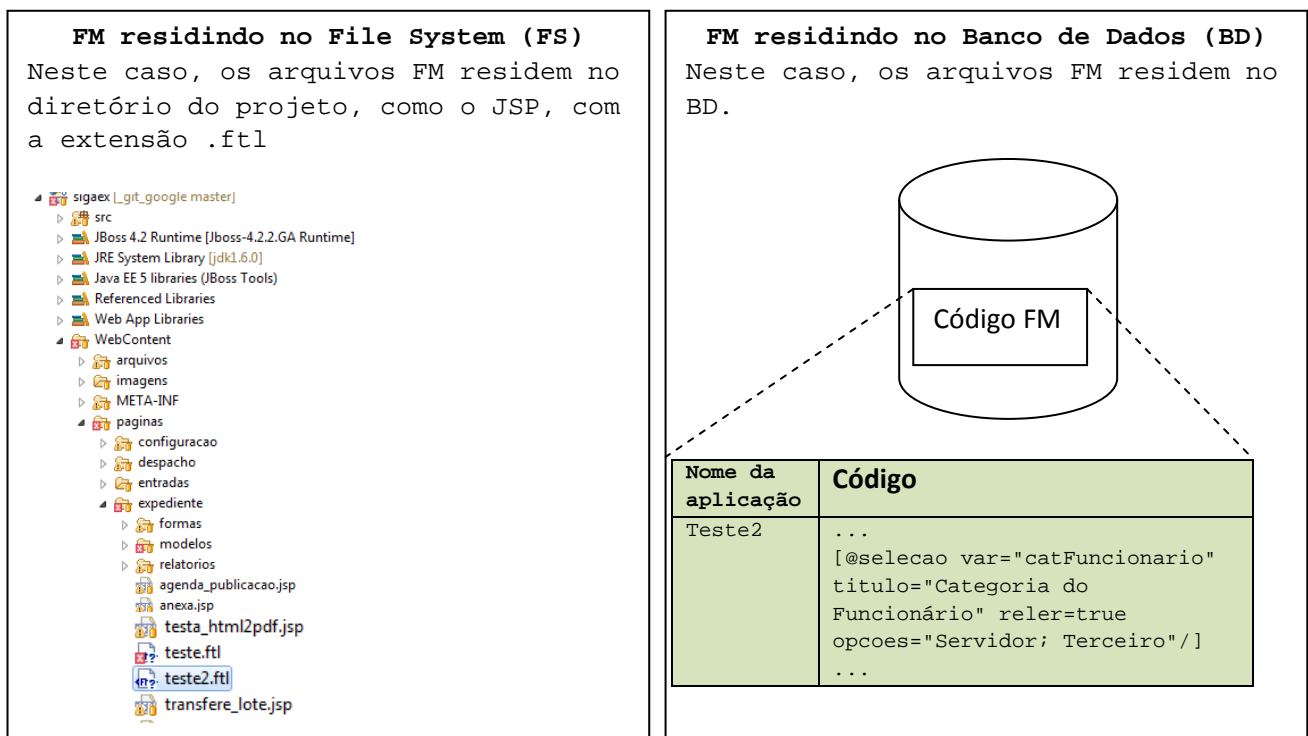
Para o código do SIGA-DOC utilizamos a IDE JBoss Developer Studio e o GIT como ferramenta de controle de versão, sendo o repositório do Google Code o repositório Master. Porém, só abordaremos a atualização do código FM neste manual e o FM reside no BD.

12.2 - O Ambiente de Desenvolvimento do FM

Para o FM NÃO temos uma IDE e nem uma ferramenta de controle de versão!!! Por quê?

É o problema do tradeoff, Ônus e Bônus, da nossa escolha.

Os dois modos de trabalho do FM. A nossa escolha recaiu sobre o BD.



TRADEOFFS		
Local onde reside o FM	Vantagens	Desvantagens
No FS		Deployment, que segue o

	Pode ser assistida por uma IDE, CVS e debug.	mesmo modus operandi de um JSP, por exemplo. Lento para realizar pequenas correções e para testar o código.
NO BD	Deployment ZERO. Uma correção é colocada no ar instantaneamente. Isto também facilita o desenvolvimento, visto que podemos testar em tempo real.	Sem IDE, sem CVS e sem Debug. Os controles são todos manuais. Atualização de código sujeito a erros. Dificuldade em depurar código.

É importante deixar claro que em um futuro próximo poderemos ter IDE, CVS e Debug para FM residindo em BD. Enquanto este tempo não chegar, podemos utilizar o Integrador V1.0 que veremos na seção 12.6.

12.2.1 - Ambientes da SJRJ

Ambientes	Endereço da aplicação SIGA no Servidor de Aplicação	Qual BD é apontado automaticamente pelo servidor de aplicação
Desenvolvimento no ambiente local * A princípio, o desenvolvedor não utilizará o ambiente local para o FM	http://localhost:8080/siga	Teste (ou Desenvolvimento) * Caso o desenvolvedor tenha o BD instalado na sua máquina ele pode apontar o servidor de aplicação para o BD Local
Desenvolvimento no ambiente de testes	http://versailles:8080/siga	Teste (ou Desenvolvimento)
Homologação	http://sigat.jfrj.jus.br	Homologação (ou Treinamento)
Produção	http://siga.jfrj.jus.br	Produção

12.2.2 - Procedimentos para criar e manter uma aplicação FM

Criando uma nova aplicação FM:

- Desenvolver no ambiente local, se for o caso;
- Desenvolver (ou migrar o código do ambiente local) no ambiente de Desenvolvimento (também conhecido como ambiente de testes);
- Migrar o código do ambiente de Desenvolvimento para o ambiente de Homologação (também conhecido como ambiente de Treinamento). Solicitar que o usuário homologue a aplicação. Se for ok, passar para a atividade seguinte, caso contrário, retornar a atividade anterior;
- Migrar o código do ambiente de Homologação para o ambiente de Produção.

Realizando manutenção em uma aplicação FM existente:

- Obter o código mais atual da aplicação que sofrerá manutenção. Teoricamente, os ambientes de Desenvolvimento, Homologação e Produção DEVEM possuir o mesmo código, porém não há nada que garanta isto. Nesta situação recomenda-se obter o código do ambiente de Produção;
- Se a manutenção é urgente deve-se seguir o downstream como mencionado em 12.1, ou seja, atualizar o código no ambiente de Produção e replicá-lo para os ambientes de Homologação e Desenvolvimento;
- Se a manutenção não for urgente, seguir o upstream (sempre preferido)
 - o Atualizar o código no ambiente de Desenvolvimento;

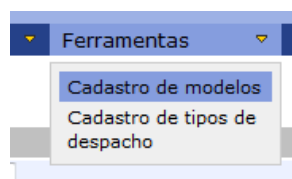
- o Migrar o código do ambiente de Desenvolvimento para o ambiente de Homologação. Solicitar que o usuário homologue a aplicação. Se for ok, passar para a atividade seguinte, caso contrário, retornar a atividade anterior;
- o Migrar o código do ambiente de Homologação para o ambiente de Produção.

12.3 - Criando uma aplicação FM

Como o FM não deixa de ser um documento, utilizamos o próprio ambiente do SIGA-DOC para criá-lo e armazená-lo.

12.3.1 - Criando uma aplicação FM

No SIGA-DOC, acessar **Ferramentas / Cadastro de Modelos / Novo**.



Edição de Modelo

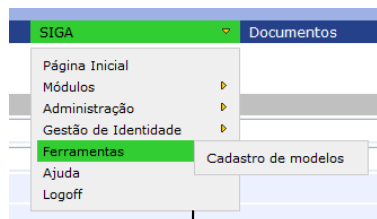
Modelo	
Nome:	Teste para manual
Descrição:	Teste para manual
Classificação:	<input type="text"/> ...
Classificação para criação de vias:	<input type="text"/> ...
Forma:	Anexo ▼
Nível de acesso:	Público ▼
Tipo do Modelo:	Freemarker ▼ <pre> 4 Este é o editor 5 6 [@grupo] 7 [@texto titulo="Ano Posse/Contratação" var="ano" largura=""] 8 [/@grupo] 9 [@grupo depende="anoAjax"] 10 [#if (ano!="")] 11 [mensagem titulo="Alerta" texto="Ano deve ser preenchido" 12 vermelho=true/] 13 [/#if] 14 [/@grupo] </pre>
<input type="button" value="Ok"/> <input type="button" value="Aplicar"/> <input type="button" value="Cancelar"/>	

Editor FM

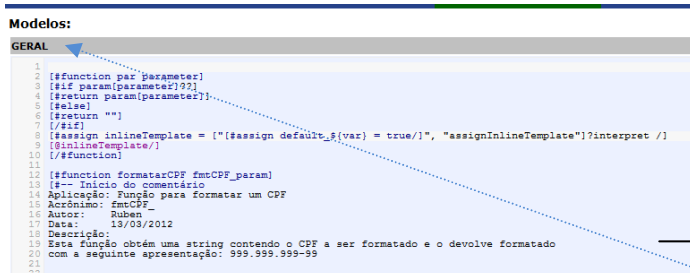
Observação: No desenvolvimento, geralmente escolhemos **"Anexo"** como Forma, apenas por convenção, desta forma, todas aplicações sendo desenvolvidas residem neste tipo de documento, até serem migrados para o tipo correto.

12.3.2 - Criando uma macro FM

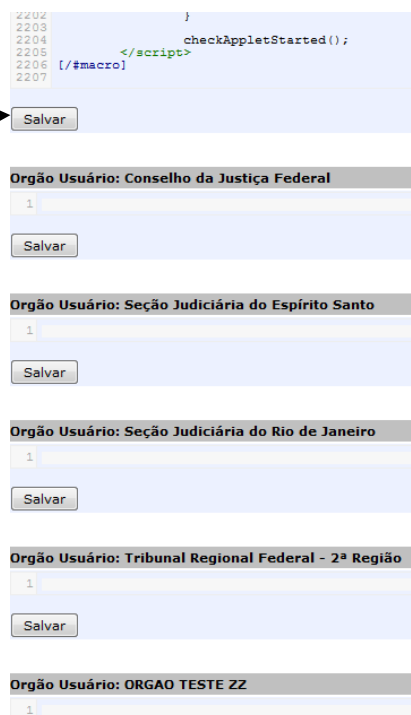
A - No SIGA-DOC, acessar **SIGA / Ferramentas / Cadastro de Modelos / Novo**



B - Escrever o código da macro



C - Salvar



É importante observar no arquivo de macros, a seção correspondente a sua instituição. A primeira seção é para todos, chamada de Geral, e é a que os desenvolvedores na SJRJ utilizam.

12.4 - Backup, Restore e comparando códigos FM

A forma de se levar um produto (código) seguindo a corrente upstream, do desenvolvimento para homologação e da homologação para produção é através de Copy e Paste, em cada ambiente do SIGA instalado.



É prática comum do desenvolvedores da infraestrutura do SIGA-DOC realizar o downstream forçado, da produção para a homologação e desenvolvimento, porque como dito, perde-se o controle das atualizações que são realizadas diretamente na produção. Como não temos um BD local (ver diagrama em 12.1 ... e até poderíamos ter, porém o Oracle pesa muito a máquina), **todos os nossos desenvolvimentos são perdidos**. Para minimizar este problema, e por questões de segurança, podemos realizar um **backup** de todas as

nossas aplicações e das macros. No fundo da tela, próximo ao botão |NOVO|, temos o botão |Exportar XML|, que gera um arquivo XML com todas as aplicações ou macros.

BACKUP

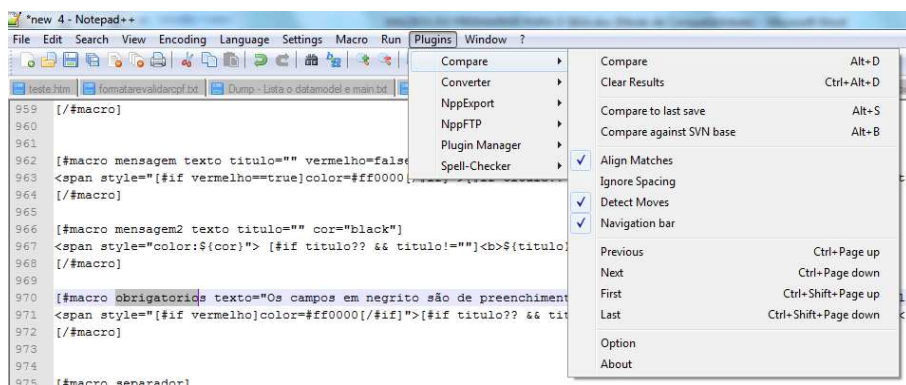
Exemplo, no caso das aplicações:

TRF-PRES Portaria da Presidência	TRF-PRES-PORTARIA Licença Saúde para Desembargador
TRF-PRES Resolução da Presidência	TRF-PRES Resolução da Presidência
TRF-Proposta e Concessão de Diárias	TRF-Proposta e Concessão de Diárias
Termo	SGP: Recebimento da Segunda Via de Crachá
Termo	SGP: Recebimento de Crachá
Termo	Termo - Modelo Livre

Novo Exportar XML



Podemos também utilizar uma ferramenta de comparação de códigos (Diff), para comparar o arquivo XML do desenvolvimento com o da produção, por exemplo. Pode-se utilizar a ferramenta free Notepad++ com o plugin Compare.

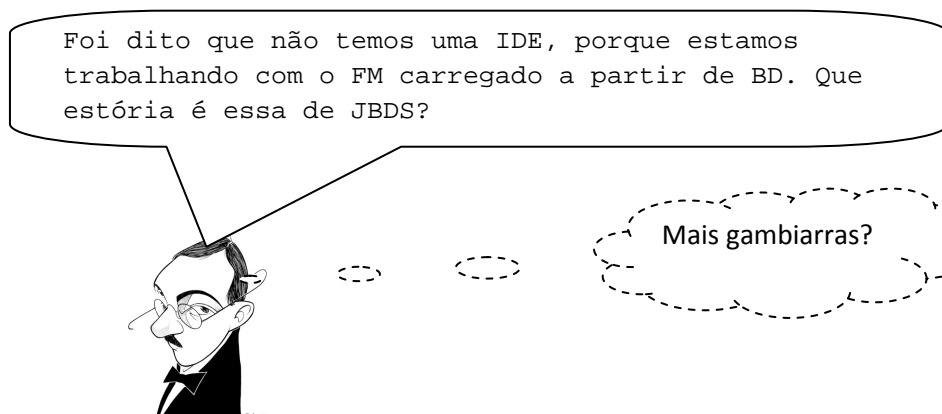


Notepad++ e o plugin Compare, permite que se compare os arquivos, por exemplo, desenv20062012.xml e producao20062012.xml. Ele abrirá 2 janelas e apresentará as diferenças, desta forma você pode tomar uma decisão.

RESTORE

Infelizmente não temos uma ferramenta, tipo importar. O restore tem que ser feito como Copy e Paste, a partir do arquivo XML. Pode-se copiar e colar todas as macros ao mesmo tempo, porém no caso das aplicações, temos que fazer uma a uma.

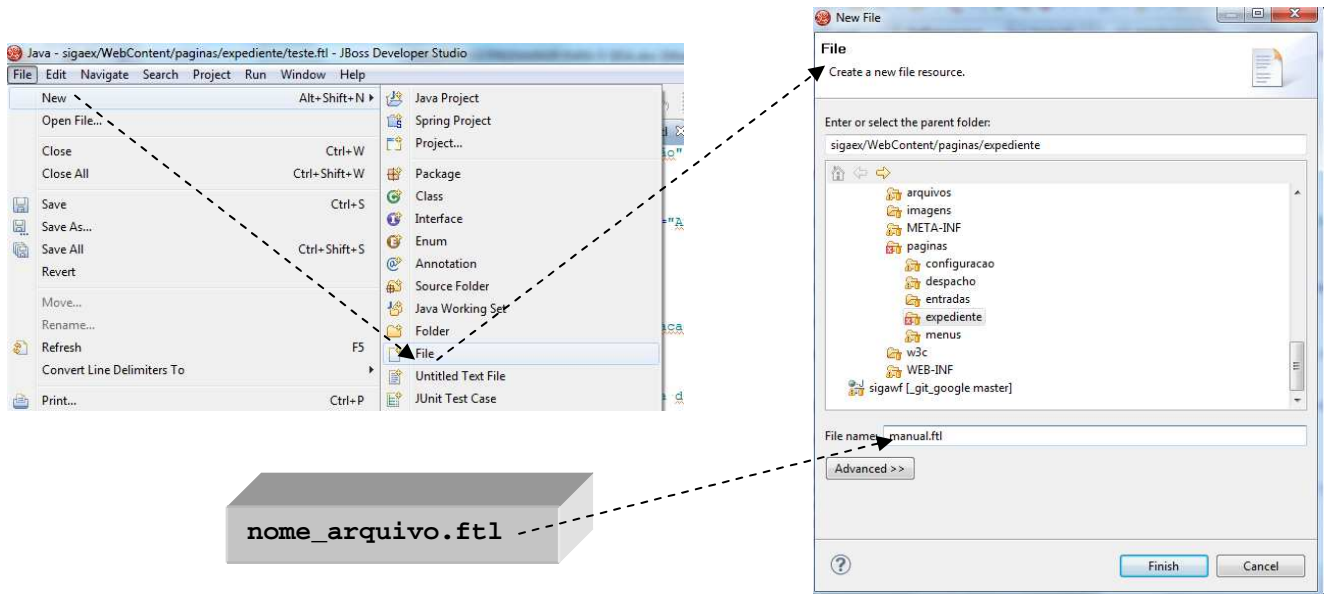
12.5 - Sintaxe HighLight para o FM no JBoss Developer Studio (JBDS)



Isto é verdade, porém o IDE JBDS trabalha com arquivos .ftl. Neste caso podemos escrever a aplicação neste ambiente, usufruir do seu editor com intuito de obtermos uma aplicação sintaticamente correta, e depois, copiá-la e colá-la no editor do SIGA-DOC. No futuro, tentaremos configurar o JBOSS local para que ele rode também a aplicação, enquanto isso, poderemos utilizar o Integrador que será visto na próxima seção.

Para obter detalhes de como instalar o plugin Freemarker no Eclipse visite o capítulo [Anexo 9 - Instalando o plugin do Freemarker no Eclipse](#)

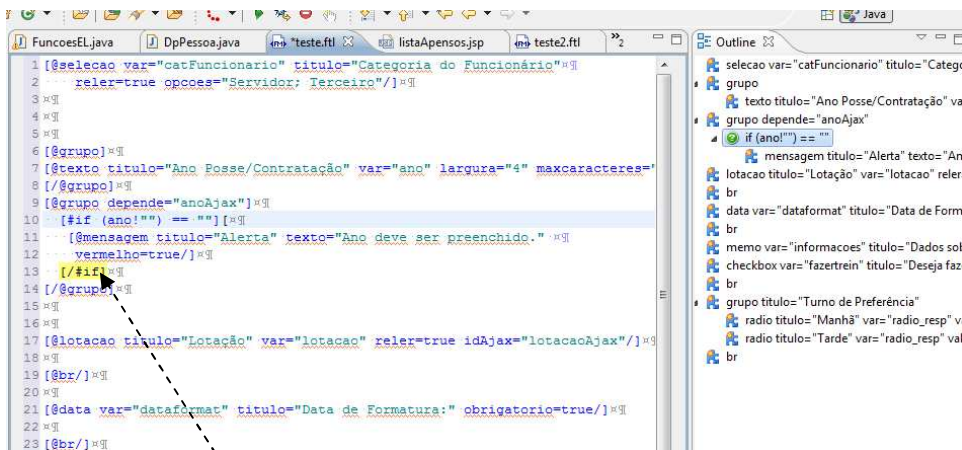
12.5.1 - Criando um arquivo .ftl




12.5.2 - Tela Principal do editor

Área do código

Outliner

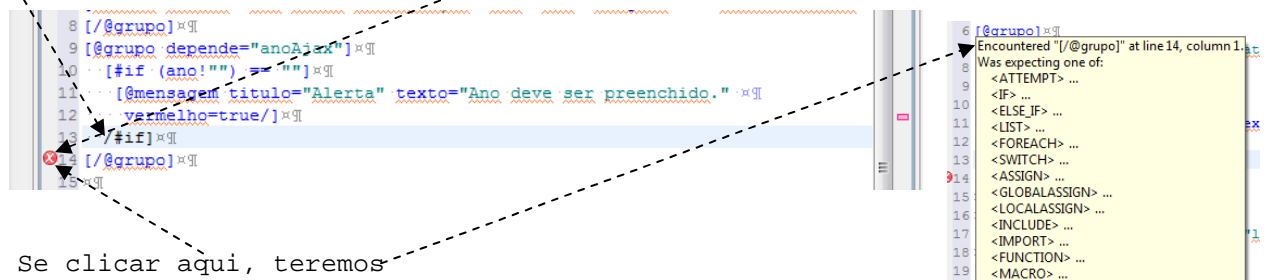


Clicando no `[/#IF]`, em amarelo, que é um IF de fechamento, ele passa para o IF de abertura e vice-versa, facilitando entender os blocos de código.

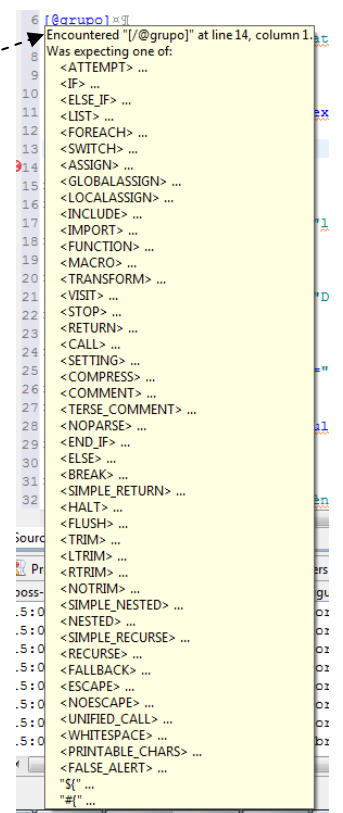
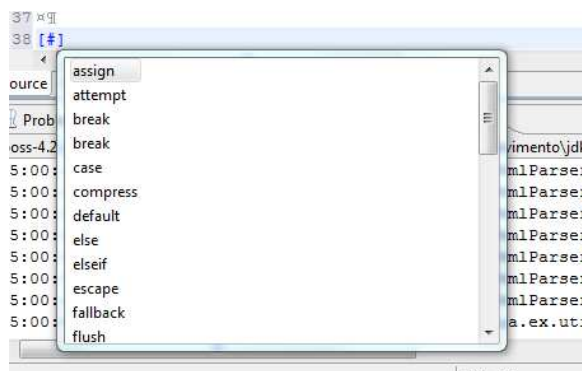
O outliner estrutura o código e para cada comando, IF, CASE etc., coloca um ícone específico, como o  para o IF.

12.5.3 - Acusando o erro

Observar o erro, falta o [, e o apontamento



12.5.4 - Sintaxe dos comandos



12.6.1 - Introdução

Os modelos freemarker na base de dados do SIGA-DOC promovem zero deployment, porém o desenvolvimento é prejudicado. Por outro lado, o desenvolvimento no JBDS é excelente, porém necessita de deployment. É aí que entra o Integrador, combinando o melhor dos dois mundos: desenvolvimento no JBDS com zero deployment.

O Integrador é um aplicativo (modelo do SIGA-DOC) desenvolvido com os seguintes objetivos em mente:

INTEGRADOR

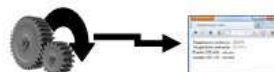
CARGA (Sem a necessidade de copy / paste)

A carga se dá a partir do JBOSS, do projeto/pasta `/sigaex/freemarker/`



- ❖ Por default, `template.ftl` é carregado para o editor. Caso não exista, o editor aparecerá vazio;
- ❖ Posteriormente, pode-se informar outro modelo a ser carregado;

EXECUÇÃO (On the fly. Sem a necessidade de cadastrar o modelo)



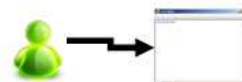
- ❖ O modelo pode ser executado de forma normal, como em produção;
- ❖ Porém, pode ser executado em modo DUMP. Neste caso, um dump do data-model e variáveis do template é realizado no início e outro no final da execução do modelo freemarker. Dois botões serão inseridos na aplicação para visualização dos dumps. Útil como ferramenta de debug.

REGISTRO NO SIGA-DOC (Sem sair do ambiente, sem copy / paste)



- ❖ O Modelo pode ser cadastrado (registrado) na base de desenvolvimento do SIGA-DOC diretamente.

EDIÇÃO (Sem sair do ambiente, sem copy / paste)

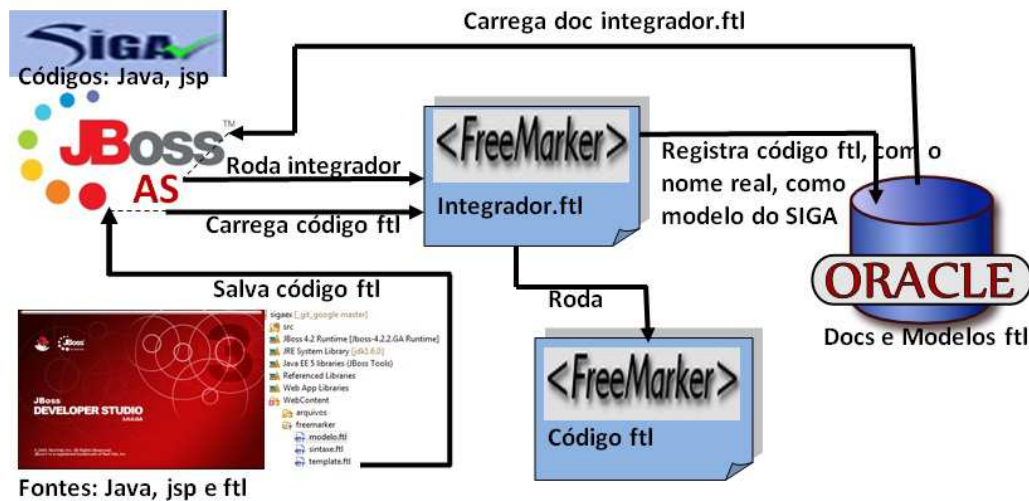


- ❖ Pode-se colar ou escrever um código ou comandos diretamente no editor. Ótimo para testar comandos, funções, pequenas aplicações e etc. Pode-se colocar *traps* na aplicação sem afetar o código original. Muito útil para debug.

Mas como funciona e por que estamos privilegiando o desenvolvimento no JBDS?

Pelos motivos expostos nas seções anteriores, o qual faço um resumo abaixo, tanto do funcionamento quanto das vantagens em utilizar o JBDS.

FREEMARKER NO JBOSS DEVELOPER STUDIO

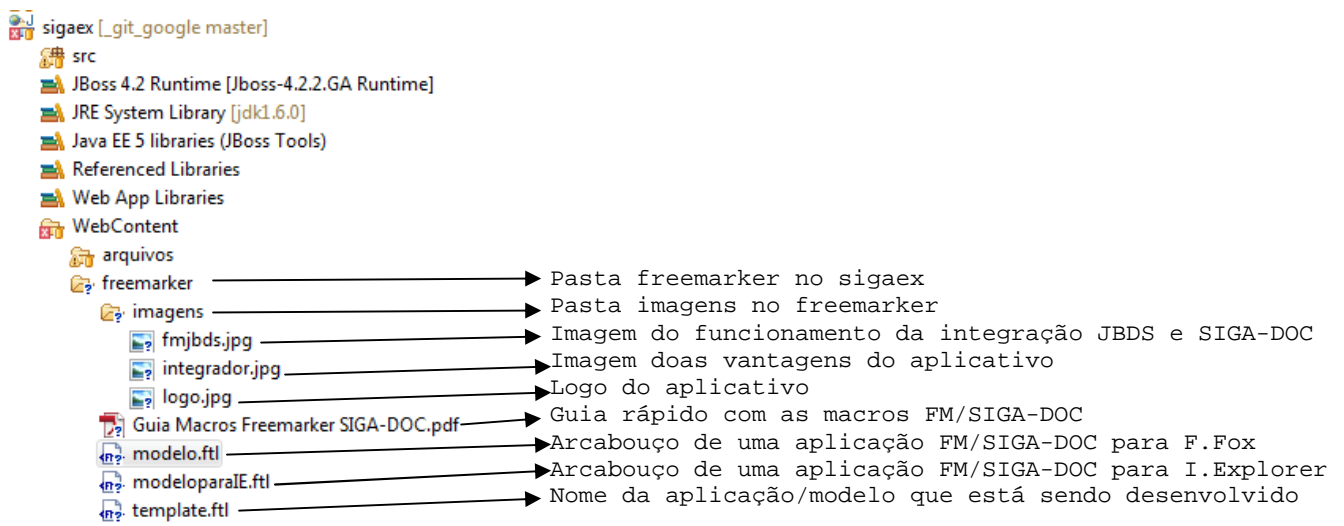


Vantagens

- Editor ftl no JBDS é infinitamente melhor;
- Os templates ficam em work área local. Nunca são substituídos;
- Controle de versão dos templates pelo Git;
- Desenvolvimento offline, sem timeout. Não depende do JBAS, SIGA-DOC (SDc) e Oracle;
- Ambiente voltado ao desenvolvimento de código, facilitando e acelerando o mesmo.

Mas só funciona com a IDE JBDS e o Servidor de Aplicação JBOSS AS? Embora não testado, acredito que funcione com qualquer IDE e com qualquer Servidor de Aplicação, visto que o acomplamento com estes produtos é muito fraco, ou melhor, inexistente.

12.6.2 - Setup do Ambiente



Explicação dos objetos:

Pasta freemarker: é necessário criá-la debaixo do projeto sigaxe.

Pasta imagens: é necessário criá-la debaixo da pasta freemarker. Conterá as imagens do aplicativo que serão carregadas quando o mesmo for iniciado.

Guia da macros feemarker / SIGA-DOC: um resumo do manual contendo apenas as macros, com intuito de servir de guia ao desenvolvimento. Aqui, o desenvolvedor obterá as respostas que necessita, exemplos e etc.

Modelo.ftl: um esqueleto (V0) de uma aplicação freemarker para o SIGA-DOC. Deve-se utilizar este modelo para o Fire Fox.

ModeloparaIE.ftl: idem ao anterior, porém para o Internet Explorer.

Template.ftl: este deve ser o nome do aplicativo que está sendo desenvolvido. Não é obrigatório utilizar este procedimento, porém o aplicativo Integrador já carrega este aplicativo na área de edição.

12.6.3 - Funcionalidades do Aplicativo

12.6.3.1. - Chamada do aplicativo

Tipo: Anexo

Tipo: IntegraJBDS-SIGADOC Versão 1.0.0.0

12.6.3.2 - Tela Inicial

INTEGRADOR V 1.0 JS Almost 100%

Os modelos freemarker na base de dados do SIGA-DOC promovem zero deployment, porém o desenvolvimento é prejudicado. Por outro lado, o desenvolvimento no JBDS é excelente, porém necessita de deployment. É aí que entra o Integrador, combinando o melhor dos dois mundos: desenvolvimento no JBDS com zero deployment.

CLIQUE "aqui" PARA OBTER MAIS INFORMAÇÕES SOBRE O APLICATIVO

OPÇÕES

Executar o código ☒ Modo Normal ☐ Modo Dump

Registrar (cadastrar) o código na base do SIGA-DOC

Carregar outro modelo, com o nome de .ftl

Editar Linhas: 30 Colunas: 190 Cor da Fonte: Azul Tamanho da Fonte: 10 Família da Fonte: Lucida Console Cor do Fundo: Amarelo Claro Estilo da Fonte: Negrito

```
[@entrevista]
[selecao var="catFuncionario" titulo="Categoria do Funcionário"
reler=true opcoes="Servidor; Terceiro"]

[grupo]
[texto titulo="Ano Posse/Contratação" var="ano" largura="4" maxcaracteres="4" obrigatorio="Sim" reler="ajax" idAjax="anoAjax" relertab="sim"]
[/grupo]
[grupo depende="anoAjax"]
[if (ano!="")]
[mensagem titulo="Alerta" texto="Ano deve ser preenchido."
vermelho=true]
[/if]
[/grupo]
```

ÁREA DE EDIÇÃO

É importante observar que, por default, o aplicativo já carrega o modelo chamado template.ftl do projeto/sigaex/freemarker na área de edição. Desta forma, é interessante que o aplicativo atual, que está sendo desenvolvido, possua este nome. Depois que for testado e carregado na base do SIGA-DOC, pode-se renomeá-lo com o nome real. Isto não é uma obrigação, visto que podemos após a inicialização do aplicativo indicar outro modelo a ser carregado.

12.6.3.3 - Ajuda

Ao clicar na imagem / logo do aplicativo (da tela inicial) será apresentado uma ajuda conforme apresentado abaixo.



Ao se clicar na imagem acima, voltamos para a tela inicial.

12.6.3.4 - Opção / Executar

Executar o código ☒ Modo Normal ☐ Modo Dump

Permite que se execute o código que está na área de edição. Pode-se rodar em modo:

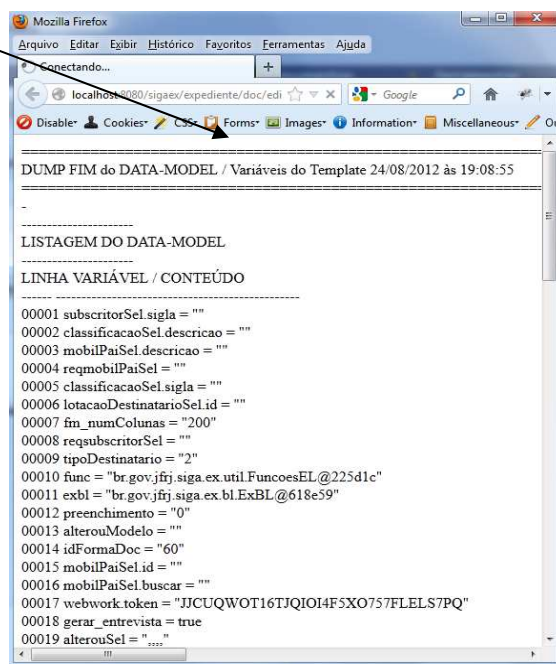
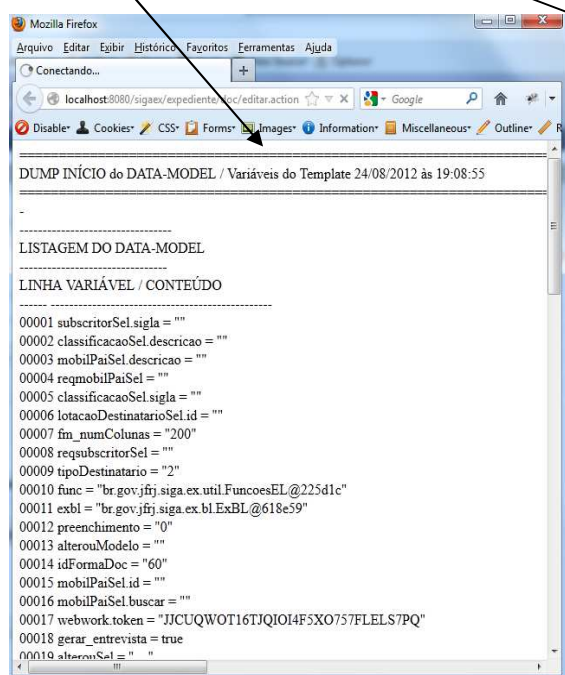
Normal: ou seja, rodá-lo como se em produção estivesse.;

Dump: neste caso, são inseridos dois comandos freemarker (duas chamadas de macros) no código, uma no início ([@dumpvarantes/]) e outra no final ([@dumpvardepois/]). Estas macros tirarão uma foto do data-model + variáveis do template e as armazenarão para posterior consulta na aplicação sendo executada. Será automaticamente inserido os seguintes botões na aplicação;

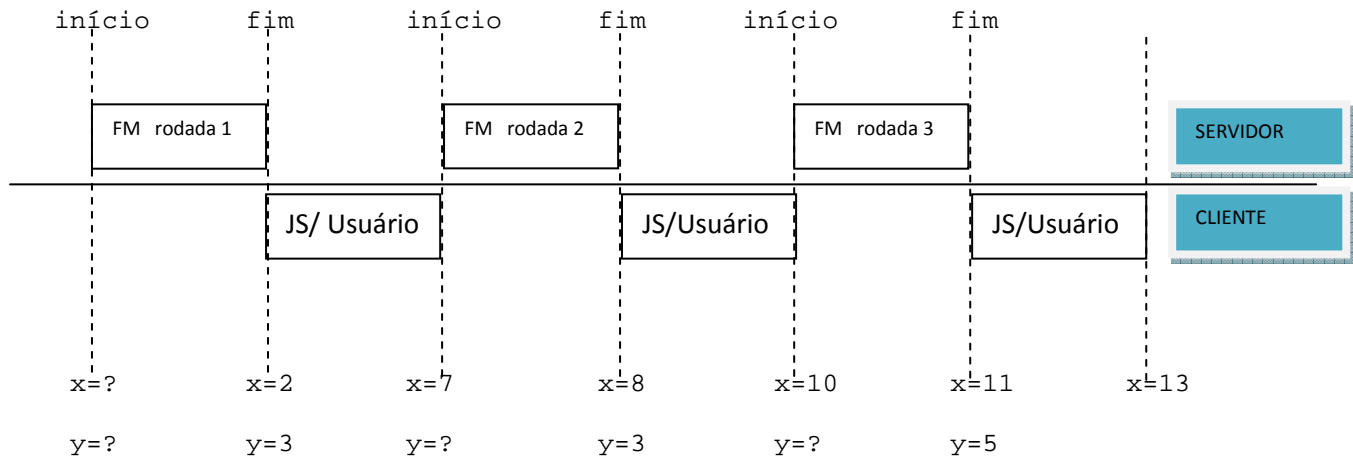
DUMP

Início da execução

Fim da execução



Para analisar o estado (conteúdo) das variáveis ao longo do processamento. Pode-se guardar os conteúdos entre cada rodada do FM no servidor, ou seja, a cada vez que a tela é enviada ou relida (reler=true), com intuito a identificar quem / o que está promovendo tais mudanças nas variáveis.

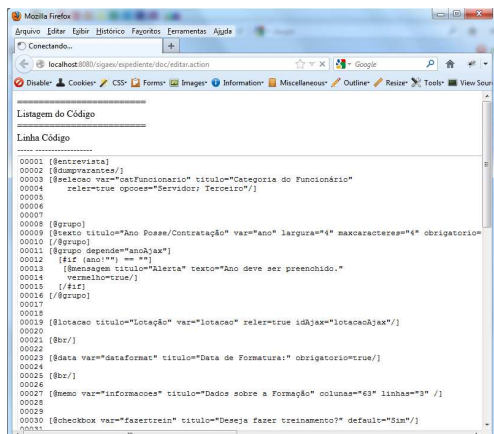


Pode-se observar a evolução da variável x .

- No início não existia, e depois o FM associou o valor 2 e enviou a tela;
- O usuário ou algum aplicativo JS alterou o valor para 7;
- O FM alterou o valor para 8 e enviou a tela;
- e assim por diante ...

As variáveis do data-model só podem ser lidas pelo FM, nunca alteradas. Quem altera o valor das variáveis do data-model é o aplicativo JAVA que executa o FM e carrega o template.

O problema é que o editor do Integrador é aberto, como veremos mais abaixo. Você pode desenvolver um código ad-hoc, copiar e colar um código qualquer ou alterar o código na área de edição, para inserir traps (tipo: passei por aqui, conteúdo de x é ...), por exemplo. Desta forma, seria perdida a referência da numeração, e caso ocorra um erro (o freemarker indica a linha que ocorreu) seria difícil identificar a linha que ocorreu o mesmo. Esta janela permite você identificar a linha.



12.6.3.5 - Opção / Registrar

Registrar (cadastrar) o código na base do SIGA-DOC

Permite que o código seja registrado (cadsatrado) no na base do SIGA-DOC. A seguinte tela será apresentada:

INTEGRADOR - Registrar (Cadastrar) o template na base do SIGA-DOC Desenvolvimento

Edição de Modelo

Modelo	
Nome:	<input type="text"/>
Descrição:	<input type="text"/>
Forma:	Anexo ▾
Nível de acesso:	Público ▾
Tipo do Modelo:	Freemarker ▾
Clique aqui para cadastrar o template	
<pre>[@entrevista] [@selecao var="catFuncionario" titulo="Categoria do Funcionário" reler=true opcoes="Servidor; Terceiro"/]</pre>	

O nome é o que aparecerá como Modelo e a forma como Tipo na tela Documento do SIGA-DOC

Destinatário:	Órgão Integrado ▾	...
Tipo:	Memorando ▾	
Modelo:	Memorando ▾	

Após a inclusão, será exibida a tela com a lista dos modelos.

12.6.3.6 - Opção / Carregar

Carregar outro modelo, com o nome de .ftl

Permite que se carregue outro modelo / template da localização /sigaex/freemarker. Lembre-se que o template.ftl é carregado como default na inicialização do Integrador. A partir daqui podemos carregar outro modelo para a área de edição.

12.6.3.7 - Opção / Editar

Editar

Linhas: Colunas: Cor da Fonte: Tamanho da Fonte: Família da Fonte: Cor do Fundo: Estilo da Fonte:

Por default, a área de edição é readonly. Clicando em Editar, abre-se a área de edição para alteração. Pode-se customizar o editor em relação a cor, tamanho da fonte e etc.

Cenário de utilização:

Você está desenvolvendo o template no JBDS e carrega-o no editor. Porém antes de rodar, você deseja adicionar traps, tipo: 'passei no Then do If do `matric=="1212"`', 'passei no case xyz', 'Valor da variável x =' `${x}` e etc. Ou seja, adicionar código de depuração, que só serve para a rodada. Qualquer alteração no código na área de edição não afeta o original no JBDS, deixando-o intacto.

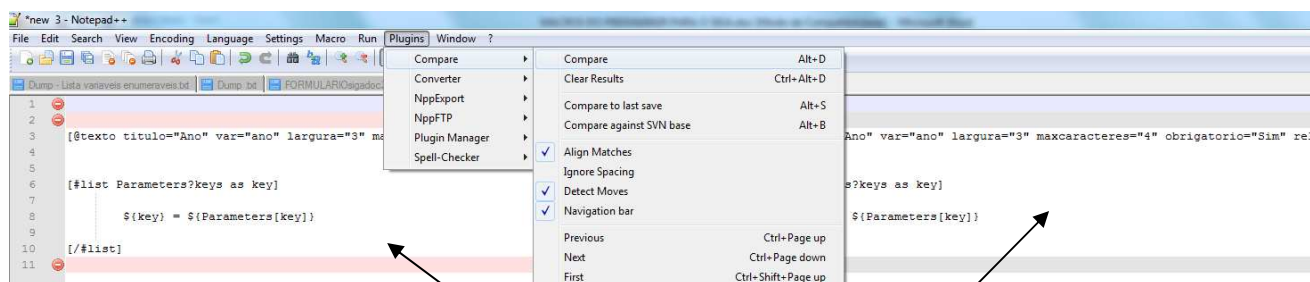


Você até pode registrar no SIGA-DOC o código que está na área de edição, mesmo que ele seja diferente do código no JBDS, porém não é uma boa prática. Deve-se sempre consertar os bugs no código original no JBDS e carregá-lo novamente no Integrador.



Será que o meu código no JBDS está igual ao registrado na base do SIGA-DOC?

Utilize o Notepad++, plugin Compare.



Códigos a serem comparados