PLATAFORMA DE DESENVOLVIMENTO PINHÃO PARANÁ
SISTEMA SCRIBAPINHÃO
MANUAL DE ACOPLAMENTO

Junho – 2006

Sumário de Informações do Documento		
Tipo do Documento: Manual		
Título do Documento: Manual de Acoplagem ao Sistema Hospedeiro		
Estado do Documento: Elaborado		
Responsáveis: Thiago Meneghello - Revisado por: Helio Silvio Piccinatto, Elisabeth Hoffmann e Cíntia Evangelista.		
Palavras-Chaves: Escriba, Workflow, Acoplagem		
Resumo: Esse documento contém o manual de acoplagem para sistemas hospedeiros utilizarem o sistema Escriba		
Número de páginas: 56		
Software utilizados: OpenOffice 2.0		
Versão	**Data**	**Mudanças**
1.0	07/06/06	Elaboração do Manual
1.1	12/06/06	Primeira revisão
1.1.1	23/03/07	Inclusão de Documentação de Taglibs - Leslie
1.2	18/12/2007	Inclusão de serviços do componente cliente da versão 1.2
1.3	22/09/2008	Atualizado serviços do componente cliente da versão 1.3
1.4	17/11/2008	Atualizada serviços do componente cliente da versão 1.4

SUMÁRIO

1 INTRODUÇÃO

Este manual destina-se a fornecer informações aos responsáveis pelo desenvolvimento de sistemas baseados na Plataforma de Desenvolvimento Pinhão Paraná, que serão terão acoplados ao sistema o componente cliente do ScribaPinhão para controle de conteúdos.

O sistema ScribaPinhão nos fornece uma facilidade no controle de conteúdos, permitindo a configuração das informações necessárias para cada tipo de dado e fornecendo uma ferramenta para persistência e busca dessas informações de maneira simplificada, através de chamadas via WebService.

Todas as ações realizadas no repositório de conteúdos passa por um filtro interno do ScribaPinhão para a validação de permissão. Dessa forma, o ScribaPinhão controla o acesso dos usuários internamente, deixando a aplicação livre dessa preocupação. Os níveis de acesso a uma determinada informação está dividido em quatro categorias: **Leitura**, **Alteração**, **Exclusão**, **Permissão**, que estão explicados no documento *ScribaPinhao_politica_segurança_server*.

Nos capítulos subseqüentes, serão apresentados os procedimentos de acoplagem do sistema ScribaPinhão, via seu componente cliente, ao sistema hospedeiro. Internamente na CELEPAR, dúvidas a respeito destes procedimentos devem ser verificadas junto a GTI. Também serão abordados procedimentos relacionados à implementação, os quais deverão ser consultados junto a GIC.

2 ACOPLANDO O SISTEMA HOSPEDEIRO

A acoplagem do componente cliente do ScribaPinhão a um sistema hospedeiro deve passar por alguns procedimentos, são eles:

- Cópia dos arquivos do sistema ScribaPinhão para o diretório do projeto do sistema hospedeiro. Neste passo alguns arquivos deverão ser importados para a aplicação. Este passo será detalhado no tópico 2.1.
- Configuração dos tipos de dados utilizados pelo sistema hospedeiro através do ScribaPinhão ADM (interface administrativa), este passo será detalhado no tópico 2.2.

Após estes procedimentos, seu sistema poderá utilizar as funcionalidades do ScribaPinhão para desenvolver aplicações com controle de conteúdos.

2.1 Cópia de arquivos

O ScribaPinhão versão cliente está disponível em forma de um arquivo compactado em formato padrão JAR, que deve ser acoplado após o download do projeto mínimo do CVS.

Na página http://www.frameworkpinhao.pr.gov.br encontram-se os arquivos ".jar" que devem ser copiados da página para a pasta WEB_INF/lib do sistema hospedeiro, são eles:

- scriba_client_x_x_x.jar
- axis.jar
- axis-ant.jar
- axis-schema.jar
- commons-discovery-x.x.jar
- commons-logging-x.x.x.jar
- jaxrpc.jar
- log4j-x-x-x.jar
- saaj.jar

- wsdl4j-x-x-x.jar
- properties-plugin.jar
- commons-http-client-x.x
- commons-codec x.x

Após a cópia de arquivos, pode-se utilizar a biblioteca scriba_client para o acesso as informações do repositório.

2.1.1 Configurando através de repositório de componentes MAVEN

Se o projeto do sistema utilizador do ScribaPinhão utiliza o controle de versão de componentes pelo Maven, ele pode ter acesso ao componente cliente através da seguinte configuração de dependência no seu arquivo pom.xml:

```
...
  <repositories>
    <repository>
      <id>central</id>
      <url>http://maven.celepar.parana:8080/maven2/repo</url>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>central</id>
      <url>http://maven.celepar.parana:8080/maven2/repo</url>
    </pluginRepository>
  </pluginRepositories>
...
  <dependencies>
    <dependency>
      <groupId>scribapinhao</groupId>
      <artifactId>scribapinhao-client</artifactId>
      <!-- artifactId>scribapinhao-client</artifactId -->
      <version>1.4</version>
    </dependency>
...
  </dependencies>
```

As demais dependências de outras bibliotecas citadas também devem ser inseridas.

2.2 Configuração de área do sistema

Esta etapa não é essencial para a utilização do sistema ScribaPinhão, e deve ser feita somente se os tipos de dados definidos pelo ScribaPinhão não forem suficientes e/ou seja

necessária uma área de armazenamento para o sistema hospedeiro.

A definição de novos tipos de dados deve ser feita através do ScribaPinhão ADM, onde está incluso um arquivo ".XML" com as novas definições. Esse arquivo ".XML" deve seguir o mesmo padrão definido no documento *ScribaPinhao_conceituacao.pdf*.

A criação de uma área de armazenamento específica para o sistema hospedeiro deve ser feita através do ScribaPinhão ADM, da seguinte forma: deve ser cadastrado um novo sistema com os dados referentes ao sistema hospedeiro, e a partir do sucesso no cadastro será criada a área inicial para armazenamento de conteúdo com o mesmo nome do sistema hospedeiro, a partir desta área o sistema hospedeiro poderá trabalhar com seus conteúdos. Maiores informações sobre a utilização da interface de administração dos repositórios ScribaPinhão.

2.3 Configuração para obter conexão com serviços ScribaPinhão

Para poder se conectar ao servidor ScribaPinhao via webservice, é necessário disponibilizar ao menos uma das configurações a seguir:

A) (mais indicada) Configurar um arquivo do servidor de aplicação Jboss, que se encontra no diretório JBOSS_HOME/server/default/deploy, chamado "properties-service.xml". Deve-se descomentar a linha dentro da tag <attribute name="Properties">", incluindo a seguinte propriedade "gov.pr.celepar.escriba.adm.service.url", com o valor dependendo do ambiente desejado para a aplicação:

A.1) Para sistemas utilizando repositório ScribaPinhão de ambiente de DESENVOLVIMENTO:

```
<!--
gov.pr.celepar.escriba.adm.service.url=http://www.gic.desenvolvimento.eparana
.parana/gtf-escriba-adm/services/EscribaAdmService -->
```

A.2) Para sistemas utilizando repositório ScribaPinhão de ambiente de PRODUÇÂO:

```
<!-- gov.pr.celepar.escriba.adm.service.url=http://www.escriba-
adm.pr.gov.br/escriba-adm/services/EscribaAdmService -->
```

B) Configurar um arquivo ".xml" para conexão a uma base de dados (datasource), e disponibilizá-lo no diretório de deploy do servidor de aplicação Jboss.

B.1) Para sistemas utilizando repositório ScribaPinhão de ambiente de DESENVOLVIMENTO:

```
<datasources>
 <local-tx-datasource>
  <jndi-name>jdbc/ScribaAdmClienteDS</jndi-name>
  <connection-url>
jdbc:postgresql://bancos.pinhao.desenvolvimento.eparana.parana/pinhao
</connection-url>
  <driver-class>org.postgresql.Driver</driver-class>
  <user-name>sa_escriba_client</user-name>
  <password>stranger</password>
  <min-pool-size>1</min-pool-size>
  <max-pool-size>1</max-pool-size>
  <check-valid-connection-sql>SELECT 1+1</check-valid-connection-sql>
 </local-tx-datasource>
</datasources>
```

B.2) Para sistemas utilizando repositório ScribaPinhão de ambiente de PRODUÇÂO:

```
<datasources>
 <local-tx-datasource>
  <jndi-name>jdbc/ScribaAdmClienteDS</jndi-name>
  <connection-url>
jdbc:postgresql://bancos.pinhao.producao.eparana.parana:5434/pinhao
</connection-url>
  <driver-class>org.postgresql.Driver</driver-class>
  <user-name>sa_escriba_client</user-name>
  <password>CONSULTAR GTI</password>
  <min-pool-size>1</min-pool-size>
  <max-pool-size>1</max-pool-size>
  <check-valid-connection-sql>SELECT 1+1</check-valid-connection-sql>
 </local-tx-datasource>
</datasources>
```

Sendo que o repositório ScribaPinhão a ser acessado vai depender do nome da instância do

servidor ScribaPinhão que deve ser passado para o método "EscribaServicoFacade.getInstance(nomeContextoAplicacaoScribaServer)."

Atenção: o parâmetro "nomeContextoAplicacaoScribaServer" a ser utilizado pela aplicação em PRODUÇÂO deve ser pego com a GTI pois depende da configuração feita através do sistema administrador. Para a aplicação em DESENVOLVIMENTO pode ser utilizado neste parâmetro o valor "gtf-escriba", que fará uso da instância do servidor de repositório ScribaPinhão no ambiente de DESENVOLVIMENTO. Caso a aplicação necessite trabalhar com MAIS DE UMA FONTE DE REPOSITÓRIO ScribaPinhão deve ser passado o valor de parâmetro correspondente ao repositório necessário.

3 COMUNICAÇÃO ENTRE SISTEMA HOSPEDEIRO E SCRIBAPINHÃO

O sistema hospedeiro necessita comunicar-se com o ScribaPinhão para consultar informações referentes a conteúdos, que podem ser arquivos ou pastas. Esta comunicação deve ser feita preferencialmente centralizada dentro de uma classe DAO e o retorno deve ser tratado e convertido para classes POJO do sistema hospedeiro.

Para o acesso as informações do repositório ScribaPinhao, deve-se utilizar a classe **EscribaServicoFacade** através do método EscribaServicoFacade.getInstance(String nomeContextoAplicacaoScriba), e tomar o cuidado para que as chamadas sejam feitas somente após a criação de uma sessão entre a thread do usuário na aplicação hospedeira com a aplicação ScribaPinhao pelo método *novaSessao*, e sempre no fim dos tratamentos a serem feitos pela aplicação hospedeira deve ser fechada a sessão com o método *fechaSessao*. Para as chamadas de métodos que efetuem alterações no repositório e necessitem ser comitados para serem persistidos, deve ser feita a chamada ao método *salvaSessao*.

É uma boa prática aproveitar uma abertura de sessão com o Scriba para todas as necessidades de iteração da aplicação com o repositório, por exemplo a cada processamento de um request da aplicação procurar utilizar uma mesma sessão até a entrega final do resultado para o browser cliente, isso quem deve controlar é a própria aplicação.

A seguir temos um exemplo de uma chamada ao método *buscarEscribaNode* para trazer as informações do repositório e uma chamada ao método *criarDiretório* para criar uma pasta no repositório, lembrando também que qualquer chamada que altera informações do repositório deve ser seguida pela chamada do método *salvaSessao* para persistir os dados no repositório.

Ex 1:

```
try {
        //Cria nova sessão e busca informação do Node

EscribaServicoFacade eSF = EscribaServicoFacade.getInstance("gtf-escriba");
eSF.novaSessao(NOME_USUARIO, SENHA);
        EscribaNode eNode = eSF.buscarEscribaNode(CAMINHO);
```

```
} catch (Exception e) {
        //Tratamento da Exception
} finally {
        try {
        //Fechamento da sessão
         if(eSF != null && eSF.hasSessao())
          eSF.fechaSessao();
        } catch (Exception e) {}
}
```

Ex 2:

```
try {
        //Cria nova sessão, cria diretório e persiste as alterações
 EscribaServicoFacade eSF = EscribaServicoFacade.novaSessao(NOME_USUARIO, SENHA);
 EscribaServicoFacade.criarDiretorio(CAMINHO, NOME_PASTA);
 EscribaServicoFacade.salvaSessao();
} catch (Exception e) {
        //Tratamento da Exception
} finally {
        try {
        //Fechamento da sessão
         if(eSF != null && eSF.hasSessao())
          eSF.fechaSessao();
        } catch (Exception e) {}
}
```

3.1 A Classe EscribaServicoFacade

Para usar a classe EscribaServicoFacade, basta importar os pacotes indicados nesse manual, e fazer as chamadas dos métodos da classe EscribaServicoFacade diretamente após a obtenção da instância da mesma.

A classe EscribaServicoFacade está localizada no pacote:
gov.pr.celepar.escriba.client.EscribaServicoFacade.

Dica: Consulte o anexo deste manual para saber todos os métodos disponíveis na classe EscribaServicoFacade, EscribaNode e EscribaUtil.

4 DOCUMENTAÇÃO DE TAGLIB

4.1 Conceitos

Taglib é uma biblioteca de tags preparada para executar uma ação predeterminada. No caso do ScribaPinhão, existe a taglib "***<escriba:navegacao>***" que tem por finalidade disponibilizar uma estrutura de navegação dentro do repositório do ScribaPinhão.

4.2 Roteiro de Utilização da taglib <escriba:navegacao>

A utilização da taglib envolve a configuração do ambiente e inclusão da tag na(s) página(s) relacionadas. Abaixo serão apresentados os passos para configurar o ambiente:

1. É necessário configurar um servlet no arquivo web.xml que fica no diretório "context/WEB-INF" com as seguintes tags, que devem ser colocadas **antes** da configuração do struts.

```
<!-- Servlet para montagem da Navegação ScribaPinhao -->
<servlet>
 <servlet-name>EscribaNavegavaoServlet</servlet-name>
 <servlet-class>
gov.pr.celepar.escriba.client.servlet.NavegacaoServlet
 </servlet-class>
</servlet>
<servlet-mapping>
 <servlet-name>EscribaNavegavaoServlet</servlet-name>
 <url-pattern>/escriba_navegacao.do</url-pattern>
</servlet-mapping>
```

   O arquivo "*escriba_navegacao.do"* é um servlet que está dentro do ScribaPinhão. Para que esse servlet possa ser acessado dentro de uma aplicação que utilize o framework é necessário cadastrar uma função que habilite as permissões dentro do *Sentinela*. O nome "*escriba_navegacao.do*", dado aqui, pode ser um outro conforme desejado para utilização dentro da aplicação.

2. Cadastrar dentro do *Sentinela* o servlet "***escriba_navegacao.do"*** na área de ***funções***

genéricas.

3. Colocar o arquivo "***escriba_dtree.js"*** e o "***prototype.js"*** dentro da aplicação. O local padrão para esse arquivo é o diretório "***context/js/generic"*** .

4. Cadastrar no *Sentinela* os arquivos "***escriba_dtree.js"*** e "***prototype.js"*** na seção de ***excessões*** para o projeto atual.

5. Incluir abaixo da pasta "WEB-INF/context/src" do projeto o arquivo "***scribaServlet.properties"***, cuja descrição de itens segue:

```
1. #tipos de node predefinidos para serem utilizados pelo servlet de
   navegacao
   tipoNode.1=escriba:base
2. #url para montar link de busca de item da arvore de navegacao
   (neste caso o javascript deve saber disso)
   linkExibir.1=/action.do?action=processarRequestParaExibicao
3. #icones default para itens da arvore
   iconeDefault.1=icon_novo.png
4. #icones para item de paginacao da arvore
   iconeMais.1=icon_download.gif
5. #classes utilizadas para seleção de ícone a apresentar no item da
   arvore
   iconeSelectorClass.1=gov.pr.celepar.projeto.???.ClasseImplIconeSele
   ctor
6. #classes utilizadas para filtragem da exibição do item da arvore
   itemFilterClass.1=gov.pr.celepar.projeto.???.ClasseImplFiltroItem
7. #ordenação para os itens da arvore (NÃO IMPLEMENTADO!!!)
   orderBy.1=dcaminho
8. #indica se retorna o parâmetro nome do node (deve ser tratado via
   javascript o nome subtraindo o caminho do caminhoPai e
   descodificando)
   retorna.nome.node=false
9. #indica se retorna o parâmetro caminhoPai do node (deve ser tratado
   via javascript o caminho pai pegando a última porção do caminho
   completo do item após o último '/' (%2F))
   retorna.caminho.pai.node=false
10. #indica o número de registros por pagina da arvore
   qtdePagina=20
11. #indica a classe que implementa interface ItemDownload
   itemDownload=gov.pr.celepar.projeto.???.ClasseImplItemDownload
12.#indica o nome do contexto padrão da aplicação de repositório
   Scriba a ser acessada pela aplicação. Utilize no caso da aplicação
   hospedeira somente necessitar de uma única fonte de repositório. De
   qualquer maneira a aplicação hospedeira deve prever que o servlet
   vai renderizar itens com o caminho tendo a precedência deste nome
   de contexto, por ex: @gtf-escriba@/pai/filho, ou @gtf-
   escriba@12A55-....(uuid). A aplicação deve retirar este início
   através da chamada a classe
   EscribaUtil.retiraContextoDeCaminho(String caminho)
```

```
nomeContextoScriba=gtf-escriba
```

A seguir é ilustrado como utilizar a taglib dentro de uma página html. Da mesma forma que anteriormente, abaixo estão os passos a seguir:

1. Incluir a chamada à taglib do ScribaPinhão:

```
<!-- inclui a taglib do ScribaPinhão -->
<%@ taglib uri="http://celepar.pr.gov.br/taglibs/escriba.tld" prefix="escriba" %>
```

2. Incluir os arquivos ***escriba_dtree.js*** e ***prototype.js*** no cabeçalho da página.

   Nota: O arquivo *prototype.js* deve ser o fornecido pelo ScribaPinhao.

```
<!--define local onde estão os javascript -->
<c:url var="script_dtree" value="/js/generic/escriba_dtree.js" />
<c:url var="script_prototype" value="/js/generic/prototype.js" />
<!--utiliza variáveis c:url para que o contexto do servidor seja adicionado automaticamente ao
caminho informado na variável -->
<script type="text/JavaScript" src="${script_dtree}"></script>
<script type="text/JavaScript" src="${script_prototype}"></script>
```

3. Chamar a taglib dentro da página (exemplo):

```
<escriba:navegacao    nomeRaiz="Repositório"
                      caminho="/"
                      linkExibir="${link_exibir}"
                      iconeDefault="${icon_default}"
                      divTitulo="item_titulo"
                      divConteudo="item"
                      qtdePagina="20"
                      iconeMais="${link_exibir}"
              iconeSelectorClass="gov.pr.celepar.escriba.iconeSelector.EscribaIconeSelector">
       <escriba:ícone tipo="escriba:base" src="${icon_escriba_base}" />
       <escriba:ícone tipo="escriba:arquivo" src="${icon_escriba_arquivo}" />
       <escriba:ícone tipo="escriba:pasta" src="${icon_escriba_pasta}" />
</escriba:navegacao>
```

Alguns dos parâmetros da taglib podem ser indicados por padrão no arquivo scribaServlet.properties citado anteriormente, e todos são explicados a seguir:

nomeRaiz: Nome que será colocado como Raiz da árvore de diretórios. Pode ser deixado vazio.

linkRaiz: Link a ser utilizado para o rótulo do raiz do repositório.

caminho: Localização do diretório raiz a ser exibido.

Ex: /Lattes torna o diretório /Lattes o raiz da árvore exibida.

tipoNode: Tipo de nó a ser utilizado como filtro do repositório (padrão = escriba:base). Obs: é possível criar novos tipos de nó e aumentar o número de opções existentes.

linkExibir: Link para ativar chamada de exibição de conteúdo. Utilizado para definir qual action da aplicação irá receber a tupla "link + caminho do item" e retornar com o conteúdo apropriado.

iconeDefault: Link para o ícone padrão a ser exibido para o item a ser apresentado na pagina html.

divTitulo: Id da tag div na página html que irá exibir o título do item. O título é sempre renderizado como sendo o nome do item selecionado.

divConteudo: Id da tag div que irá conter o conteúdo do item a ser exibido. O conteúdo a ser renderizado é o resultado da chamada ao link encontrado no parâmetro linkExibir.

qtdePagina: Número de itens a serem mostrados em caso de paginação (quando houverem mais itens que o estipulado).

iconeMais: Ícone a ser exibido quando houver paginação.

servlet: Endereço onde o servlet EscribaNavegacao está mapeado. O padrão é "*escriba_navegacao.do*".

iconeSelectorClass: Classe que implementa a interface IconeSelector para selecionar qual ícone será renderizado. Essa classe permite diferenciar o tipo de ícone que será exibido de acordo com o tipo de nodo. É necessário implementar uma classe para renderizar o ícone para cada nó que será exibido na tela. Pode ser substituída por um número correspondente a indicação no arquivo scribaServlet.properties.

itemFilterClass: Classe que implementa a interface ItemFilter que irá filtrar os itens que não devem ser exibidos. Pode ser substituída por um número correspondente a indicação

no arquivo scribaServlet.properties. Caso a classe retorne **TRUE** o item ***NÃO será exibido***.

Nota Importante: É possível passar parâmetros para a classe ItemFilter utilizando a seguinte sintaxe:

Ex: itemFilterClass="classe@param1,param2,param3".

Onde classe refere-se ao nome da classe e param1, param2 e param3 são os parâmetros a serem passados para a classe. Essa característica se torna importante por permitir a definição dinâmica dos filtros a serem aplicados (usando variáveis como parâmetros).

Dentro da taglib escriba:navecacao, podemos utilizar outra taglib, escriba:icone que irá definir os ícones que serão mostrados de acordo com o tipo de nó carregado. Os parâmetros desta taglib são:

tipo: Indica o tipo do nó que utilizará a imagem especificada (escriba:base, escriba:arquivo, escriba:pasta, etc...).

src: Caminho da imagem a ser exibida.

5 ANEXOS

5.1 Métodos da classe – EscribaFacade

gov.pr.celepar.escriba.client
Class EscribaFacade
java.lang.Object
gov.pr.celepar.escriba.client.EscribaFacade
CLASSE DEPRECIADA NA VERSÃO 1.4. UTILIZAR CLASSE ESCRIBASERVICOFACADE a partir de agora.
`public class EscribaFacade extends java.lang.Object`

5.2 Métodos da classe – EscribaServicoFacade

gov.pr.celepar.escriba.client
Class EscribaFacade
java.lang.Object
gov.pr.celepar.escriba.client.EscribaServicoFacade

`public class EscribaServicoFacade extends java.lang.Object`

Since: 1.4

- Possibilita que a aplicação cliente se utilize de uma instância desta classe para acesso a uma fonte de repositório ScribaPinhao, sendo assim várias instâncias podem ser utilizadas pela aplicação, cada uma obtendo serviços de um repositório ScribaPinhao distinto.

Métodos herdados da classe java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalhes dos Métodos

getInstance

`public static EscribaServicoFacade getInstance(java.lang.String nomeContextoScribaServer)`

Devolve instância de EscribaServicoFacade preparada para acessar serviços no servidor Scriba de nomeContextoScribaServer informado. Esse parâmetro corresponde ao nome do arquivo ".war" (sem informar a extensão) referente a aplicação de repositório ScribaPinhão.

Parâmetros:
`nomeContextoScribaServer` – Deve ser informada o nome da instância da aplicação

ScribaPinhao a ser acessada, com isso internamente será recuperada a url correspondente aos serviços Web, conforme configurado na aplicação administradora do ScribaPinhao.

---

novaSessao

```
public static void novaSessao(java.lang.String usuario,
 java.lang.String senha) throws java.lang.Exception
```

Cria uma nova sessão com o repositório do Escriba. Caso exista uma sessão aberta este método retornará um Exceção

Parâmetros:
usuario – String com o login
senha – não utilizado atualmente, informar "".
Throws: java.lang.Exception

---

hasSessao

```
public static boolean hasSessao()
```

Este método verifica se já existe uma sessão aberta.

Returns: boolean

---

fechaSessao

```
public static void fechaSessao() throws java.lang.Exception
```

Fecha a sessão que foi aberta pelo método novaSessao.

Throws: java.lang.Exception

---

salvaSessao

```
public static void salvaSessao() throws java.lang.Exception
```

Persiste as alterações feitas pelas chamadas do EscribaFacade na sessão.

Throws: java.lang.Exception

---

getAtributo

```
public static java.lang.Object getAtributo(java.lang.String caminho,
java.lang.String nome) throws java.lang.Exception
```

Busca o valor do atributo de um Node, deve-se passar o caminho/uuid onde o Node se encontra e o nome da propriedade para buscar o atributo. Informando o caminho como path

fica com melhor performance do que com uuid.

Deve ser testado se é devolvido um Object[] pois pode retornar atributo multivalorado.

Parameters:
caminho -
nome -
Returns: Object
Throws:java.lang.Exception

---

getTamanhoAtributo

public static java.lang.Long **getTamanhoAtributo**(java.lang.String caminho, java.lang.String nome) throws java.lang.Exception

Retorna o tamanho do atributo de um Node, geralmente utilizado para buscar o tamanho de dados binários, para não precisar trazer toda a informação do servidor apenas para descobrir o tamanho do arquivo, deve-se passar o caminho/uuid onde o Node se encontra e o nome da propriedade para buscar o tamanho do atributo.

Parameters:
caminho -
nome -
Returns: Long
Throws:java.lang.Exception

---

setAtributos

public static void **setAtributos**(EscribaNode eNode) throws java.lang.Exception

Salva todos os atributos alterados localmente no EscribaNode, e cria os EscribaNodes filho que foram incluídos localmente e não existem no servidor. Este método grava os atributos de todos os EscribaNodes carregados de forma recursiva.

Parameters:
eNode -
Throws:java.lang.Exception

---

setAtributos

public static void **setAtributos**(EscribaNode[] eNodes) throws java.lang.Exception

Salva todos os atributos alterados localmente nos EscribaNodes, e cria os EscribaNodes filho que foram incluídos localmente e não existem no servidor. Este método grava os atributos de

todos os EscribaNodes carregados de forma recursiva. Utilizar caso não seja necessário atribuir valores byte[] em atributos.

Parameters:
eNode -
Throws:java.lang.Exception
Since:1.3

---

getNodeType

public static EscribaNodeType **getNodeType**(java.lang.String nodeTypeName) throws java.lang.Exception

Busca a definição de um tipo específico.

Parameters:
nodeTypeName - Nome da definição do tipo do nó.
Returns: EscribaNodeType
Throws:java.lang.Exception

---

getNodeTypes

public static java.lang.String[] **getNodeTypes**() throws java.lang.Exception

Busca os Tipos de Nodes primários cadastrados no servidor.

Returns: String[]
Throws:java.lang.Exception

---

getMixinTypes

public static java.lang.String[] **getMixinTypes**() throws java.lang.Exception

Busca os Tipos de Nodes "mixin" cadastrados no servidor.

Returns: String[]
Throws:java.lang.Exception

---

getMixinTypes

public static java.lang.String[] **getMixinTypes**(java.lang.String caminho) throws java.lang.Exception

Busca os Tipos de Nodes "mixin" para o node passado como parâmetro.

Parameters:
node -
Returns: String[]
Throws:java.lang.Exception

---

getNodeProperties

```
public static java.lang.String[] getNodeProperties(java.lang.String tipoNode)
throws java.lang.Exception
```

Busca os nomes dos atributos do Tipo do Node Obs: O retorno desse método é um array de String com uma quantidade par de elementos, e os índices i (sendo o i um número par) são os nomes dos atributos e os índices i+1 são os tipos dos atributos i;

Ex: String[] { "prop:ativo", "BOOLEAN", "prop:descricao", "STRING" };
TIPOS A SEREM RETORNADOS: STRING;NAME;PATH;REFERENCE;BINARY;LONG;DOUBLE;DATE;BOOLEAN; NAO DEFINIDO;
A classe EscribaUtil provê os valores int para cada tipo de dado.

Parameters:
sessionId -
nomeNode -
Returns: String[]
Throws: java.lang.Exception

---

getPropriedadesDeTipo

```
public static java.lang.String[] getPropriedadesDeTipo(java.lang.String
tipoNode) throws java.lang.Exception
```

Busca os nomes dos atributos do Tipo do Node Obs: O retorno desse método é um array de String com uma quantidade par de elementos, e os índices i (sendo o i um número par) são os nomes dos atributos e os índices i+1 são os tipos dos atributos i representados por um valor numérico;

Ex: String[] { "prop:ativo", "1", "prop:descricao", "2" };
TIPOS A SEREM RETORNADOS: STRING;NAME;PATH;REFERENCE;BINARY;LONG;DOUBLE;DATE;BOOLEAN; NAO DEFINIDO;

A classe EscribaUtil provê os valores int para cada tipo de dado.

Este método serve como alternativa ao getNodeProperties com a diferença que este retorna os

tipos para serem buscados pela classe EscribaUtil.

Parameters:
sessionId -
nomeNode -
Returns: String[]
Throws:java.lang.Exception

---

getPalavrasNaoIndexadas

public static java.lang.String[] **getPalavrasNaoIndexadas**() throws java.lang.Exception

Retorna array de palavras que estão definidas no repositório para não serem indexadas.

Isso pode auxiliar em alguma funcionalidade da aplicação cliente no caso para o filtro de busca a ser executada.

Returns: String[]
Throws:java.lang.Exception
Since:1.3

---

buscarGruposUsuario

public static java.lang.String[] **buscarGruposUsuario**(java.lang.String caminho, boolean somenteAtivos) throws java.lang.Exception

Busca os uuids dos grupos onde o usuário faz parte, resolvendo os grupos na hierarquia superior.

Caso desejar que somente os grupos ativos sejam retornados deve ser informado false no parâmetro somenteAtivos.

Parameters:
caminho -
somenteAtivos -
Returns: String[]
Throws:java.lang.Exception
Since:1.3

---

criarNode

public static EscribaNode **criarNode**(java.lang.String caminho, EscribaNode eNode) throws java.lang.Exception

Cria um novo EscribaNode no servidor do escriba abaixo do caminho informado e salva as propriedades atribuídas no EscribaNode, caso esse EscribaNode possua filhos este método

cria todos os filhos recursivamente.

Parameters:
caminho -
eNode -
Returns: EscribaNode
Throws:java.lang.Exception

---

renomearNode

public static void **renomearNode**(java.lang.String caminho, java.lang.String nomeAntigo, java.lang.String nomeNovo) throws java.lang.Exception

Altera o nome de um EscribaNode no servidor.

Parameters:
caminho -
nomeAntigo -
nomeNovo -
Throws:java.lang.Exception

---

moverNode

public static void **moverNode**(java.lang.String caminho, java.lang.String nome, java.lang.String caminhoNovo) throws java.lang.Exception

Move o EscribaNode de um local para outro no servidor.

Parameters:
caminho -
nome -
caminhoNovo -
Throws:java.lang.Exception

---

excluirNode

public static void **excluirNode**(java.lang.String caminho) throws java.lang.Exception

Exclui um EscribaNode do servidor.

Parameters:
caminho -
Throws:java.lang.Exception

---

importaZip

public static void **importaZip**(java.lang.String caminho, byte[] data) throws java.lang.Exception

Importa uma estrutura de arquivos e pastas dentro de um arquivo ZIP para o escriba, este

método cria Nodes do tipo escriba:pasta para os diretórios e escriba:arquivo para os arquivos, gravando também o seu conteúdo.

Parameters:
caminho -
data -
Throws:java.lang.Exception

---

importaZip

```
public static void importaZip(java.lang.String caminho, java.io.InputStream is) throws java.lang.Exception
```

Importa uma estrutura de arquivos e pastas dentro de um arquivo ZIP para o escriba, este método cria Nodes do tipo escriba:pasta para os diretórios e escriba:arquivo para os arquivos, gravando também o seu conteúdo.

Parameters:
caminho -
fis -
Throws:java.lang.Exception

---

criarDiretorio

```
public static EscribaNode criarDiretorio(java.lang.String caminho, java.lang.String nome) throws java.lang.Exception
```

Cria um Node do tipo escriba:pasta no servidor.

Parameters:
caminho -
nome -
Returns: EscribaNode
Throws:java.lang.Exception

---

criarDiretorio

```
public static EscribaNode criarDiretorio(java.lang.String caminho, java.lang.String nome, java.lang.String[] mixinTypes) throws java.lang.Exception
```

Cria uma pasta no repositório (node do tipo escriba:pasta).

Parameters:
caminho -
nome -
mixinTypes -
Returns: EscribaNode
Throws:java.lang.Exception

criarArquivo

```
public static EscribaNode criarArquivo(java.lang.String caminho, java.lang.String nome, java.lang.String mimeType, java.lang.String encoding, byte[] data) throws java.lang.Exception
```

Cria um Node do tipo escriba:arquivo no servidor, gravando as propriedades do arquivo e o seu conteúdo.

Parameters:
caminho -
nome -
mimeType -
encoding -
data -
Returns: EscribaNode
Throws: java.lang.Exception

---

criarArquivo

```
public static EscribaNode criarArquivo(java.lang.String caminho, java.lang.String nome, java.lang.String mimeType, java.lang.String encoding, byte[] data, java.lang.String[] mixinTypes) throws java.lang.Exception
```

Cria um node do tipo escriba:arquivo no repositório

Parameters:
caminho -
nome -
mimeType -
encoding -
data -
Returns: EscribaNode
Throws: java.lang.Exception

---

criarArquivo

```
public static EscribaNode criarArquivo(java.lang.String caminho, java.lang.String nome, java.lang.String mimeType, java.lang.String encoding, java.io.InputStream is) throws java.lang.Exception
```

Cria um Node do tipo escriba:arquivo no servidor, gravando as propriedades do arquivo e o seu conteúdo.

Parameters:
caminho -
nome -
mimeType -
encoding -
is -

Returns: EscribaNode
Throws: java.lang.Exception

---

criarArquivo

```
public static EscribaNode criarArquivo(java.lang.String caminho, java.lang.String nome, java.lang.String mimeType, java.lang.String encoding, java.io.InputStream is, java.lang.String[] mixinTypes) throws java.lang.Exception
```

Cria um node do tipo escriba:arquivo.

Parameters:
caminho -
nome -
mimeType -
encoding -
fis -
Returns: EscribaNode
Throws: java.lang.Exception

---

gravarAssinatura

```
public static void gravarAssinatura(java.lang.String caminho, byte[] data) throws java.lang.Exception
```

Grava Assinatura Digital no item do repositório

Parameters:
caminho - caminho do item no repositório
data - retorno de assinatura pego do Tabelião
Throws: java.lang.Exception

---

gravarAssinatura

```
public static void gravarAssinatura(java.lang.String caminho, java.io.InputStream is) throws java.lang.Exception
```

Grava Assinatura Digital no item do repositório

Parameters:
caminho - caminho do item do repositório
is - retorno de assinatura pego do Tabelião
Throws: java.lang.Exception

---

existeEscribaNode

```
public static boolean existeEscribaNode(java.lang.String caminho) throws java.lang.Exception
```

Verifica a existência do node possuidor do caminho (mais rápido) ou do uuid informado.

Parameters:
caminho -
Returns:boolean
Throws:java.lang.Exception

---

existeEscribasNodes

public static java.lang.Boolean[] **existeEscribasNodes**(java.lang.String[] caminhos) throws java.lang.Exception

Verifica a existência do node possuidor do caminho (mais rápido) ou do uuid informado e retorna array de boolean com a mesma seqüência, informando para cada posição se o node existe ou não.

Parameters:
caminho - array com caminho/uuid dos nodes a serem verificados.
Returns:boolean
Throws:java.lang.Exception
Since:1.3

---

buscarEscribaNode

public static EscribaNode **buscarEscribaNode**(java.lang.String caminho) throws java.lang.Exception

Busca um EscribaNode pelo seu caminho ou uuid.

Parameters:
caminho -
Returns: EscribaNode
Throws:java.lang.Exception

---

buscarEscribasNodes

public static EscribaNode[] **buscarEscribasNodes**(java.lang.String[] caminhos) throws java.lang.Exception

Busca EscribaNode's pelo seu caminho ou uuid conforme posição do array recebido.

Retorna seqüência de escribaNode que pode ser diferente da seqüência de caminhos passados caso algum caminho não exista.

Caso um dos caminhos informados não exista será lançada exceção por isso a aplicação cliente deve saber da existência dos itens com antecedência.

Parameters:
caminhos[] -
Returns: EscribaNode

Throws: java.lang.Exception
Since: 1.3

---

getAtributos

```
public static java.lang.Object[] getAtributos(java.lang.String[] nomes, java.lang.String caminho) throws java.lang.Exception
```

Retorna o valor dos atributos de nomes informados, presentes no node de caminho/uuid informado.

A seqüência de retorno de valores se mantém conforme a seqüência de nomes informada.

O chamador deste método deve conhecer os tipos esperados dos valores de cada atributo.

Informando o caminho como path fica com melhor performance do que com uuid.

Deve ser testado se é devolvido um Object[] em uma posição do array de retorno, pois pode retornar atributo multivalorado.

De qualquer forma a aplicação chamadora deve ter conhecimento sobre o domínio de retornos possíveis por atributo.

Não se aconselha buscar propriedades BINARY com este método, utilize o método getAtributo para isso.

Parameters:
nomes[] -
caminho -
Returns: EscribaNode
Throws: java.lang.Exception
Since: 1.3

---

getAtributosPorCaminhos

```
public static java.lang.Object[] getAtributosPorCaminhos(java.lang.String[] nomes, java.lang.String[] caminhos) throws java.lang.Exception
```

Retorna o valor dos atributos de nomes informados, presentes nos nodes de caminho/uuid informados.

A seqüência de retorno de valores se mantém conforme a seqüência de nomes informada para cada node presente nos caminhos passados.

Por exemplo: nome{"prop:nome","jcr:uuid"} e caminho{"/A","/B"}, o retorno será: {"nomeA","uuidA","nomeB","uuidB"}.

O chamador deste método deve conhecer os tipos esperados dos valores de cada atributo.

Informando o caminho como path fica com melhor performance do que com uuid.

Deve ser testado se é devolvido um Object[] em uma posição do array de retorno, pois pode retornar atributo multivalorado.

De qualquer forma a aplicação chamadora deve ter conhecimento sobre o domínio de retornos possíveis por atributo.

Não se aconselha buscar propriedades BINARY com este método, utilizar o método getAtributo para isso.

Não existindo alguma propriedade vai gerar exceção.

Parameters:
nomes[] -
caminhos[] -
Returns: EscribaNode
Throws: java.lang.Exception
Since: 1.3

---

addMixinsSetAtributosPorCaminhos

```
public static void addMixinsSetAtributosPorCaminhos(java.lang.String[] caminhos, java.lang.String[] mixins, java.lang.String[] atributos, java.lang.Object[] valores) throws java.lang.Exception
```

Efetua a adição dos tipos mixins fornecidos e dos atributos com seus respectivos valores em cada item localizado nos caminhos fornecidos.

A aplicação deve se certificar da existência dos itens pois caso algum não seja retornado vai ser lançada exceção.

Para que essa adição tenha efeito a sessão deve ser salva.

Parameters:
caminhos[] - array de caminhos de itens existentes no repositório
mixins[] - array de nomes de tipos mixins a serem atribuídos nos itens
atributos[] - nomes dos atributos a serem setados nos itens (em decorrência da atribuição dos mixins), se for null somente vai atribuir os mixins nos caminhos.
valores[] - valor dos atributos. A ordem dos valores deve seguir a ordem dos atributos para cada caminho de item.
ex:
caminhos{"/A","/B"},mixin{"tipo:mixin"},atributos{"prop:um","prop:dois"},valores{valorPropUmA,valorPropDoisA,valorPropUmB,valorPropDoi

sB}

Throws: java.lang.Exception

Since: 1.3

---

addMixinsSetAtributosPorXPath

```
public static void addMixinsSetAtributosPorXPath(java.lang.String xPath, java.lang.String[] mixins, java.lang.String[] atributos, java.lang.Object[] valores) throws java.lang.Exception
```

Efetua a adição dos tipos mixins fornecidos e dos atributos com seus respectivos valores em cada item retornado pela expressão de busca.

Para que essa adição tenha efeito a sessão deve ser salva.

Parameters:

xPath - expressão xpath de busca válida
mixins[] - array de nomes de tipos mixins a serem atribuídos nos itens
atributos[] - nomes dos atributos a serem setados nos itens (em decorrência da atribuição dos mixins), se for null somente vai atribuir os mixins
valores[] - valor dos atributos na mesma ordem dos atributos. Deve ser informado um valor para cada atributo, e os valores serão aplicados em todos itens retornados.

Throws: java.lang.Exception

Since: 1.3

---

pesquisarCountRepositorioPorXPath

```
public static java.lang.Integer pesquisarCountRepositorioPorXPath(java.lang.String xpath) throws java.lang.Exception
```

Faz uma pesquisa e retorna a quantidade de registros encontrados pela linguagem de busca XPATH.

Parameters:

sql -

Returns: Integer

Throws: java.lang.Exception

---

pesquisarRepositorioPorXPath

```
public static EscribaNode[] pesquisarRepositorioPorXPath(java.lang.String xpath) throws java.lang.Exception
```

Faz uma pesquisa no repositório pela linguagem de busca XPATH.

Parameters:

xpath -
Returns: EscribaNode[]
Throws:java.lang.Exception

---

pesquisarRepositorioPorXPath

public static EscribaNode[] **pesquisarRepositorioPorXPath**(java.lang.String xpath, java.lang.String[] flags) throws java.lang.Exception

Faz uma pesquisa no repositório pela linguagem de busca XPATH utilizando algumas pós-execuções definidas em flags.

As flags podem ser: (se for enviar ordenação no flag, colocar em posição anterior ao limit no array para o correto funcionamento):

- order by dcaminho : (Não é possível ordenar diretamente por caminho -@jcr:path- na expressão xpath, aqui no caso por ordem decrescente)

- order by caminho : (Não é possível ordenar diretamente por caminho -@jcr:path- na expressão xpath)

- order by nome : (Não é possível ordenar diretamente por nome -@jcr:name- na expressão xpath, preferível utilizar a propriedade @prop:nome no order by da expressão)

- limit x, y : (Indica a paginação (x=nro registro inicial, y=nro registro final), se somente informado essa flag,a ordenação ocorrerá diretamente pela busca -melhor performance-, caso contrário o Scriba fará varredura nos registros retornados).

Parameters:
xpath -
flags -
Returns: EscribaNode[]
Throws:java.lang.Exception

---

pesquisarCountRepositorioPorSql

public static java.lang.Integer **pesquisarCountRepositorioPorSql**(java.lang.String sql) throws java.lang.Exception

Faz uma pesquisa e retorna a quantidade de registros encontrados pela linguagem de busca SQL

Parameters:

sql -
Returns: Integer
Throws: java.lang.Exception

---

pesquisarRepositorioPorSql

```
public static EscribaNode[] pesquisarRepositorioPorSql(java.lang.String sql)
throws java.lang.Exception
```

Faz uma pesquisa no repositório pela linguagem de busca SQL

Parameters:
sql -
Returns: EscribaNode[]
Throws: java.lang.Exception

---

pesquisarRepositorioPorSql

```
public static EscribaNode[] pesquisarRepositorioPorSql(java.lang.String sql,
java.lang.String[] flags) throws java.lang.Exception
```

Faz uma pesquisa no repositório pela linguagem de busca SQL utilizando algumas pós-execuções definidas em flags As flags podem ser: (se for enviar ordenação no flag, colocar em posição anterior ao limit no array para o correto funcionamento)

- order by dcaminho : (Não é possível ordenar diretamente por caminho -jcr:path- na expressão sql, aqui no caso por ordem decrescente)

- order by caminho : (Não é possível ordenar diretamente por caminho -jcr:path- na expressão sql)

- order by nome : (Não é possível ordenar diretamente por nome -jcr:name- na expressão sql, preferível utilizar a propriedade prop:nome no order by da expressão)

- limit x, y : (Indica a paginação (x=nro registro inicial, y=nro registro final), se somente informado essa flag, a ordenação ocorrerá diretamente pela busca -melhor performance-, caso contrário o Scriba fará varredura nos registros retornados).

Parameters:
sql -
Returns: EscribaNode[]
Throws: java.lang.Exception

---

lockEscribaNode

```
public static void lockEscribaNode(java.lang.String caminho, boolean isDeep)
throws java.lang.Exception
```

Deixa o node em estado de lock no repositório.

Parameters:
caminho -
isDeep -
Throws: java.lang.Exception

---

lockEscribaNode

```
public static void lockEscribaNode(java.lang.String caminho, boolean isDeep,
long timeout) throws java.lang.Exception
```

Deixa o node em estado de lock no repositório.

Parameters:
caminho -
isDeep -
timeout -
Throws: java.lang.Exception

---

unlockEscribaNode

```
public static void unlockEscribaNode(java.lang.String caminho) throws
java.lang.Exception
```

Retira estado de lock de node no repositório.

Parameters:
caminho -
Throws: java.lang.Exception

---

addMixinType

```
public static void addMixinType(java.lang.String caminho, java.lang.String
mixinType) throws java.lang.Exception
```

Adiciona uma propriedade de MixinType a um Node.

Parameters:
caminho -
mixinType -
Throws: java.lang.Exception

---

removeMixinType

```
public static void removeMixinType(java.lang.String caminho, java.lang.String
mixinType) throws java.lang.Exception
```

Remove um MixinType de um Node.

Parameters:
caminho -
mixinType -
Throws:java.lang.Exception

---

removeMixinsTypesDeCaminhos

```
public static void removeMixinsTypesDeCaminhos(java.lang.String[] caminhos, java.lang.String[] mixinTypes) throws java.lang.Exception
```

Remove MixinTypes informados de itens encontrados nos caminhos passados.

Os itens devem existir bem como os mixins a serem removidos, senão lança exceção.

Parameters:
caminhos - String[]
mixinTypes - String[]
Throws:java.lang.Exception
Since:1.3

---

removeMixinsTypesPorXPath

```
public static void removeMixinsTypesPorXPath(java.lang.String xPath, java.lang.String[] mixinTypes) throws java.lang.Exception
```

Remove MixinTypes informados de itens encontrados pela expressão de busca xpath.

Parameters:
xPath -
mixinTypes - String[]
Throws:java.lang.Exception
Since:1.3

---

reRegistraNodeTypes

```
public static void reRegistraNodeTypes(byte[] data) throws java.lang.Exception
```

Registra um conjunto de Tipos de Nodes existentes e atualiza os já registrados na base do escriba, utilizando anexos para o envio de dados.

Parameters:
data -
Throws:java.lang.Exception

---

registraNodeTypes

```
public static void registraNodeTypes(byte[] data) throws java.lang.Exception
```

Registra um conjunto de Tipos de Nodes não existentes no escriba e que estão definidos no arquivo XML passado como parâmetro, utilizando anexo.

Parameters:
data -
Throws:java.lang.Exception

---

buscaNodesComPermissaoPorUsuarioUUID

public static EscribaNode[] **buscaNodesComPermissaoPorUsuarioUUID**(java.lang.String uuid) throws java.lang.Exception

Busca todos os Nodes que possuem permissões que estão referenciando o uuid do usuário passado como parâmetro. Não significa que exista nível maior que zero definido de permissão para o uuid.

Parameters:
uuid -
Returns: EscribaNode[]
Throws:java.lang.Exception
Since:1.3

---

buscaNodesComPermissaoPorGrupoUUID

public static EscribaNode[] **buscaNodesComPermissaoPorGrupoUUID**(java.lang.String uuid) throws java.lang.Exception

Busca todos os Nodes que possuem permissões que estão referenciando o uuid do grupo passado como parâmetro. Não significa que exista nível maior que zero de permissão definida.

Parameters:
uuid -
Returns: EscribaNode[]
Throws:java.lang.Exception
Since:1.3

---

checkIn

public static void **checkIn**(java.lang.String caminho) throws java.lang.Exception

Efetua checkIn do node.

Parameters:
caminho -
Throws:java.lang.Exception

---

checkInPorCaminhos

```
public static void checkInPorCaminhos(java.lang.String[] caminhos) throws
java.lang.Exception
```

Efetua checkIn dos nodes presentes nos caminhos (ou uuid).

Os caminhos devem ser existentes senão gera exceção.

Parameters:
caminho[] -
Throws:java.lang.Exception
Since:1.3

---

checkInPorXPath

```
public static void checkInPorXPath(java.lang.String xPath) throws
java.lang.Exception
```

Efetua checkIn dos nodes retornados pela expressão de busca fornecida.

A expressão de busca deve ser válida.

Parameters:
xPath -
Throws:java.lang.Exception
Since:1.3

---

checkOutPorCaminhos

```
public static void checkOutPorCaminhos(java.lang.String[] caminhos) throws
java.lang.Exception
```

Efetua checkout dos Nodes de caminhos (ou uuid) informados. Os caminhos devem ser existentes senão gera exceção.

Parameters:
caminhos[] -
Throws:java.lang.Exception
Since:1.3

---

checkOutPorXPath

```
public static void checkOutPorXPath(java.lang.String xPath) throws
java.lang.Exception
```

Efetua checkout dos Nodes retornados pela expressão xpath de busca fornecida.

A expressão de busca deve ser válida.

Parameters:

xPath -
Throws:java.lang.Exception
Since:1.3

---

checkOut

public static void **checkOut**(java.lang.String caminho) throws java.lang.Exception

Efetua checkout do Node.

Parameters:
caminho -
Throws:java.lang.Exception

---

restore

public static void **restore**(java.lang.String caminho, java.lang.String versao) throws java.lang.Exception

Restaura valor do node para a versão existente informada.

Parameters:
caminho -
versao -
Throws:java.lang.Exception

---

restoreBase

public static void **restoreBase**(java.lang.String caminho) throws java.lang.Exception

Restaura valor do node para a última versão gravada do node.

Parameters:
caminho -
Throws:java.lang.Exception

---

restoreBasePorCaminhos

public static void **restoreBasePorCaminhos**(java.lang.String[] caminhos) throws java.lang.Exception

Restaura valor dos nodes presentes nos caminhos informados (devem ser existentes) para a última versão gravada do node.

Parameters:
caminho -
Throws:java.lang.Exception
Since:1.3

---

restoreBasePorXPath

```
public static void restoreBasePorXPath(java.lang.String xPath) throws java.lang.Exception
```

Restaura valor dos nodes presentes nos retornados pela expressão xpath para a última versão gravada do node.

Parameters:
xPath -
Throws:java.lang.Exception
Since:1.3

---

getCaminhoPorUuid

```
public static java.lang.String getCaminhoPorUuid(java.lang.String UUID) throws java.lang.Exception
```

Método para retornar o caminho do node a partir do UUID.

Parameters:
UUID - uuid do node
Returns:retorno - caminho do node com o UUID passado como parâmetro.
Throws:java.lang.Exception

---

getNivelPermissaoUsuario

```
public static int getNivelPermissaoUsuario(java.lang.String caminho) throws java.lang.Exception
```

Método para retornar o nível de permissão do usuário corrente no node do caminho/uuid do node.

Parameters:
caminho - caminho do node
Returns:retorno - permissões do pai do node
Throws:java.lang.Exception

---

tratarCodificacaoTextoPesquisa

```
public static java.lang.String tratarCodificacaoTextoPesquisa (java.lang.String texto) throws java.lang.Exception
```

Método para compatibilizar a codificação do texto utilizado na pesquisa com o jackrabbit.

(evitar interpretação errada de caracteres especiais, útil para indicação de path).

Parameters:
xpath -
Returns: String

Throws: `java.lang.Exception`

---

listarNameSpaces

```
public static java.lang.String[] listarNameSpaces() throws java.lang.Exception
```

Retorna array de String contendo namespaces contidos no repositório, a cada par o prefixo e uri relativos a um name space.

Ex: [prefixoNS1 ; uriNS1 ; prefixoNS2 ; uriNS2]

Returns: `String[]`
Throws: `java.lang.Exception`

---

registrarNameSpaces

```
public static void registrarNameSpaces(java.lang.String[] nameSpaces) throws java.lang.Exception
```

Registra em workspace da sessão no repositório, os namespaces contidos no array de String (prefixo e uri).

Informar no padrão: [prefixoNS1 ; uriNS1 ; prefixoNS2 ; uriNS2].

Parameters:
nameSpaces -
Throws: `java.lang.Exception`

---

sincronizarUsuarios

```
public static void sincronizarUsuarios() throws java.lang.Exception
```

Utilizado para disparar rotina de sincronização de estrutura de usuários do Scriba com a fonte de dados de usuários definida.

Deve ser disparada por usuário definido como root caso contrário vai gerar exceção por não poder criar/alterar estrutura de usuários.

Throws: `java.lang.Exception`

---

getDataAtualServidorScriba

```
public static java.util.Calendar getDataAtualServidorScriba() throws java.lang.Exception
```

Retorna a data (Calendar) atual do servidor do Scriba.

Returns: java.util.Calendar
Throws: java.lang.Exception
Since: 1.3

5.3 Métodos da classe – EscribaNode

gov.pr.celepar.escriba.client
Class EscribaNode
java.lang.Object
  gov.pr.celepar.escriba.client.EscribaNode

Interfaces Implementadas:
java.io.Serializable

```
public class EscribaNode extends java.lang.Object implements java.io.Serializable
```

Since: 1.3

- Retirado supressão de avisos de compilação onde possível em métodos
- Especificado tipo de objetos Map para String,Object.
- Reorganizado separação entre métodos públicos e privados.
- Adicionado métodos getNomeCriadorEAlteradoPor, getAtributos, getEscribasNodes, getAtributosServer, getNodesServer.

Since: 1.4

- Incluído atributo EscribaServicoFacade que se trata da instância utilizada no momento da recuperação do EscribaNode do repositório ScribaPinhao. A aplicação com isso consegue fazer operações com este EscribaNode ciente de onde ele foi obtido.

Métodos herdados da classe java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Detalhes do Construtor

EscribaNode

```
public EscribaNode()
```

EscribaNode

```
public EscribaNode(byte[] data) throws java.lang.Exception
```

Construtor do EscribaNode.

Parameters:
data -
Throws: java.lang.Exception

EscribaNode

public **EscribaNode**(java.io.InputStream is) throws java.lang.Exception

Construtor do EscribaNode.

Parameters:
data -
Throws:java.lang.Exception

EscribaNode

public **EscribaNode**(java.util.Map<java.lang.String,java.lang.Object> map) throws java.lang.Exception

Construtor do EscribaNode.

Parameters:
map -
Throws:java.lang.Exception

EscribaNode

public **EscribaNode**(java.lang.String nome, java.lang.String tipo) throws java.lang.Exception

Construtor do EscribaNode.

Parameters:
nome -
tipo -
Throws:java.lang.Exception

Detalhes dos Métodos

getUsuario

public java.lang.String **getUsuario**()

Retorna o usuário que fez a solicitação do EscribaNode.

Returns: String

---

getESF

public EscribaServicoFacade **getESF**()

Retorna a instância do EscribaServicoFacade que recuperou o EscribaNode do repositório ScribaPinhão que fez a solicitação do EscribaNode.

Returns: String

---

getNome

```
public java.lang.String getNome()
```

Retorna o nome do EscribaNode.

Returns: String

---

getTipo

```
public java.lang.String getTipo()
```

Retorna o tipo do EscribaNode.

Returns: String

---

getMixinTypes

```
public java.lang.String[] getMixinTypes() throws java.lang.Exception
```

Retorna nomes de tipo mixin presentes no EscribaNode.

Returns: String[]
Throws:java.lang.Exception

---

getPath

```
public java.lang.String getPath()
```

Retorna o caminho no servidor do EscribaNode.

Returns: String

---

getNomeCriador

```
public java.lang.String getNomeCriador()
```

Retorna o nome do criador do node no repositório.

Returns: String

---

getNomeAlteradoPor

```
public java.lang.String getNomeAlteradoPor()
```

Retorna o nome do último alterador do node no repositório.

Returns: String

---

getNomeCriadorEAlteradoPor

```
public java.lang.String[] getNomeCriadorEAlteradoPor()
```

Retorna na primeira posição o nome do usuário criador e na segunda posição o nome do usuário que alterou o node.

Returns: String[]
Since:1.3

---

isCheckedOut

```
public java.lang.Boolean isCheckedOut()
```

Verifica situação referente a checkin/checkout do node.

Returns: Boolean

---

setCheckedOut

```
public void setCheckedOut(java.lang.Boolean checkedOut)
```

Define situação de checkin/checkout do node.

Parameters:
checkedOut -

---

isLocked

```
public java.lang.Boolean isLocked()
```

Verifica situação de lock do node.

Returns: Boolean

---

setLocked

```
public void setLocked(java.lang.Boolean locked)
```

Define situação de lock do Node.

Parameters:

locked -

---

addAtributoServer

```
public void addAtributoServer(java.lang.String nome, boolean valor)
```

Adiciona atributos que existem na parte Server, este método não deve ser chamado pela parte Client. Utilizado para controle de informações em CACHE.

Parameters:
nome -
valor -

---

getAtributos

```
public java.lang.String[] getAtributos()
```

Retorna os nomes dos atributos existentes no EscribaNode.

Returns: String[]

---

setAtributo

```
public void setAtributo(java.lang.String nome, java.lang.Object valor)
```

Grava ou altera um atributo no EscribaNode.

Parameters:
nome -
valor -

---

getAtributo

```
public java.lang.Object getAtributo(java.lang.String nome) throws java.lang.Exception
```

Retorna o valor de um atributo.

Parameters:
nome -
Returns: Object
Throws: java.lang.Exception

---

getAtributos

```
public java.lang.Object[] getAtributos(java.lang.String[] nomes) throws
```

`java.lang.Exception`

Retorna o valor de atributos do node que sejam informados de uma vez.

Não incluir chamada de atributos de tipo byte[].

Parameters:
nome -
Returns:`Object`
Throws:`java.lang.Exception`
Since:`1.3`

---

isAtributoCarregado

`public boolean isAtributoCarregado(java.lang.String nome)`

Verifica se um atributo está carregado no EscribaNode (CACHE)

Retorna: TRUE – caso o atributo está em memória, ou FALSE – caso o atributo existe mas não está em memória.

Parameters:
nome -
Returns:`boolean`

---

addNodesServer

`public void addNodesServer(java.lang.String nome)`

Adiciona nodes que existem na parte Server, este método não deve ser chamado pela parte Client. Utilizado para controle de informações dos nodes em CACHE.

Parameters:
nome -
valor -

---

getEscribaNodes

`public java.lang.String[] getEscribaNodes()`

Retorna os nomes dos EscribaNodes Filhos.

Returns:`String[]`

---

addEscribaNode

```
public void addEscribaNode(EscribaNode node)
```

Adiciona um EscribaNode Filho.

Parameters:
node -

---

removeEscribaNode

```
public void removeEscribaNode(java.lang.String nome)
```

Remove um EscribaNode Filho.

Parameters:
nome -

---

getEscribaNode

```
public EscribaNode getEscribaNode(java.lang.String nome) throws java.lang.Exception
```

Retorna o EscribaNode Filho.

Parameters:
nome -
Returns: EscribaNode
Throws: java.lang.Exception

---

getEscribasNodes

```
public EscribaNode[] getEscribasNodes(java.lang.String[] nomes) throws java.lang.Exception
```

Retorna os EscribaNodes Filhos.

Parameters:
nome -
Returns: EscribaNode[]
Throws: java.lang.Exception
Since: 1.3

---

getEscribaNodePai

```
public EscribaNode getEscribaNodePai() throws java.lang.Exception
```

Retorna o node pai do EscribaNode no repositório.

Returns: EscribaNode
Throws:java.lang.Exception

---

isEscribaNodeCarregado

```
public boolean isEscribaNodeCarregado(java.lang.String nome)
```

Verifica se um node está carregado no EscribaNode (CACHE)

Retorna: TRUE – caso o node está em memória, ou FALSE – caso o node existe mas não está em memória.

Parameters:
nome -
Returns:boolean

---

getPermissao

```
public java.lang.Integer getPermissao()
```

Retorna a permissão que o usuário que solicitou o node possui sobre o mesmo.

Returns: Integer

---

getPermHerda

```
public boolean getPermHerda() throws java.lang.Exception
```

Verifica a propriedade HERDA do EscribaNode

Returns:boolean
Throws:java.lang.Exception

---

setPermHerda

```
public void setPermHerda(boolean perm) throws java.lang.Exception
```

Altera a propriedade HERDA do EscribaNode. Obs: Para que as alterações realizadas por essa chamada sejam persistidas, o método salvarPermissao() deve ser chamado.

Parameters:
perm -
Throws:java.lang.Exception

---

getPermUUIDUsuarios

```
public java.lang.String[] getPermUUIDUsuarios() throws java.lang.Exception
```

Retorna os UUIDs dos usuários que possuem permissões a este EscribaNode.

O fato de existir um node com o nome do uuid pertencente a algum usuário do repositório, não significa que o mesmo possui algum nível de permissão sobre o node.

Para saber o nível de permissão de algum usuário, com uuid retornado sobre o node, deve ser pego via getPermUsuario.

Returns:`String[]`
Throws:`java.lang.Exception`

---

getNomeUsuario

```
public java.lang.String getNomeUsuario(java.lang.String uuid) throws java.lang.Exception
```

Retorna o nome do usuário referenciado através do UUID do seu node dentro do repositório.

Parameters:
uuid -
Returns:`String`
Throws:`java.lang.Exception`

---

getPermUsuario

```
public int getPermUsuario(java.lang.String uuid) throws java.lang.Exception
```

Retorna a permissão que o usuário UUID possui neste EscribaNode.

Parameters:
uuid -
Returns:`int`
Throws:`java.lang.Exception`

---

setPermUsuario

```
public void setPermUsuario(java.lang.String uuid, int permissao) throws java.lang.Exception
```

Cria ou altera a permissão do usuário UUID Obs: Para que as alterações realizadas por essa chamada sejam persistidas, o método salvarPermissao() deve ser chamado.

Parameters:
uuid -
permissao -
Throws:`java.lang.Exception`

---

removePermUsuario

```
public void removePermUsuario(java.lang.String uuid) throws java.lang.Exception
```

Remove a permissão de um usuário a este EscribaNode.

Parameters:
uuid -
Throws:java.lang.Exception

---

getPermUUIDGrupos

```
public java.lang.String[] getPermUUIDGrupos() throws java.lang.Exception
```

Retorna os UUIDs dos grupos que possuem permissões a este EscribaNode.

Returns: String[]
Throws:java.lang.Exception

---

getNomeGrupo

```
public java.lang.String getNomeGrupo(java.lang.String uuid) throws java.lang.Exception
```

Retorna o nome do grupo referenciado através do UUID.

Parameters:
uuid -
Returns: String
Throws:java.lang.Exception

---

getPermGrupo

```
public int getPermGrupo(java.lang.String uuid) throws java.lang.Exception
```

Retorna a permissão do Grupo através do UUID.

Parameters:
uuid -
Returns:int
Throws:java.lang.Exception

---

setPermGrupo

```
public void setPermGrupo(java.lang.String uuid, int permissao) throws java.lang.Exception
```

Cria ou altera a permissão de um grupo através do UUID Obs: Para que as alterações realizadas por essa chamada sejam persistidas, o método salvarPermissao() deve ser chamado.

Parameters:
grupo -
permissao -
Throws:java.lang.Exception

---

removePermGrupo

```
public void removePermGrupo(java.lang.String uuid) throws java.lang.Exception
```

Remove a permissão de um grupo a este EscribaNode.

Parameters:
uuid -
Throws:java.lang.Exception

---

salvarPermissao

```
public void salvarPermissao() throws java.lang.Exception
```

Salva as alterações feitas nas permissões de usuários e grupos.

Throws:java.lang.Exception

---

saveToMap

```
public java.util.Map<java.lang.String,java.lang.Object> saveToMap() throws java.lang.Exception
```

Converte o EscribaNode para Map.

Returns:
Map
Throws:java.lang.Exception

---

saveToByteArray

```
public byte[] saveToByteArray() throws java.lang.Exception
```

Utilizado para conversão interna pelo objeto.

Throws:java.lang.Exception

---

saveToDataOutputStream

```
public void saveToDataOutputStream(java.io.DataOutputStream dos) throws java.lang.Exception
```

Utilizado para conversão interna pelo objeto.

Throws:java.lang.Exception

loadFromMap

```
public void loadFromMap(java.util.Map<java.lang.String,java.lang.Object> map)
throws java.lang.Exception
```

Carrega as propriedades do EscribaNode de um MAP.

Parameters:
map -
Throws: java.lang.Exception

---

loadFromDataInputStreamPublic

```
public void loadFromDataInputStreamPublic(java.io.DataInputStream dis) throws
java.lang.Exception
```

Utilizado para conversão interna pelo objeto.

Throws: java.lang.Exception

---

mostraNode

```
public void mostraNode(boolean recursivo) throws java.lang.Exception
```

Parameters:
recursivo -
Throws: java.lang.Exception

5.4 Métodos da classe – EscribaUtil

gov.pr.celepar.escriba.client
Class EscribaUtil
java.lang.Object
  gov.pr.celepar.escriba.client.EscribaUtil

```
public class EscribaUtilextends java.lang.Object
```

Versão 1.3

- Criado atributo LOGIN_USUARIO_ANONIMO para uso centralizado por todas as classes do projeto e aplicacões cliente.

Versão 1.4

- Incluído constantes para níveis de permissão.
- Criado método getSuperTipos(String tipo, EscribaServicoFacade eSF) e depreciado getSuperTipos(String tipo).
- Criado método inputStreamToByteArray(InputStream is).
- Alterado método getCaminhoPaiDeCaminho para tratar caso se depare com algo como "@contexto@/", que agora significa o caminho root (/) do repositório no contexto informado.
- Criado getContextoDeCaminho, insereContextoEmCaminho, retiraContextoDeCaminho

Detalhes dos Métodos

nomeDeValor

```
public static java.lang.String nomeDeValor(int tipo)
```

Retorna o nome do tipo especificado.

Parameters:
tipo - o valor do tipo
Returns: String - o nome do tipo especificado.
Throws:java.lang.IllegalArgumentException - se não existir o tipo informado.

---

valorDeNome

```
public static int valorDeNome(java.lang.String nome)
```

Retorna o valor numérico constante do tipo com o nome recebido.

Parameters:
nome - o nome do tipo da propriedade.
Returns:int - o valor numérico constante.
Throws:java.lang.IllegalArgumentException - se o nome informado não

for válido

---

url

```
public static java.lang.String url(java.lang.String url) throws java.io.UnsupportedEncodingException
```

Codifica a string conforme padrão de linguagem do servidor.

Parameters:
url -
Returns: String
Throws:java.io.UnsupportedEncodingException

---

encode

```
public static java.lang.String encode(java.lang.String txt)
```

Codifica a string para padrão html.

Parameters:
txt -
Returns: String

---

getNomeDeCaminho

```
public static java.lang.String getNomeDeCaminho(java.lang.String caminho)
```

Retorna a parte da string recebida de caminho referente a última parte que no caso se trata do nome de um item no repositório.

Parameters:
caminho -
Returns: String

---

getCaminhoPaiDeCaminho

```
public static java.lang.String getCaminhoPaiDeCaminho(java.lang.String caminho)
```

Retorna a parte da string recebida de caminho referente ao nível pai.

Parameters:
caminho -
Returns: String

getMimeType

```
public static java.lang.String getMimeType(java.lang.String nomeArquivo)
```

Retorna o mimeType de acordo com a extensão do nome de Arquivo.

Utilizar este método quando não conseguir descobrir o mimetype pelo header do arquivo.

Parameters:
nomeArquivo -
Returns: String

---

getSuperTipos

```
public static java.util.List<java.lang.String> getSuperTipos(java.lang.String
tipo, EscribaServicoFacade eSF) throws java.lang.Exception
```

Retorna a lista dos supertipos, da definição do tipo com o nome fornecido, presentes em cache local.

Significando que o tipo de nome fornecido herda todas as definições dos tipos constantes na lista de supertipos.

Deve haver sessão aberta com o Repositório para o objeto EscribaServicoFacade passado.

Parameters:
tipo - nome do tipo de node.
ESF – Instância de serviços do repositório ScribaPinhao a recuperar os tipos.
Returns: List<String>
Throws:java.lang.Exception
Since:1.4

---

inputStreamToByteArray

```
public static java.lang.byte[] inputStreamToByteArray(java.io.InputStream iS)
throws java.lang.Exception
```

Retorna byte array contido no iS em chuncks de 1024 bits

Parameters:
iS – InputStream válido
Returns:byte[]
Throws:java.lang.Exception
Since:1.4

---

getContextoDeCaminho

```
public static java.lang.String getContextoDeCaminho(java.lang.String caminho)
throws java.lang.Exception
```

Retorna o contexto da aplicação de repositório ScribaPinhão contido no início do parâmetro de caminho recebido. O padrão esperado para a retirada desta informação é: @contexto@caminho, ou @contexto@uuid.

Parameters:
caminho – String.
Returns: String
Throws:java.lang.Exception
Since:1.4

---

insereContextoEmCaminho

```
public static java.lang.String insereContextoEmCaminho(java.lang.String
caminho, java.lang.String contextoScribaPinhao) throws java.lang.Exception
```

Insere o contexto da aplicação de repositório ScribaPinhão no início do parâmetro de caminho recebido. O padrão para a inclusão desta informação é: @contexto@caminho, ou @contexto@uuid.

Parameters:
caminho – String.
ContextoScribaPinhao – String
Returns: String
Throws:java.lang.Exception
Since:1.4

---

retiraContextoDeCaminho

```
public static java.lang.String retiraContextoDeCaminho(java.lang.String
caminho) throws java.lang.Exception
```

Devolve o caminho sem a informação do contexto da aplicação de repositório ScribaPinhão no início do parâmetro de caminho recebido. O padrão para a retirada desta informação é: @contexto@caminho, ou @contexto@uuid.

Parameters:
caminho – String.
Returns: String
Throws:java.lang.Exception
Since:1.4