如果经常会发生变化的数据我们可以定义在配置文件上。 比如说：数据库的用户名与密码。

应用程序首选项(**配置信息**)存储

**什么是首选项？**

**首选项**Preferences 指**配置信息**

**首选项存储**就是指对**配置信息的存储**

**有什么方式存储？**

有两种，

**第一种**

直接将配置信息存储在孤立的**属性文件**中，使用属性映射（property map）的数据结构来存储属性的键值对信息，属性之间没有层次结构。Properties类实现了属性映射数据结构，继承于HashTable类。

在该属性文件中，配置信息都是文本信息，都是String

/\*专门使用一个类来组织配置信息\*/

**写一个属性映射列表**：

Properties settings = new Properties(); //创建一个无默认值的空属性列表

settings.setProperty(“width”, “200”); //内部调用put，这里只是为了形参限制为String类型

settings.setProperty(“title”, “ixenos”);

**将属性映射列表保存到文件中**：

FileOutputStream out = new FileOutputStream(“program.properties”); //相对路径

Settings.store(out, “Program Properties”); //第二个参数是对这个文件的注释

在文件中将显示：

#Program Properties

#Sun Sept 4 07:31:52 2016

width=200

title=ixenos

**从文件中加载属性**：

FileInputStream in = new FileInputStream(“program.properties”);

Settings.load(in);

习惯上，Properties需要将程序配置存储在用户**主目录的某个子目录下**，**子目录名**通常**由一个圆点开始**：如 **.corejava**

String userDir = System.getProperty(“user.home”); //获得用户主目录字符串表示

**Properties类为应用程序指定默认值的两种方式**：

**getProperty时指定**：

String title = settings.getProperty(“title”, “Default Title”); //当键值不存在时返回默认值

**在构造器指定，传入一个二级属性映射**：

Properties defaultSettings = new Properties();

defaultSettings.setProperty(“width”, );

defaultSettings.setProperty();

defaultSettings.setProperty();

…

Properties settings = new Properties(defaultSettings); //为查询表提供默认值

Properties类缺点：**1.没有层次结构，没有标准的为配置文件命名的规则**

**2.配置文件不能放置在主目录中**

**3.要求键是唯一的**

**树形结构可以解决（如Preferences导出的XML格式）**

**第二种存储配置信息的方式：**

Preferences首选项API

Preferences类提供了一个与平台无关的中心知识库，其底层实现是native的

Preferences的中心知识库具有**树状结构**，这样每个节点的路径就与包名形式一致，因此建议**配置节点路径与程序中的包名匹配**，如/com/ixenos/myapp

中心知识库中的**每一个节点都有一个独立的map**，可以用来存放数字、String、byte[]，但不适宜存放JDK序列化的对象（二进制流），因为API设计者认为序列化格式不适于长期存放数据，不过可以将序列化对象存放在字节数组中。

为了增加灵活性，**系统中有多棵并行的树**。

每个程序使用者拥有一棵树，同时，系统中还存一棵树，称为系统树，用于存放全体用户的公共信息

**获取特定节点**的两种方式：

第一种**基于节点所在位置**：

Preferences root = Preferences.userRoot(); //先获取用户或系统的根

//Preferences root = Preferences.systemRoot();

Preferences node = root.node(“/com/ixenos/myapp”); //节点由节点路径获得

第二种**基于类所在位置**：

Preferences node = Preferences.userNodeForPackage(some.getClass()); //也可以直接填入Class对象；通常obj就是this引用

// Preferences node = Preferences.userSystemForPackage(some.getClass());

一旦获得节点，就可以使用一系列 xxx getXXX(String key, xxx default) 方法访问键值表，右边的形参是对应类型的默认值；对应的用put(String key, xxx value)方法填入数据

**Preferences类使配置数据易于迁移**

可以调用下列方法将子树或者一个节点的全部值显示输出

void exportSubtree(OutputStream out)

void exportNode(OutputStream out)

**数据将以XML格式存储，很关键**

可以调用下列方法将他们导入到另一个中心知识库中

void importPreferences(InoutStream in)

例如：

<?xml version=”1.0” encoding=”UTF-8”?>

<!DOCTYPE perferences SYSTEM “http://java.sun.com/dtd/preferences.dtd”>

<preferences EXTERNAL\_XML\_VERSION=”1.0”>

<root type=”user”>

<map/>

<node name=”com”>

<map/>

<node name=”ixenos”>

<map/>

<node name=”corejava”>

<map>

<entry key=”left” value=”11”>

<entry key=”top” value=”9”>

<entry key=”width” value=”22”>

<entry key=”height” value=”52”>

<entry key=”title” value=”Halo”>

</map>

</node>

</node>

</node>

</root>

</preferences>