

Die Squeak Versionsverwaltung Monticello

Eine kleine Einführung

Stand 04.05.06
von Patty Gadegast

Dies ist eine kurze Einführung für den Umgang mit Monticello, dem Squeak Versionsmanagementsystem. Wozu man ein solches Versionsverwaltungssystem benötigt, wird jeder spätestens nach dem ersten größeren Software-Projekt wissen, bei dem er mit mehreren anderen Programmierern zusammengearbeitet hat und die erstellten Quellcode-Schnipsel zusammenfügen durfte. Daher wird der Zweck eines solchen kooperativen Systems nicht weiter diskutiert und die Notwendigkeit vorausgesetzt.

An wen richtet sich diese Einführung? Und was sollte man schon wissen?

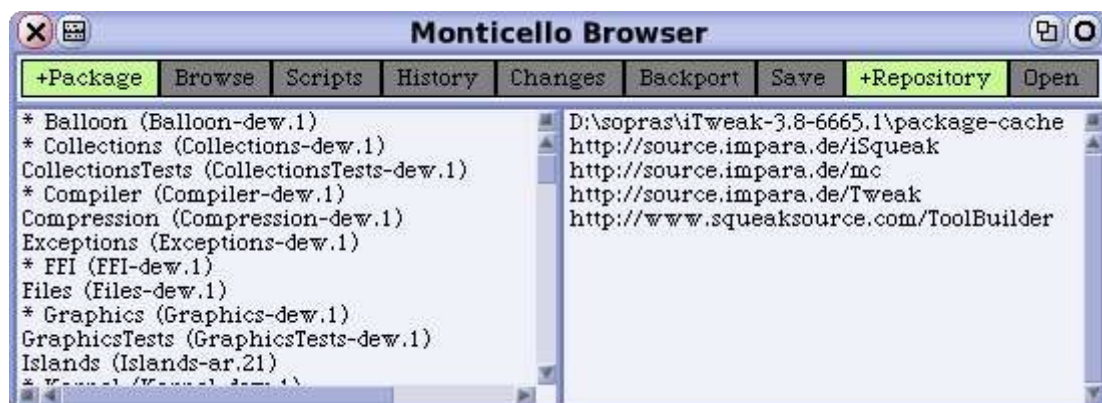
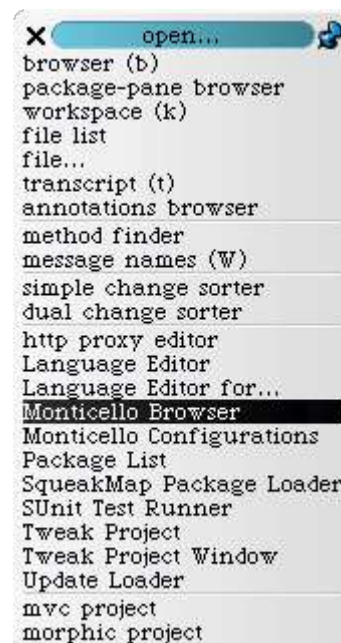
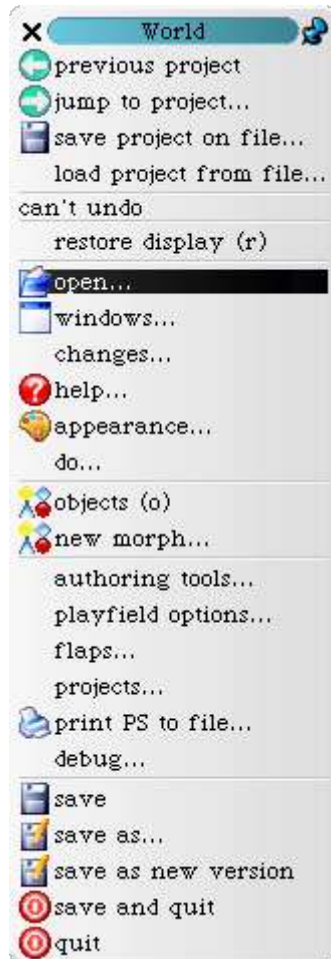
Diese Einführung ist für diejenigen gedacht, die zum ersten Mal ein größeres Softwareprojekt mit Squeak und mehreren Beteiligten realisieren wollen. Die Einführung erhebt keineswegs den Anspruch auf Vollständigkeit, sie dient lediglich dazu, einen Einstieg in den Umgang mit Monticello zu ermöglichen und über die erste Hürde des leeren Bildschirms der Squeak-Entwicklungsumgebung hinweg zu helfen. Eigentlich sollte diese Hürde sogar schon genommen worden sein, bevor Monticello zum Tragen kommt, es müssen Daten, also Quellcode da sein, bevor dieser das Bedürfnis weckt, ausgetauscht werden zu müssen ...
Grundlegende Kenntnisse der Programmierung in Squeak und der Handhabung der Entwicklungsumgebung Squeak werden also vorausgesetzt.

Was kann Monticello und was kann es nicht?

Monticello ist ein von Squeak zur Verfügung gestelltes Versionsverwaltungssystem von Squeak-Sourcecode, für die Verwaltung anderer Daten wie Assets, Sounds etc. müssen andere Systeme genutzt werden (z.B. Subversion).

Wo finde ich den Monticello-Browser?

...die einfachste Möglichkeit: im Weltmenü im Öffnen-Menü wird auch der Monticello-Browser angeboten. Wählt man diesen aus, so öffnet sich der Monticello-Browser...



Wer, wie, was – Repositories, Pakete und Versionen

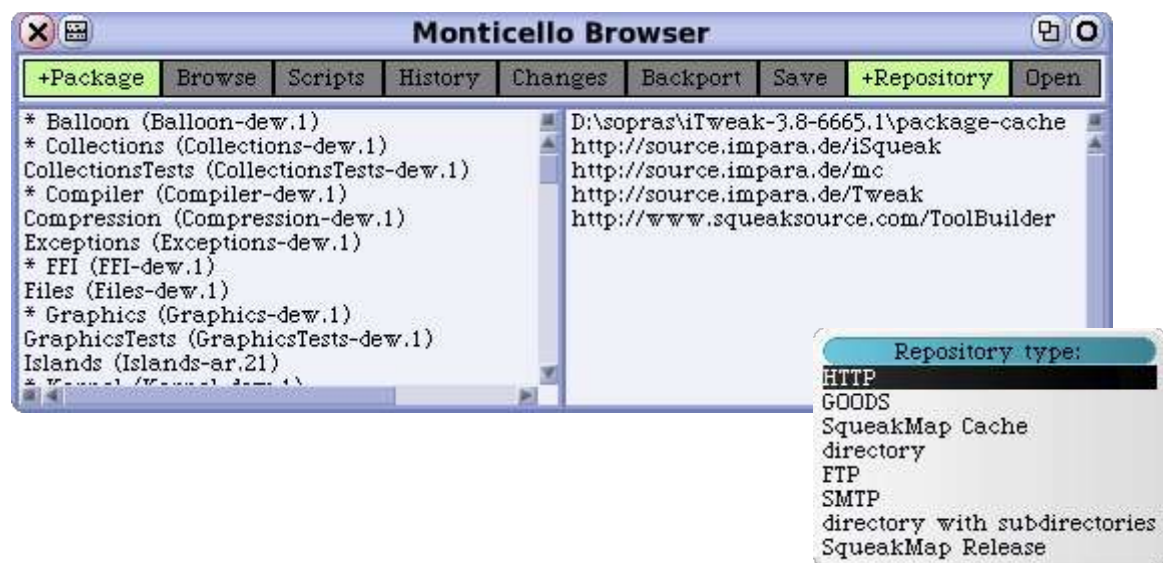
Der Monticello Browser ermöglicht den Zugriff auf sogenannte Repositories, Depots in denen der Sourcecode aufbewahrt wird. Diese Repositories befinden sich auf der eigenen Festplatte oder irgendwelchen Servern.

In den Repositories liegen die Daten in Form von Paketen.

Um existierende Datenpakete, genauer Sourcecode-Pakete, zu laden bzw. um eigene Softwarepakete zu sichern und zu verwalten, muss erst einmal der Zugang zu einem Repository geschaffen werden.

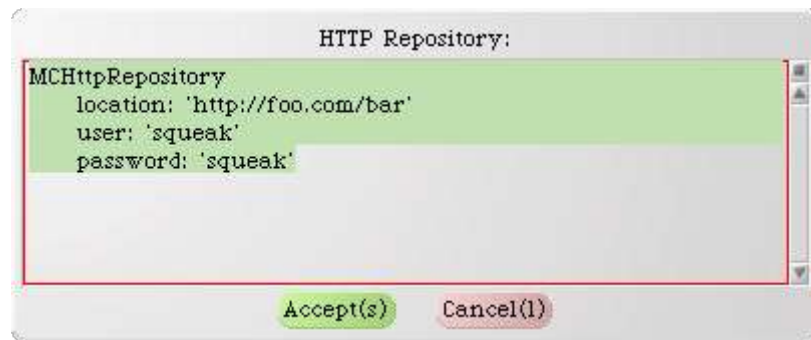
So wird ein neues Repository hinzugefügt

Auf den Button *Repository hinzufügen* (+Repository) klicken und dort einen Repository-Typ auswählen. Hier erkläre ich jetzt alles anhand eines *HTTP-Repositories*, da einige der wichtigen Repositories (<http://source.impara.de/Tweak> und <http://www.squeaksource.com/ToolBuilder>) eben HTTP-Repositories sind. Es gibt jedoch auch noch andere Typen.



Es öffnet sich ein weiteres Fenster, in dem die Zugangsdaten für das Repository eingetragen werden müssen: Adresse, Nutzernamen und Passwort...

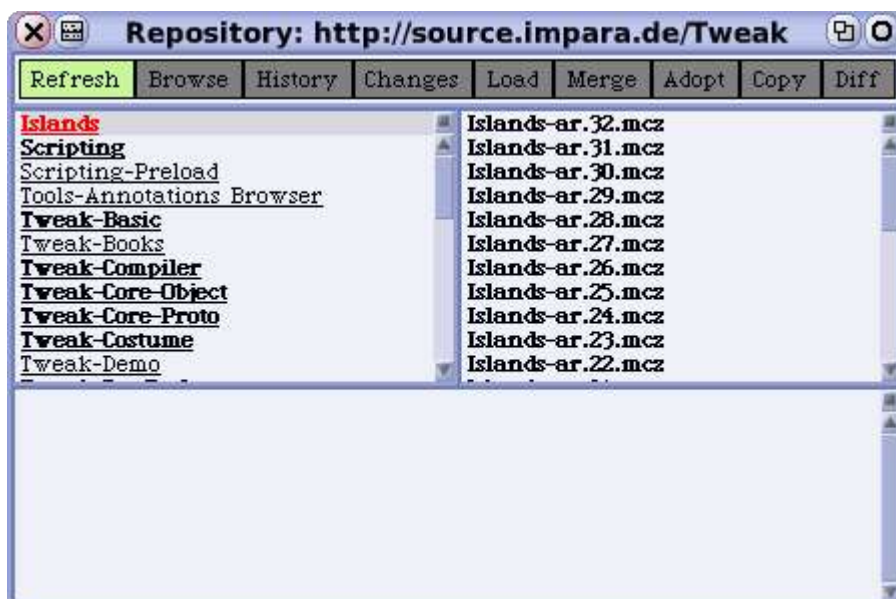
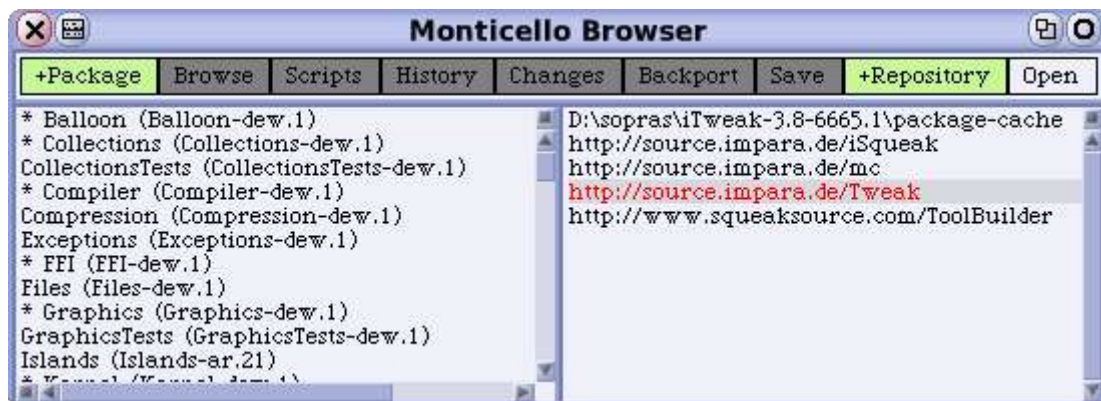
Bei fremden Repositories lässt man jedoch Nutzernamen und Passwort leer, zumal man sie auch oft nicht weiss...



Wenn das Repository angelegt ist, Adresse, Nutzernamen und Passwort korrekt sind, so wird einem der Zugriff auch nicht länger verwehrt. Sesam öffne dich...

Und wie kommt nun der Sourcecode aus dem Repository in mein Image?

Man nehme den Monticello-Browser, markiere das entsprechende Repository im rechten Fenster (man sollte schon wissen, wo was drin ist, sonst muss man eben alles durchsuchen...) und spreche die magische Formel: Sesam öffne dich...



Notation – Was will mir der Repository-Browser sagen?

Im linken Fenster werden alle Monticello-Pakete angezeigt, die in dem Repository abgelegt sind. Im rechten Fenster werden alle gespeicherten Versionen des ausgewählten Paketes angezeigt.

- unterstrichene Pakete und Versionen sind auf dem lokalen System bereits geladen, sind also bereits im Image vorhanden
- **fett** hervorgehobene Pakete und Versionen sind aktuellere Versionen von bereits geladenen Paketen oder Paketversionen aus dem lokalen Image
- normale (nicht unterstrichen und nicht fett!) Pakete sind nicht im lokalen Image vorhanden, normale Paketversionen sind ältere Versionen und sollten Vorgänger der aktuellsten Version sein

Monticello Pakete laden und mergen – Was mache ich wann (und warum)?

Wie bei so vielen Dingen im Leben, gibt es ein paar einfache Regeln – Konventionen! Halte dich dran, dann ersparst du dir so manchen Ärger...

1. Fremde Pakete, die nur genutzt werden sollen, werden nur **geladen**.

An ihnen solltest du auch nicht so ohne weiteres irgendetwas verändern. Das kannst du zwar auf deinem , meistens darfst du es auch gar nicht.

2. Eigene Pakete, an denen du mit anderen zusammen arbeitest, werden **gemerged**.

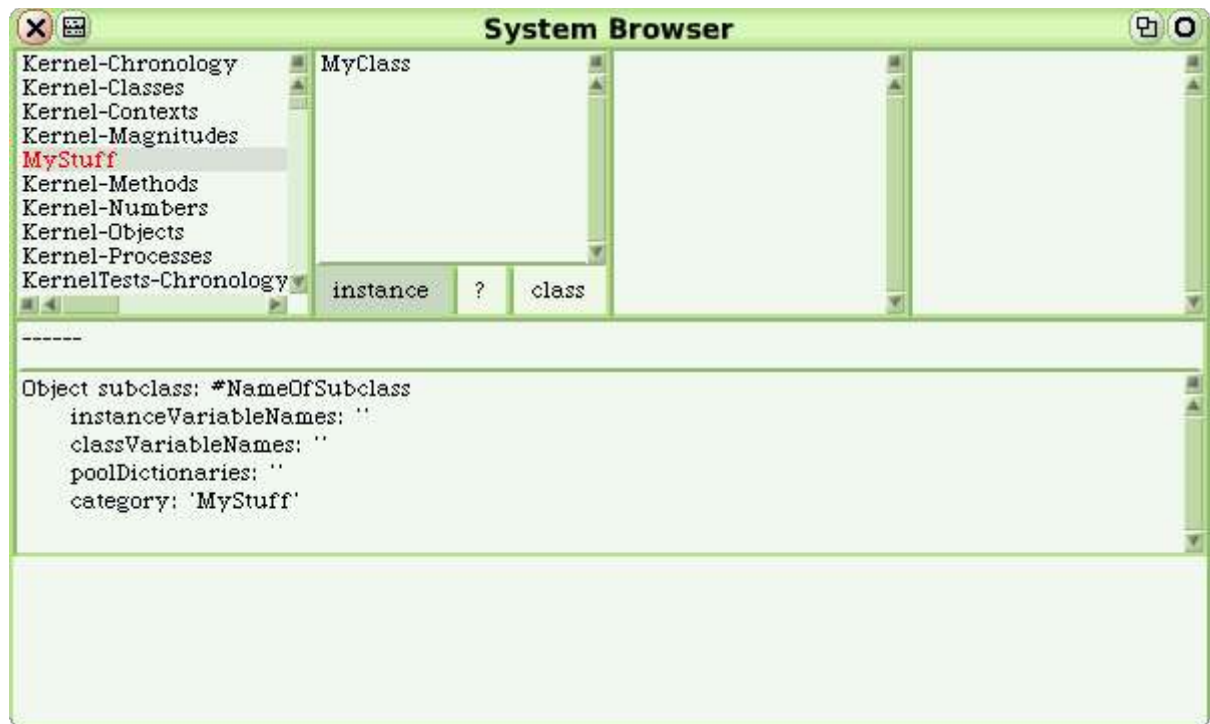
Dein lokales System sollte immer auf dem aktuellsten Stand gehalten werden. Nur so vermeidest du doppelte und vergebliche Arbeit, weil zwei Leute an derselben Sache arbeiten oder einer sich durch einen Fehler debuggt, den ein anderer schon längst gefixt hat. Oder auch einfach ein mühsames zusammenpuzzeln am Ende, bei dem oftmals auch einiges verloren geht.

Vom Sourcecode zum Monticello Paket auf dem Repository...

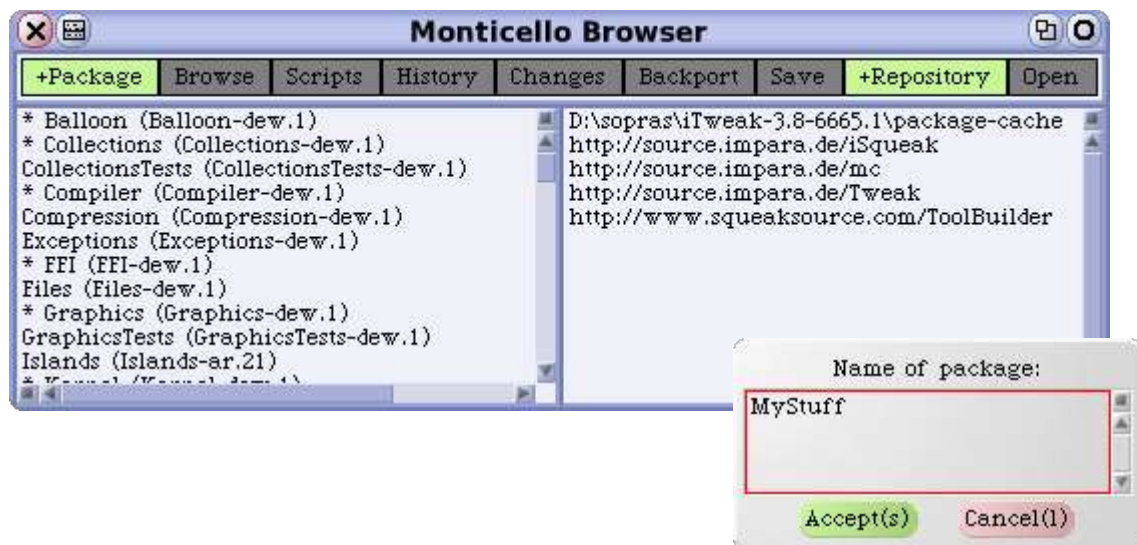
Pakete schnüren – Wie bekomme ich meinen eigenen Code ins Repository

Um dein eigenes Paket auf dem Repository anzulegen, musst du erst einmal eines auf deinem lokalen System zusammenschnüren.

Dein Quellcode sollte sich in einer Systemkategorie oder in mehreren mit einem gemeinsamen Präfix im Namen befinden.



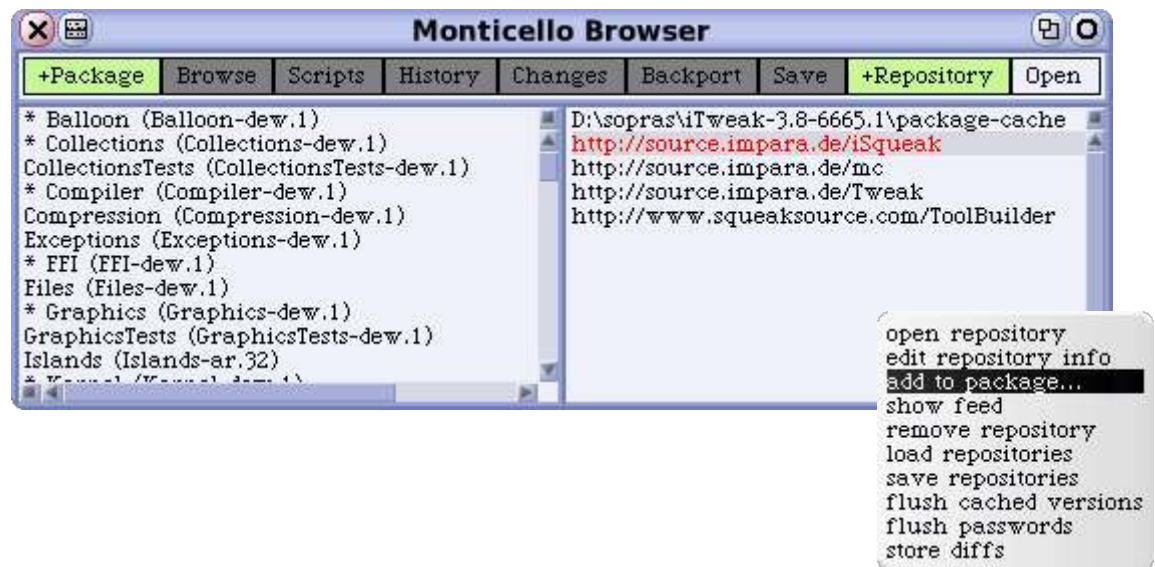
Im Monticello-Browser wird dann ein neues Paket angelegt. Klicke dazu auf den Button Paket hinzufügen (+Package) und benenne es nach der Systemkategorie oder dem Präfix der Kategorien.



Ist das Paket im Monticello angelegt, ist es geschnürt, aber noch keineswegs abgeschickt. Es muss noch zur Post gebracht werden und aufgegeben. Das Paket zur Post bringen heißt es ein Repository zuzuordnen, es aufzugeben heißt es hochzuladen – wie im richtigen Leben also ;).

1. Der Gang zur Post - das Paket einem Repository hinzufügen:

Das Repository auswählen, mit einem Rechte-Maustaste-Klick öffnet sich ein Menü. Dort lässt sich dem Repository ein Paket hinzufügen.



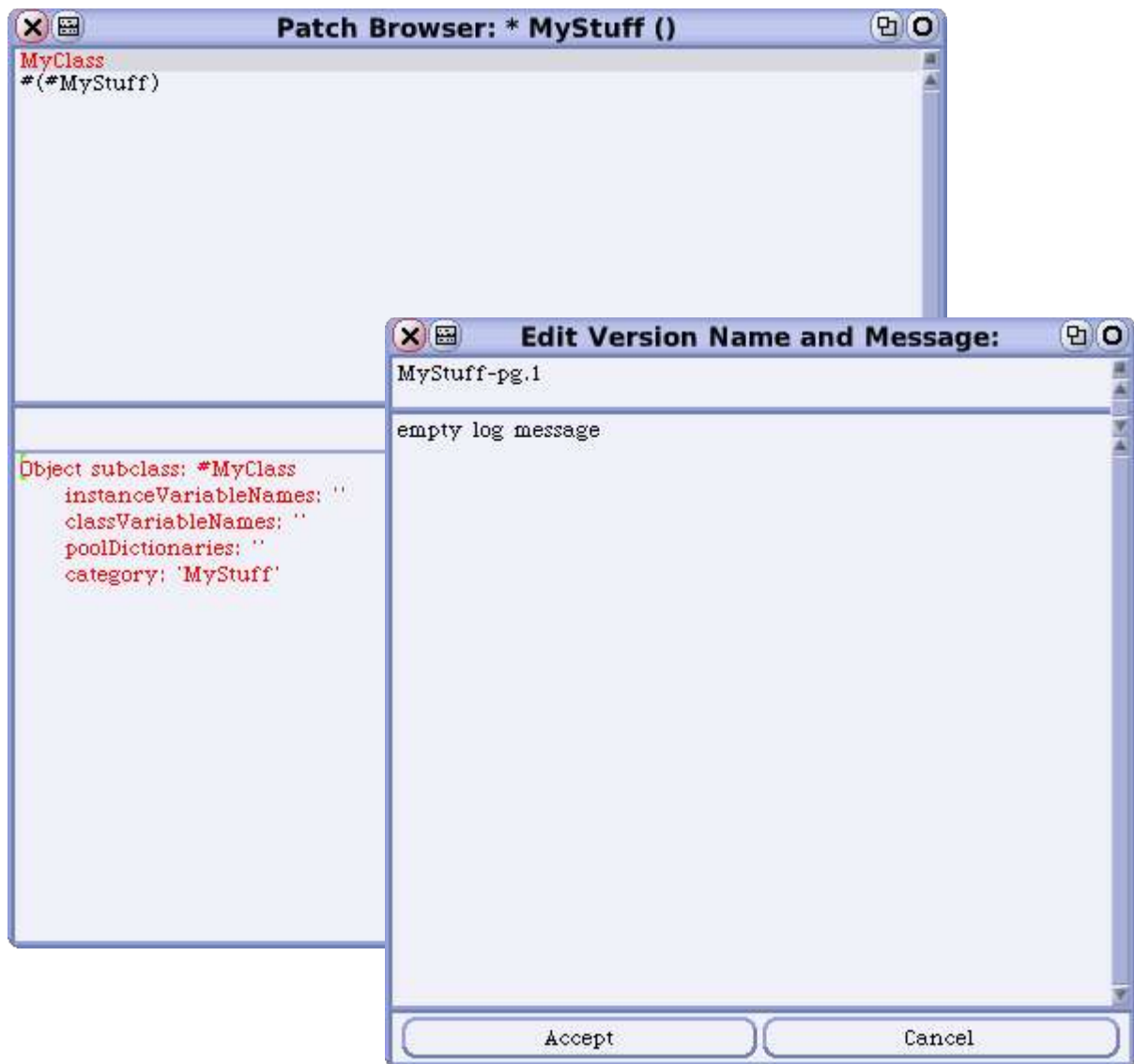
...und dort nur noch das Paket auswählen...

1. Paketinformationen – der Versionskommentar

Nun liegt das Paket bei der Post, jetzt muss es nur die nette Dame am Schalter überwinden und ab geht die Reise. „Was wollen sie denn verschicken, junger Mann/ junge Frau?“ - der Versionskommentar.

Im Monticello Browser das Paket und sein Repository markieren und sich die Änderungen (Changes) ansehen. Es geht ein Fenster auf, das einem alle vorgenommen Änderungen in der lokalen Version im Vergleich zu der aktuellsten auf dem Repository-Server anzeigt. Dieses ist hilfreich um den Versionskommentar zu schreiben, denn an alle Änderungen kann man sich doch beim besten Willen nicht erinnern...

Um das Paket auf den Repository-Server hochzuladen, muss es auf dem Repository gespeichert werden. Man drücke Speichern im Monticello-Browser und et voilà es öffnet sich ein weiteres Fenster... dort möge man seinen Versionskommentar hineinschreiben.



Warum soll ich die Versionen kommentieren?

...damit der nächste weiß, was du geändert und hinzugefügt hast, damit keiner Probleme angeht, die du schon gelöst hast oder deine Testklassen etwa für fertig hält und ins System einbaut....

Einfach um sich wiederum eine Menge Ärger zu ersparen....

Ist man fertig mit seinem wohl formulierten Kommentar, der Poesie genüge getan und man sich wieder dem schnöden Programmieren zuwenden darf oder will, so akzeptiere man sein Opus Magnum.

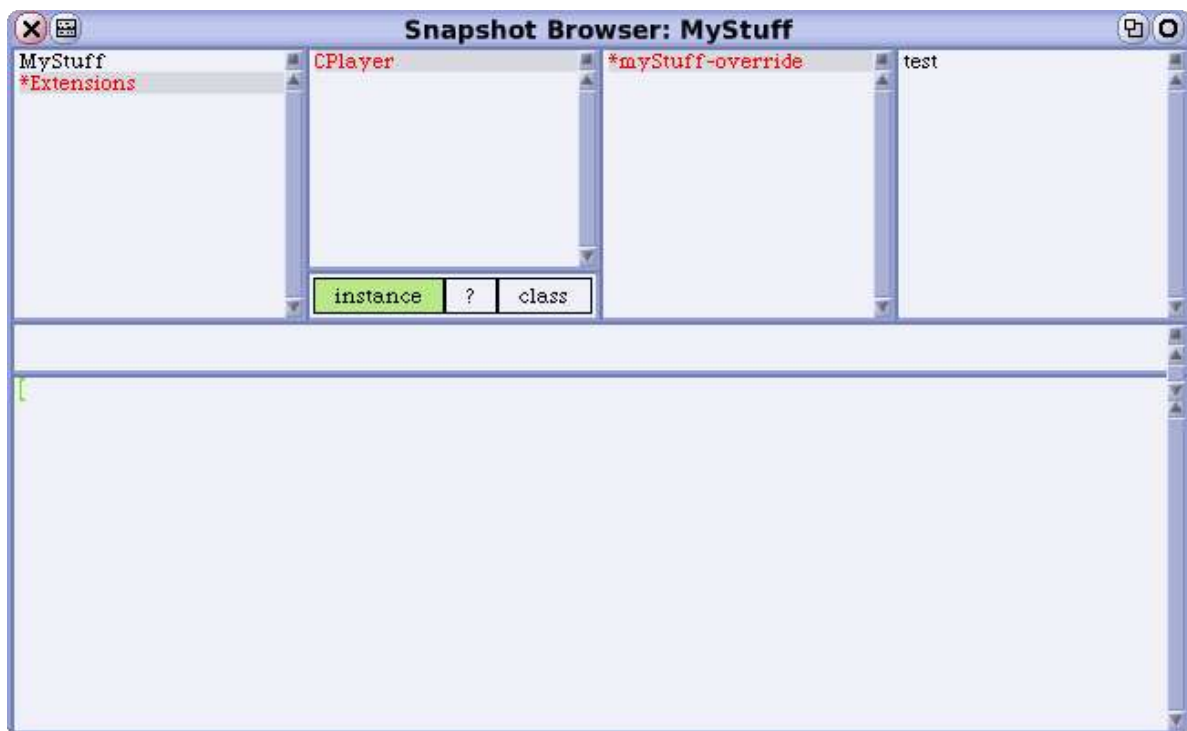
Schließlich wird das Paket hochgeladen, also gespeichert...Und ab geht die Post!

Änderungen an fremden Klassen

Manchmal ist es erforderlich Änderungen für das eigene Projekt vorzunehmen, die sich nicht ausschließlich in den eigenen Klassen realisieren lassen und fremde Klassen modifiziert werden müssen. Diese Änderungen sind zum einen problematisch, weil sie ebenfalls ausgetauscht werden müssen, aber eben projektspezifisch sind und in einem anderen Zusammenhang evtl. zu Problemen führen könnten. Sie sollten nur in Ausnahmefällen verwendet werden, wenn es wirklich unumgänglich ist!

Solche Änderungen werden als Extensions an das Monticello-Paket angehängt. Dafür müssen sie in separaten Methodenkategorien vorliegen, die die Form **packageName-override* haben müssen.

Vor dem Speichern einer Version solltest du das Paket ruhig noch einmal überprüfen. Hierfür gibt es einen speziellen Browser, der sich über den Monticello-Browser öffnen lässt. Das betreffende Paket markieren und browsen...



Aktualisieren und Mergen von Paketen

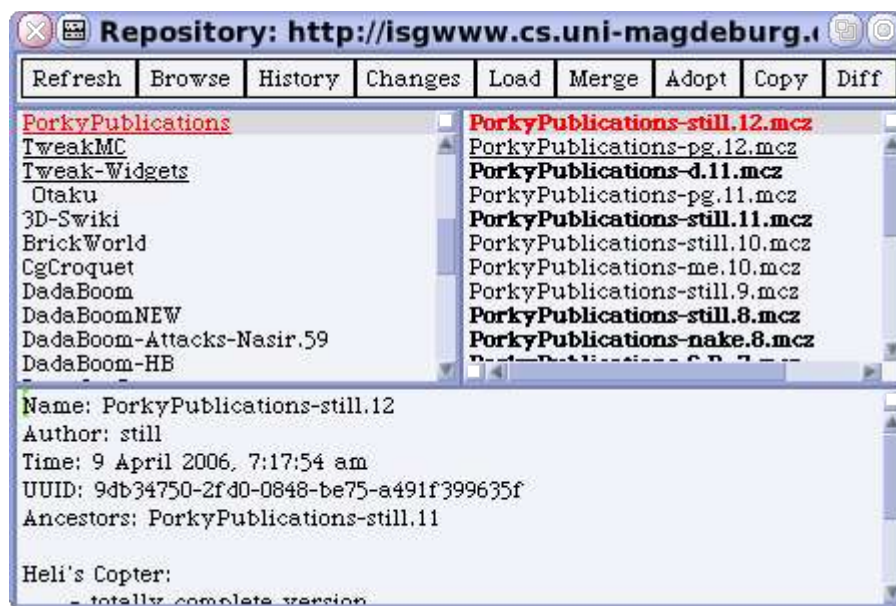
Fürs erste wird vorgegangen wie bereits beschrieben: Repository auswählen, öffnen und ein Paket auswählen...

Dieses darf jetzt aber nicht einfach geladen werden, eigene Änderungen würden dabei verloren gehen. Squeak warnt einen dann zwar, aber entweder man nimmt die Daten und überschreibt seine Neuerungen oder man lässt es beim alten...oder man merged den Sourcecode.

Normalerweise sollte es keine Probleme beim Mergen geben, aber manchmal passiert es eben doch, das zwei Leute dieselbe Methode oder Klasse angefasst haben...dann gibt's Konflikte!

Konflikte beim mergen – Warum muss man sich immer streiten, jetzt auch noch mit seinem Rechner?

Wenn zwei Leute an den selben Sachen arbeiten, gibt es sehr wahrscheinlich Konflikte - viele Köche verderben eben doch den Brei...



Was also tun, wenn's Probleme gibt?

Die Methoden, in denen es Konflikte gibt, werden **fett** hervorgehoben, die solltest du dir genauer ansehen, die anderen lassen sich problemlos mergen...



Neuerungen sind **rot** eingefärbt, alte Sachen, die entfernt werden würden, **blau** und durchgestrichen. Nun kann jeder Konflikt einzeln betrachtet werden und überlegt, ob die Änderung beibehalten oder verworfen werden soll. Änderungen am Sourcecode, um die Konflikte gleich richtig zu beseitigen, sind allerdings in diesem Fenster nicht möglich. Hierfür muss eine Systembrowser o.ä. geöffnet werden. Du kannst hierfür einfach die konfliktreiche Methode markieren und nach den Implementoren browsen... Gegebenenfalls nimm die neue Version an und füge deine eigenen Änderungen erneut in dem separaten Systembrowser hinzu (Man denke hierbei an die History, die vom gesamten Sourcecode gespeichert wird!).

Alternativ zum Betrachten aller Konflikte im Einzelnen, ist es auch möglich einfach alle Neuerungen zu akzeptieren oder alte Versionen beizubehalten, bzw. auf die lokale oder auf die serverseitige Versionen zu bestehen.

Ich rate jedoch dazu sich jeden Konflikt anzusehen und zu entscheiden, damit keine bösen Überraschungen nach dem Mergen geschehen...

Und vergiss nicht deine neuen Änderungen auf das Repository hochzuladen, oder wie oft willst du dir den Spaß machen und deine Sachen neu schreiben?!

Goldene Regel:

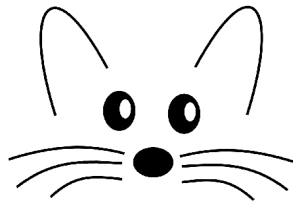
Merge immer erst die neueste Version vom Repository-Server ins eigene Image hinein, bevor du deine Änderungen hoch lädst, das erspart dir (und deinen Mitprogrammierern) viele Probleme!

So, das sollte es gewesen sein... für mehr Informationen wirf einen Blick auf die folgenden Webseiten, dort findest du das offizielle User Manual von Monticello sowie eine Menge Sachen zur Funktionsweise etc. Aber jetzt solltest du erst einmal in der Lage sein mit Monticello zu arbeiten...

- <http://www.wiresong.ca/Monticello/>
 - <http://minnow.cc.gatech.edu/squeak/monticello>
-

Fertig – es wurde ja auch Zeit!

So, nun endlich ran an die Arbeit!
Und frohes Schaffen!



PS: Fragen und Anmerkungen bitte an den Autor dieses Opus Magnum: peppermint-p@web.de