*Окончание. Начало в № 2`2010*

# Основы систем управления версиями файлов на примере Subversion (SVN)

**Денис ШЕХАЛЕВ**
shdv@micran.ru

## Работа с SVN-свойствами

В предыдущих частях мы рассмотрели основные операции, с которыми разработчик сталкивается каждый день: просматривает репозиторий, работает с проектом в рабочей копии, фиксирует изменения, делает слияние ветвей разработки и т. д. Но этом рассмотрение основ SVN можно было бы закончить, но, как и любой другой инструмент, SVN обладает возможностями тонкой настройки и дополнительными опциями — для автоматизации его использования. Это реализовано через механизм SVN-свойств папок и файлов, находящихся под контролем репозитория. Существует достаточно большое количество SVN-свойств файлов/папок, описывающих те или иные ситуации, но мы рассмотрим только базовые свойства, которые желательно использовать при работе.

### *Разделение бинарных и текстовых файлов*

SVN — это система контроля версий исходных кодов. И в первую очередь она ориентирована на работу с текстовыми файлами. В том случае, если система работает с бинарным файлом, результаты команд сравнения, слияния и разрешения конфликтов дают непредсказуемый результат.

Чтобы все было корректно, нужно указать SVN, что данный файл — не текстовый и к нему нельзя применять методы обработки текстовых файлов. Делается это с помощью свойства ***svn:mime-type***. Например, вы решили добавить к документации графический файл формата **.png* (рис. 74). Для того чтобы сообщить SVN, что этот файл не должен рассматриваться как бинарный, нужно в SVN-свойствах файла (рис. 75) добавить свойство ***svn:mime-type*** со значением ***application/octet-stream*** (рис. 76). После фиксации изменений в репозитории этот файл будет рассматриваться и обрабатываться как не текстовый файл.

***Важное замечание.*** *Помните, что при возникновении конфликтов с бинарными файлами SVN не содержит программ, которые смогут вам показать, чем отличаются сравниваемые файлы, это нужно будет сделать вручную. (Согласно анонсам последних версий TortoiseSVN появилась программа сравнения графических файлов.)*

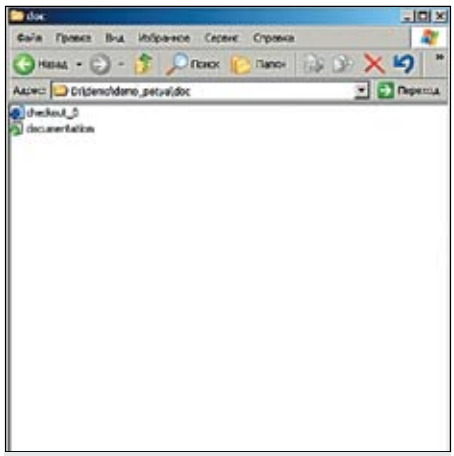

**Рис. 74.** Состояние рабочей копии после добавления графического файла checkout_0.png

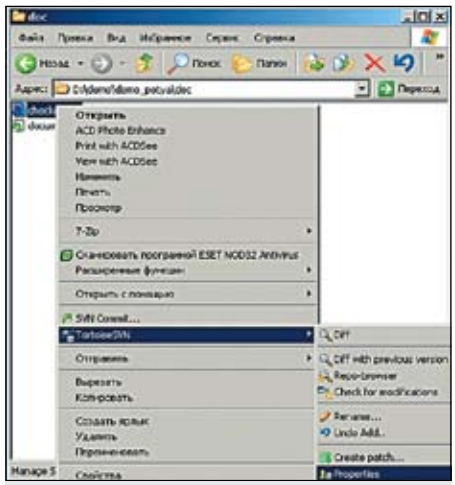

**Рис. 75.** Вызов диалогового окна SVN-свойств файла

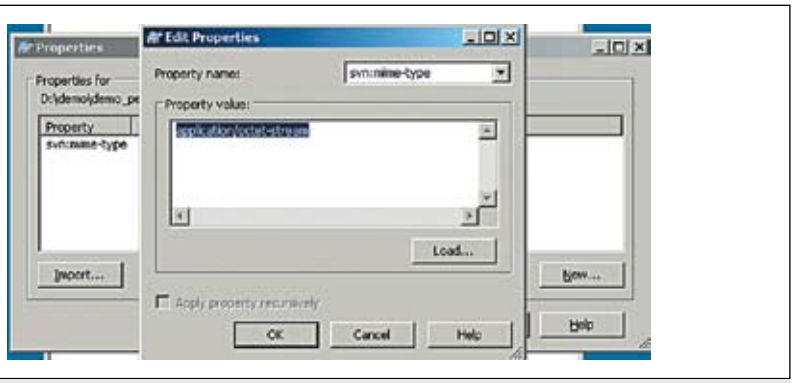


**Рис. 76.** Диалоговое окно установки SVN-свойства файла svn:mime-type

***Совет.*** *Некоторые распространенные расширения файлов TortoiseSVN распознает и ставит* ***svn:mime-type*** *автоматически. Но лучше лишний раз проверить, чем потом выяснять, почему после слияния ветвей или устранения конфликта графический файл вдруг перестал открываться или бинарный файл прошивки процессора вдруг перестал работать.*

### *Ссылки на другие проекты и репозитории*

Достаточно часто при работе над смежными или похожими проектами возникает следующая ситуация: в проектах используется одинаковый код, но размер этого кода недостаточен для выделения его в отдельный проект. Например, это могут быть небольшие компоненты общего применения (модули скремблеров, интерфейсов и т. д.).

Обычно разработчики решают эти вопросы копированием общих модулей из другого проекта. Но при устранении ошибок или модификаций в этих модулях им приходится вручную синхронизировать эти модули между проектами. В итоге возникает ситуация, когда в проектах лежат функционально одинаковые модули, но с разными интерфейсами, ошибками и т. д. и т. п. Это приводит к путанице и не вносит порядка в процесс разработки.

Подобная ситуация возникает и при включении модулей проекта более низкой иерархии в проект более высокого уровня. Это не зависит от того, используется SVN или нет. Нужно изменять подход к решению данной проблемы.

Существует два варианта, как это можно сделать в SVN.

**Неправильный вариант**

Создаем рабочую копию, в ней создаем папки, делаем ***Checkout*** файлов и папок из нужных проектов. Разработчик должен делать это вручную каждый раз при создании рабочей копии. При этом всем, кто работает над данным проектом, нужно помнить, где и что он взял. Этот метод не обладает автоматизмом: это самый большой недостаток данного подхода.

**Правильный вариант**

Назначить в свойствах папки ссылку на проект в репозитории. Этот метод обладает автоматизмом. После назначения свойства при создании рабочей копии или экспорте проекта все необходимые файлы будут скопированы из репозитория автоматически.

Рассмотрим только второй вариант. Предположим, что в проекте ***demo_project*** необходимо использовать наработки из проекта ***demo_project1***. Командой ***Checkout*** создаем рабочую копию проекта ***demo_project***. До назначения SVN-свойств рабочая копия выглядит так, как показано на рис. 77.

Для задания ссылки на проект ***demo_project1*** нужно определить свойство папки ***svn:externals*** (рис. 78). В этом свойстве указать папку-приемник и папку-источник. Поместим ссылку на проект ***demo_project1***, который находится в репозитории ***demo_repo***, в папку ***link*** (рис. 79). Назначение свойства — это изменение, и оно требует фиксации. Фиксируем изменения в репозитории командой ***Commit*** (рис. 80). Но для того чтобы получить файлы по ссылке, после фиксации изменения свойств папки нужно выполнить команду ***Update***. Только после этого файлы будут скопированы (рис. 81).

Обратите внимание: папка со ссылкой на проект создается автоматически и находится под контролем SVN. Следовательно, если вы внесете изменения в файлы или папки проекта по ссылке, то при фиксации изменения будут зафиксированы в проекте ***demo_project1***. Если кто-то другой внес изменения в файлы или папки (например, поправил ошибки в коде), то с помощью команды ***Update*** вы получите последние изменения.

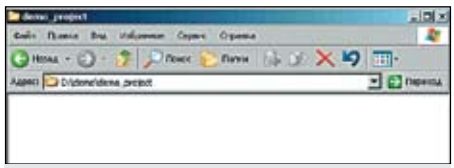

**Рис. 77.** Состояние рабочей копии проекта demo_project перед созданием ссылки на проект demo_project1

**Рис. 78.** Вызов диалогового окна SVN-свойств папки

***Подсказка.*** *Количество ссылок может быть любым. Но все ссылки перечисляются в одном свойстве* ***svn:externals*** *по следующему шаблону:*

```
<папка-приемник1> пробел <папка источник1>
<папка-приемник2> пробел <папка источник2>
<папка-приемник3> пробел <папка источник3>
```

***Автоматическая подстановка ключевых слов***

SVN умеет выполнять подстановку ключевых слов — элементов полезной динамической информации о файле под контролем SVN. SVN «знает» ключевые слова (таблица).

Для того чтобы SVN выполнял в файле подстановку, ключевые слова должны быть оформлены справа и слева символами *$*. Но добавить ключевые слова в файл недостаточно, нужно включить в SVN поддержку их обработки. Для этого нужно установить свойство ***svn:keywords*** на папку или файл с указанием, какие именно ключевые слова включены в список автозамены (рис. 82).

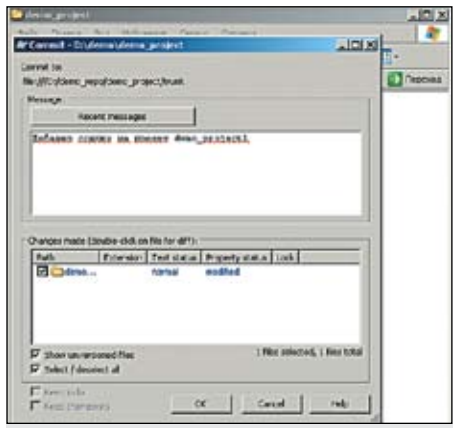

**Рис. 80.** Диалоговое окно для задания комментария при фиксации изменений SVN-свойства папки

**Таблица.** Ключевые слова

| | |
|---|---|
| Date | Время последнего изменения файла в репозитории |
| Revision | Номер последней правки, в которой файл был изменен в репозитории |
| Author | Имя пользователя, который последним изменил этот файл в репозитории |
| HeadURL | Полный URL к последней версии файла в репозитории |
| Id | Компактная комбинация всех вышеприведенных слов |

Изменение свойств файла или папки тоже требует фиксации в репозитории, как и любое другое изменение. Если подстановка больше не требуется, нужно снять свойство с папки или файла.

***Подсказка.*** *Очень удобно использовать данные ключевые слова в заголовках файлов. Вот пример такого заголовка:*

```
// Project      : Test project
// Project Nick : Testik
// Revision     : $Revision$
// Author       : $Author$
// Date         : $Date$
// Description  : demo
```

Результат работы автозамены приведен на рис. 83.

## Что же в итоге?

Мы рассмотрели основы работы с системой управления версиями файлов SVN. Многие, кто владеет SVN, скажут, что приведенная информация не точна или не раскрывает всех


**Рис. 79.** Диалоговое окно установки SVN-свойства файла svn:externals

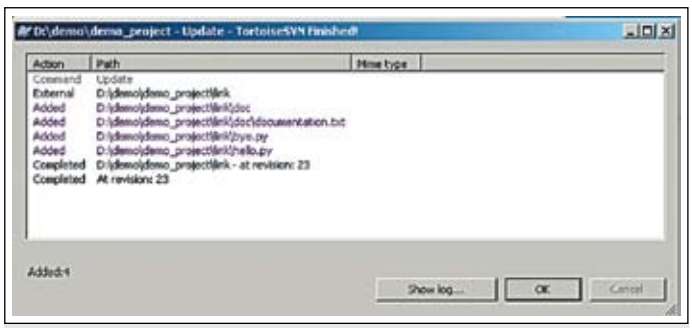


**Рис. 81.** Окно протокола синхронизации рабочей копии с репозиторием, если в SVN-свойствах папок есть ссылки на другие проекты

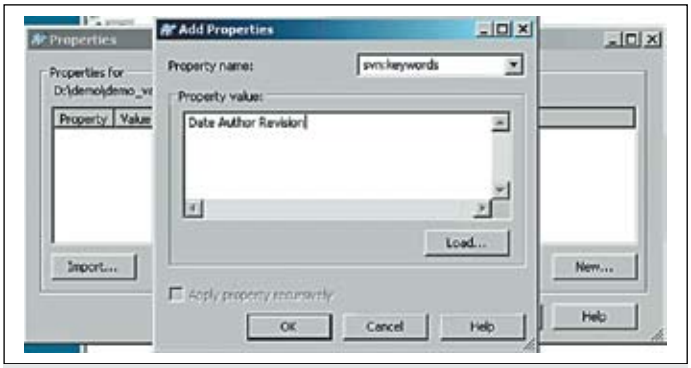


**Рис. 82.** Диалоговое окно установки SVN-свойства файла svn: keywords

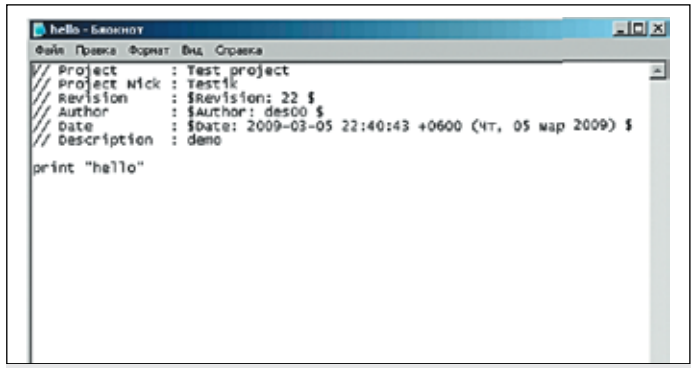


**Рис. 83.** Результат автозамены ключевых слов при создании рабочей копии

тонкостей его работы (в особенности в области слияния ветвлений и разрешения конфликтов). Они будут правы. Другие скажут, что у SVN много недостатков (централизованный репозиторий, проблемы при слиянии ветвлений) и что есть более качественные системы управления версиями файлов. И они тоже будут правы.

Нет смысла спорить. Потому что цель этой статьи показать, что в системах контроля версий, и в SVN в частности, нет ничего сложного. Для эффективного ее использования достаточно знать порядок осуществления основных операций. Выигрыш от использования SVN будет много больше стоимости времени, затраченного на изучение его основ, которые мы на простых примерах рассмотрели в статье.

В качестве последнего совета перечислим некоторые ограничения, которые возникают при работе с SVN. Изначально это система управления версиями текстовых файлов, она создавалась для обработки текстовой информации. Конечно, разместить под контроль SVN можно любую информацию и любые файлы, но нужно помнить о следующих ограничениях:

1. Средства сравнения файлов в SVN работают только для текстовой информации. Изменения не текстовых файлов будут зафиксированы, но место, где именно произошли изменения, придется искать самостоятельно. Если только вы не обладаете талантом свободного чтения бинарного кода прошивки и можете выяснить, что именно в ней изменилось.
2. Автоматически ***TortoiseSVN*** распознает не все расширения файлов. В этом случае файл считается текстовым. И при операциях сравнения, слияния, возникновении конфликта вы можете получить совершенно не тот результат, который ожидаете. Не забывайте проверять свойства файлов, которые не должны считаться текстовыми (например, *.pcb файлы PCAD, *.vqm файлы и т. д.).
3. Любое, пусть даже самое малое, изменение файлов проекта (добавили лишний пробел) будет зафиксировано SVN и потребует фиксации. Поэтому избегайте случайной модификации файлов при их просмотре. Перед Commit проверяйте файлы на результат изменений. Используйте Revert для отката изменений файлов. Замена символов табуляции на пробелы — тоже изменение. Поэтому правильно настройте свой редактор.
4. По этой же причине не рекомендуется хранить в SVN временные файлы и файлы сообщений работы компиляторов, синтезаторов и т. д. Храните только те файлы, информация в которых нужна для сборки проекта.
5. Неразумно хранить в SVN программное обеспечение, используемое для сборки проекта, документацию на микросхемы, используемые в проекте, и т. д. Для этих целей используйте систему ссылок на ресурсы предприятия или производителя и указывайте все в сопроводительной документации.

Что касается автора, то вопрос «Использовать или нет систему контроля версий?» даже не ставится. Использовать. В любом, пусть даже самом небольшом проекте. Во-первых, небольшой проект может вылиться в более серьезный. Во-вторых, в репозитории проект точно не потеряется, конечно, если у вас не выйдет из строя жесткий диск или не произойдет его внезапного форматирования. А в-третьих, практика написания комментариев при фиксации изменений в какой-то мере заменяет систему документирования и не занимает много времени.

На работе автор использует сетевой репозиторий на коллектив разработчиков из ~20 человек, дома — локальный репозиторий для своих проектов. При минимуме усилий на фиксацию проектов в репозитории автор получает простоту, легкость и удобство управления ими. Возможности SVN автора полностью устраивают. Читателям же рекомендуем просмотреть несколько систем управления версиями файлов и выбрать ту, которая им понравится.