

Bug! E agora?

Como fazer para rastrear o problema? Que ferramentas podem me auxiliar?



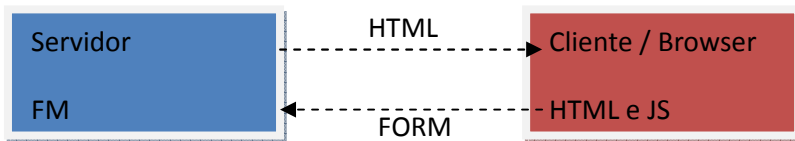
Inexoravelmente sua aplicação não rodará de plano.

Bom, isso pode verdade. Porém, ao ferranteas tem sido processo de



parecer pessimismo, mas é a pura longo dos anos diversas desenvolvidas para nos ajudar no depuração.

Como programadores de interface temos dois ambientes que devemos nos ater:



10.1 - Depuração no servidor

10.1.1- FM

Infelizmente, no FM não temos um ambiente de debug, como no JSP, o que torna o processo de resolução mais demorado, cansativo e até, frustrante.

A técnica, hoje, consiste de se colocar traps (do tipo: `${minhavar}` ou "passei neste IF") no programa para exibir o conteúdo das variáveis que desejamos investigar ou sabermos por onde estamos passando (trace). Muitas vezes desejamos exibir várias variáveis em diversos pontos, o que é um trabalho maçante. Para amenizar o problema, desenvolvi a macro `@dumpall`, cuja descrição segue abaixo.

10.1.1.1 - A MACRO `@dumpall` (sem parâmetro)

Esta macro lista as variáveis (objetos), os respectivos tipos e os conteúdos (o conteúdo somente se a variável for escalar: numérica, string, date e boolean). É uma ferramenta útil para fins de debug, visto que não há necessidade de se ficar colocando *traps* do tipo `${varquequeropesquisar}` no código.

Obs: existe a macro `@dump`, que é uma forma simplificada do `@dumpall`, pois só lista as variáveis do formulário (que é uma parte do data-model).

Tipificação (tipos de objetos listados pela macro):

```
is_method
is_enumerable
is_hash_ex      //hash extendido
is_number
is_string
is_boolean
is_date
is_transform
is_macro
is_hash
is_node
```

Data Model, o Hash Root, o hash (classe) func, o hash (classe) exbl, o hash (classe) doc, as variáveis (criadas via #assign) do template, as variáveis locais das macros / function criadas via #local

DATA MODEL	AMBIENTE DE EXECUÇÃO
<pre> root + Func (hash) + Exbl (hash) + Doc (hash) + Outras variáveis </pre>	<pre> +-----+ Global +-----+ +-----+ Template sendo executado Main +-----+ Macro ou Function Local +-----+ +-----+ </pre>

- ⇒ Cada template roda no NAMESPACE default chamado Main, onde constam as variáveis definidas via diretiva Assign.
- ⇒ Nas macros e functions são permitidas a criação de variáveis locais via diretiva Local
- ⇒ As variáveis definidas via diretiva Global podem ser utilizadas em todo o ambiente do freemarker
- ⇒ O DATA MODEL e o Hash Root são idênticos

Aplicabilidade: Supor a aplicação abaixo. Tudo bem, ela é muito simples e conseguimos debugá-la no olho, porém serve ao propósito de exemplo. Supor que estamos com problema na variável `dataformat`. Para tanto, vamos introduzir ma macro `@dumpall` após a declaração da variável.

```
[@selecao var="catFuncionario" titulo="Categoria do Funcionário"  
    reler=true opcoes="Servidor; Terceiro"/]  
[  
@br/  
@texto titulo="Ano Posse/Contratação" var="ano" largura="4" maxcaracteres="4"  
obrigatorio="Sim" reler="ajax" idAjax="anoAjax"/>  
@grupo depende="anoAjax"  
    [#if (ano!="") == ""]  
        @mensagem titulo="Alerta" texto="Ano deve ser preenchido."  
        vermelho=true/  
    [/#if]  
[/@grupo]  
@lotacao titulo="Lotação" var="lotacao" reler=true idAjax="lotacaoAjax"/>  
[  
@br/  
@data var="dataformat" titulo="Data de Formatura:"/>  
[@dumpall/]  
[  
@br/  
@memo var="informacoes" titulo="Dados sobre a Formação" colunas="63" linhas="3"  
/  
@checkbox var="fazertrein" titulo="Deseja fazer treinamento?" default="Sim"/>  
@br/  
@grupo titulo="Turno de Preferência"  
@radio titulo="Manhã" var="radio_resp" valor="1" default="Manhã" /]  
@radio titulo="Tarde" var="radio_resp" valor="0" /]  
]
```

[/@grupo]

Após rodar a aplicação, conseguiremos a tela abaixo, logo após ao campo dataformat da entrevista. Os campos em turquesa foram destacados por mim, e correspondem aos campos do formulário de entrevista. Os destaques em amarelo são para ser analisados com atenção:

Cada template roda no NAMESPACE default chamado Main, onde constam as variáveis definidas via diretiva Assign

Nas macros e functions são permitidas a criação de variáveis locais via diretiva Local

As variáveis definidas via diretiva Global podem ser utilizadas em todo o ambiente do freemarker

O DATA MODEL e o Hash Root são idênticos

===== VARIÁVEIS DO DATA MODEL =====

```
lotacao_lotacaoSel.id = ""
subscritorSel.sigla = ""
root = ?? (hash)
classificacaoSel.descricao = ""
lotacao_lotacaoSel.descricao = ""
mobilPaiSel.descricao = ""
reqmobilPaiSel = ""
fazertrein = "Sim"
classificacaoSel.sigla = ""
lotacaoDestinatarioSel.id = ""
ano = ""
reqsubscritorSel = ""
tipoDestinatario = "2"
func = ?? (hash)
exbl = ?? (hash)
preenchimento = "0"
alterouModelo = ""
template = "[@selecao var="catFuncionario" titulo="Categoria do Funcionário" reler=true
opcoes="Servidor; Terceiro"/] [ @br/ ] [ @texto titulo="Ano Posse/Contratação" var="ano"
largura="4" maxcaracteres="4" obrigatorio="Sim" reler="ajax" idAjax="anoAjax"/ ] [ @grupo
depende="anoAjax" ] [ #if (ano!="") == "" ] [ @mensagem titulo="Alerta" texto="Ano deve ser
preenchido." vermelho=true/ ] [ /#if ] [ /@grupo ] [ @lotacao titulo="Lotação" var="lotacao"
reler=true idAjax="lotacaoAjax"/ ] [ @br/ ] [ @data var="dataformat" titulo="Data de
Formatura:"/ ] [ @br/ ] [ @memo var="informacoes" titulo="Dados sobre a Formação" colunas="63"
linhas="3" / ] [ @checkbox var="fazertrein" titulo="Deseja fazer treinamento?"
default="Sim"/ ] [ @br/ ] [ @grupo titulo="Turno de Preferência" ] [ @radio titulo="Manhã"
var="radio_resp" valor="1" default="Manhã" / ] [ @radio titulo="Tarde" var="radio_resp"
valor="0" / ] [ /@grupo ] [ @br/ ] [ @dumpall/ ] "
idFormaDoc = "60"
reqlotacao_lotacaoSel = "sim"
fazertrein_chk = "Sim"
mobilPaiSel.id = ""
mobilPaiSel.buscar = ""
webwork.token = "NV5YPPID3TEUA02T3S3RHIUNT2K390V4"
gerar_entrevista = true
alterouSel = "lotacao,,,,,"
lotacaoDestinatarioSel.descricao = ""
lotacao_lotacaoSel.buscar = ""
desativarDocPai = ""
nmArqMod = ?? (enumerável)
titularSel.buscar = ""
vars =
"catFuncionario,ano,lotacao_lotacaoSel.id,lotacao_lotacaoSel.sigla,lotacao_lotacaoSel.desc
ricao,dataformat,informacoes,fazertrein,radio_resp"
mobilPaiSel.sigla = ""
serverAndPort = "localhost:8080"
informacoes = ""
titularSel.descricao = ""
reqtitularSel = ""
idTpDoc = "1"
lotacaoDestinatarioSel.sigla = ""
```

A variável **template** contém a aplicação FM

A variável **vars** contém o nome de todas as variáveis do formulário de entrevista da parte variável

```

radio_resp = ""
subscritoresSel.buscar = ""
nmMod = "RubenTeste"
reqclassificacaoSel = ""
param = ?? (hash)
classificacaoSel.id = ""
campos =
"despachando,criandoAnexo,idTpDoc,dtDocString,nivelAcesso,eletronico,subscritoresSel.id,subs
tituicao,titularSel.id,nmFuncaoSubscritores,tipoDestinatario,lotacaoDestinatarioSel.id,preen
chimento,classificacaoSel.id,descrDocumento"
lotacao_lotacaoSel.sigla = ""
nomePreenchimento = ""
obrigatorios = "ano"
titularSel.id = ""
subscritoresSel.id = ""
classificacaoSel.buscar = ""
reqlotacaoDestinatarioSel = ""
_FALSEE.substituicao = "false"
subscritoresSel.descricao = ""
catFuncionario = "Servidor"
entrevista = "1"
idMod = "742"
criandoAnexo = "false"
lotacaoDestinatarioSel.buscar = ""
descrDocumento = ""
despachando = "false"
postback = "1"
sigla = ""
lotaTitular = ?? (hash)
nivelAcesso = "1"
titularSel.sigla = ""
doc = ?? (hash)
nmFuncaoSubscritores = ""
webwork.token.name = "webwork.token"
dataformat = ""
dtDocString = ""

```

A variável **campos** contém o nome de todas as variáveis do formulário de entrevista da parte fixa

A variável **obrigatorios** contém o nome de todas as variáveis do formulário de entrevista que foram definidas como obrigatório

===== TEMPLATE SENDO EXECUTADO (NAMESPACE MAIN)=====



ATENÇÃO !

Aqui serão listados os nomes de todas as macros e functions, assim como dos Assign (não é o assign dentro da macro ou function, e sim dos assign feitos diretamente no arquivo geral - Funcionando como constantes) definidos no arquivo geral que contém as macros.

```

extensaoAssinadorLote = ?? (macro)
rodape = ?? (macro)
assinatura = ?? (macro)
cabecalhoEsquerdaPrimeiraPagina = ?? (macro)
rodapeNumeracaoADireita = ?? (macro)
cabecalho = ?? (macro)
vertipo = ?? (macro)
extensaoBuscaTextual = ?? (macro)
estiloBrasaoAEsquerda = ?? (macro)
numeroDJE = ?? (macro)
br = ?? (macro)
temRadio_radio_resp = true
tituloDJE = ?? (macro)
formatarCPF = ?? (macro)
lotacao = ?? (macro)
_secretario_rh = ?? (hash)
formulario = ?? (macro)
rodapeNumeracaoCentralizada = ?? (macro)
endereçoDiretorGeral = "Ilmo(a). Sr(a). Diretor(a)-Geral"
portaria = ?? (macro)
assentamento_funcional = ?? (macro)
key = "key"

```

```

[#assign _secretario_rh = {
"genero":"F",
"vocativo":"Senhora",
"vocativo_carta":"Sra. Diretora da
Subsecretaria de Gestão de Pessoas",
"endereço":"Sra. Dra.",
"nome":"<DEFINIR_NOME>",
"cargo":"<DEFINIR_CARGO>",
"orgao":"<DEFINIR_ORGAO>",
"endereço":"Avenida Almirante Barroso,
78 - Centro - Rio de Janeiro/RJ - CEP:
20031-004"} /]

```

Obs: Se quisermos acessar um dado deste hash utilizamos a notação: `_secretario_rh.vocativo`, por exemplo, para obter "Senhora".

Quando a macro `lotacao` foi invocada, esta chama a macro `selecionavel` que define estas duas variáveis com assign, que conterá dados da última macro chamada (`lotacao` ou função ou pessoa)

```

finalizacao = ?? (macro)
tipoSel = "_lotacao"
varName = "lotacao_lotacaoSel.descricao"
fmtvldCPF = ?? (macro)
grupo = ?? (macro)
msgid = ?? (macro)
texto = ?? (macro)
atualizaoculto = ?? (macro)
estiloBrasaoCentralizado = ?? (macro)
_presidente = ?? (hash)
assinaturaCentro = ?? (macro)
editor = ?? (macro)
corpoBIE = ?? (macro)
dadosComplementares = ?? (macro)
quebraPagina = ?? (macro)
primeiroCabecalho = ?? (macro)
assinaturaBIE = ?? (macro)
cabecalhoEsquerda = ?? (macro)
mioloDJE = ?? (macro)
assinaturaMovCentro = ?? (macro)
inlineTemplate = ?? (transform)
separador = ?? (macro)
dump = ?? (macro)
memo = ?? (macro)
oficio = ?? (macro)
validarCPF = ?? (macro)
rodapeClassificacaoDocumental = ?? (macro)
div = ?? (macro)
data = ?? (macro)
fixcrlf = ?? (macro)
caixaSelecao = ?? (macro)
botoesExtensaoAssinador = ?? (macro)
par = ?? (macro)
primeiroRodape = ?? (macro)
obrigatorios = ?? (macro)
funcao = ?? (macro)
requerimento2 = ?? (macro)
enderecamentoPresidente = "Exmo. Sr. Juiz Federal - Diretor de Foro"
memorando = ?? (macro)
oculto = ?? (macro)
aberturaBIE = ?? (macro)
cabecalhoCentralizadoPrimeiraPagina = ?? (macro)
entrevista = ?? (macro)
letra = ?? (macro)
mensagem2 = ?? (macro)
atualizacampo = ?? (macro)
requerimento = ?? (macro)
mensagem = ?? (macro)
selecionavel = ?? (macro)
fechoBIE = ?? (macro)
pessoa = ?? (macro)
enderecamentoDiretorDeRH = "Ilma. Sra. Diretora da Subsecretaria de Gestão de Pessoas"
documento = ?? (macro)
resumo = ?? (macro)
_secretario_geral = ?? (hash)
cabecalhoCentralizado = ?? (macro)
identificacao = ?? (macro)
selecao = ?? (macro)
topico = ?? (macro)
sec = ?? (macro)
item = ?? (macro)
checkbox = ?? (macro)
estiloBrasaoCentralizadoSEC = ?? (macro)
radio = ?? (macro)
extensaoEditor = ?? (macro)
extensaoAssinador = ?? (macro)
XStandard = ?? (macro)

```

```

[#assign _presidente = {
"genero":"M",
"vocativo":"Excelentíssimo Senhor",
"vocativo_carta":"Exmo. Sr. Juiz Federal
- Diretor do Foro",
"enderecamento":"Exmo. Sr. Dr.",
"nome":"<DEFINIR_NOME>",
"cargo":"<DEFINIR_CARGO>",
"orgao":"<DEFINIR_ORGAO>",
"endereço":"Avenida Almirante Barroso,
78 / 13º andar - Centro - Rio de
Janeiro/RJ - CEP: 20031-004"} /]

```

Esta variável é utilizada pelas macros checkbox, radio e seleção. Ela conterá trechos de código FM que são interpretados e executados dinamicamente

```

[#assign _secretario_geral = {
"genero":"F",
"vocativo":"Senhora",
"vocativo_carta":"Sra. Diretora da
Secretaria Geral",
"enderecamento":"Sra. Dra.",
"nome":"<DEFINIR_NOME>",
"cargo":"<DEFINIR_CARGO>",
"orgao":"<DEFINIR_ORGAO>",
"endereço":"Avenida Almirante Barroso,
78 - Centro - Rio de Janeiro/RJ - CEP:
20031-004"} /]

```

===== VARIÁVEIS DO HASH ROOT (Idêntica do DATA_MODEL) =====

... Não listarei aqui

===== VARIÁVEIS DO HASH (DA CLASSE) FUNC =====

```
hashCode = ?? (method)
quebraLinhas = ?? (method)
contains = ?? (method)
has = ?? (method)
calculaData = ?? (method)
formatarCPF = ?? (method)
lotacaoPessoa = ?? (method)
... aqui neste documento não listarei todos métodos
```

===== VARIÁVEIS DO HASH (DA CLASSE) EXBL =====

```
cancelarJuntada = ?? (method)
excluirMovimentacao = ?? (method)
receberEletronico = ?? (method)
processar = ?? (method)
pedirPublicacao = ?? (method)
... aqui neste documento não listarei todos métodos
```

===== VARIÁVEIS DO HASH (DA CLASSE) DOC =====

```
conteudoBlobForm = "?? (method)"
setExNivelAcesso = "?? (method)"
isAssinadoEletronicoPorTodosOsSignatarios = "?? (method)"
hashCode = "?? (method)"
getAnoEmissao = "?? (method)"
... aqui neste documento não listarei todos métodos
```

===== VARIÁVEIS DEFINIDAS COMO LOCAL =====

```
dump_var = "variável local criada para não gerar erro na rotina de listar as variáveis locais"
```

10.1.1.2 - Depurando o template (aplicação) FM utilizando o utilitário Integrador

Ver seção "12.6 - Integrador" para obter todos os detalhes de como depurar um template FM utilizando este utilitário.

10.1.2- Depurando os métodos JAVA

Como já vimos em outras seções, um template FM pode chamar métodos JAVA. Desta forma, podemos utilizar o ambiente da IDE (JBDS, Eclipse ...) para depurar estes métodos.

Supor que tenhamos a seguinte aplicação FM:

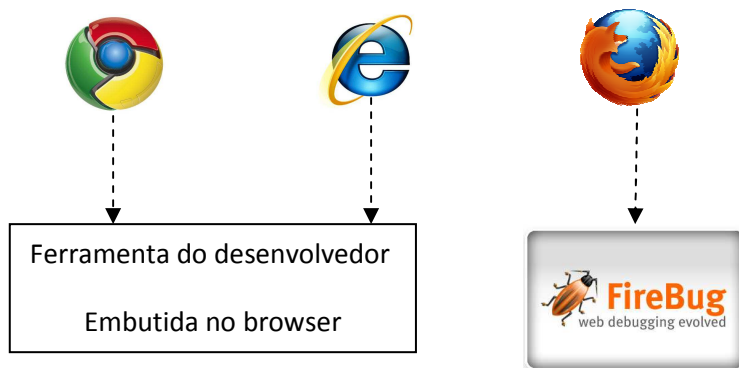
```
[#assign suprid = func.calculaData("22","08/09/2012")]
${suprid}
```

A aplicação FM acima executa o método calculaData() na classe func (FuncoesEl.java). Para mais detalhes sobre este método e outros, consultar a seção "5.1 - VARIÁVEIS DO HASH FUNC - CLASSE FuncoesEl.java".

Ver "Anexo 3 - Item 7 - Debugando um método JAVA" para obter mais detalhes de como depurar um método na IDE JBOSS Developer Studio.

10.2 - Depuração no cliente

Aqui nós temos boas, ou melhor, excelentes ferramentas para análise / depuração do cliente. Estas ferramentas estão embutidas no próprio browser. No Chrome e no IE, temos a ferramenta do desenvolvedor que já vem instalada. No FF (FireFox) nós temos o Firebug, que temos que baixar e instalar no browser.



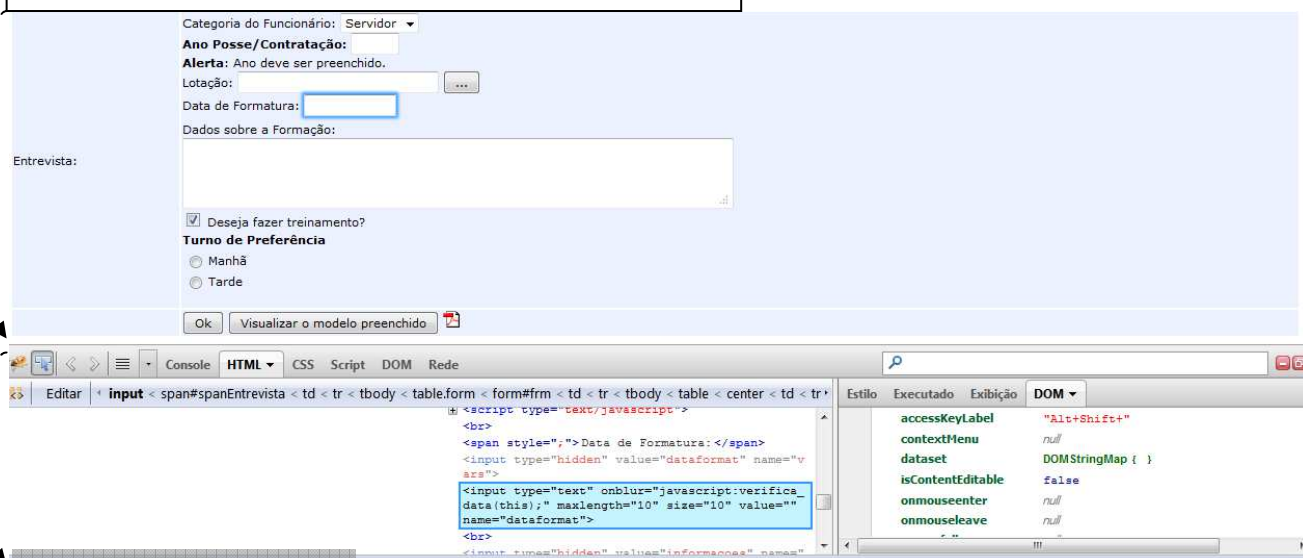
10.2.1 - Firebug

A minha preferência é o Firebug, que pode ser obtido em: (<http://getfirebug.com>).

O que podemos analisar no cliente com estas ferramentas?

Para responder a esta pergunta é interessante exibirmos a teal do depurador, que se abre abaixo da página WEB que está sendo depurada. Está longe deste manual explicar o pleno funcionamento desta ferramenta que é extremamente complexo. Farei somente um destaque da coisas importantes.

Tela do entrevista do exemplo anterior, visto em 10.1



Tela do depurador

Clicando no campo da entrevista, o FireBug (FB) já exibe o HTML correspondente, no caso, o cursor está no campo dataformat (Data de Formatura). Reciprocamente, se clicarmos no HTML ele exibe o campo da entrevista correspondente.

Na aba principal do FB temos:

10.2.2 - HTML / Estilo (CSS)

Com o cursor no Ano da Posse, podemos ver na esquerda o HTML correspondente e na direita o CSS (cascateado e herdado) que vigora para o campo em questão.

(preencher o campo acima com palavras-chave, sempre usando substantivos, gênero masculino e singular)

Categoria do Funcionário: Servidor

Ano Posse/Contratação:

Alerta: Ano deve ser preenchido.

Lotação:

Console HTML CSS Script DOM Rede

pan#spanEntrevista < td < tr < tbody < table>form < form#frm < td < tr < tbody < table < center < td < tr >

```
<select onchange="javascript:sbmt();" name="catFuncionario"><br><input type="hidden" value="ano" name="vars"><input type="hidden" value="ano" name="obrigatorios"><span style="font-weight:bold;">Ano Posse/Contratação:</span><input type="text" maxlength="4" size="4" onchangen="javascript:sbmt('anoAjax');" value="" name="ano"><div id="divanoAjax" depende="anoAjax"><input type="hidden" value="lotacao_lotacaoSel.id" name="vars"><input type="hidden" value="lotacao_lotacaoSel
```

Estilo Executado Exibição DOM

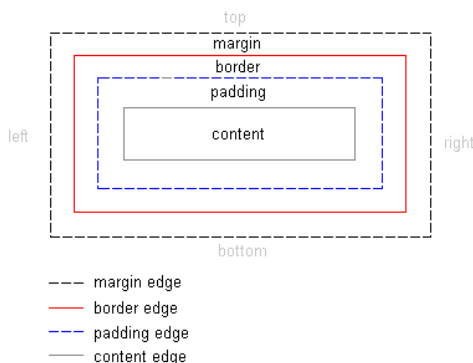
body, p, td, input, button, select { font-size: 11px; }
body, p, td, h1, h2, h3, input, button, select, newsarticle { font-family: Verdana,Arial,Helvetica,sans-serif; }

Herdado de td
body, p, td, input, button, select { font-size: 11px; }
body, p, td, h1, h2, h3, input, button, select, newsarticle {

10.2.3 - HTML / Exibicao (LayOut) O BOX MODEL

O box model (modelo das caixas) em CSS, descreve os boxes (as caixas) geradas pelos elementos HTML. O box model, detalha ainda, as opções de ajuste de margens, bordas, padding e conteúdo para cada elemento. Abaixo um diagrama representando a estrutura de construção do box model:

O box model em CSS



Com o cursor no Ano da Posse, podemos ver na esquerda o HTML correspondente e na direira o Box Model do campo. Observar como o campo foi enquadrado na régua.

0 50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900 950

Categoria do Funcionário: Servidor ▼

Ano Posse/Contratação:

Alerta: Ano deve ser preenchido.

Lotação:

Console HTML CSS Script DOM Rede

span#spanEntrevista <td <tr <tbody <table.form <form#frm <td <tr <tbody <table <center <td <tr <

```
<input type="hidden" value="ano" name="vars">
<input type="hidden" value="ano" name="obrigatorios">
<span style="font-weight:bold;">Ano
Posse/Contratação:</span>
<input type="text" maxlength="4" size="4" onchange="javascript:
sbmt('anoAjax');" value="" name="ano">
<div id="divanoAjax" depende="anoAjax;">
<input type="hidden" value="lotacao_lotacaoSel.
id" name="vars">
<input type="hidden" value="lotacao_lotacaoSel.
sigla" name="vars">
<input type="hidden" value="lotacao_lotacaoSel.
descricao" name="vars">
```

Estilo Executado Exibição ▼ DOM

margin

borda 1

enchimento 2

1 2 41 x 15 2 1

2

1

posição: static

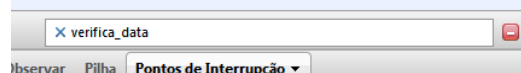
box-sizing: border-box;

z: auto

10.2.4 - Script

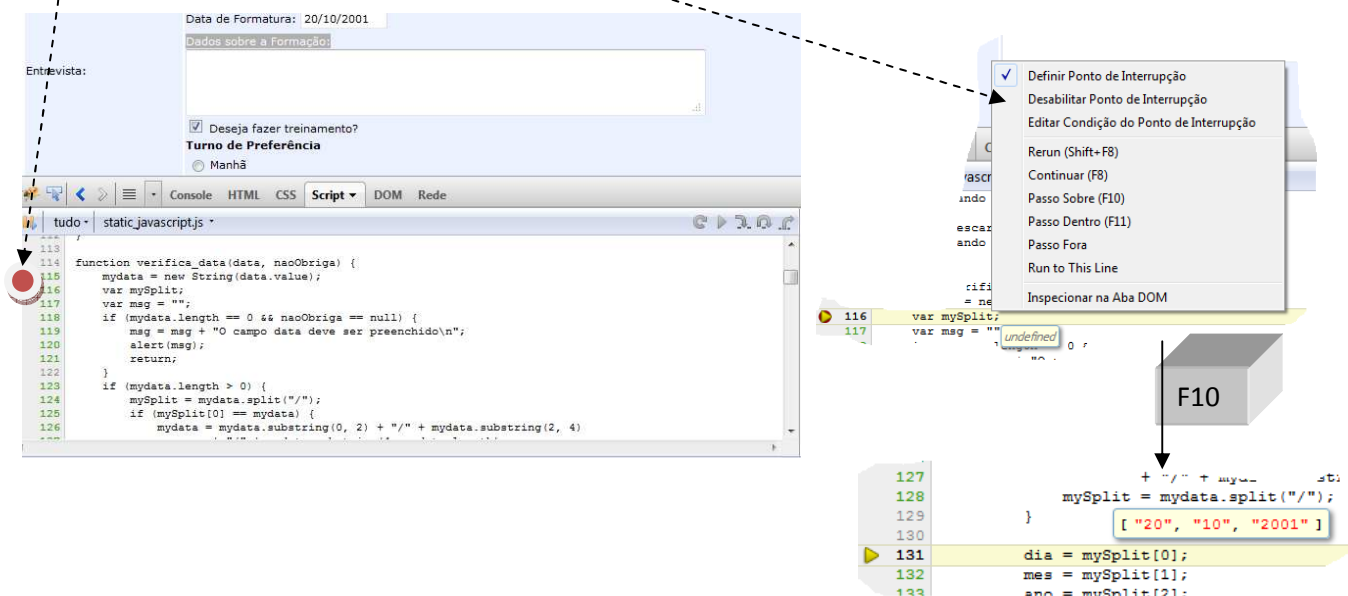
O campo Data de Formatura possui um JS associado a ele, e o mesmo é executado no evento **onchange**. É só observar no HTML da tela do item 10.2.1. A rotina se chama **verifica_data**.

Na aba Script, selecionei o arquivo de script JS chamado **static_javascript.js**. Por quê? Porque é nele que está definido a função que desejamos depurar. Existem 2 formas de descobrir onde a função reside: uma é neste manual, na seção 3.8 e a outra é consultando no próprio FB, selecionando os arquivos .JS na aba Script (ou na aba DOM) e pesquisando via área de pesquisa.



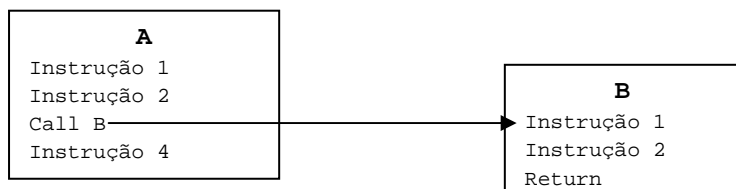
Defini um ponto de interrupção (breakpoint) para analisar a variável **mySplit** (linha 116).

Na entrevista preenchi o campo com a data 20/010/2001. Após o preenchimento a tela abaixo é apresentada, com a parada no local previsto. Com o botão direito do mouse na linha, surgem as seguintes opções:



Na linha 116 o conteúdo de **mySplit** é indefinido, pois ela está sendo definida. Na linha 128, já conseguimos analisar o seu conteúdo (20, 10, 2001).

Funções do depurador JS:



F8 (continue) - Continuar: continua a executar a aplicação até o próximo breakpoint, caso exista, ou até ao final do código.

Shift + F8 (Rerun) - Volta a executar a função do início.

Terminate - Interrompe o servidor em modo Debug

Resume - Executa o código até o próximo breakpoint, caso exista, ou até o final do código.

Step over - Passo sobre: vai executando linha-a-linha, porém se o seu código chama outro método / função (incluindo os métodos / funções da linguagem, tais como: `parse.int`, `string ...`) ele não entra. Se na figura acima o debug (em A) está na instrução `Call B`, ele executará B, porém não entrará no código de B.

Step **over** proceeds to the next line in your current scope (i.e. it goes to the next line), without descending into any method calls on the way. This is generally used for following the logic through a particular method without worrying about the details of its collaborators, and can be useful for finding at what point in a method the expected conditions are violated.

Step into - Passo Dentro. Idêntico ao `step over`, porém ele entrará nos métodos / funções chamados. Se na figura acima o debug (em A) está na instrução `Call B`, ele entrará em B, e continuará executando linha-a-linha. A única forma de sair de B é chamar o `Step out` / `Step return`.

Step **into** will cause the debugger to descend into any method calls on the current line. If there are multiple method calls, they'll be visited in order of execution; if there are no method calls, this is same as `step over`. This is broadly equivalent to following every individual line of execution as would be seen by the interpreter.

Step Out / Step Return - termina a execução do método / função e retorna ao chamador na instrução de chamada. Se na figura acima o debug (em A) está na instrução `Call B`, e o usuário está modo `Step into`, ele entrará em B, e continuará executando linha-a-linha. A única forma de sair de B diretamente (sem executá-lo) é chamar o `Step out` / `Step return`, que fará o debug retornar para A na instrução `Call B`. Se usuário fornecer `Step over`, ele executará sem entrar em B, se der `Step into`, entrará novamente em B, e executará linha-a-linha.

Step out proceeds until the next "return" or equivalent - i.e. until control has returned to the preceding stack frame. This is generally used when you've seen all you need to at thispoint/method, and want to bubble up the stack a few layers to where the value is actually used.

Imagine the following code, which entered through `main()` and is now on the first line of `bar`:

Exemplo:

```
function main() {
  val s = foo();
  bar(s);
}

function foo() {
  return "hi";
}

function bar(s) {
```

```

    val t = s + foo(); // <== Debugger is currently here
    return t;
}

```

Then:

- **Step into (F11)** will proceed into the foo call, and the current line will then become the return "hi"; line within foo.
- **Step over** will ignore the fact that another method is being invoked, and will proceed to the return t; line (which lets you quickly see what t is evaluated as).
- **Step out** will finish the execution of the rest of the bar method, and control will return to the last line of the main method.

Resumo:

Step Into	Step Over	Step Out
<ul style="list-style-type: none"> ➤ Step Into will cause the debugger to go into the next function call and break there. ➤ Go into the subroutine and wait for next action. ➤ Change the debugger context to run into the function the code is stopped on. If the code cannot step into the function, this is the same as Step Over. 	<ul style="list-style-type: none"> ➤ Step Over will tell the debugger to execute the next function and break afterwards. ➤ Jump over the subroutine without waiting again. ➤ Execute the code the debugger is stopped on, but stay within the current function. 	<ul style="list-style-type: none"> ➤ Step Out will tell the debugger to finish the current function and break after it. ➤ If you are in the subroutine, you will leave it without waiting again ➤ Execute code until the end of the current function, and resume debugging once it has returned.

10.2.5 - DOM

Existem dois tipos de objetos e funções:

- Aqueles que são parte do padrão DOM, e
- Aqueles definidos pelo usuário através de JS / DOM.

Firebug exibe os objetos e funções criados pelos usuários, via scripts, em negrito, no topo da lista dos elementos DOM.

O sistema de cores da Aba DOM:

Black are properties and green are methods. Bold means the member was declared "by the user" meaning the members aren't from the default

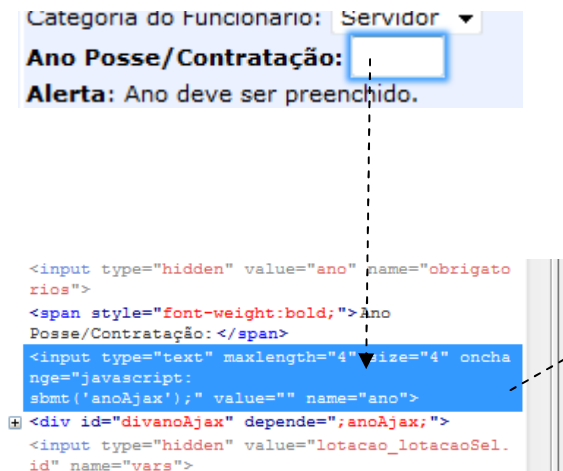
Numbers are blue, strings are red. Objects appear as a "instance preview" in which the type name and the member names are green and the member values

window	
a1	undefined
ajax	Object { x=function(), serialize=function(), send=function(), more... }
get	function ()
gets	function ()
post	function ()
send	function ()
serialize	function ()
submit	function ()
update	function ()
x	function ()
ap1	div#imouter1.imcm
atag	a#ulaitem1z2#
b1	undefined
carregando	false
cm_obj	Object { }
conexaoTimer	undefined
counter	0
dd	<TextNode textContent="" ">

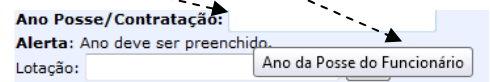
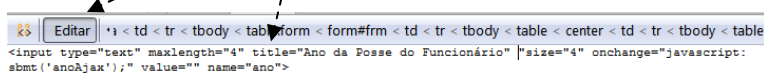
Navegando na Hierarquia: Document/Head ou Document/Body:

document	Document editar.action
currentScript	null
head	head
mozFullScreen	false
mozFullScreenElement	null
mozFullScreenEnabled	true
mozHidden	false
mozSyntheticDocument	false
mozVisibilityState	"visible"
onmouseenter	null
onmouseleave	null
onmozfullscreenerror	null
parentElement	null
scripts	[script /sigaex/.../ajax.js, script /sigaex/.../cript.js, script, 13 more...]
contains	contains ()
mozCancelFullScreen	mozCancelFullScreen ()
mozSetImageElement	mozSetImageElement ()
onmouseover	function ()
releaseCapture	releaseCapture ()
URL	"http://localhost:8080/s...iente/doc/editar.action"
activeElement	body
alinkColor	" "
anchors	[]
applets	[]
attributes	null
baseURI	"http://localhost:8080/s...iente/doc/editar.action"
bgColor	" "
body	body
characterSet	"UTF-8"

Vamos a um exemplo. Com o cursor no Ano da Posse, na esquerda temos o HTML e na direita os atributos DOM, incluindo: o nodo pai, nodos filhos, o array de atributos do elemento e etc.



Podemos alterar em tempo real as propriedades do elemento. Por exemplo: vamos colocar um title com o conteúdo "Ano da Posse do Funcionário". Podemos clicar no outerHTML da aba DOM, ou na aba HTML, clicar no elemento e com o botão direito selecionar "Editar HTML". Podemos então incluir: title="Ano da Posse do Funcionário", clicar em Editar e PRONTO. Se clicarmos o cursor no campo Ano, já veremos a caixa com o título. Tudo em tempo real, sem reload da página.



Através da aba HTML ou DOM, pode-se:

- Alterar o elemento;
- Incluir um novo elemento;
- Excluir um elemento;

Tudo dinamicamente, sem reload da página.

Como saber a estrutura DOM de um elemento?

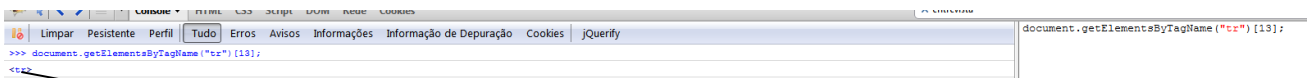
Às vezes, é importante sabermos qual é o DOM de um elemento para tabalhar com o JS.

Exemplo: na tela de entrevista, para desenvolver o aplicativo Integrador (Capítulo 12 – Ambiente de Desenvolvimento), necessitei saber o DOM associado ao elemento <td>Entrevista:</td>.

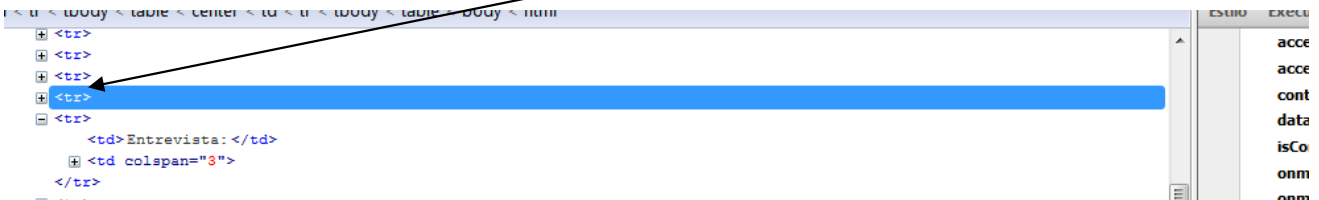
- Primeiro acessei a aba HTML e realizei uma pesquisa com Entrevista;
- Ao achar, cliquei no elemento para obter o XPath (/html/body/table/tbody/tr/td/center/table/tbody/tr/td/form/table/tbody/tr [13]/td). Observem o tr[13].;



- Então tentei acessar o elemento emitindo o seguinte comando no console:
`document.getElementsByTagName("tr")[13];`



Clicando em <TR>, no levará ao elemento na tela



- Observei que o elemento é anterior ao que desejo acessar, então tento o seguinte comando: `document.getElementsByTagName("tr")[14]`, o que me leva ao <TR> Correto. Porém desejo acessar o <td>, e emito o seguinte comando: `document.getElementsByTagName("tr")[14].childNodes[0]`, supondo que ele fosse o primeiro elemento do <TR>;
- Não obtive sucesso, e tentei o comando: `document.getElementsByTagName("tr")[14].childNodes[1]`, obtendo sucesso. Para me garantir, solicitei o innerHTML e o outerHTML.

```
document.getElementsByTagName("tr")[14].childNodes[1].innerHTML
"Entrevista:"
```

```
document.getElementsByTagName("tr")[14].childNodes[1].outerHTML
"<td>Entrevista:</td>"
```

10.2.6 - Rede

Aqui pode-se observar todo o fluxo de dados entre o cliente e servidor. Os métodos, GET ou POST, o HTML recebido pelo cliente, os dados passados ao servidor, o tempo de cada requisição, gargalos e etc.

CASO 1: Requisições durante um AJAX

Vamos ver o que acontece quando o Ano da Posse é preenchido, visto que neste caso, o AJAX será ativado.

Neste exemplo, vamos preencher o campo Ano da Posse:

The first screenshot shows a web form with fields for 'Categoria do funcionário', 'Ano Posse/Contratação' (set to 2011), 'Lotação', and 'Data de Formatura'. Below the form is the 'Dados sobre a Formação' section. The second screenshot shows the 'Rede' (Network) tab with a single request 'POST editar.action' from 'localhost:8080' with a status of '200 OK' and a size of '80.5 KB'. The third screenshot shows the 'Resposta' (Response) tab for the same request, displaying the HTML content of the response, which includes the same form as in the first screenshot, but with the 'Ano Posse/Contratação' field now filled with '2011'.

O que foi postado para o servidor. Observar as variáveis enviadas, incluindo a data da posse

HTML recebido pelo cliente

Observar que a data foi fornecida, logo não há a mensagem de erro

No AJAX, temos somente 1 Request

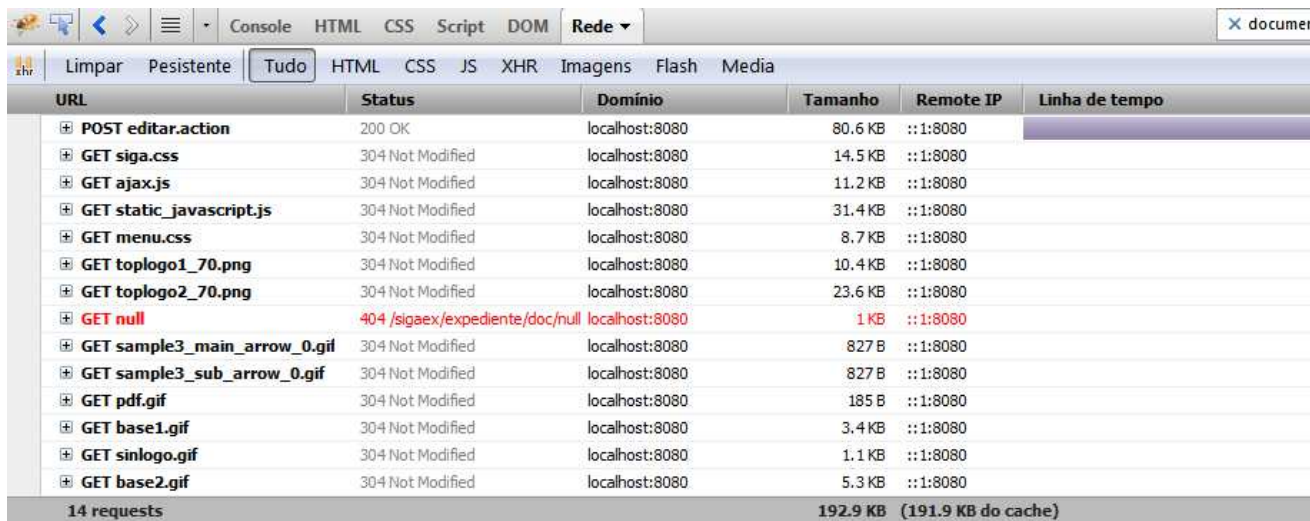
The screenshot shows the 'Script' tab in the developer tools. A break point is set on the line 'xmlhttp.send(xmlhttp.serialize(frm));'. A 'STOP' sign icon is placed over the 'xmlhttp.send' text. The 'Replacer...Response' pane shows the current state of the 'ajax.js' file.

Se voltarmos a preencher o campo Ano da Posse, com o XHR habilitado, a aplicação parará no xmlhttp.send, como mostrado na tela abaixo.

Observe que a aba mudou para Script automaticamente. Neste ponto pode se fazer uma análise do código, variáveis e etc.

The screenshot shows the 'Script' tab in the developer tools. A break point is set on the line 'xmlhttp.send(xmlhttp.serialize(frm));'. The 'Replacer...Response' pane shows the current state of the 'ajax.js' file.

CASO 2: O que ocorre durante um F5 (Reload de página) ou com campo que possua `reler=true`?



URL	Status	Domínio	Tamanho	Remote IP	Linha de tempo
POST editar.action	200 OK	localhost:8080	80.6 KB	::1:8080	
GET siga.css	304 Not Modified	localhost:8080	14.5 KB	::1:8080	
GET ajax.js	304 Not Modified	localhost:8080	11.2 KB	::1:8080	
GET static_javascript.js	304 Not Modified	localhost:8080	31.4 KB	::1:8080	
GET menu.css	304 Not Modified	localhost:8080	8.7 KB	::1:8080	
GET toplogo1_70.png	304 Not Modified	localhost:8080	10.4 KB	::1:8080	
GET toplogo2_70.png	304 Not Modified	localhost:8080	23.6 KB	::1:8080	
GET null	404 /sigaex/expediente/doc/null	localhost:8080	1 KB	::1:8080	
GET sample3_main_arrow_0.gif	304 Not Modified	localhost:8080	827 B	::1:8080	
GET sample3_sub_arrow_0.gif	304 Not Modified	localhost:8080	827 B	::1:8080	
GET pdf.gif	304 Not Modified	localhost:8080	185 B	::1:8080	
GET base1.gif	304 Not Modified	localhost:8080	3.4 KB	::1:8080	
GET sinlogo.gif	304 Not Modified	localhost:8080	1.1 KB	::1:8080	
GET base2.gif	304 Not Modified	localhost:8080	5.3 KB	::1:8080	
14 requests			192.9 KB	(191.9 KB do cache)	

14 Requests, entre eles: Gifs, Pngs, JS , CSS são carregados novamente.

10.2.7 - Console

A console apresenta os erros de HTML e JS. Pode-se também entrar com comandos na console, ou seja, comandos JS. Pode-se carregar uma aplicação JS a partir da console.

Exemplo: Entrar com o comando: `document.getElementById("frm")`. O comando foi entrado na janela da direita, depois executar, e o resultado com o elemento HTML cujo id é "frm" aparece na janela da esquerda.

