Table des matières

[Utilisation du support Git dans l’IDE NetBeans 1](#_Toc476755444)

[Initialisation d’un répertoire Git 2](#_Toc476755445)

[Cloner un répertoire Git 4](#_Toc476755446)

[Ajouter des fichiers 6](#_Toc476755447)

[Edition des fichiers 7](#_Toc476755448)

[Affichage des modifications apportées dans le Source Editor 7](#_Toc476755449)

[Affichage des informations sur l’état du fichier 7](#_Toc476755450)

[Comparaison des fichiers révisés 11](#_Toc476755451)

[Annuler les changements 14](#_Toc476755452)

[Commiting des sources vers un répertoire 15](#_Toc476755453)

[Travailler avec des Branches 17](#_Toc476755454)

[Créer une branche 18](#_Toc476755455)

[Vérification 20](#_Toc476755456)

[Fusion 24](#_Toc476755457)

[Supprimer une branche 26](#_Toc476755458)

[Utilisation des dépôts distants (Remote Repositories) 27](#_Toc476755459)

[Fetching 27](#_Toc476755460)

[Pulling 29](#_Toc476755461)

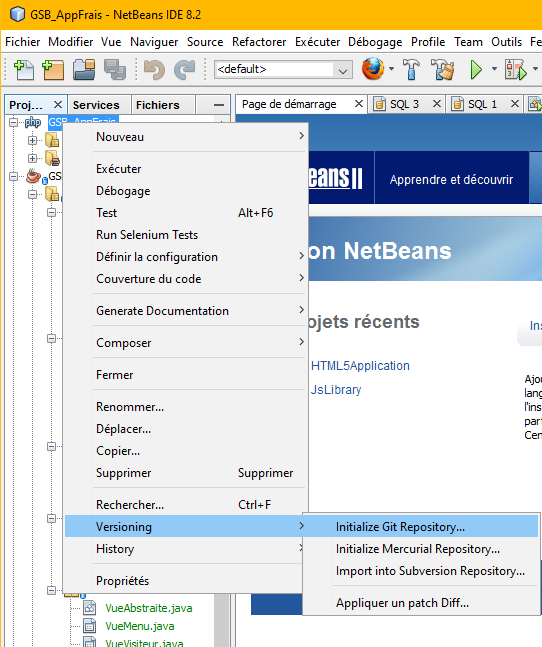
[Pushing 31](#_Toc476755462)

# Utilisation du support Git dans l’IDE NetBeans

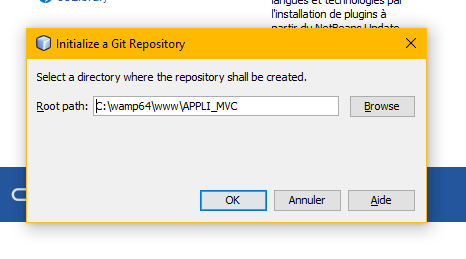
Ce didacticiel a pour but de montrer comment effectuer des tâches de versioning de base dans l'IDE en vous guidant dans le flux de travail standard lorsque vous utilisez le support GIT de l'IDE. Il montre comment configurer un projet versionné et effectuer des tâches de base sur les fichiers versionnés tout en vous initiant à certaines des fonctionnalités spécifiques à GIT, incluses dans l'IDE.

## Initialisation d’un répertoire Git

Dans la fenêtre des Projets, faire une clique-droit sur le projet -> Versioning -> Initialize Git Repository.



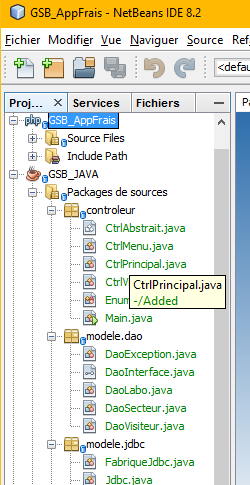
Spécifier le chemin d’accès au répertoire dans lequel vous allez stocker vos fichiers.



Cliquer sur OK.

Un dossier caché .git est alors créée dans le dossier sélectionné, qui sera votre répertoire Git dans lequel toutes les données du projet sont stockées.

Tous les fichiers du projet sont marqués Added dans l’arbre de travail.

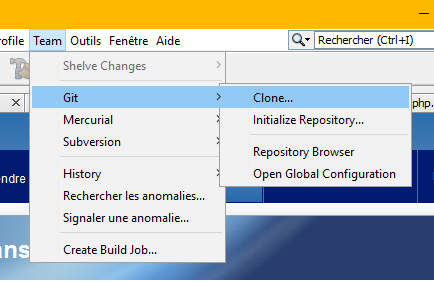


Après avoir initialisé le répertoire Git, vous pouvez soit ajouter des fichiers, soit les transmettre directement au répertoire Git.

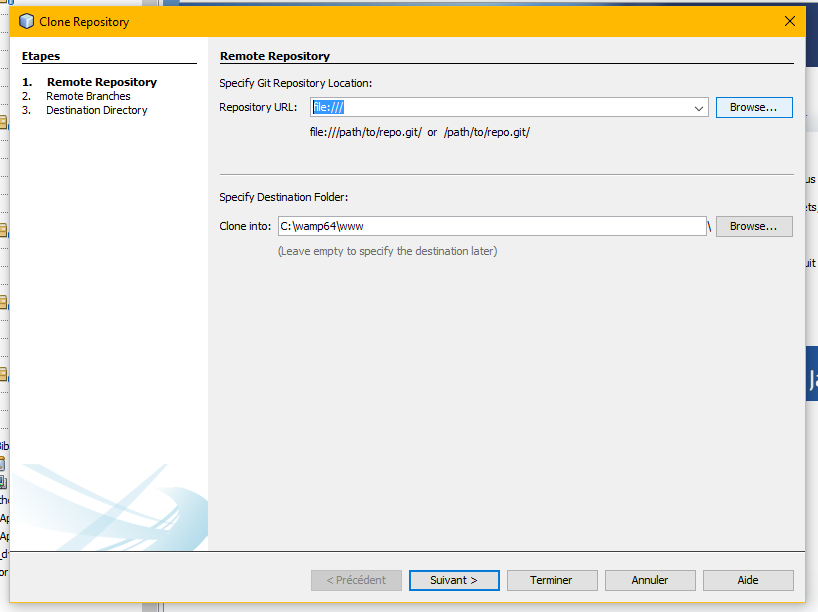
## Cloner un répertoire Git

Pour obtenir une copie d’un répertoire Git déjà existant, vous devez le cloner. Assurez-vous de connaître l’URL du répertoire Git avant de lancer l’assistant Clone Repository dans l’IDE.

Choisir Team -> Git -> Clone.



L’assistant Clone Repository s’affiche.



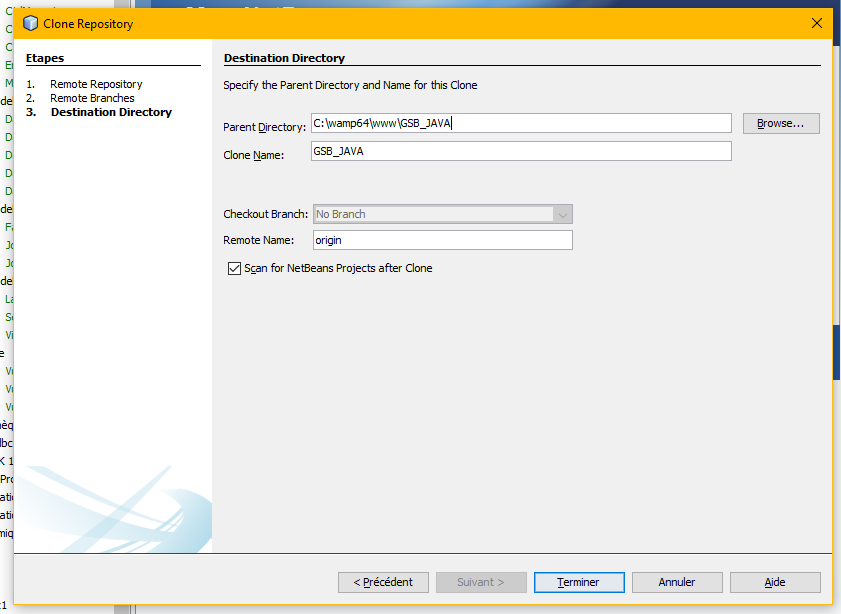
Dans Repository URL, spécifiez le chemin d’accès à un emplacement du répertoire Git.

Cliquez sur suivant.

Dans la page Remote Branches, sélectionnez la ou les branches du répertoire à récupérer (téléchargée) dans votre dépôt local. Cliquez sur suivant.

Dans la page Destination Directory, spécifiez :

* Dans le champ Parent Directory, le chemin d’accès au répertoire destiné au référentiel cloné sur votre disque dur (ou cliquez sur le bouton Bowse et accédez au répertoire). Le champ Parent Directory est pré-rempli avec le chemin d’accès au répertoire NetBeansProjects par défaut où tous les projets NetBeans sont stockés.
* Dans le champ Clone Name, le nom du dossier local dans lequel le projet d’origine sera cloné. Par défaut, Clone Name est rempli avec le nom du répertoire Git actuel.
* Dans le champ Checkout Branch, sélectionnez la branche à extraire dans l’arbre de travail.
* Dans le champ Remote Name, le nom qui représente le référentiel d’origine étant cloné. Origin est l’alias par défaut du référentiel cloné. C’est une valeur recommandée.
* Laissez la case à cocher Scan for NetBeans Projects after Clone pour activer le balayage après la fin du clonage. (Le plugin recherche des projets NetBeans dans les ressources clonées et propose d’ouvrir les projets trouvés.)



Cliquez sur Finish.

Après le clonage d’un répertoire Git, le dossier .git metadata est créé à l’intérieur du dossier que vous avez sélectionné dans l’assistant.

## Ajouter des fichiers

Pour commencer le suivi d’un nouveau fichier et également pour mettre en évidence les modifications apportées à un fichier déjà suivi dans le répertoire Git, vous devez l’ajouter au répertoire.

Lors de l’ajout de fichier à un répertoire Git, l’IDE compose et enregistre les snapshots de votre projet dans l’index. Après avoir effectué le commit, l’IDE enregistre ces snapshots dans HEAD. L’IDE vous permet de choisir entre les deux flux de travail (workflow) décrits dans le tableau suivant.

|  |  |  |
| --- | --- | --- |
| Description du workflow | Ajoute explicitement des nouveaux fichiers ou modifiés à l’index et valide uniquement ceux qui sont mis en scène dans l’Index du HEAD | Ignore l’ajout de nouveaux fichiers ou modifiés dans l’Index et valide les fichiers requis directement sur le HEAD |
| Etapes à suivre pour le workflow | 1) Dans la fenêtre Projets, cliquez avec le bouton droit sur le fichier que vous souhaitez ajouter.  2) Dans le menu contextuel, choisissez Git -> Add.  Cela ajoute le contenu du fichier à l’index avant de valider.  3) Dans la fenêtre Projects, faites clique-droit sur le fichier à valider.  4) Dans la boîte de dialogue Commit, sélectionnez le bouton Changes between HEAD and Index https://netbeans.org/images_www/articles/74/ide/git/changes-head-index.png.  Cela affiche la liste des fichiers déjà mis en scène.  5) Confirmez le ou les fichiers. | 1) Dans la fenêtre Projects, faites clique-droit sur le fichier à valider.  2) Dans le menu contextuel, choisissez Git -> Commit.  3) Dans la boîte de dialogue Commit, sélectionnez le bouton Select the Changes between HEAD and Working Tree https://netbeans.org/images_www/articles/74/ide/git/changes-head-wt.png.  Cela affiche la liste des fichiers qui ne sont pas mis en scène.  4) Confirmez le ou les fichiers. |

L’action fonctionne de manière récursive si elle est appelée sur des dossiers tout en respectant la structure du contenu de dossier NetBeans IDE.

## Edition des fichiers

Une fois que vous avez un projet Git ouvert dans l’IDE, vous pouvez commencer à apporter des modifications aux sources. Comme pour tout projet ouvert dans l’IDE NetBeans, vous pouvez ouvrir des fichiers Source Editor en double-cliquant sur leurs nœuds, tels qu’ils apparaseent dans les fenêtres de l’IDE (par exemple, fenêtres Projetcts (Ctrl-1), Files (Ctrl-2), Favorites (Ctrl-3)).

Lorsque vous travaillez avec des fichiers sources dans l’IDE, vous disposez de différents composants d’interface utilisateur qui vous aident à utiliser des commandes de contrôle de version :

### Affichage des modifications apportées dans le Source Editor

Lorsque vous ouvrez un fichier dans le Source Editor de l’IDE, vous pouvez afficher les modifications en temps réel de votre fichier au fur et à mesure que vous le modifiez par rapport à la version de base du répertoire Git. Au fur et à mesure que vous travaillez, l’IDE utilise le codage couleurs dans les marges de Source Editor pour transmettre les informations suivantes :

* + Bleu indique les lignes qui ont été modifiées depuis la révision précédente.
  + Vert indique les lignes qui ont été ajoutées depuis la révision précédente.
  + Rouge indique les lignes qui ont été supprimées depuis la révision précédente.

La marge gauche du Source Editor indique les changements survenant ligne par ligne. Lorsque vous modifiez une ligne donnée, les modifications sont immédiatement affichées dans la marge de gauche.

La marge de droite du Source Editor vous donne une vue d’ensemble des modifications apportées à votre fichier dans son ensemble, de haut en bas. Le codage des couleurs est généré immédiatement lorsque vous apportez des modifications à votre fichier.

### Affichage des informations sur l’état du fichier

Quand vous travaillez dans les vues Projets (Ctrl-1), Fichiers (Ctrl-2), Favoris (Ctrl-3) ou Versioning, l’IDE fournit plusieurs fonctionnalités visuelles qui aident à afficher les informations d’état de vos fichiers. Dans l’exemple ci-dessous, notez que le badge (par exemple le badge bleu), la couleur du nom de fichier et l’étiquette d’état adjacente, coïncident tous les uns avec les autres pour vous fournir un moyen simple mais efficace de garder une trace des informations de version sur vos fichiers.

* + Badges et codes couleur

Les badges sont appliqués aux nœuds de projet, de dossier et de package et vous indiquent l’état des fichiers contenus dans ce nœud.

Le tableau suivant affiche le jeu de couleurs utilisé pour les badges :

|  |  |
| --- | --- |
| UI Component | Description |
| Badge Bleu (Blue badge) | Indique la présence de fichiers qui ont été modifiés, ajoutés ou supprimés dans votre arbre de travail. Pour les packages, ce badge s’applique uniquement au package lui-même et non à ses sous-paquets. Pour les projets ou les dossiers, le badge indique les modifications de cet élément ou de l’un des sous-dossiers contenus. |
| Badge Rouge (Red badge) | Marque les projets, les dossiers ou les packages contenant des fichiers conflictuels. Pour les packages, ce badge s’applique uniquement au package lui-même et non à ses sous-paquets. Pour les projets ou les dossiers, le badge indique les conflits au sein de cet élément ou de l’un des sous-dossiers contenus. |

Le codage couleur est appliqué aux noms de fichiers afin d’indiquer leur état actuel par rapport au référentiel :

|  |  |  |
| --- | --- | --- |
| Couleur | Exemple | Description |
| Pas de couleur (noir) | Black text | Indique que le fichier n’a pas de modifications. |
| Bleu | Blue text | Indique que le fichier a été modifié localement. |
| Vert | Green text | Indique que le fichier a été ajouté localement. |
| Rouge | Red text | Indique que le fichier est dans un conflit de fusion. |
| Gris | Gray text | Indique que le fichier est ignoré par Git et ne sera pas inclus dans les commandes de versioning (par exemple, Update and Commit). Les fichiers ne peut pas être ignorés s’ils sont versionnés. |

* + Etiquettes d’état des fichiers

L’IDE affiche deux valeurs d’était pour un fichier :

Un état décrivant les différences entre les fichiers Working Tree et Index,

Un état décrivant les différences entre les fichiers index state et current HEAD commit.

Les étiquettes d’état des fichiers fournissent une indication textuelle de l’état des fichiers versionnés dans les fenêtres de l’IDE.

|  |  |
| --- | --- |
| Etiquette d’état | Signification |
| - | Non modifié |
| A | Ajouté |
| U | Mis à jour mais non fusionné |
| M | Modifié |
| D | Supprimé |
| I | Ignoré |
| R | Renommé |

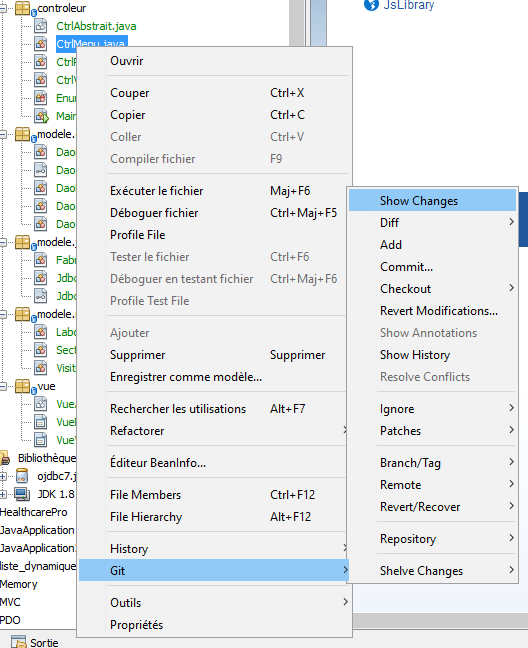
Par défaut, l’IDE affiche des informations d’état et de dossiers en texte gris à droite des fichiers.

File labels displayed next to file names

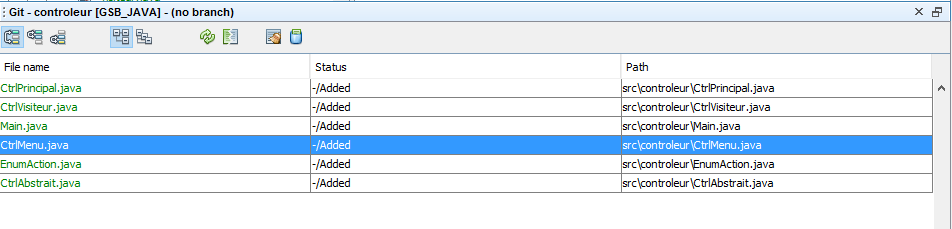
Les fichiers en conflit de fusion disposent de l’état non fusionné qui est généralement annoté par la couleur rouge jusqu’à ce que les fichiers n’aient résolu le conflit.

* Les étiquettes d’état des fichiers peuvent être activées ou désactivées en choisissant View -> Show Versioning Labels dans le menu principal.
  + Git Versioning View

Le Git Versionning view vous fournit une liste en temps réel de toutes les modifications apportées aux fichiers dans un dossier sélectionné de votre arbre de travail local. Il s’ouvre par défaut dans le panneau inférieur de l’IDE, avec la liste des fichiers ajoutés, supprimés ou modifiés.

Pour ouvrir la Version, sélectionnez un fichier ou un dossier versionné (par exemple, dans la fenêtre Projects, Files, Favorites) et choisissez Git -> Show Changes dans le menu contextuel ou choisissez Team -> Show Changes dans le menu principal. 

La fenêtre suivante apparaît :



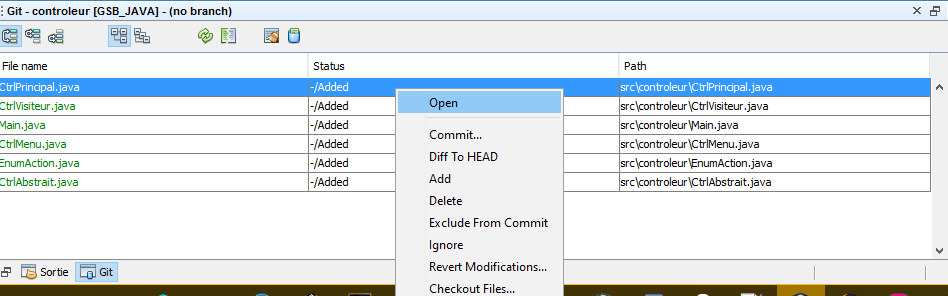
Par défaut, le Versioning view affiche une liste de tous les fichiers modifiés dans le package ou le dossier séléctionné dans votre arbre de travail.

En utilisant les boutons de la barre d’outils, vous pouvez choisir d’afficher la liste des fichiers qui ont des différences entre Index et HEAD, Working Tree et HEAD. Vous pouvez également cliquer sur les en-têtes de colonne au-dessus des fichiers listés pour trier les fichiers par nom, statut ou emplacement.

La barre d’outils du Versionning view comprend également des boutons qui vous permettent d’invoquer les tâches Git les plus courantes sur tous les fichiers affichés dans la liste. Le tableau suivant répertorie les commandes Git disponibles dans la barre d’outils du Versionning view :

|  |  |  |
| --- | --- | --- |
| Icône | Nom | Fonction |
| Changes between HEAD and Working Tree icon | Changes between HEAD and Working Tree | Affiche une liste de fichiers déjà mis en scène ou modifiés / créés et pas encore mis en scène. |
| Changes between HEAD and Index icon | Changes between HEAD and Index | Affiche une liste de fichiers mis en scène. |
| Changes between Index and Working Tree icon | Changes between Index and Working Tree | Afficher les fichiers qui ont des différences entre leurs états de scène et Working Tree. |
| Refresh Statuses icon | Refresh Statuses | Rafraîchit l’état des fichiers et dossiers sélectionnés. Les fichiers affichés dans le Versionning view peuvent être actualisés pour faire apparaître les modifications qui peuvent avoir été faites en externe. |
| Open Diff icon | Open Diff | Ouvre le Diff Viewer vous offrant une comparaison côte-à-côte de vos copies locales et des versions maintenues dans le référentiel. |
| Checkout Paths icon | Revert Modifications | Affiche la boîte de dialogue Revert Modifications (Rétablir les Modifications). |
| Commit Changes icon | Commit Changes | Affiche la boîte de dialogue Commit. |

Vous pouvez accéder à d’autres commandes Git dans le Versioning view en sélectionnant une ligne de table correspondant à un fichier modifié et en choisissant une commande dans le menu contextuel :



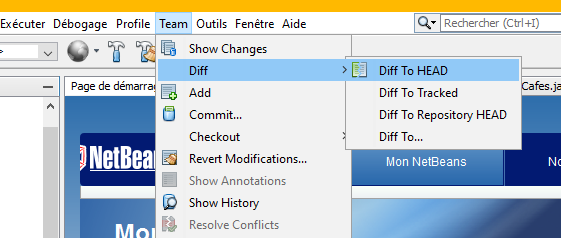
### Comparaison des fichiers révisés

La comparaison des versions de fichiers est une tâche courante lorsque vous travaillez avec des projets versionnés. L’IDE vous permet de comparer les révisions en utilisant la commande Diff.

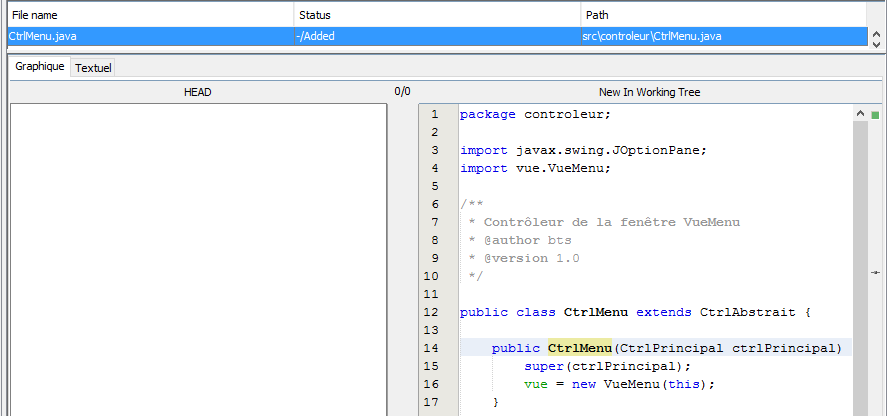
Remarque : Plusieurs modes de comparaison – Diff To HEAD, Diff To Tracked et Diff To – sont disponibles dans l’IDE.

Sélectionnez un fichier ou un dossier versionné (par exemple, dans la fenêtre Projects, Files ou Favorites).

Choisissez Team -> Diff -> Diff to HEAD dans le menu principal.



Un graphique Diff Viewer s’ouvre pour les fichiers sélectionnés et les révisions dans la fenêtre principale de l’IDE. Le Diff Viewer affiche deux copies dans des panneaux cote-à-côte. La copie la plus courante apparaît sur le côté droit, donc si vous comparez une révision du répertoire à votre arbre de travail, l’arborescence de travail s’affiche dans le panneau de droite :



Le Diff Viewer utilise le code de couleur utilisé ailleurs pour afficher les modifications de contrôle de version.

Dans la capture d’écran affichée ci-dessus,

le bloc vert indique le contenu ajouté à la révision la plus récente,

le bloc rouge indique que le contenu de la révision précédente a été supprimé de la version ultérieure,

le bloc bleu indique que des changements se sont produits dans les lignes en surbrillance.

Remarque : D’autres révisions peuvent être sélectionnées dans les listes Diff et déroulante sous la barre d’outils Diff Viewer.

La barre d’outil Diff Viewer comprend également des boutons qui vous permettent d’invoquer les tâches Git les plus courantes sur tous les fichiers affichés dans la liste. Le tableau suivant répertorie les commandes Git disponibles dans la barre d’outils du Diff Viewer :

|  |  |  |
| --- | --- | --- |
| Icône | Nom | Fonction |
| Changes between HEAD and Working Tree icon | Changes between HEAD and Working Tree | Affiche une liste de fichiers déjà mis en scène ou modifiés / créés et pas encore mis en scène. |
| C:\Users\LATOMA~1\AppData\Local\Temp\changes-head-index.png | Changes between HEAD and Index | Affiche une liste de fichiers mis en scène. |
| Changes between Index and Working Tree icon | Changes between Index and Working Tree | Afficher les fichiers qui ont des différences entre leurs états de scène et Working Tree. |
| Go to Next Difference icon | Go to Next Difference | Affiche la différence suivante dans le fichier. |
| Go to Previous Difference icon | Go to Previous Difference | Affiche la différence précédente dans le fichier. |
| Refresh Statuses icon | Refresh Statuses | Rafraîchit l’état des fichiers et dossiers sélectionnés. Les fichiers affichés dans la fenêtre Versioning peuvent être actualisés pour refléter les modifications qui peuvent avoir été faites en externe. |
| Checkout Paths icon | Revert Modifications | Affiche la boîte de dialogue Revert Modifications (Rétablir les Modifications). |
| Commit Changes icon | Commit Changes | Affiche la boîte de dialogue Commit. |

Si vous effectuez un diff sur votre copie locale dans le Working Tree, l’IDE vous permet d’effectuer des modifications directement à partir du Diff Viewer. Pour ce faire, vous pouvez soit positionner votre curseur dans le volet droit du Diff Viewer et modifier votre fichier en conséquence, soit utiliser les icônes en ligne qui s’affichent à côté de chaque changement mis en évidence :

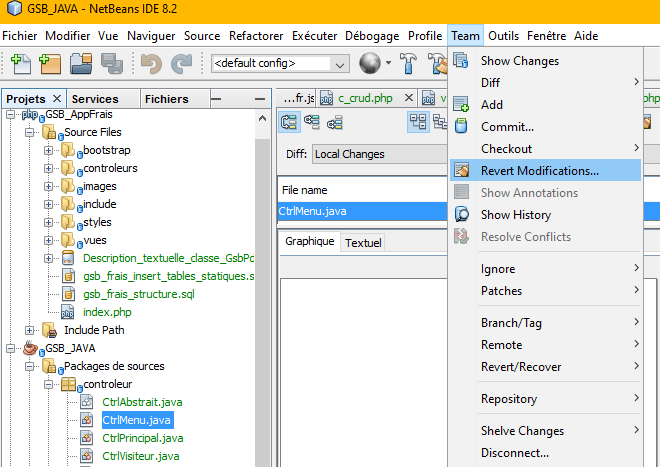
|  |  |  |
| --- | --- | --- |
| Icône | Nom | Fonction |
| Replace icon | Replace | Insère le texte mis en surbrillance dans votre copie du Working Tree. |
| Move all icon | Move All | Rétablit l’ensemble de la copie locale de Working Tree. |
| Remove icon | Remove | Supprime le texte mis en surbrillance de la copie locale de Working Tree. |

### Annuler les changements

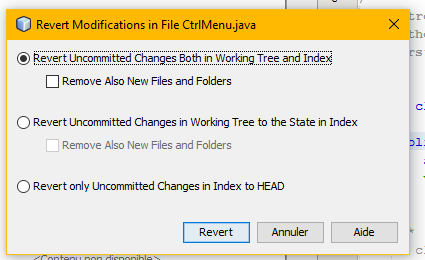
Pour jeter les modifications locales apportées aux fichiers sélectionnés dans votre arbre de travail et les remplacer par celles de l’index ou HEAD :

Sélectionnez un fichier ou un dossier versionné (par exemple, dans la fenêtre Projects, Files, ou Favorites).

Choisissez Team -> Revert Modification dans le menu principal.



La boîte de dialogue Revert Modifications s’affiche.



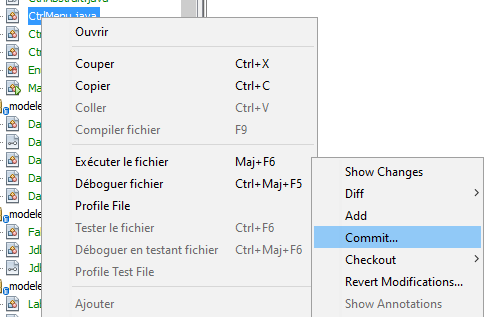
Spécifiez les options supplémentaires et cliquez sur Revert.

## Commiting des sources vers un répertoire

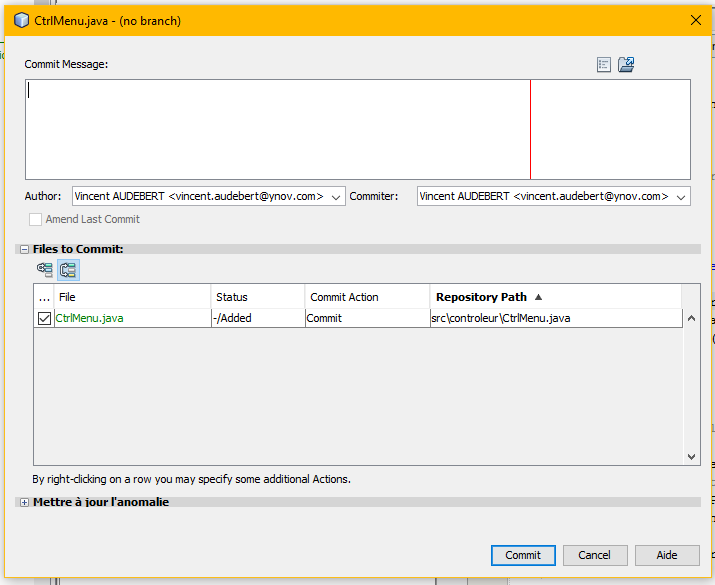
Pour commit les fichiers dans le répertoire Git :

Dans la fenêtre Projects, clique-droit sur le fichier à commit.

Dans le menu contextuel, choisissez Git -> Commit.



La boîte de dialogue commit s’affiche.



La boîte de dialogue Commit contient les composants suivants :

Zone de texte Commit Message destinée à décrire la modification effectuée.

Les listes déroulantes Author et Commiter qui permettent de différencier ceux qui ont apporté le changement et qui ont physiquemnt engagé le fichier si nécessaire.

Files to Commit qui répertorie :

Tous les fichiers modifiés.

Tous les fichiers qui ont été supprimés dans le Working Tree (localement).

Tous les nouveaux fichiers (c’est-à-dire les fichiers qui n’existent pas encore dans le répertoire Git).

Tous les fichiers que vous avez renommés.

Deux boutons à bascule qui commutent le mode dans lequel le commit actuel doit être exécuté sont disponibles ici :

|  |  |  |
| --- | --- | --- |
| Icône | Nom | Fonction |
| Changes between HEAD and Working Tree icon | Changes between HEAD and Working Tree | Affiche une liste de fichiers déjà mis en scène ou modifiés / créés et pas encore mis en scène. |
| C:\Users\LATOMA~1\AppData\Local\Temp\changes-head-index.png | Changes between HEAD and Index | Affiche une liste de fichiers mis en scène. |

Remarque : Pour spécifier ici s’il faut exclure des fichiers individuels de la validation, désélectionnez la case à cocher de la première colonne intitulée Commit ou cliquez avec le bouton droit sur une ligne de fichier dans la colonne Commit Action et choisissez Exclude from commit dans le menu contextuel.

Pour afficher le Diff Viewer, cliquez avec le bouton droit sur une ligne de fichier dans la colonne Commit Action et choisissez Diff dans le menu contextuel.

Remarque : Vous devez installer le plugin JIRA ou Subversion pour démarrer le suivi des problèmes dans l’IDE.

Tapez un message de validation dans la zone de texte Commit Message. Vous pouvez également effectuer l’une des opérations suivantes :

Cliquez sur l’icône Recent Messages (Recent messages icon) située dans le coin supérieur droit pour afficher et sélectionner dans une liste de messages que vous avez déjà utilisés.

Cliquez sur l’icône Load Template (Select template icon) située dans le coin supérieur droit pour sélectionner un modèle de message.

Après avoir spécifié des actions pour des fichiers individuels, cliquez sur Commit.

L’IDE exécute le commit et stocke vos snapshots dans le répertoir. La barre d’état de l’IDE, située en bas à droite de l’interface, affiche alors que l’action a lieu. Lors d’une validation réussie, les badges de contrôle de version disparaissent dans les Projects, Files et Favorites, et le code couleur des fichiers validés revient au noir.

## Travailler avec des Branches

Le support Git de l’IDE vous permet de gérer différentes versions d’une base de code à l’aide de branches.

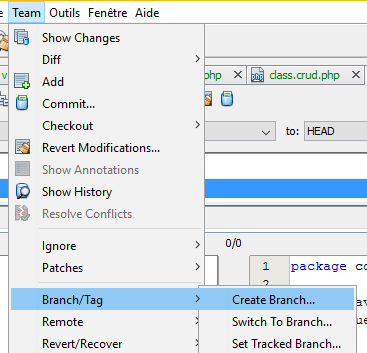
Lorsque vous travaillez avec des branches dans l’IDE, les actions suivantes sont prises en charge : Création, Vérification, Fusion, Suppression.

### Créer une branche

Pour créer une branche locale, si vous souhaitez travailler sur une version distincte de votre système de fichiers à des fins de stabilisations ou d’expérimentation sans perturber le coffre principal, procédez comme suit :

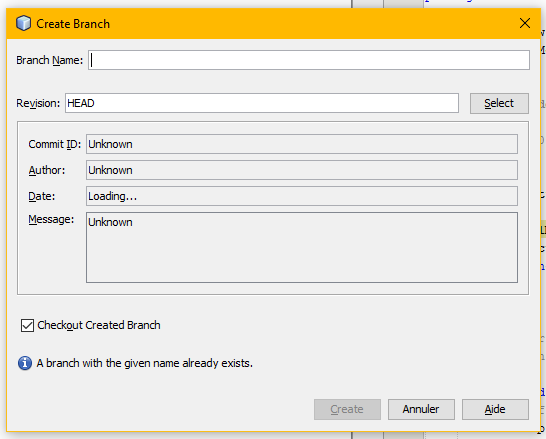
Dans la fenêtre Projects ou Files, choisissez un projet ou un dossier dans le répertoire dans lequel vous souhaitez créer la branche.

Dans le menu principal, choisissez Team -> Branch/Tag -> Create Branch.



Remarque : En alternative, cliquez sur le bouton droit de la souris sur le projet ou le dossier versionné, puis choisissez Git -> Branch/Tag -> Create Branch dans le menu contextuel.

La boîte de dialogue Create Branch s’affiche.



Dans le champ Branch Name, entrez le nom de la branche en cours de création.

Tapez une révision spécifique de l’élément sélectionné en entrant un commit ID, une branche existante ou un nom de tag dans le champ Revision ou appuyez sur Select pour afficher la liste des révisions maintenues dans le répertoire.

(Facultatif) Dans la boîte de dialogue Select Revision, développez Branch et choisissez la branche requise, spécifiez le commit ID dans la liste adjacente et appuyez sur Select.

Examinez les informations Commit ID, Author, Message, spécifiques à la révision en cours de ramification et cliquez sur Create.

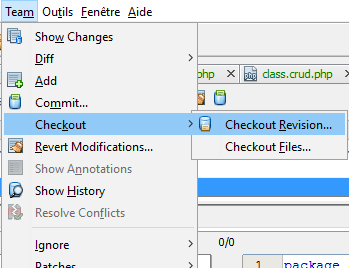
La branche est ajoutée au dossier Branches/Local du répertoire Git.

### Vérification

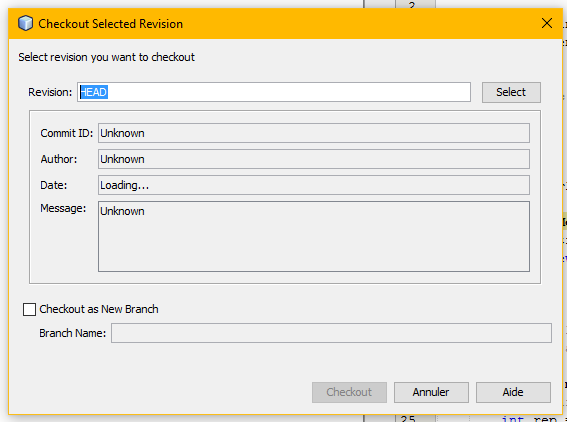
Si vous avez besoin d’éditer des fichiers sur une branche qui existe déjà, vous pouvez consulter la branche pour copier les fichiers dans votre Working Tree.

Pour vérifier une révision, procédez comme suit :

Choisissez Team -> Checkout -> Checkout Revision dans le menu principal.



La boîte de dialogue Checkout Selected Revision s’affiche.



Spécifiez la révision requise en entrant un commit ID, une branche existante ou un nom dans le champ Revision ou appuyez sur Select pour afficher la liste des révisions maintenues dans le répertoire.

    Remarque: Si la révision spécifiée fait référence à une validation valide qui n'est pas marquée d'un nom de branche, votre HEAD devient détaché et vous n'êtes plus sur aucune branche.

    Examinez les champs Commit ID, Author, Message, spécifiques à la révision en cours de vérification.

    Pour créer une nouvelle branche hors de la révision vérifiée, choisissez l'option Commander en tant que nouvelle branche et entrez le nom dans le champ Branch Name.

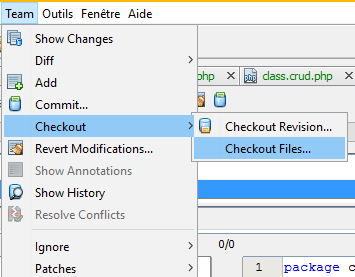
    Appuyez sur Checkout pour vérifier la révision.

    Les fichiers du Working Tree et de l'index sont mis à jour pour correspondre à la version de la révision spécifiée.

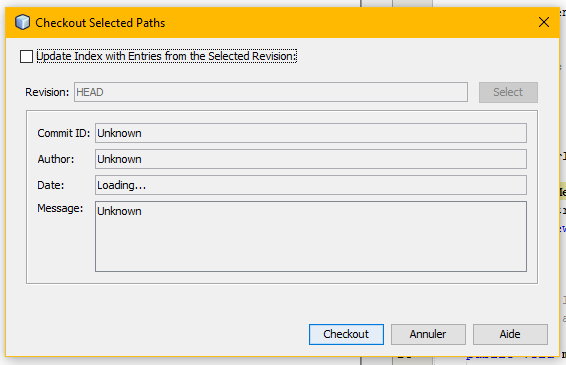
Remarque: Si vous souhaitez basculer vos fichiers vers une branche existante (par exemple vers un commit qui n'est pas en haut de l'une de vos branches), vous pouvez utiliser la commande Team -> Git -> Branch -> Switch To Branch, la branche dans la boîte de dialogue Switch to Selected Branch, le vérifier comme une nouvelle branche (facultatif), et appuyez sur Switch.

L'IDE prend en charge le contrôle contextuel des fichiers, des dossiers ou des projets actuellement sélectionnés dans l'IDE. Pour extraire certains fichiers (pas une branche) de l'index, procédez comme suit:

Choisissez Team -> Checkout -> Checkout Fies dans le menu principal.



La boîte de dialogue Checkout Selected Paths s’affiche.

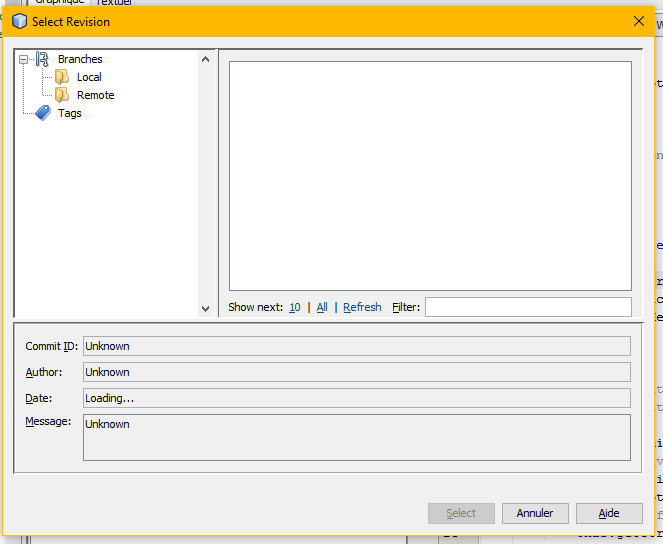


Choisissez Update Index with Entries from the Selected Revision.

Si cette option est sélectionnée, l’index est mis à jour avec l’état dans la révision sélectionnée avant la vérification elle-même (c’est-à-dire les fichiers sélectionnés à la fois dans le Working Tree et l’index sont mis à jour).

Spécifiez la révision requise entrant un Commit ID, une branche existante, ou un nom dans le champ Revision ou appuyez sur Select pour afficher la liste des révisions maintenues dans le répertoire.

Dans la boîte de dialogue Select Revision, développez Branches et choisissez la branche requise, spécifiez le numéro de révision dans la liste adjacente si nécessaire, puis appuyez sur Select.

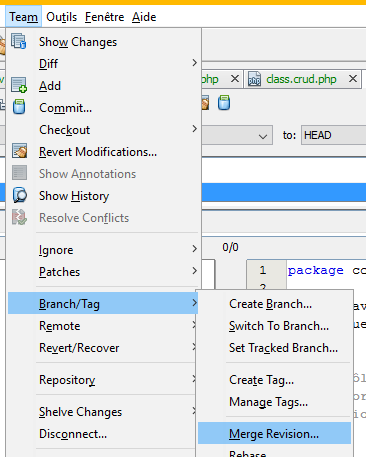


Appuyez sur Checkout pour terminer.

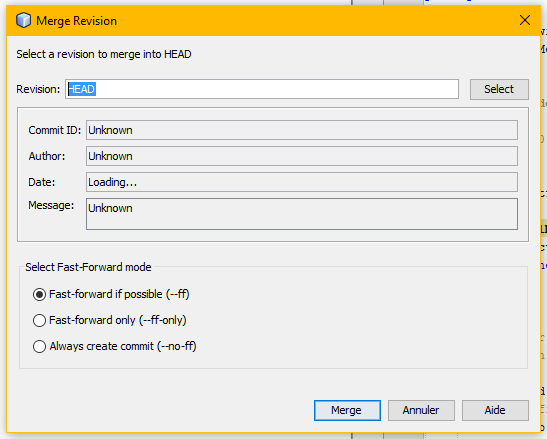
### Fusion

Pour transférer les modifications d’une révision de répertoire au Working Tree, procédez comme suit :

Choisissez Team -> Branch/Tag -> Merge Revision dans le menu principal.



La boîte de dialogue Merge Revision s’affiche.



Spécifiez la révision requise en entrant un Commit ID, une branche existante ou un nom dans le cham Revision ou appuyez sur Select pour afficher la liste des révisions maintenues dans le répertoire.

Dans la boîte de dialogue Select Revision, développez Branches et choisissez la branche requise, spécifiez le Commit ID dans la liste adjacente si nécessaire et appuyez sur Select.

Appuyez sur Merge.

Une fusion à trois voies entre la branche actuelle, le contenu du Working Tree et la branche spécifiée est effectuée.

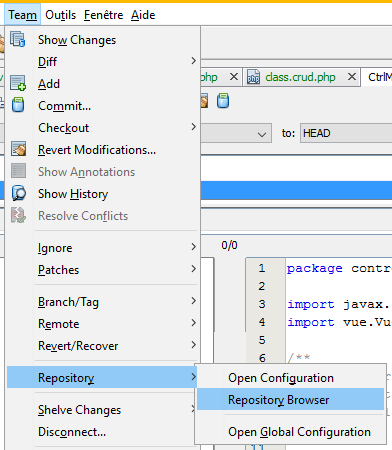
Remarque : Si un conflit de fusion se produit, le fichier conflictuel est marqué d’un badge rouge pour l’indiquer.

Remarque : Après la fusion, vous devez toujours valider les modifications afin qu’elles soient ajoutées à la HEAD.

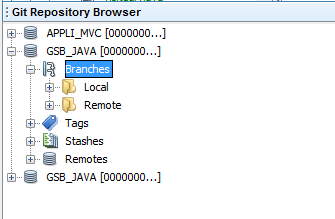
### Supprimer une branche

Pour supprimer une branche locale inutile, procédez comme suit :

Choisissez Team -> Repository -> Repository Browser dans le menu principal.



Dans le Git Repository Browser, choisissez une branche à supprimer.



Remarque : La branche doit être inactive, c’est-à-dire qu’elle n’est pas actuellement extraite dans le Working Tree.

Faites clique-droit sur la branche sélectionnée et choisissez Delete Branch dans le menu contextuel.

Dans la boîte de dialogue Delete Branch, appuyez sur OK pour confirmer la suppression de la branche.

La branche est supprimée du répertoire local ainsi que du Git Repository Browser.

## Utilisation des dépôts distants (Remote Repositories)

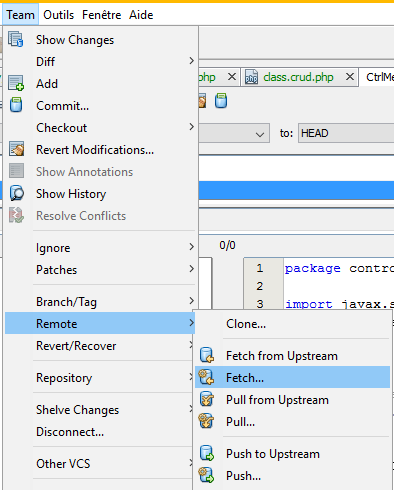
Lorsque vous travaillez avec d’autres développeurs, vous devez partager votre travail, ce qui implique d’extraire, de pousser et de transférer des données vers et depuis des répertoires distants hébergés sur Internet ou sur le réseau.

### Fetching

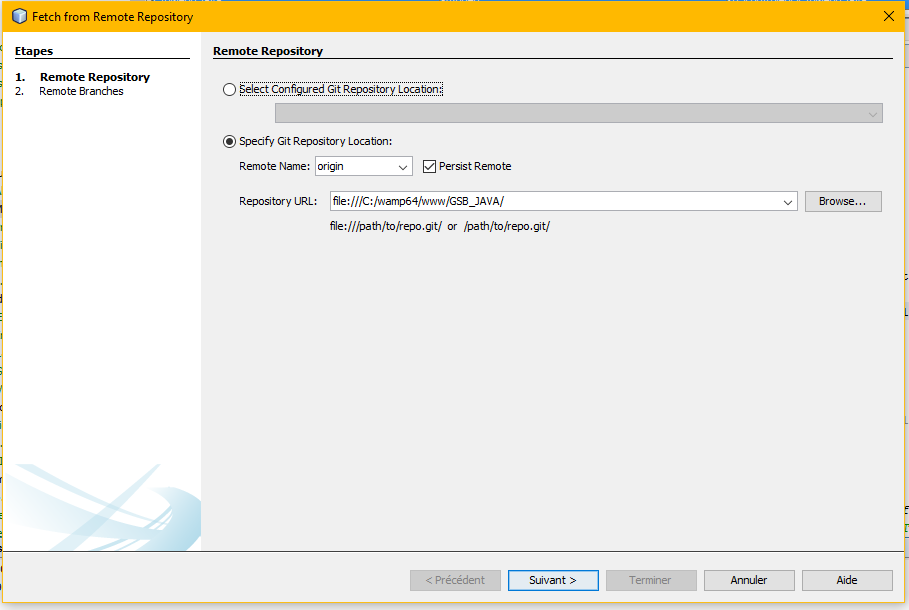
Fetching obtient les modifications du répertoire distant d’origine que vous n’avez pas encore. Il ne change jamais aucune de vos branches locales. Fetching récupère toutes les branches des dépôts distants, que vous pouvez fusionner dans votre branche ou tout simplement inspecter à tout moment.

Pour récupérer les mises à jour, procédez comme suit :

Choisissez Team -> Remote -> Fetch.



L’assistant Fetch from Remote Repository s’affiche.



Dans la page Remote Repository de l’assistant, Select Configured Git Repository Location pour utiliser le chemin d’accès au répertoire configuré précédemment, ou l’option Specify Git Repository Location pour définir le chemin d’accès à un référentiel distant inaccessible, son nom, Login, mot de passe et configuration proxy si nécessaire.

Cliquez sur suivant.

Dans la page Remote Branches de l’assistant, choisissez les branches pour récupérer les modifications et cliquez sur Terminer.

Une copie locale d’une branche distante est créée. La branche sélectionnée est mise à jour dans le répertoire Branches -> Remote du Git Repository Browser.

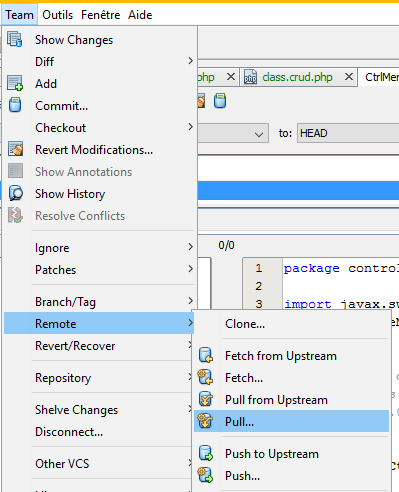
Ensuite, les mises à jour récupérées peuvent être fusionnées dans une branche locale.

### Pulling

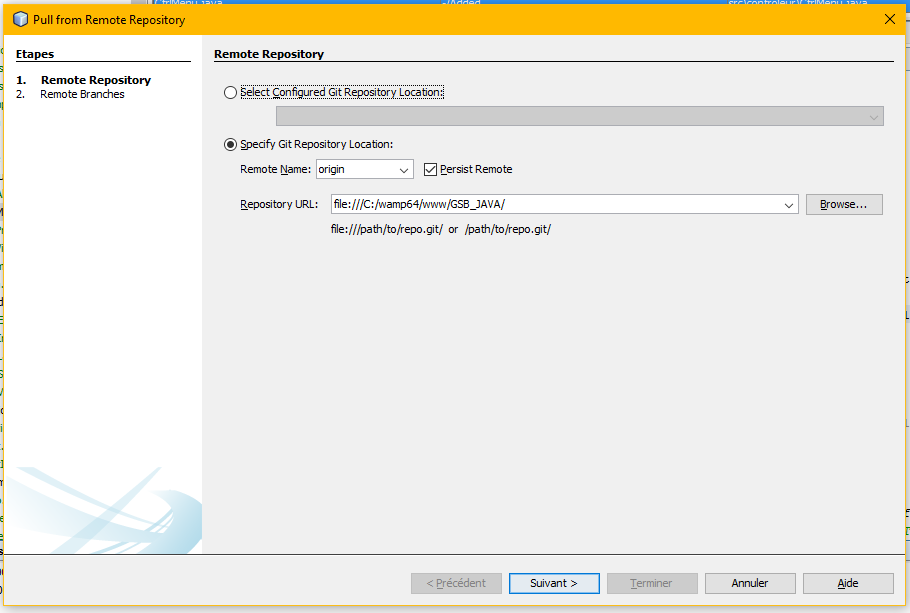
Lors de l’extraction de certaines mises à jour à partir d’un dépôt Git distant, les modifications sont extraites de celui-ci et fusionnées dans le HEAD actuel de votre répertoire local.

Pour effectuer le pulling, procédez comme suit :

Choisissez Team -> Remote -> Pull.



L’assistant Pull from Remote Repository s’affiche.



Dans la page Remote Repository de l’assistant, Select Configured Git Repository Location pour utiliser le chemin d’accès au répertoire configuré précédemment, ou l’option Specify Git Repository Location pour définir le chemin d’accès à un référentiel distant inaccessible, son nom, Login, mot de passe et configuration proxy si nécessaire.

Cliquez sur suivant.

Dans la page de l’assistant Remote Branches, choisissez les branches à extraire et cliquez sur Terminer.

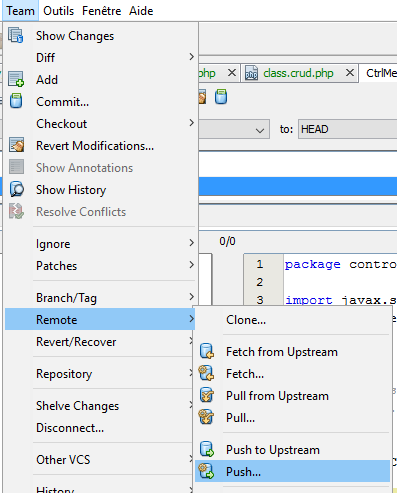
Votre répertoire local est synchronisé avec le répertoire d’origine.

### Pushing

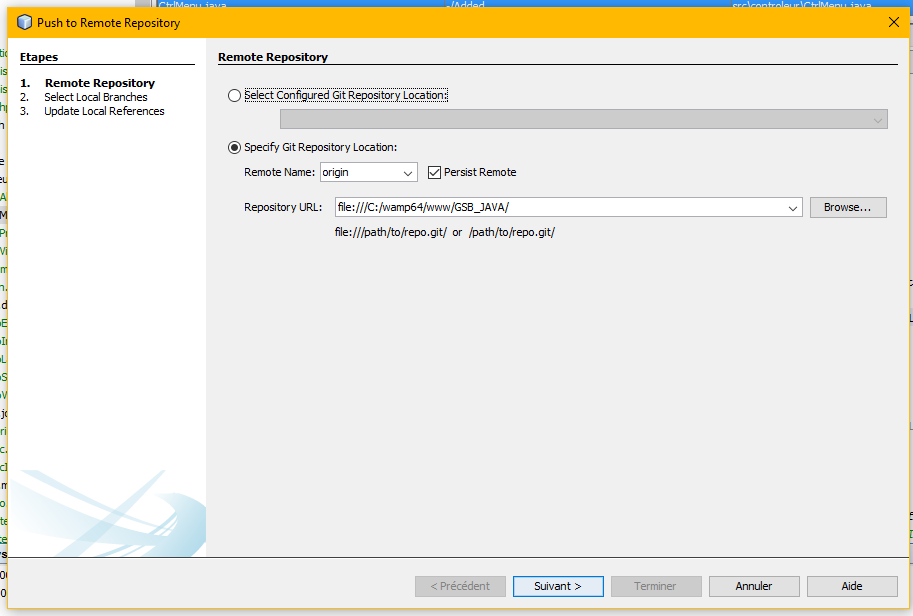
Pour apporter des modifications à partir de votre dépôt Git local dans un répertoire Git public, procédez comme suit :

Remarque : Lorsque vous allez push vos commits dans un répertoire et que d’autres modifications ont été push en même temps, vous devez d’abord les fusionner.

Choisissez Team -> Remote -> Push.



L’assistant Push to Remote Repository s’affiche.



Dans la page Remote Repository de l’assistant, Select Configured Git Repository Location pour utiliser le chemin d’accès au répertoire configuré précédemment, ou l’option Specify Git Repository Location pour définir le chemin d’accès à un référentiel distant inaccessible, son nom, Login, mot de passe et configuration proxy si nécessaire.

Cliquez sur suivant.

Dans la page Select Local Branches, choisissez les branches pour push vos modifications, puis cliquez sur Suivant.

Dans la page Update Local References, choisissez les branches à mettre jour dans le répertoire Remotes du répertoire local et cliquez sur Terminer.

La branche de répertoire distant spécifiée est mise à jour avec l’état le plus récent de votre branche locale.