Блог GunSmoker-a

...when altering one's mind becomes as easy as programming a computer, what does it mean to be human?..

9 сентября 2011 г.

Сериализация - общие сведения

Оглавление серии.

Этот пост содержит основные определения серии.

Сериализация

Операция записи каких-то данных в файл заключается в их **сериализации** (от англ. serialize - переводить в последовательную форму) в последовательность байт (и записи результата в файл).

Обратной к операции сериализации является операция **десериализации** — восстановление начального состояния структуры данных из байтовой последовательности.

Эта байтовая последовательность записывается или загружается в/из файл(а). Соответственно, сериализация используется для передачи объектов по сети и для сохранения их в файлы. Это просто название для операции "выкладывания" ("упаковки") данных в файл. Десериализация - операция "распаковки" данных из файла. Если вы сериализуете какие-то данные, а затем десериализуете их, то получите эквивалентный набор данных, имеющих тот же смысл, что и оригинал. При некоторых дополнительных условиях это будет полная копия оригинала.

Любой из схем сериализации присуще то, что кодирование данных последовательно по определению, и извлечение любой части сериализованной структуры данных требует, чтобы весь объект был считан от начала до конца и воссоздан. Во многих приложениях такая линейность полезна, потому что позволяет использовать простые интерфейсы ввода/вывода общего назначения для сохранения и передачи состояния объекта.

Маршалинг

Почти синонимом термина "сериализация" является **маршалинг** (от англ. marshal — упорядочивать). Обратный процесс - **демаршалинг** (аналог десериализации). Маршалинг - более общее понятие, чем сериализация. Всякая сериализация является частным случаем маршалинга. Отличие маршалинга от сериализации в том, что сериализация предполагает упаковку лишь данных программы. Скажем, матрицы чисел, записи о сотруднике в картотеке или таблицы высот игрового уровня. Когда говорят про маршалинг, подразумевают сериализацию не только данных программы, но и её программного состояния, мета-информации. К примеру, сериализация объекта с полями запишет лишь данные объекта (поля). Маршалинг же объекта запишет не только информацию о его данных (полях), но и информацию по восстановлению структуры объекта - класс объекта, либо же его мета-информацию для реконструирования типа.

Маршалируя и демаршалируя объект вы создаёте его полную копию. При этом принимающая сторона может ничего не знать про объект.

Маршалинг используется в различных механизмах RPC, где есть необходимость в передаче данных между процессами и потоками, а также для обмена данными с движками скриптов. Во всех этих случаях

маршалинг используется, чтобы "упаковать" вызов метода, его параметры, передать сообщение на другую сторону для выполнения (другой процесс или машину в случае RPC или скрипт в случае скриптового движка), и провести обратный процесс (демаршалинг) для распаковки ответа в результат вызова метода.

Персистентность

Персистентность - это свойство чего-либо поддерживать постоянное (перманентное) состояние. Скажем, применительно к объекту персистентность обычно означает способность этого объекта сериализовать (и десериализовать) самого себя.

Способы сериализации данных в Delphi

В любом случае, если вы мало что поняли из сказанного выше, вам нужно вынести только одну вещь: "сохранение данных в файл - это сериализация данных в файл, загрузка данных из файла - это десериализация".

Итак, для наших целей в качестве примеров мы будем рассматривать загрузку и сохранение таких данных:

1. Строка или набор строк - текстовые данные
2. Массив чисел: `array of Extended` - набор однородных данных
3. Пары имя-значение - набор ассоциативных данных
4. Запись - набор неоднородных данных
5. Набор (массив) из записей - иерархический набор данных
6. Массив из записей внутри записи - составные данные
7. Какие-то другие примеры данных, иллюстрирующие особенности конкретных подходов

Давайте сразу же приведём пример объявления этих структур:

1. Одно значение: `Double`
2. Одно значение переменного размера: `String`
3. Набор однородных значений: `array of Double`
4. Набор однородных значений переменного размера: `array of String`
5. Запись - набор неоднородных данных:

```
type
  TData = record
    Signature: LongWord;
    Size: LongInt;
    Comment: String;
    CRC: LongWord;
  end;
```

6. Набор (массив) из записей - иерархический набор данных:

```
type
  TPerson = record
    Name: String;
    Age: Integer;
    Salary: Currency;
  end;

  TPersons = array of TPerson;
```

7. Массив из записей внутри записи - составные данные:

```
type
  TCompose = record
    Signature: LongInt;
    Person: TPerson;
    Count: Integer;
    Related: TPersons;
```

```
7     end;
8
9     TComposes = array of TCompose;
```

8. Какие-то другие примеры данных, иллюстрирующие особенности конкретных подходов

Тэги Delphi, Статья

Комментариев нет:

Отправить комментарий

Можно использовать некоторые HTML-теги, например:

Жирный
Курсив
Ссылка

Вам необязательно регистрироваться для комментирования - для этого просто выберите из списка "Анонимный" (для анонимного комментария) или "Имя/URL" (для указания вашего имени и (опционально) ссылки на сайт). Все прочие варианты потребуют от вас входа в вашу учётку (поддерживается OpenID).

Пожалуйста, по возможности **используйте "Имя/URL" вместо "Анонимный"**. URL можно просто не указывать.

Ваше сообщение может быть помечено как спам спам-фильтром - не волнуйтесь, оно появится после проверки администратором.

Введите комментарий...

Подпись комментария: Аккаунт Goog ▼

Публикация Просмотр

Ссылки

Создать ссылку

Следующее Главная страница Предыдущее

Подписаться на: Комментарии к сообщению (Atom)

Поиск по блогу

Google™ Пользовательский поиск
Поиск

Архив

Архив ▼

Тэги

Delphi (185) Статья (72) задачки (38) ты можешь это сделать (30) начинающим (26) обработка ошибок (26) случайные

Подписка

мысли (26) EurekaLog (22) блог (17) прочее (16) Windows (14) не делай так (11) роботы/киберпанк (9) журнал (6) 7 (4) Tiburon (4) Vista (4) Королевство Delphi (4) TasksEx (3) x64 (2) Коты (2) кроссплатформенность (1) работа (1)