**Properties**

В Java принято делать программы гибкими и легко настраиваемыми. Иногда еще говорят легко конфигурируемыми (от конфигурация).

Например, ваша программа раз в час копирует файлы из некоторой директории, архивирует их и отсылает Вам на email. Для этого программе надо знать директорию, откуда брать файлы и email, куда их посылать. Такие данные принято хранить не в коде программы, а в отдельных файлах свойств – **properties**-файлах.

Кроме того многие разработчики сталкиваются с необходимостью чтения конфигурационных (\*.ini, \*.prop, \*.conf, etc.) файлов в разрабатываемых приложениях. С помощью класса Properties можно очень легко загрузить ini-файл и прочитать его свойства.

Данные в таком файле хранятся в виде пар ключ-значение, разделенные знаком равно.

**Пример**

Файл data.properties

directory=c:/text/downloads

email=zapp@javarush.ru

Слева от знака равно – имя (ключ), справа – значение.

— Т.е. это что-то типа текстового представления HashMap?

— В общем-то, да.

Для удобной работы с такими файлами в Java есть специальный класс – Properties. Класс Properties унаследован от Hashtable<Object,Object>. Его даже можно рассматривать как HashTable, который умеет загружать себя из файла.

Вот его методы:

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| void **load**(Reader reader) | Загружает свойства из файла, представленного объектом Reader |
| void **load**(InputStream inStream) | Загружает свойства из файла, представленного объектом InputStream |
| void **loadFromXML**(InputStream in) | Загружает свойства из XML-файла |
| Object **get**(Object key) | Возвращает значение по ключу. Метод унаследован от HashTable |
| String **getProperty**(String key) | Возвращает значение свойства (строку) по ключу |
| String **getProperty**(String key, String defaultValue) | Возвращает значение свойства по ключу или defaultValue, если такого ключа нет |
| Set<String> **stringPropertyNames**() | Возвращает список всех ключей |

Т.е. фактически Вам нужно выполнить всего две операции – загрузить в объект **Properties** данные из какого-нибудь файла, а затем получить эти свойства с помощью метода **getProperty**(). Ну и не забывайте, что можете пользоваться объектом **Properties** как **HashMap**.

Вот пример:

**Код**

//файл, который хранит свойства нашего проекта

File file = **new** File("c:/data.properties");

//создаем объект Properties и загружаем в него данные из файла.

**Properties** properties = **new** Properties();

properties.load(**new** FileReader(file));

//получаем значения свойств из объекта Properties

String email = properties.getProperty("email");

String directory = properties.getProperty("directory");

//получаем числовое значение из объекта Properties

**int** maxFileSize = Integer.parseInt(properties.getProperty("max.size", "10000"));

— Ага. Т.е. мы создаем объект Properties, затем передаем в него файл. В метод load, а затем просто вызываем getProperty. Так?

— Ага.

— А Вы еще говорили, что им можно пользоваться как HashMap? Что Вы имели в виду?

— Класс Properties унаследован от Hashtable, а это – тот же HashMap, просто все методы его синхронизированы. Вот так можно просто вывести на экран все значения из файла свойств:

**Код**

//получаем файл со свойствами

File file = **new** File("c:/data.properties");

//создаем объект Properties и загружаем в него данные из файла.

**Properties** properties = **new** Properties();

properties.load(**new** FileReader(file));

//проходимся по всем ключам и печатаем все их значения на консоль

**for** (String key : properties.stringPropertyNames())

{

System.out.println(properties.get(key));

}

**Еще один пример**

Файл конфигурации:

SOME\_INT\_VALUE = 2

SOME\_DOUBLE\_VALUE = 1.2

SOME\_STRING\_VALUE = foo

SOME\_INT\_ARRAY = 1;2;3

Загрузка конфигурации с помощью класса Properties.

public class Example1

{

private static int SOME\_INT\_VALUE = 1;

private static double SOME\_DOUBLE\_VALUE;

private static String SOME\_STRING\_VALUE;

private static int[] SOME\_INT\_ARRAY;

public Example1() throws IOException

{

Properties props = new Properties();

props.load (new FileInputStream (new File ("config/example.ini") ) );

SOME\_INT\_VALUE = Integer.valueOf(props.getProperty("SOME\_INT\_VALUE", "1"));

SOME\_STRING\_VALUE = props.getProperty("SOME\_STRING\_VALUE");

SOME\_DOUBLE\_VALUE = Double.valueOf(props.getProperty("SOME\_DOUBLE\_VALUE", "1.0"));

// Предположим, что в настройках находится список целых чисел через точку с запятой

String[] parts = props.getProperty("SOME\_INT\_ARRAY").split(";");

SOME\_INT\_ARRAY = new int[parts.length];

for (int i = 0; i < parts.length; ++i)

{

SOME\_INT\_ARRAY[i] = Integer.valueOf(parts[i]);

}

}

public static void main(String[] args) throws IOException

{

new Example1();

}

}