



– Porphyre 2002 –

Reading Trails

Conception Client-Serveur

Version 1.0

Référence du document : RH-RTConcCLS-v1.0

Date de création : 17/06/02

Dernière modification : 10/07/02

Etat du document : validé

Nombre de pages : 13

Rédacteur : Rémi HUYNH

Équipe : Informatique

Validation : Chef de Projet

Responsable thèse

SOMMAIRE

1.	INTRODUCTION	3
1.1.	OBJET DU DOCUMENT	3
1.2.	TERMINOLOGIE.....	3
2.	STRUCTURE GENERAL DES CLASSES DE LA GESTION DES PARCOURS DE LECTURE	4
3.	CLASSES DE DONNEES : PACKAGE PORPHYRY.SERVER.CALCUL	6
3.1.	CLASSE STEP	6
3.2.	CLASSE TRAIL	6
4.	CLASSES D'INTERFACE : PACKAGE PORPHYRY.PRESENTER	7
4.1.	CLASSE TRAILBROWSER	7
5.	CLASSES POUR LA COMMUNICATION CLIENT-SERVEUR	9
5.1.	PACKAGE PORPHYRY.VIEW.....	9
5.2.	PACKAGE PORPHYRY.SERVER.USERMANAGEMENT.....	9
5.3.	PACKAGE PORPHYRY.SERVER.CALCUL.....	9
6.	IHM	10
6.1.	CLASSE USER (EXISTANTE)	10
6.2.	PARTIE IHM DE LA CLASSE TRAILBROWSER : PANNEAUX	11
6.3.	PARTIE IHM DU TRAILBROWSER : CONTROLES ET ACTIONS	12
7.	CONCLUSION.....	14

1. Introduction

Le projet Porphyre 2002 est un sous-projet du projet Porphyre qui est un projet d'aide à la consultation de documents scientifiques et de bibliothèque numérique dont la première application est la mise en ligne de la chronique de fouilles de l'Ecole Française d'Athènes.

La version 2002 devra intégrer la gestion d'un parcours de lecture (voir définition ci-après) à l'ensemble Porphyre 2001 existant.

1.1. Objet du document

Le but de ce document est de définir le « COMMENT », c'est à dire ce qu'il faut faire dans le cadre du sous projet Reading Route du projet Porphyre 2002.

L'implantation réseau étant de type trois tiers (client – serveur d'application – serveur de données), on a distingué différentes parties à spécifier dont voici contenant le client et le serveur.

1.2. Terminologie

RMI : Remote Method Invocation, permet la communication entre un serveur d'application et le client

2. Structure générale des classes de la gestion des parcours de lecture

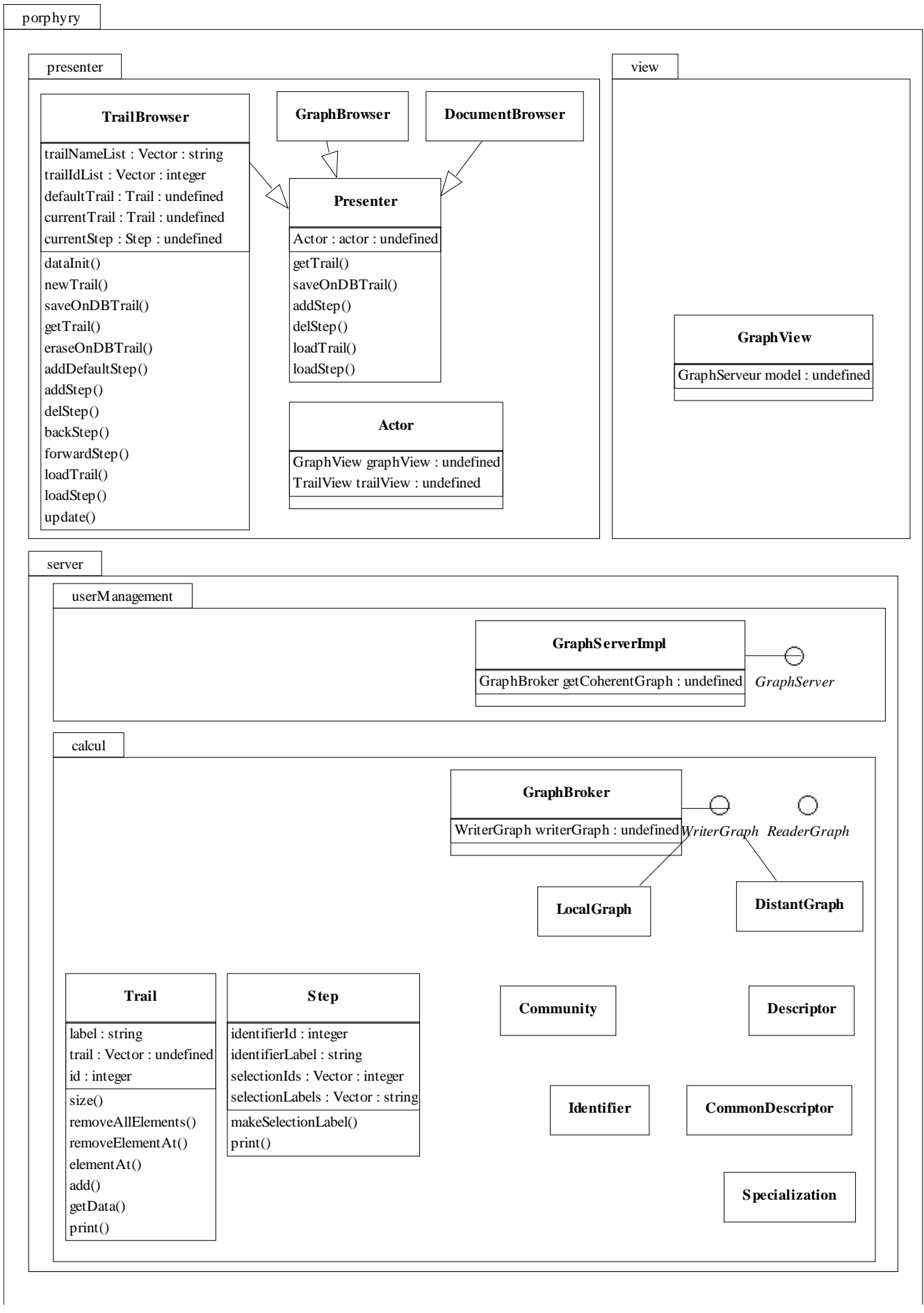
Remarque :

La communication client-serveur reprend le fonctionnement RMI de la communication entre le serveur de graphe et le client natif. Ainsi, la communication entre un nouveau serveur de Trail et le client natif se fera en suivant la même architecture, qui est décrite en page suivante.

Le package `porphyry.presenter` contient toutes les classes d'interface du client Porphyre. Il servira donc pour l'ajout de la classe `TrailBrowser` contenant toute l'interface nécessaire à la gestion des parcours de lecture.

Les packages `porphyry.view` et `porphyry.server` permettront la mise en place de l'architecture RMI.

Sur le schéma ci dessous, pour plus de lisibilité, n'apparaissent pas les méthodes concernant la mise à jour des attributs protégés de chaque classe (`getXXX` et `setXXX`), ainsi que les attributs et méthodes utilisés pour l'interface.



3. Classes de données : Package porphyry.server.calcul

3.1. Classe Step

Attributs :

Type	Nom	Description
protected int	identifieurId	id de l'identifieur pointé par l'étape de lecture
protected String	identifieurLabel	common_label de l'identifieur pointé par l'étape de lecture
protected Vector	selectionIds	Liste des id des descripteur menant au document pointé
protected Vector	selectionLabels	Liste des labels des descripteur menant au document pointé

Notons qu'il n'est pas nécessaire de stocker dans les classes l'ordre des *step*, étant donné qu'ils sont classés dans la classe *Trail* grâce au Vector trail.

Méthodes :

Type	Nom	Description
public String	makeSelectionLabel()	Renvoie la concaténation des labels des descripteurs de la sélection
public void	print()	Permet l'affichage pour le débogage du step

3.2. Classe Trail

Attributs :

Type	Nom	Description
protected String	label	label du trail
protected Vector	trail	Contient les steps contenus dans le trail
protected int	id	id du trail (0 pour le trail par défaut et l'id de la base de données pour les autres)

Méthodes :

Type	Nom	Description
public int	size()	Renvoie le nombre de steps contenus dans le trail
public void	removeElementAt(int)	Permet l'enlèvement d'un step
public void	removeAllElements()	Permet la remise à zéro d'un trail
public Step	elementAt(int)	Permet de récupérer un step
public void	add(Step)	Permet l'ajout d'un élément
public String [][]	getData()	Permet de récupérer les données nécessaires à l'affichage dans le tableau d'un trail
public void	print()	Permet d'afficher les données d'un trail pour débogage





4. Classes d'interface : Package porphyre.presenter

4.1. Classe TrailBrowser

Attributs :





Type	Nom	Description
protected Vector	trailNameList	liste des noms des trails à mettre dans la combobox (commence toujours par DefaultTrail)
protected Vector	trailIdList	liste des ids des trails correspondant à trailNameList (commence toujours par 0)
protected Trail	defaultTrail	Le trail par défaut
protected Trail	currentTrail	Le trail courant
protected Step	currentStep	Le step courant

Méthodes de gestion des parcours :

Type	Nom	Description
protected void	dataInit	- Initialise les données - Crée en local un parcours de lecture par défaut
protected void	 newTrail	- Remise à zéro du trail par défaut
protected void	 saveOnDBTrail	- Affichage d'une fenêtre de dialogue demandant le label du trail - Sauvegarde sur le serveur de données du parcours par défaut avec le label préalablement entré
protected void	 getTrail	- si un document est ouvert : permet la récupération de l'ensemble des {trailId, trailLabel} qui correspondent aux parcours du graphe de l'utilisateur (et de ceux auxquels il est abonné) passant par le même document. - si aucun document n'est ouvert, permet la récupération de tous les trails de l'utilisateur - Mise à jour de la combobox contenant le nom des trails
protected void	 eraseOnDBTrail	- Efface sur le serveur le parcours sélectionné dans la combobox






Méthodes de gestion des étapes de lecture :

Type	Nom	Description
protected void	loadTrail	- A la sélection d'un trail dans la combobox : affiche les étapes de lecture (vignette plus concaténation des labels des descriptor menant au document) dans le stepTable
protected void	loadStep	- A la sélection d'un step dans le stepTable : ouvre le document correspondant en ouvrant chaque Descriptor

protected void	addDefaultStep	- A l'ouverture d'un document : ajoute au parcours par défaut le step correspondant au document courant
protected void	 addStep	- Au clic sur le bouton addStepButton, si le parcours courant n'est pas le parcours par défaut, ajoute au parcours en cours le step correspondant au document courant
protected void	 delStep	- Au clic sur le bouton delStepButton, si le parcours courant n'est pas le parcours par défaut, enlève au parcours en cours le step sélectionné
protected void	 backStep	- Au clic sur le bouton backStepButton : ouvre le document correspondant au step précédent et place le step sélectionné sur le step précédent dans le stepTable
protected void	 forwardStep	- Au clic sur le bouton forwardStepButton : ouvre le document correspondant au step suivant et place le step sélectionné sur le step suivant dans le stepTable

5. Classes pour la communication Client-Serveur

Pour la communication entre le serveur de Trail et le client natif, il faudra modifier plusieurs classes qui permettent la mise en place d'une architecture de communication RMI. Cette enchaînement de classe permet de transmettre l'invocation de méthode depuis l'IHM client jusqu'au serveur, à savoir pour la gestion des trails les six fonctions suivantes :

Type	Nom	Description
protected void	 saveOnDBTrail	- Affichage d'une fenêtre de dialogue demandant le label du trail - Sauvegarde sur le serveur de données du parcours par défaut avec le label préalablement entré
protected void	 getTrail	- si un document est ouvert : permet la récupération de l'ensemble des {trailId, trailLabel} qui correspondent aux parcours du graphe de l'utilisateur (et de ceux auxquels il est abonné) passant par le même document. - si aucun document n'est ouvert, permet la récupération de tous les trails de l'utilisateur - Mise à jour de la combobox contenant le nom des trails
protected void	 eraseOnDBTrail	- Efface sur le serveur le parcours sélectionné dans la combobox
protected void	loadTrail	- A la sélection d'un trail dans la combobox : affiche les étapes de lecture (vignette plus concaténation des labels des descriptor menant au document) dans le stepTable
protected void	 addStep	- Au clic sur le bouton addStepButton, si le parcours courant n'est pas le parcours par défaut, ajoute au parcours en cours le step correspondant au document courant
protected void	 delStep	- Au clic sur le bouton delStepButton, si le parcours courant n'est pas le parcours par défaut, enlève au parcours en cours le step sélectionné

5.1. Package porphyry.view

La classe **GraphView**, qui d'obtenir une vue des graphes, devra permettre une vue des Trail venus du server.

5.2. Package porphyry.server.userManagement

La classe d'interface **GraphServer**, dont l'implémentation est la classe **GraphServerImpl**, qui est le serveur à proprement dit, permettra aussi la récupération des Trails.

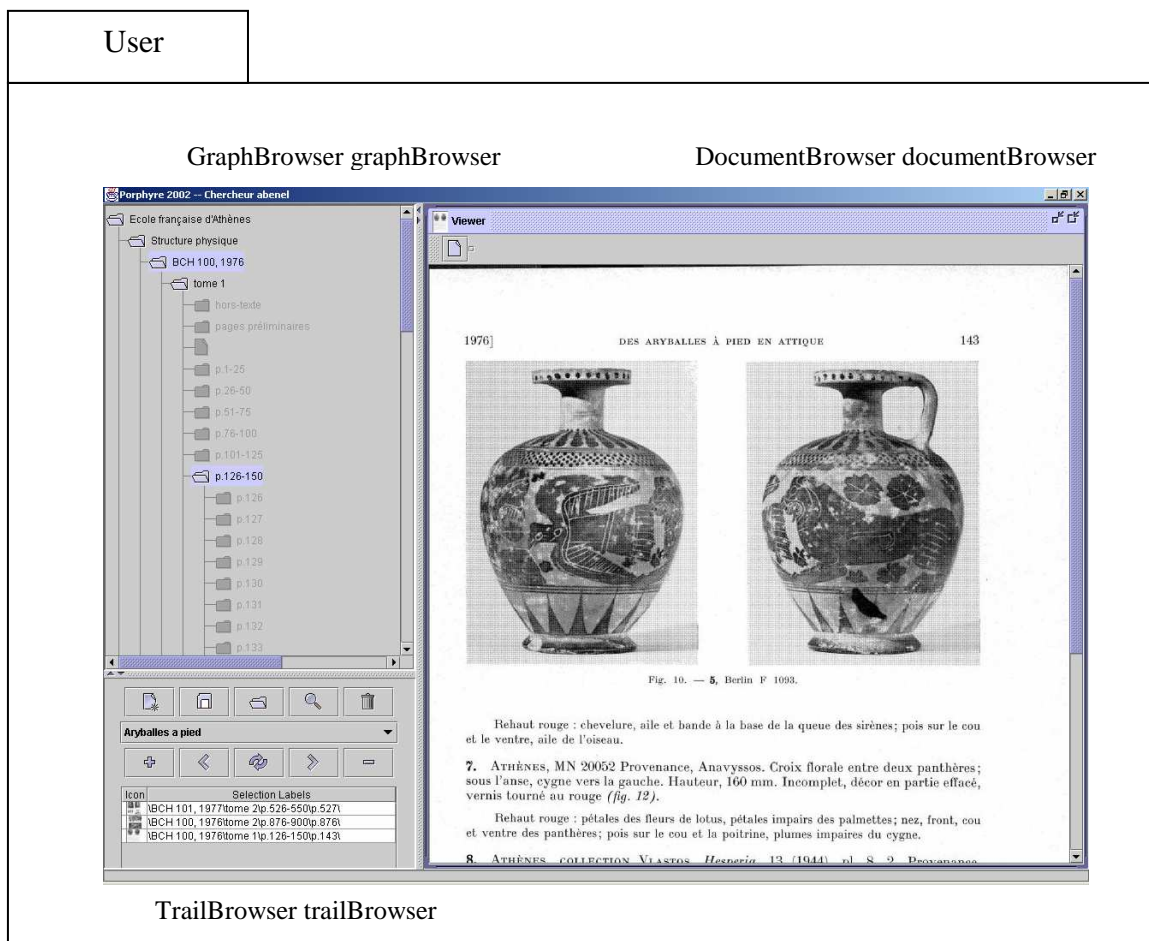
5.3. Package porphyry.server.calcul

La classe **GraphBroker** qui permet de travailler avec plusieurs **LocalGraph** par le biais de la classe interface **WriterGraph**.

6. IHM

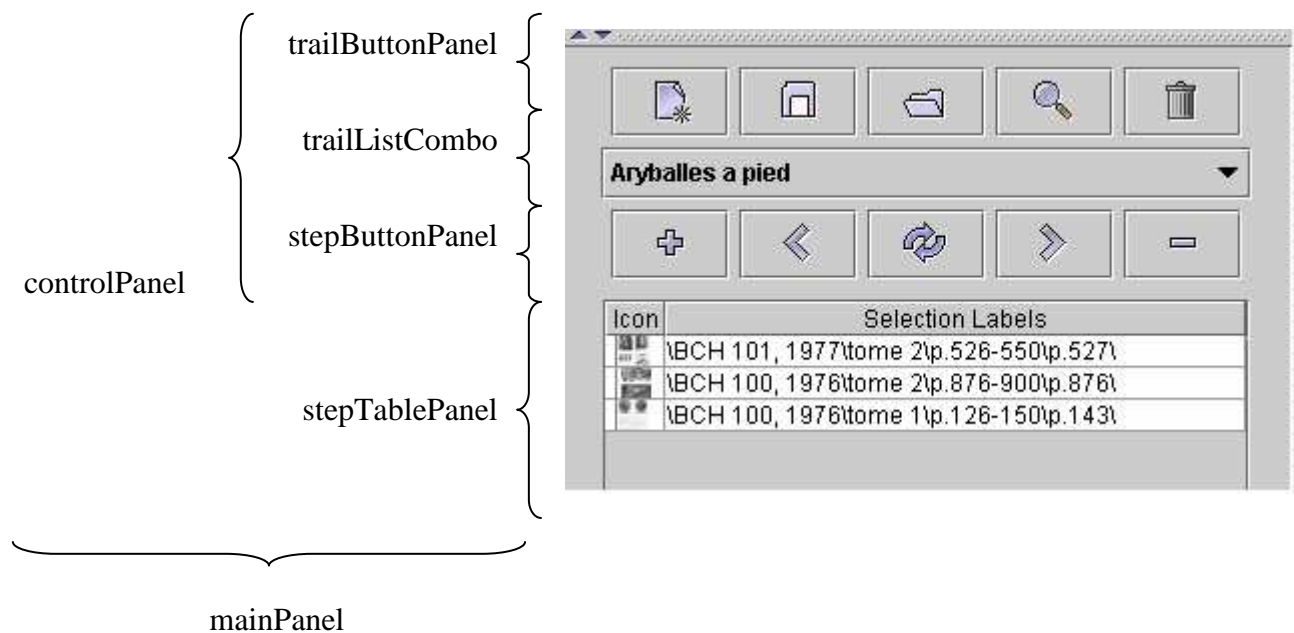
6.1. Classe User (existante)

La classe **TrailBrowser** est instanciée, au même titre que les classes **GraphBrowser** et **DocumentBrowser**, dans la classe **User** qui contient la méthode de création de l'application cliente (main). Elle permet la création de l'interface de gestion des parcours de lecture.



6.2. Partie IHM de la classe TrailBrowser : Panneaux

La classe TrailBrowser, au niveau IHM, se compose d'un assemblage de panneaux, tous initialisés au sein de la méthode interfaceInit(). Voici le détail de ces panneaux :

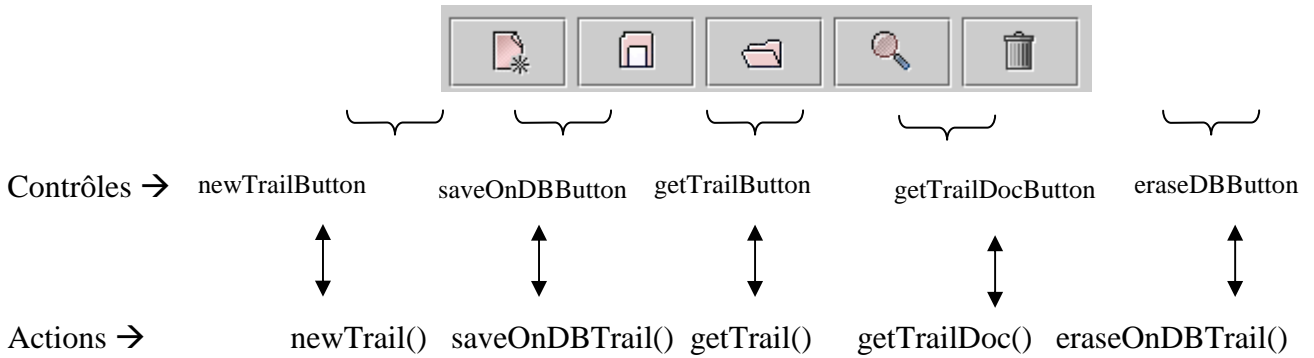


Assemblage des panneaux du TrailBrowser

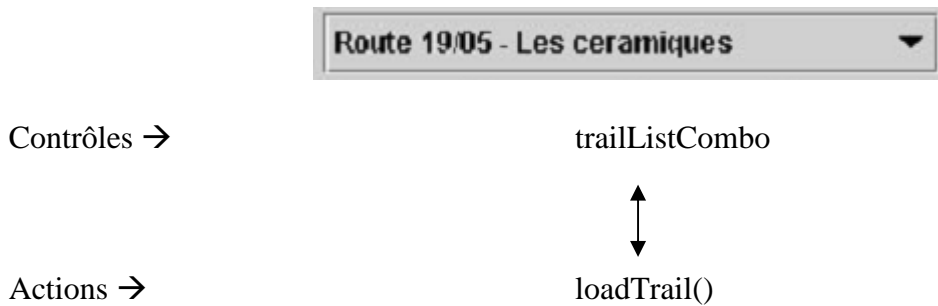
* Sur ce schéma apparaît le titre au lieu de la vignette, car cette fonctionnalité n'a pas encore été réalisée.

6.3. Partie IHM du TrailBrowser : Contrôles et Actions

Panneau de bouton trailButtonPanel :



ComboBox trailListCombo :



Panneau de bouton stepButtonPanel :

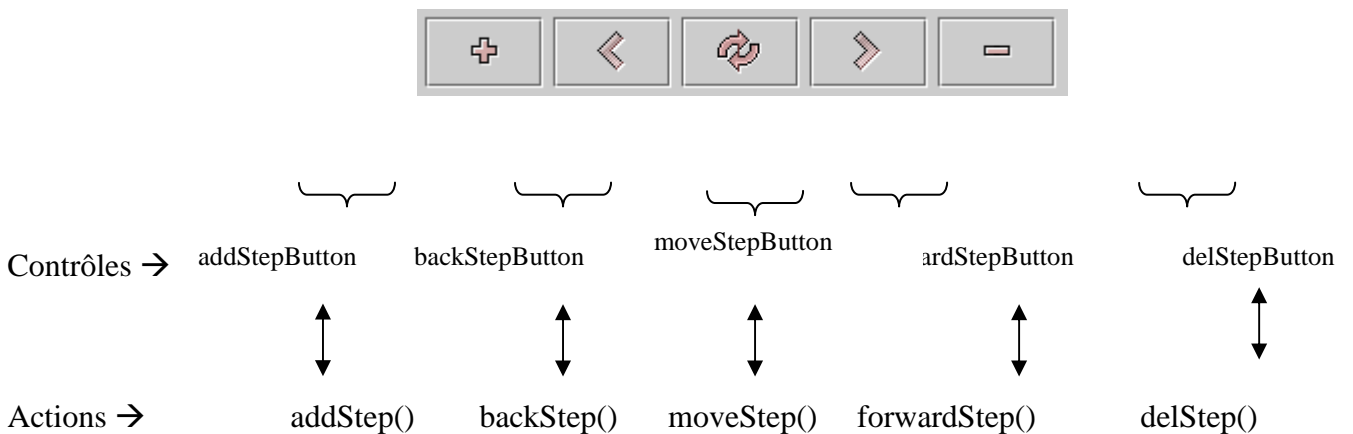





Tableau StepTable :

Icon	Selection Labels
	\\BCH 101, 1977\tome 2\p.526-550\p.527\
	\\BCH 100, 1976\tome 2\p.876-900\p.876\
	\\BCH 100, 1976\tome 1\p.126-150\p.143\

Contrôles →

stepTable



Actions →

loadStep()

* Sur ce schéma apparaît le titre au lieu de la vignette, car cette fonctionnalité n'a pas encore été réalisée.

7. Conclusion

Voici le document de conception du client natif JAVA et de sa communication avec le serveur qui reprend et incrémente, en ce qui concerne l'IHM et les fonctionnalités le travail effectué lors des spécifications.