SmartPackager 1.0

Описание

Библиотека по быстрой упаковке и распаковке данных в массив байт. В отличии от стандартных сериализаторов не хранит никакой дополнительной информации, благодаря чему достигается высокая скорость упаковщика и минимальный вес данных. Благодаря удобному интерфейсу использования позволяет избежать множества ошибок при упаковке данных вручную, а также экономя время на реализацию данного функционала, кроме того, реализованы и протестированы оптимальные алгоритмы упаковки. Рекомендуется для использования в проектах, где необходима высокая скорость передачи потоковой информации, а также экономия размера информации, которая будет передана, например:

* Сохранение настроек, большого объёма данных или кэша
* Использование при построении сервисно-ориентированной архитектуры (упаковка данных для коммуникаций между сервисов)
* Программы и игры с сетевым взаимодействием.

Функционал

Реализована поддержка типов:

1. DateTime
2. TimeSpan
3. string

Иметься возможность добавления поддержки различных типов данных путём создания и реализации класса для каждого типа с унаследованным интерфейсом IPackagerMethod<Тип данных>. Пример реализации упаковки для типа [DateTime](#_Реализация_упаковки_для), [string](#_Реализация_упаковки_для_1).

Если вы не уверенны что упаковщик добавил ваш сгенерированный тип данных, то вы можете это проверить, вызвав в классе PackMethods метод GetPackMethods(), или проверить авто сгенерированные (учтите, что сначала надо вынудить автогенератор сгенерировать тип а только потом смотреть) типы вызвав GetPackagerMethodsUnmanagedTypes() или для управляемых типов GetPackagerMethodsManagedTypes(). В случае подключении других сборок с типами и реализацией упаковки вы можете попробовать вызвать функцию SetupAgainPackMethods(), которая заново попытается найти все реализации пользовательских упаковок.

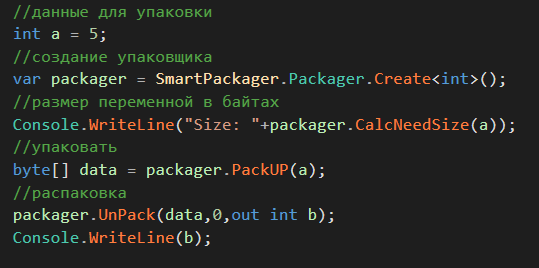
Программа имеет автогенераторы универсальных классов для упаковки очевидно реализуемых данных (автогенераторы могут комбинируют методы при упаковки сложных структур данных, например классов с массивами структур):

1. Неуправляемые структуры с базовыми типами или другими такими же структурами. Если структура содержит любое ссылочное поле, то бишь класс, к которому относиться даже string и любые массивы то эта структура не может быть упакована данным методом. Данный метод является одним из самых быстрых и упаковывает все поля включая закрытые.
2. Массивы. На данный момент не поддерживается упаковка многомерных массивов. Возможна упаковка одномерных массивов (самый оптимальный вариант по скорости и размеру), так и массива с массивами – но является более затратным методом. Не допускается упаковка не инициализированных массивов (null). Стоит отметить, что скорость вычисления размера массива при упаковке зависит от типа – структура с фиксированным размером будет вычислена единожды и просто умножаться на размер массива, в то время как для других данных придётся вычислять размер каждой переменной.
3. Классы, и управляемые структуры. Самый медленный тип для упаковки, так как используют методы рефлексии для чтения и записи полей. Допускается упаковка не инициализированных данных (null) – при распаковке будут так же иметь не инициализированное состояние. По умолчанию упаковщик не работает с закрытыми членами, но иметься возможность указать атрибут для класса или структуры что бы упаковщик начал отслеживать закрытые поля:

Automatic.SearchPrivateFieldsAttribute. Стоит отметить, не стоит использовать циклические ссылки между классами так как это приведёт бесконечному циклу и зависанию программы.

Использование

Для работы с упаковщиком существует класс Packager, который генерирует универсальный тип с функциями упаковки, существует насколько перегрузок упаковщика позволяющие упаковывать в один массив до 20 разных типов. Пример упаковки/распаковки переменной типа int:



# Реализация упаковки для типа DateTime



# Реализация упаковки для типа string

Примечание – данные будут представлены в виде <количество><символы>

