Федеральное государственное автономное образовательное
учреждение высшего образования
«Научно-образовательная корпорация ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №3 по основам программной инженерии

Вариант 789349871

Выполнили:
Степанов Арсений Алексеевич
Шпак Всеволод Васильевич

Группа:
P3209

Преподаватель:
Пименов Данила Дмитриевич

Санкт-Петербург, 2024г

Задание

Написать сценарий для утилиты Maven, реализующий компиляцию, тестирование и упаковку в jar-архив кода проекта из лабораторной работы №3 по дисциплине "Веб-программирование".

Каждый этап должен быть выделен в отдельный блок сценария; все переменные и константы, используемые в сценарии, должны быть вынесены в отдельный файл параметров; MANIFEST.MF должен содержать информацию о версии и о запускаемом классе.

Сценарий должен реализовывать следующие цели (targets):

1. **compile** – компиляция исходных кодов проекта.
2. **build** – компиляция исходных кодов проекта и их упаковка в исполняемый jar-архив. Компиляцию исходных кодов реализовать посредством вызова цели compile.
3. **clean** – удаление скомпилированных классов проекта и всех временных файлов (если они есть).
4. **test** – запуск junit-тестов проекта. Перед запуском тестов необходимо осуществить сборку проекта (цель build).
5. **native2ascii** – преобразование native2ascii для копий файлов локализации (для тестирования сценария все строковые параметры необходимо вынести из классов в файлы локализации).
6. **scp** – перемещение собранного проекта по scp на выбранный сервер по завершению сборки. Предварительно необходимо выполнить сборку проекта (цель build)
7. **xml** – валидация всех xml-файлов в проекте.
8. **music** – воспроизведение музыки по завершению сборки (цель build).
9. **doc** – добавление в MANIFEST.MF MD5 и SHA-1 файлов проекта, а также генерация и добавление в архив javadoc по всем классам проекта.
10. **history** – если проект не удаётся скомпилировать (цель compile), загружается предыдущая версия из репозитория git. Операция повторяется до тех пор, пока проект не удастся собрать, либо не будет получена самая первая ревизия из репозитория. Если такая ревизия найдена, то формируется файл, содержащий результат операции diff для всех файлов, измёненных в ревизии, следующей непосредственно за последней работающей.
11. **diff** – осуществляет проверку состояния рабочей копии, и, если изменения касаются классов, указанных в файле параметров выполняет commit в репозиторий git.

12. **team** – осуществляет получение из svn-репозитория 2 предыдущих ревизий, их сборку (по аналогии с основной) и упаковку получившихся jar-файлов в zip-архив. Сборку реализовать посредством вызова цели build.

13. **report** – в случае успешного прохождения тестов сохраняет отчет junit в формате xml, добавляет его в репозиторий svn и выполняет commit.

14. **alt** – создаёт альтернативную версию программы с измененными именами переменных и классов (используя задание replace/replaceregexp в файлах параметров) и упаковывает её в jar-архив. Для создания jar-архива использует цель build.

15. **env** – осуществляет сборку и запуск программы в альтернативных окружениях; окружение задается версией java и набором аргументов виртуальной машины в файле параметров.

Исходный код

Репозиторий на GitHub

Вывод

Мы научились писать собственные сценарии для утилиты Maven используя Mojo и использовать их для автоматизации задач различного спектра в своих проектах