Практикум. Сериализация и десериализация объектов

Сейчас вы измените код класса так, чтобы обеспечить эффективную сериализацию, и обновите приложение, чтобы оно выполняло сериализацию и десериализацию этого класса.

Упражнение 1. Включение возможности сериализации класса

Вы должны изменить класс, чтобы разработчики могли сохранять соответствующие объекты на диске для дальнейшего использования и передачи по сети другому .NET-приложению.

1. Скопируйте на жесткий диск папку Lesson1-Serialize-People
2. Просмотрите класс Person. Какие изменения нужно внести, чтобы этот класс можно было сериализовать?

Нужно добавить атрибут Serializable.

1. Добавьте в класс пространство имен System.Runtime.Serialization.
2. Добавьте к классу Person атрибут Serializable, а затем соберите проект, чтобы убедить­ся, что он успешно компилируется.

Упражнение 2. Сериализация объекта

Сейчас вы напишете код для сохранения объекта на диске, используя наиболее эффек­тивный метод.

1. Откройте проект Serialize-People, который вы изменяли в упражнении 1.
2. Импортируйте пространства имен System.IO, System.Runtime.Serialization и System.Run-time.Serialization .Formatters.Binary в файл, содержащий функцию Main.
3. Добавьте в метод Serialize код для сериализации объекта sp в файл Person.dat, распо­ложенный в текущей папке. Код будет выглядеть примерно так:

// C#

private static void Serialize(Person sp) {

// Создаем файл для сохранения данных

FileStream fs = new FileStream("Person.Dat", FileMode.Create);

// Создаем объект BinaryFormatter для выполнения сериализации BinaryFormatter bf = new BinaryFormatter();

// Используем объект BinaryFormatter для сериализации данных в файл bf.Serialize(fs, sp);

// Закрываем файл fs.Close();

}

1. Соберите проект, исправьте ошибки.
2. Откройте командную строку в папке приложения и протестируйте его, выполнив сле­дующую команду:

Serialize-People Топу 1923 4 22

6. Откройте файл и просмотрите сериализованные данные, чтобы убедиться, что вве-  
денное имя успешно перехвачено. Проверьте также дату и возраст; однако эти дан-  
ные труднее обнаружить в текстовом редакторе.

Упражнение 3. Десериализация объекта :

Вы прочитаете с диска сериализованный объект, используя BinaryFormatter.

1. Откройте проект Serialize-People, который вы изменяли в упражнениях 1 и 2.
2. Добавьте код в метод Deserialize основной программы, чтобы десериализовать объект dsp из файла Person.dat, расположенного в текущей папке. Этот код выглядит при­мерно так:

// C#

private static Person DeserializeO {

Person dsp = new Person();

// Открываем файл для чтения данных

FileStream fs = new FileStream("Person.Dat", FileMode.Open);

// Создаем объект BinaryFormatter для выполнения десериализации

BinaryFormatter bf = new BinaryFormatter();

// Используем объект BinaryFormatter для десериализации данных из файла

dsp = (Person)bf.Deserialize(fs);

// Закрываем файл fs.Close();

return dsp;

}

1. Соберите проект, исправьте ошибки
2. Откройте командную строку в папке приложения и выполните следующую команду без параметров:

Serialize-People

Заметьте, что команда Serialize-People отображает имя, дату рождения и возраст, взя­тые из ранее сериализованного объекта Person.

Упражнение 4. Оптимизация класса для десериализации

Ваша задача — изменить класс, чтобы повысить эффективность сериализации.

1. Откройте проект Serialize-People, с которым вы работали в упражнениях 1-3.
2. Измените класс Person, чтобы предотвратить сериализацию члена с данными о воз­расте. Для этого добавьте атрибут NonSerialized, как показано в следующем фрагмен­те кода:

// C#

[NonSerialized] public int age;

1. С и запустите проект без параметров командной строки. Заметьте, что команда Serialize- People отображает имя и дату рождения, полученные из ранее сериализо­ванного объекта Person. Однако вместо возраста отображается ноль.
2. Измените класс Person, чтобы реализовать в нем интерфейс IDeserializationCallback, как показано в следующем фрагменте кода:

// C#

namespace Serialize\_People {

[Serializable]

class Person : IDeserializationCallback

5. Добавьте в класс Person метод IDeserializationCallback.OnDeserialization. Код будет вы-  
глядеть примерно так:

// C#

void IDeserializationCallback.OnDeserialization(Object sender) {

// После десериализации вычисляем возраст

CalculateAge();

}

6. Постройте и запустите проект без параметров командной строки. Заметьте, что ко-  
манда Serialize- People отображает имя, дату рождения и возраст, полученные из ра-  
нее сериализованного объекта Person. Теперь возраст отображается корректно, так  
как вычисляется сразу после десериализации.