— Gruppenarbeit 1 (P): Mercurial —

Aufgabe 1: Alle **(Mercurial Tutorium)**

Was sind Versionskontrollsysteme?

Da Sie die praktischen Übungen dieser Vorlesung überwiegend in Gruppenarbeit lösen werden, benötigen Sie ein Werkzeug, das Sie beim gemeinsamen Bearbeiten der Quellcodes und beim Austausch der Arbeitsergebnisse unterstützt. Insbesondere brauchen Sie ein Werkzeug, dass es Ihnen ermöglicht, einzelne Teilaufgaben unabhängig voneinander zu bearbeiten, ohne dass Sie sich Ihre Arbeitsergebnisse per E-Mail schicken und manuell abgleichen müssen. Natürlich sind derartige Anforderungen für jedes Softwareprojekt gegeben, an dem mehrere Entwickler arbeiten, weshalb der Einsatz von Versionskontrollsystemen heute Standard bei der Softwareentwicklung ist.

Die Idee hinter der Versionskontrolle ist, dass eigentlich jede Software aus mehreren Quellcodedateien besteht, zu denen nicht nur der gerade aktuelle Inhalt sondern auch eine komplette Historie aller Änderungen verwaltet werden soll, so dass man zu jeder Zeit nachvollziehen kann, welche Änderungen zu welchem Zeitpunkt vorgenommen wurden. Zum Beispiel ermöglicht es ein Versionskontrollsystem herauszufinden, wann bestimmte Zeilen einer Datei hinzugefügt oder entfernt wurden und von wem die Änderung vorgenommen wurde. Ebenso ermöglichen es heutige Systeme auch, alternative Versionen einer oder mehrerer Dateien zu verwalten, zum Beispiel wenn ein neues Feature erprobt werden soll, ohne die Originaldateien zu verändern, oder wenn eine alte Version der Software selbst nach Erscheinen einer neuen Version weiterhin gewartet werden muss.[1]

Eines der ersten Versionskontrollsysteme war RCS, das bereits Anfang der 1980er-Jahre entwickelt wurde und sich darauf beschränkte, die Historie einzelner, lokaler Dateien zu verwalten. Hierfür legte das Programm einfach zu jeder Datei eine gleichnamige Datei mit der Endung `,v` an, welche sämtliche Versionsinformationen enthielt.[2] Dieses System konnte natürlich nur von einem Benutzer an einem Computer benutzt werden, weshalb später mit CVS ein Netzwerkaufsatz für RCS erschien. Dieser ermöglichte es immerhin, die Versionsdateien auf einem zentralen Server abzulegen, so dass von nun an mehrere Entwickler gleichzeitig arbeiten konnten.

Eine logische Weiterentwicklung von CVS war unter Anderem Subversion, das es erstmals ermöglichte, ganze Verzeichnisse mitsamt allen Dateien und Unterverzeichnissen zu versionieren, was bisher nicht möglich war. Dabei setzt auch Subversion auf einem zentralen Server auf, der sämtliche Historiendaten (die sog. Revisionen) verwaltet. Als Entwickler haben Sie daher immer nur eine Arbeitsversion der Quellcodes auf Ihrem Computer. An dieser nehmen Sie alle Änderungen vor und wenn Sie fertig sind, übertragen Sie alle Änderungen an den Server. Dieser Vorgang wirt `commit` genannt und führt dazu, dass auf dem Server eine neue Revision der Quellcodes entsteht. Umgekehrt müssen Sie natürlich alle Änderungen, welche von den anderen Teammitgliedern vorgenommen wurden, auf Ihren Rechner übertragen, was `checkout` genannt wird.

Im Prinzip würde Subversion ausreichen, damit Sie gemeinsam die Übungsaufgaben dieser Vorlesung bearbeiten können. Allerdings besitzt Subversion den Nachteil, dass es nur mit einer aktiven Internetverbindung funktioniert, da Sie sonst keine Änderungsinformationen vom Server abrufen oder dorthin übertragen können. Wenn man also für längere Zeit keinen Internetzugang besitzt, zum Beispiel im Zug oder während des Urlaubs, kann man Subversion und somit die Vorteile der Versionskontrolle nicht verwenden. Aus diesem Grund werden wir in der Vorlesung Mercurial verwenden, das ein sogenanntes, verteiltes Versionskontrollsystem darstellt. Es unterscheidet sich von Subversion dadurch,

[1] Deraritge Alternativversionen werden meistens Branches genannt

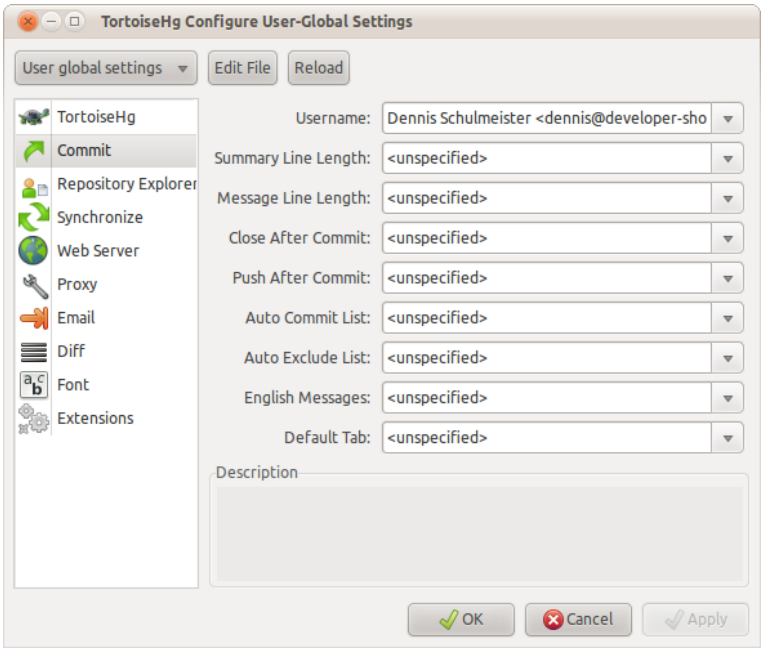
[2] Ja, es ist wirklich ein Komma. Vgl. http://www.burlingtontelecom.net/ ashawley/rcs/tichy1985rcs/html/

das ein zentraler Server für die Arbeit nicht notwendig ist. Stattdessen wird die komplette Historie auf den Computern aller Teammitglieder verwaltet, so dass für die Arbeit keine Internetverbindung nötig ist. Wie Sie in der nächsten Aufgabe sehen werden, können Sie aber trotzdem auf ein zentral gehostetes Repository zugreifen, um Ihre Quellcodes mitsamt der kompletten Änderungshistorie online abzugleichen.

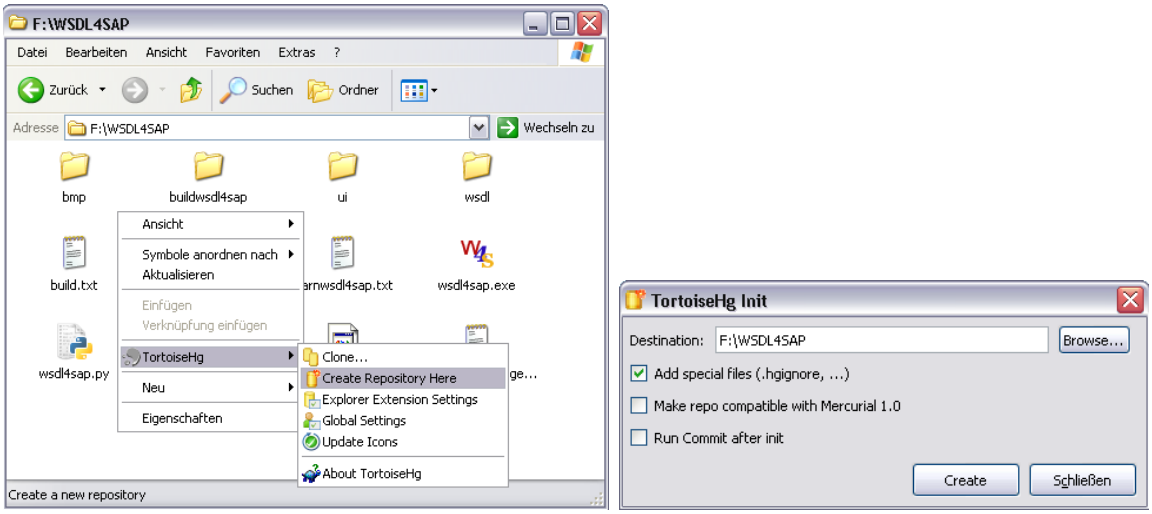
Los geht's! Hallo, Mercurial.

Besuchen Sie die Webseite http://mercurial.selenic.com/ und laden Sie sich dort TortoiseHG für Ihr Betriebssystem herunter. Es handelt sich dabei um eine graphische Version von Mercurial, die besonders unter Microsoft Windows häufig verwendet wird. Wenn Sie wollen können Sie aber auch die Kommandozeilenversion von Mercurial herunterladen. Bei den nachfolgenden Beschreibungen führen Sie dann stattdessen das entsprechende Kommando wie `hg init` oder `hg commit` aus, statt der Anleitung zu folgen. Nach der Installation von TortoiseHG finden Sie im Kontextmenü des Explorers ein neues Untermenü mit demselben Namen. Dieses beinhaltet Aufrufe der wichtigsten Befehle, so dass Sie direkt aus dem Explorer heraus Mercurial benutzen können.

Zunächst müssen Sie Mercurial erst Ihren Benutzernamen mitteilen, wobei Sie sich hier einen beliebigen Namen ausdenken können. Üblich ist allerdings die Konvention `Vorname Nachname <email@adresse.de>`. Öffnen Sie hierzu einen beliebigen Ordner und lassen Sie sich per Rechtsklick das Kontextmenü anzeigen. Dort wählen Sie den Eintrag `TortoiseHG` ⇒ `Global Settings` aus. Im daraufhin erscheinenden Fenster wechseln Sie zur Rubrik `Commit` und tragen in das Feld `Username` Ihren Benutzernamen ein. Anschließend können Sie das Fenster schließen.

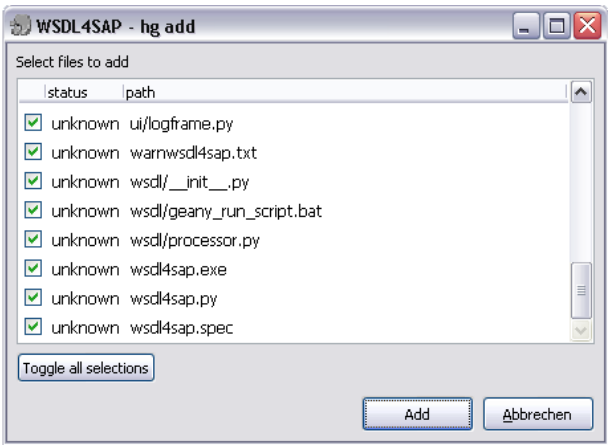


Nun wird es Zeit, dass Sie Ihr erstes Repository anlegen, in welchem Sie Dateien verwalten können. Erstellen Sie hierfür einen Ordner und kopieren Sie ein paar Dateien hinein, zum Beispiel die Musterlösung zu einer Aufgabe. Anschließend wählen Sie folgenden Menüeintrag aus: `Kontextmenü` ⇒ `TortoiseHG` ⇒ `Create Repository Here`. Es erscheint ein kleines Fenster, dass Sie mit dem Button `Create` schließen können.



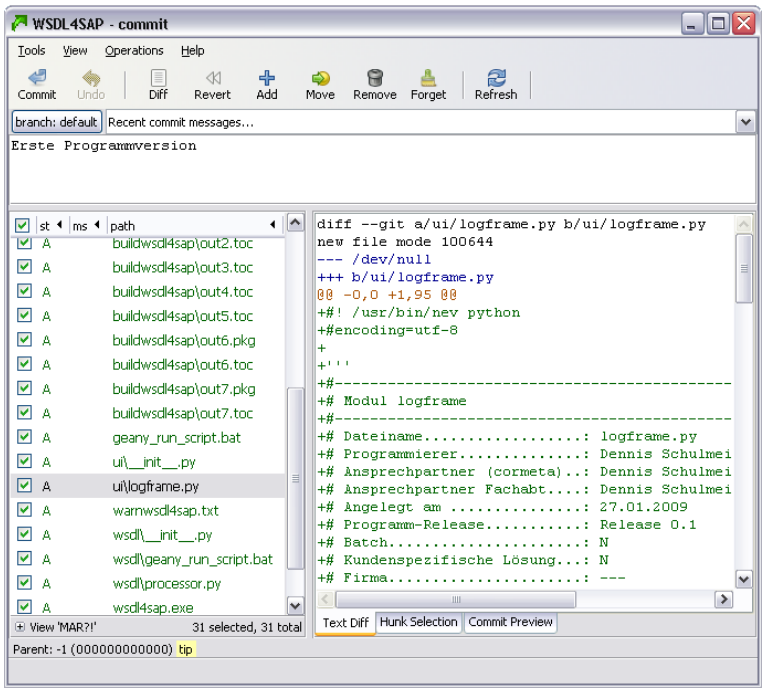
Mercurial legt daraufhin den Unterordner `.hg` und die Datei `.hgignore` an. Der Ordner `.hg` beinhaltet alle Verwaltungsdaten von Mercurial und sollte daher niemals verändert werden. Sie riskieren sonst, das Repository zu zerstören. Die Datei `.hgignore` ist hingegen zu Anfang leer und kann von Ihnen bearbeitet werden, wenn Sie bestimmte Dateien aus der Versionsverwaltung ausschließen wollen. Zum Beispiel macht es Sinn, `class`-Dateien auszuschließen, da diese Dateien nur beim Kompilieren entstehen und nicht manuell bearbeitet werden. Klicken Sie darum mit der Rechten Maustaste auf eine Datei und wählen Sie im Kontextmenü folgenden Eintrag aus: `TortoiseHG` ⇒ `Edit Ignore Filter`. In das Feld `Glob` geben Sie dann `**.class` ein und klicken neben dem Eingabefeld auf `Add`. Kompilierte Klassen werden somit von der Versionskontrolle ausgeschlossen, egal in welchem Unterverzeichnis sie sich befinden. Danach können Sie das Fenster wieder schließen.

Damit Ihre Dateien nun wirklich von Mercurial verwaltet werden, müssen Sie diese dem Repository hinzufügen. Hierfür öffnen Sie wieder das Kontextmenü einer bestehenden Datei und wählen folgenden Menüeintrag aus: `TortoiseHG` ⇒ `Add Files`. Falls Ihr Verzeichnis neue Dateien beinhaltet, die noch nicht von Mercurial verwaltet werden, erscheint ein Fenster, dass alle Dateien anzeigt, die hinzugefügt werden können. In der Regel können Sie dabei alle Häkchen gesetzt lassen, es sei denn, Sie wollten einzelne Dateien nicht hinzufügen. Wenn Sie auf `Add` drücken, verschwindet das Fenster wieder und die Dateien stehen von nun an unter Versionskontrolle.



Bevor Sie jedoch Änderungen an den Dateien vornehmen, sollten Sie erst einmal eine neue Revision anlegen, zu der Sie unter Umständen zurückkehren können, falls Sie beim Bearbeiten der Dateien

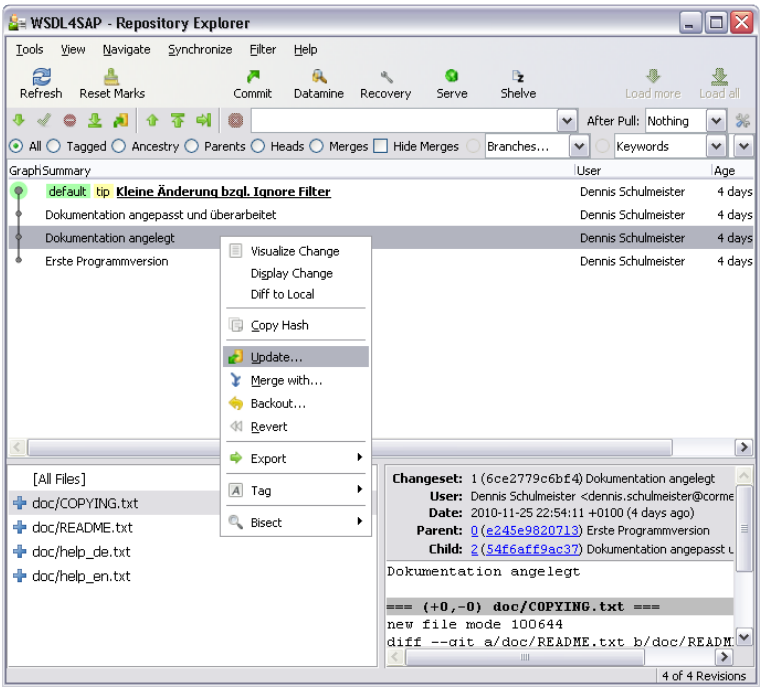
etwas falsch machen. Klicken Sie daher mit der rechten Maustaste in den freien Bereich Ihres Ordners und wählen Sie den Menüpunkt `Commit` aus. Es erscheint ein Fenster, welches alle von Ihnen vorgenommenen Änderungen anzeigt. Dabei beinhalten die Änderungen beim ersten `Commit` immer die Inhalte aller Dateien, die die Änderung im Hinzufügen der Dateien besteht. Ab dem zweiten `Commit` sehen Sie nurnoch die tatsächlich veränderten Zeilen. Geben Sie nun in das oben sichtbare Eingabefeld eine Nachricht ein, welche Ihre Änderungen beschreibt. Anschließend klicken Sie in der Toolbar auf `Commit`.



Sie haben eben eine neue Revision in Ihrem Repository angelegt. Von nun an können Sie sämtliche Dateien des Arbeitsverzeichnisses bearbeiten und Ihre Änderungen per `Commit` historisch abspeichern. Sie müssen nurnoch darauf achten, den Menüpunkt `Add Files` aufzurufen, wenn Sie eine neue Datei angelegt haben, die ebenfalls in das Repository aufgenommen werden soll. Denn Mercurial fügt dem Repository niemals von alleine neue Dateien hinzu. Diese muss immer der Anwender machen.

Nehmen Sie nun mehrmals Änderungen an den einzelnen Dateien vor und führen Sie auch mehrere Commits durch. Dabei spielt es für diese Übung keine Rolle, ob Ihre Änderungen sinnvoll sind, oder ob kompilierbarer Code dabei herauskommt. Es geht nur darum zu sehen, wie Mercurial verschiedene Revisionen Ihrer Dateien anlegt und wie Sie zu einem alten Stand zurückkehren können.

Klicken Sie wieder mit der rechten Maustaste in den freien Bereich Ihres Arbeitsverzeichnisses und wählen Sie den Menüeintrag `Hg Repository Browser` aus. Es erscheint ein Fenster, dass Ihnen graphisch alle Revisionen Ihres Repositories anzeigt. Dabei können Sie zu jeder Revision das sogenannte `Changeset` einsehen, also alle Änderungen, welche in die jeweilige Revision eingeflossen sind. Diese sehen Sie unten rechts in Form einer `Diff`-Datei. Um nun zu einer alten Revision zurückzukehren, klicken Sie diese erst mit der linken und dann mit der rechten Maustaste an und wählen im Kontextmenü `Update` aus. Es erscheint eine Sicherheitsabfrage, die Sie einfach bestätigen können. Mercurial nimmt dann alle notwendigen Änderungen vor, um das Arbeitsverzeichnis zum ausgewählten Stand zurückzurollen.



Die `Update`-Aktion selbst ist übrigens auch reversibel. Das heißt, wenn Sie einmal zu einer alten Revision zurückgekehrt sind, können Sie jederzeit auch wieder zu einer neueren Revision wechseln. Einzige Voraussetzung ist nur, dass Sie vor dem `Update` ein `Commit` ausführt haben, damit Ihre bis dahin gemachten Änderungen nicht verloren gehen.

Zu guter letzt probieren Sie noch ein kleines Gimmick von Mercurial aus – den eingebauten Webserver. Klicken Sie hierfür wieder in den freien Bereich Ihres Ordners und wählen Sie folgenden Eintrag: `TortoiseHG` ⇒ `Web Server`. Es erscheint ein Fenster, dass Sie nach der Portnummer fragt, auf welchem der Webserver antworten soll. Ebenso sehen Sie in diesem Fenster ein Protokoll aller beantworteten Zugriffe. Belassen Sie die Portnummer bei `8000` und klicken Sie in der Toolbar auf `Start Server`. Anschließend rufen Sie im Browser folgende URL auf: `http://localhost:8000`. Sie sehen nun eine Webversion Ihres Repositories, in der Sie lesend auf die enthaltenen Daten zugreifen können.

Zusammenfassung

- Für jede Aufgabe müssen Sie ein neues Repository erzeugen
- `class`-Dateien sollten Sie von der Versionsverwaltung ausschließen
- Neue Dateien müssen Sie mit `Add` dem Repository hinzufügen
- Anschließend wird jede größere Änderung mit `Commit` bestätigt
- Mit `Revert` können Sie alte Revisionen wieder herholen

In der nächsten Aufgabe kommen noch die Schritte `Clone`, `Push` und `Pull` hinzu, um ihr lokales Repository mit einer zentral gehosteten Version abzugleichen.

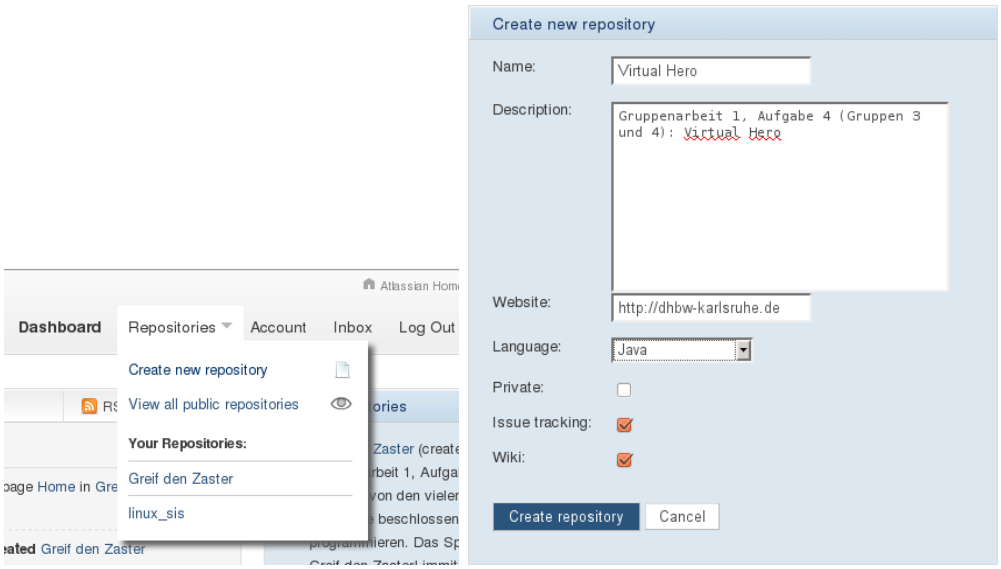
Aufgabe 2: Alle **(Bitbucket Account)**

Der Bitbucket Onlinedienst

Viel wichtiger als die lokale Aufzeichnung einer Änderungshistorie ihrer Programme ist die Möglichkeit Mercurials, auf ein im Internet gehostetes Repository zuzugreifen, so dass alle Mitglieder Ihrer Gruppe mit denselben Dateien arbeiten können. Den hierfür benötigten Server könnten Sie natürlich selbst aufsetzen, es ist jedoch einfacher, den kostenlosen Dienst von Bitbucket zu benutzen. Deshalb besteht der erste Teil dieser Aufgabe darin, dass sich alle Mitglieder Ihrer Gruppe bei `http://www.bitbucket.org` registrieren:

1. Besuchen Sie die Seite `http://www.bitbucket.org`
2. Klicken Sie auf den Link `Pricing & Signup`
3. Wählen Sie den kostenlosten Zugang aus
4. Melden Sie sich mit einem beliebigen Benutzernamen[3] an

Anschließend können Sie sofort neue Repositories anlegen, wobei Sie darauf achten sollten, dass Sie je Aufgabe nur ein Repository anlegen, auf das alle Mitglieder Ihrer Gruppe zugreifen können. Öffnen Sie hierfür das `Repositories`-Menü und klicken auf `Create new repository`. Es erscheint ein Formular, welches Sie nach dem Namen des neuen Repositories und ein paar Zusatzdaten fragt. Schicken Sie das Formular ab, um den Vorgang abzuschließen.

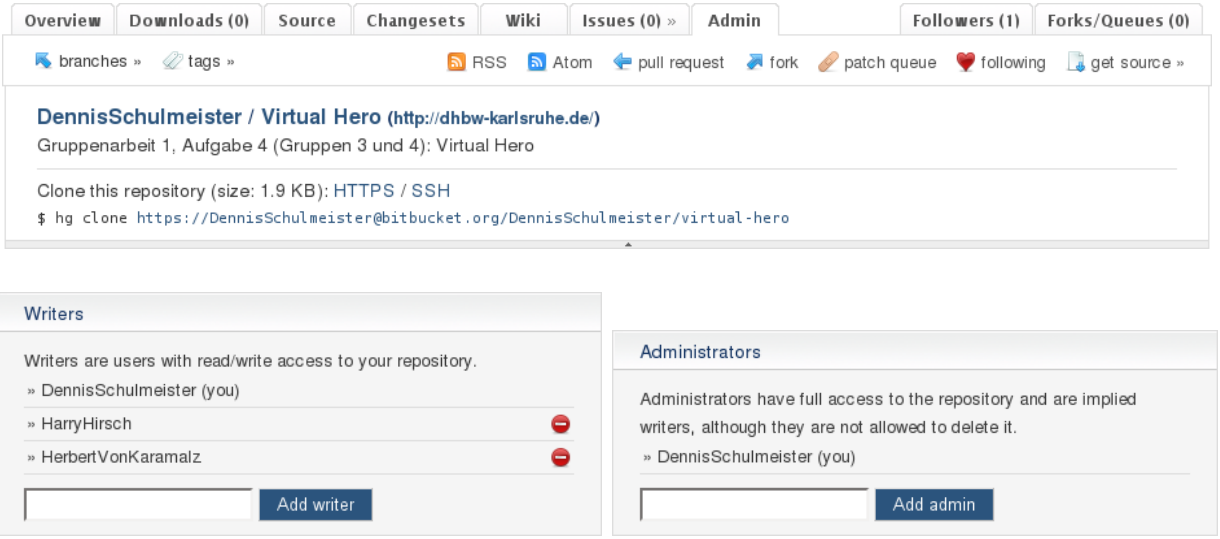


Anschließend sehen Sie das angelegte, noch leere Repository, dessen Anzeige in mehrere Tabreiter unterteilt ist. Im Reiter `Overview` sehen Sie später eine Zusammenfassung der letzten Änderungen. Die Inhalte selbst sehen Sie dann im Reiter `Source`. Ebenso sehen Sie einen Reiter `Admin`, der es Ihnen ermöglicht, verschiedene Einstellungen am Repository vorzunehmen. Diesen müssen Sie öffnen, um den anderen Mitgliedern Ihrer Gruppe Schreibrechte auf das Repository zu gewähren. Andernfalls

[3] Verwechseln Sie diesen Benutzernamen nicht mit dem Benutzernamen von Mercurial. Dieser Name identifiziert Sie nur bei Bitbucket und hat sonst keine weitere Bedeutung. Der Mercurialbenutzer hingegen identifziert Sie als Person und taucht daher in den Protokollen auf.

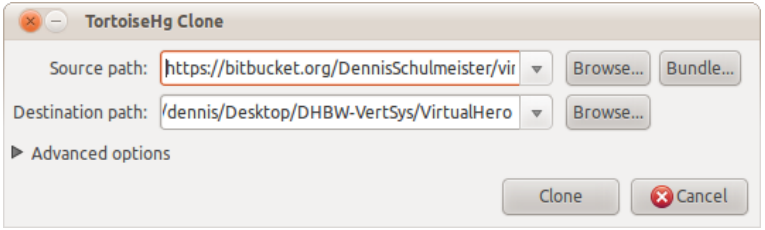
können sie zwar das Repository sehen (falls es öffentlich ist), aber keine Änderungen an seinen Inhalten vornehmen.

Öffnen Sie den `Admin`-Tab Ihres neuen Repositories und blättern Sie etwas nach unten. Rechts sehen Sie drei Kästchen namens `Readers`, `Writers` und `Administrators`. Dort müssen Sie alle Bitbucket-Benutzer eintragen, denen Sie Zugriff auf das Repository gewähren wollen. Dabei können Sie das Kästchen `Readers` nur für private Repositories ändern, da öffentliche Repositories ohnehin jedem Benutzer angezeigt werden. Fügen Sie die anderen Benutzer Ihrer Gruppe wahlweise dem Kästchen `Writers` oder `Administrators` hinzu, um ihnen Änderungsrechte einzuräumen.



Mit dem Repository arbeiten

Um nun mit dem Repository arbeiten zu können, muss jedes Mitglied Ihrer Gruppe das Repository klonen, was bedeutet, dass sich jeder eine vollständige Kopie des Repositories herunterlädt. Diesen Vorgang, den Sie nicht mit dem Reiter `Download` verwechseln dürfen, führen Sie nicht in der Webanwendung aus, sondern mit TortoiseHG auf Ihrem Rechner. Erzeugen Sie daher einen neuen Ordner auf Ihrem Rechner, der das Repository beinhalten soll. Dann öffnen Sie das Kontextmneü und wählen folgende Eintrag aus: `TortoiseHG` ⇒ `Clone ...`. Es öffnet sich ein Fenster mit zwei Eingabefeldern für `Source Path` und `Destination Path`. In das obere Feld tragen Sie die Webadresse Ihres Repositories ein, in das untere Feld das Zielverzeichnis auf Ihrem Rechner.

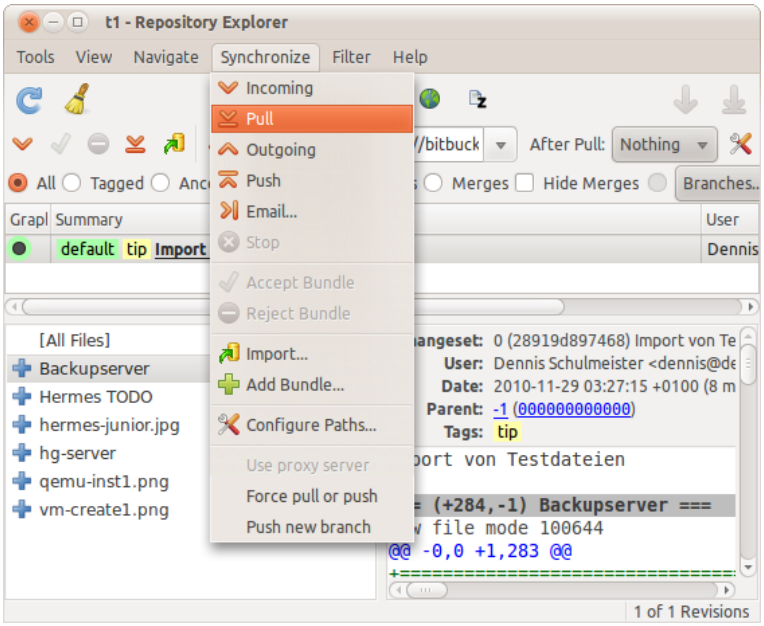


Wenn alles geklappt hat, sehen Sie nun das Unterverzeichnis `.hg`, das Ihnen anzeigt, dass Sie sich in einem lokalen Repository befinden. Dabei handelt es sich um ein gewöhnlichens Repository, dass Sie wie in Aufgabe 1 beschreiben nutzen. Das heißt, Sie können mit `Add` neue Dateien hinzufügen und mit `Commit` alle Änderungen, die Sie vorgenommen haben, lokal bestätigen. Doch zuvor sollten

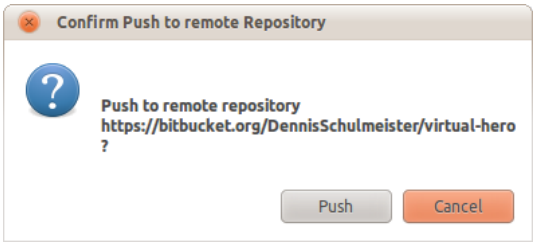
Sie den `Ignore Filter` bearbeiten, so dass die kompilierten Javaklassen nicht in das Repository aufgenommen werden.

Gemäß der Funktionsweise von Mercurial wirken sich alle Änderungen, die Sie an Ihrer Version des Repositories vornehmen, nicht automatisch auf die Version bei Bitbucket aus. Stattdessen müssen Sie, wenn Sie Ihre Ergebnisse mit dem Team teilen wollen, die Operation `Push`ausführen, um Ihren Revisionsstand sowie alle Änderungen, die zu dem Stand geführt haben, an Bitbucket übertragen. Die Anderen Teilnehmer der Gruppe müssen dann einen `Pull` ausführen, im ihre Version des Repositories mit Bitbucket abgleichen.

Beide Funktionen werden im `Repository Browser` ausgeführt, den Sie über das Kontextmenü aufrufen können. Dort wählen Sie im Menü `Synchronize` entweder `Push` oder `Pull` aus, jenachdem, ob Sie Ihre Änderungen an Bitbucket übertragen oder die Änderungen der anderen Benutzer abrufen wollen.

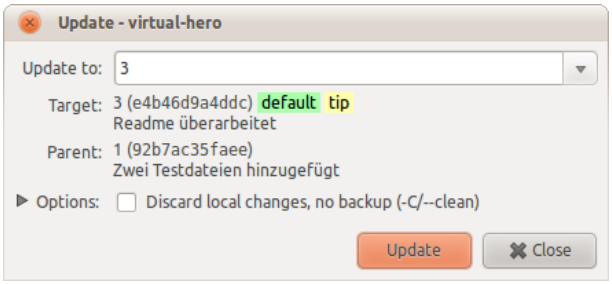


Übertragen Sie zunächst Ihre Änderungen an Bitbucket, indem Sie den Menüeintrag `Push` auswählen. Es folgt eine Sicherheitsabfrage, ob Sie Ihre Änderungen wirklich übertragen wollen. Nachdem Sie diese Meldung bestätigt haben, fängt Mercurial an, die zu übertragenden Änderungen zu ermitteln. Anschließend werden diese an Bitbucket gesendet. Hierfür müssen Sie dann Benutzernamen und Passwort von Bitbucket eingeben.



Nun wählen Sie den Menüpunkt `Synchronize` ⇒ `Pull` aus, um die von den anderen Teamkollegen durchgeführten Änderungen auf Ihren Computer zu übertragen. Falls relevante Änderungen gefunden wurden, erscheinen diese nun in der `Commit`-Hitorie des `Repository Browsers`, andernfalls taucht

in der Statuszeile nur der Hinweis `No changesets to pull` auf. Beachten Sie, dass der Abgleich mit Bitbucket nicht automatisch zur Aktualisierung der Dateien in Ihrem Repository führt. Denn technisch gesehen haben Sie nur die Änderungsinformationen von Bitbucket abgerufen, ohne die aktive Revision zu wechseln. Das heißt, Sie müssen, wie in Aufgabe 1 beschrieben, erst ein `Update` auf die neue, zu verwendende Revision machen, damit die übertragenen Änderungen aktiv werden. Dies mag zunächst umständlich erscheinen, ermöglicht es Ihnen aber, jederzeit wieder zu einer Ihrer eigenen Revisionen zurückzukehren, wenn Sie mit den Änderungen Ihrer Kollegen nicht einverstanden sind.



Probieren Sie es aus: Klonen Sie ein Repository, das ein Mitglied Ihrer Gruppe angelegt hat und nehmen Sie alle gleichzeitig Änderungen daran vor. Das heißt, fügen Sie Dateien hinzu und fangen Sie an, die Dateien der anderen Mitglieder zu bearbeiten. Vergessen Sie nicht, regelmäßig Ihre Änderungen mit einem `Commit` lokal zu bestätigen und anschließend alle Änderungen mit `Push` an Bitbucket zu übertragen, bzw. alle Änderungen von Bitbucket per `Pull`/`Update` herunterzuladen. Beobachten Sie auch, wie sich die Weboberfläche von Bitbucket nach jedem `Push` verändert.