Лабораторная работа № 2. Subversion. Ветвления.

Цель работы

Получение навыков работы с системой контроля версий Subversion (ветками и разрешением конфликтов).

Теоретическая часть

1. Ветвление

Ветка – это направления разработки, которое существует независимо от другого направления, однако имеющие с ним общую историю, если заглянуть немного в прошлое. Ветка всегда берет начало как копия чего-либо и двигается от этого момента, создавая свою собственную историю.

Создание ветки

Создать ветку очень просто — при помощи команды svn copy делаете в хранилище копию проекта.

Обратите внимание, что директория для ветви уже должна существовать в репозитории.

Пример 1. Создание директорий для ветвей (в исходном репозитории, не в копии):

$ mkdir -p branches/test-branch

$ svn commit -m "folders for branhing"

Adding branches

Adding branches/test-branch

Committing transaction...

Committed revision 3.

Команда svn copy может оперировать с двумя URL напрямую.

Пример 2. Создание ветки «test-branch» и фиксирование этого изменения:

svn copy file:///home/uname/repository/trunk file:///home/uname/repository/branches/test-branch -m "first branch"

Committing transaction...

Committed revision 4.

Эта процедура проста в использовании, так как нет необходимости в создании рабочей копии, отражающей большое хранилище. Вам вовсе можно не иметь рабочей копии.

Работа с веткой

После создания ветки проекта, можно создать новую рабочую копию для начала ее использования:

Пример 3. Создание рабочей копии ветки «test-branch»:

$ svn checkout file:///home/uname/repository/branches/test-branch

A test-branch/Makefile

A test-branch/integer.c

Checked out revision 5.

Ключевые идеи, стоящие за ветками:

В отличие от других систем управления версиями, ветки в Subversion существуют в хранилище не в отдельном измерении, а как обычные нормальные директории файловой системы. Такие директории просто содержат дополнительную информацию о своей истории.

Subversion не имеет такого понятия как ветка — есть только копии. При копировании директории результирующая директория становиться «веткой» только потому, что вы рассматриваете ее таким образом. Вы можете по-разному думать о директории, по-разному ее трактовать, но для Subversion это просто обычная директория, которая была создана копированием.

2. Копирование изменений между ветками

В проектах, имеющих большое количество участников, когда кому-то необходимо сделать долгосрочные изменения, которые возможно нарушат главную линию, стандартной процедурой является создание отдельной ветки и фиксация изменения туда до тех пор, пока работа не будет полностью завершена.

Чтобы избежать слишком большого расхождения веток, можно продолжать делиться изменениями по ходу работы. А тогда, когда ваша ветка будет полностью закончена, полный набор изменений ветки может быть скопирован обратно в основную ветку.

Пример 4. Копирование изменений между ветками: из b1 в b2:

svn merge url://to/branch/b2 url://to/branch/b1

3. Конфликты при объединении

Команда svn merge иногда не может гарантировать корректного поведения: пользователь может запросить сервер сравнить любые два дерева файлов, даже такие, которые не имеют отношения к рабочей копии! Из этого следует большое количество потенциальных человеческих ошибок.

Команда svn merge в конфликтной ситуации создает три файла с названиями filename.working, filename.left и filename.right. Здесь, термины «left» и «right» указывают на две стороны сравнения, то есть на используемые при сравнении деревья.

Способы разрешения конфликтов

Предположим что вы запустили svn update и получили:

Пример 5. Конфликтный файл при объединении изменений:

$ svn update
U INSTALL
G README

C bar.c

Updated to revision 46.

Файл, отмеченный флагом C, имеет конфликт. Это значит, что изменения с сервера пересеклись с вашими личными, и теперь вам нужно вручную сделать между ними выбор.

Всякий раз, когда возникает конфликт, для того, чтобы помочь вам заметить и решить этот конфликт, происходят, как правило, три вещи:

1) Subversion печатает флаг C во время обновления и запоминает, что файл в состоянии конфликта;

2) Если Subversion считает, что файл объединяемого типа, она помещает маркеры конфликта — специальные текстовые строки которые отделяют «стороны» конфликта — в файл, для того, чтобы визуально показать пересекающиеся области;

3) Для каждого конфликтного файла Subversion добавляет в рабочую копию до трех не версионированных дополнительных файлов:

- filename.mine (ваш файл в том виде, в каком он был в рабочей копии до обновления (если Subversion решает, что файл не объединяемый, тогда файл .mine не создается, так как он будет идентичным рабочему файлу));
- filename.rOLDREV (файл правки BASE, где BASE – правка которая была до обновления рабочей копии, то есть это файл, который у вас был до внесения изменений);
- filename.rNEWREV (файл, который ваш Subversion-клиент только что получил с сервера, когда вы обновили рабочую копию. Этот файл соответствует правке HEAD хранилища, где HEAD — специальный указатель на последний коммит в хранилище).

Здесь OLDREV - это номер правки файла в директории .svn, а NEWREV - это номер правки HEAD хранилища.

Если вы получили конфликт, у вас есть три варианта:

- Объединить конфликтующий текст «вручную» (путем анализа и редактирования маркеров конфликта в файле);
- Скопировать один из временных файлов поверх своего рабочего файла;
- Выполнить *svn revert <filename>* для того, чтобы убрать все ваши локальные изменения.

После того, как вы решили конфликт, вам нужно поставить в известность Subversion, выполнив *svn resolved*. Эта команда удалит три временных файла, и Subversion больше не будет считать, что файл находится в состоянии конфликта.

3.1. Объединение конфликтов вручную

Конфликтный файл будет выглядеть так:

<<<<<<< имя файла

ваши изменения

=======

результат автоматического слияния с репозиторием

>>>>>>> ревизия

Вам необходимо разрешить конфликты вручную редактированием данного файла или через сторонние приложения.

Когда вы произведете слияние изменений, выполните *svn resolved* , то вы готовы к фиксации изменений:

Пример 6. Фиксирование состояния разрешения конфликта для файла sandwich.txt:

$ svn resolved sandwich.txt

И, наконец, зафиксируйте изменения:

$ svn commit -m "Go ahead and use my sandwich, discarding Sally's edits."

3.2. Использование svn revert

Если вы получили конфликт и вместо анализа решаете отбросить локальные изменения и начать сначала, просто отмените их:

Пример 7. Отмена изменений в файле sandwich.txt:

$ svn revert sandwich.txt

Reverted 'sandwich.txt'

Обратите внимание, что когда вы возвращаете файл к предыдущему состоянию, вам не нужно выполнять команду *svn resolved*.

Практическая часть

Замечание: забирайте необходимые изменения у напарника и отправляйте их в репозиторий, когда это необходимо.

1. Создайте пустой проект в любой директории.
2. Создайте в ветви trunk один файл с 15-20 строками программного кода.
3. Создайте ветви с вашим именем.
4. Склонируйте к себе локальную копию вашей ветви без клонирования всего проекта.
5. Отредактируйте созданный файл так, чтобы номера изменяемых строк у вас и напарника совпадали. Зафиксируйте изменения в ваших ветках.
6. Влейте изменения ветки напарника к себе в ветку. Отразите в сообщении к коммиту, то, что это коммит с вливанием веток.
7. Разрешите возникшие конфликты.
8. Опишите ситуации, когда конфликт происходит, а когда нет.
9. Влейте изменения в ветку trunk.

10. Сделайте ещё два любых коммита в своих ветках.
11. Отмените последний коммит.

Содержание отчёта

По результатам выполнения работы оформляется отчет в соответствии с требованиями ГОСТ 7.32-2017 «Отчет о научно-исследовательской работе. Структура и правила оформления», включающий:

- титульный лист;
- цель работы;
- описание структуры хранилища во время выполнения (при выполнении операций, меняющих состояние хранилища);
- выполняемые команды с комментариями и результаты их выполнения;
- выводы.

Контрольные вопросы:

1. Опишите, в чем состоит отличие локального репозитория от удалённого.
2. Назовите способы копирования проекта к себе в удалённый репозиторий.
3. Как отредактировать сообщение у коммита?
4. Что такое ветка?
5. Что представляет собой ветка в Subversion?
6. Перечислите недостатки работы в одной ветке.
7. Как скопировать отдельные изменения между ветками?
8. Как скопировать изменения из одной ветки в другую?
9. Какие файлы создаются при конфликтной ситуации при слиянии?
10. Опишите известные вам способы разрешения конфликтов в Subversion и их отличительные особенности.