OpenCASCADE Application Framework

Data Framework Services

eryar@163.com

一、概述 *Overview*

OpenCASCADE 的数据框架对来自不同程序的数据提供了统一的处理环境。这就简化了数据交换、修改，也保证了数据统一性、稳定性。实现方法需要用到以下部分：

- 标号 ***Tha tag***
- 标签 ***The label***
- 属性 ***The attribute***

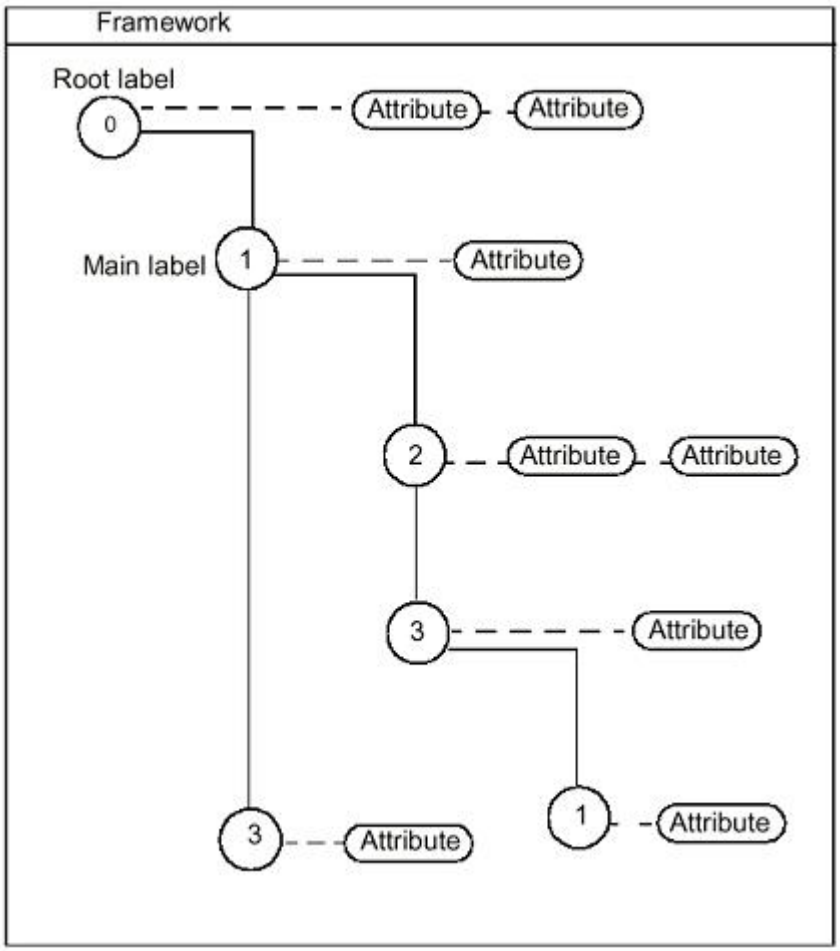


Figure 1. Contents of a document

如上图所示，框架树的第一个标签（***label***）是根标签（***root***）。每个标签（***label***）有个以整数表示的标号（***tag***）。由当前标签的标号到根标签的标号，可以得到一个惟一的标号列表，如：***0:1:2:1***。

每个标签（***label***）可以一些属性（***attribute***），这些属性可以包含数据。每个属性由 ***GUID*** 来区分。标签最重要的性质是其入口只是数据框架的一个地址。

二、标号 *The Tag*

一个标号 ***Tag*** 就是一个整数，它用两种方式标示了一个 ***label***：

- 相对标示法 ***Relative identification***：一个标签的标号只与其父标签有关系。如对于一个指定的标签，可能由四个子标签组成，其标号分别为 ***2***，***7***，***18***，***100***。使用相对标示方法，在设置属性时有安全的范围。
- 绝对标示法 ***Absolute identification***：一个标签在数据框架中位置由无歧义的、从根标签的标号到当前标签的标号用冒号表示的标号列表（***list of tags***）来表示。

不管采用哪种方法，都要注意的是这些标号的值没什么实际的意义。只是用来确定每个标签在树结构的位置，都是为了使用文档支持 ***Undo***/***Redo*** 的功能。

创建标号 ***Tag*** 的两种方式：

- ***Random delivery*** 随机创建；
- ***User-defined delivery*** 用户自定义创建；

正如字面所说，随机创建标号时，标号是由系统随机生成。用户自定义创建标号时，标号的值是创建标号函数的参数。

1. 随机创建标号法生成子标签 ***Creating child labels using random delivery of tags***

使用 ***TDF_TagSource***::***NewChild*** 来添加标签。如下代码所示，函数 ***NewChild*** 的参数 ***level2*** 也是一个 ***TDF_Label***。

Example

```
TDF_Label child1 = TDF_TagSource::NewChild (level2);
TDF_Label child2 = TDF_TagSource::NewChild (level2);
```

2. 用户自定义创建标号法生成子标签 ***Creation of a child label by user delivery from a tag***

创建子标签的另一种方式就是用户自定义创建。即在指定标号创建标签。可以使用 ***TDF_Label***::***FindChild*** 和 ***TDF_Label***::***Tag*** 来获得指定标号的子标签。

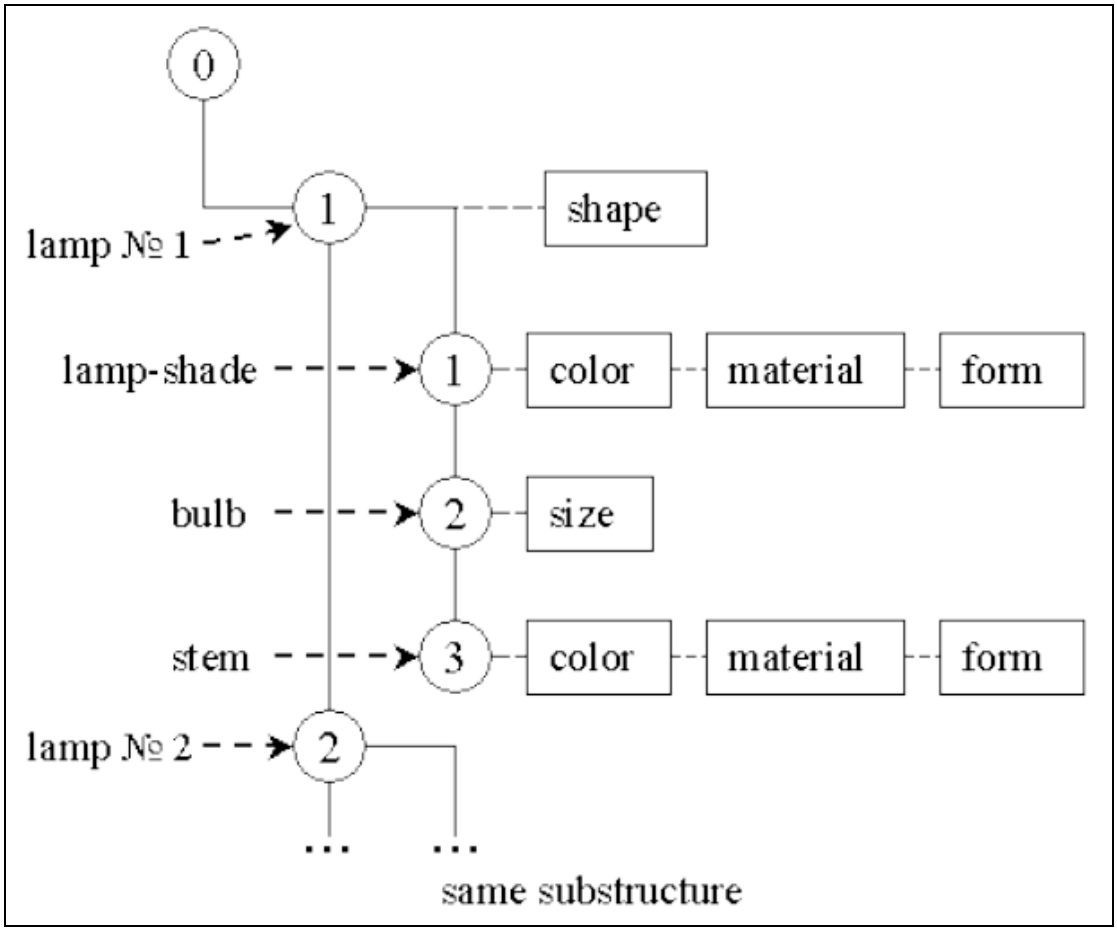
Example

```
TDF_Label achild = root.FindChild(3,Standard_False);
if (!achild.IsNull()) {
Standard_Integer tag = achild.Tag();
}
```

如上代码所示，***3*** 是需要查找的标签的标号，***Standard_False*** 用来表示若查找不到指定标号时是否创建子标签。

三、**标签** ***The Label***

标号（***Tag***）给了标签（***Label***）一个唯一的地址。数据框架中的标签是包含属性，绑定数据的容器。数据框架的本质是一个标签树，如下图所示：



数据框架中的标签不能被删除，因此，当文档打开后已经的数据框架结构不能被删除。

1. 创建标签 ***Label creation***

可以在任意层次创建标签，也可以找到标签在数据框架中的深度（***Depth***）。***TDF_Label*** 提供上述功能。

2. 创建子标签 ***Creating child labels***

在数据框架中指定标签上创建子标签使用 ***TDF_Label***::***FindChild***。如下所示：

Example

```
//creating a label with tag 10 at Root
TDF_Label lab1 = aDF->Root().FindChild(10);

//creating labels 7 and 2 on label 10
TDF_Label lab2 = lab1.FindChild(7);

TDF_Label lab3 = lab1.FindChild(2);
```

当把 ***FindChilde*** 的第二个参数设为 ***Standard_True*** 时，就确保了查找不到指定标号的标签时会创建一个标签。如下所示：

Example

```
TDF_Label level1 = root.FindChild(3,Standard_True);
TDF_Label level2 = level1.FindChild(1,Standard_True);
```

3. 访问子标签 ***Retrieving child labels***

可以使用遍历器来访问当前标签的第一层的子标签。如下所示：

Example

```
TDF_Label current;
//
for (TDF_ChildIterator it1 (current,Standard_False); it1.More(); it1.Next()) {
achild = it1.Value();
//
// do something on a child (level 1)
//
}
```

也可以访问当前标签的所有子标签，如下所示：

Example

```
for (TDF_ChildIterator itall (current,Standard_True); itall.More(); itall.Next()) {
achild = itall.Value();
//
// do something on a child (all levels)
//
}
```

使用 ***TDF_Tool*::*Entry*** 可以得到当前标签的入口字符串，如下所示：

Example

```
void DumpChildren(const TDF_Label& aLabel)
{
TDF_ChildIterator it;
TCollection_AsciiString es;
for (it.Initialize(aLabel,Standard_True); it.More(); it.Next()){
TDF_Tool::Entry(it.Value(),es);
cout << as.ToCString() << endl;
}
}
```

4. 访问父标签

访问当前标签的父标签：

Example

```
TDF_Label father = achild.Father();
isroot = father.IsRoot();
```

四、**属性** ***The Attribute***

标签本身不包含任何数据。所有数据，不管什么类型，程序的非程序的数据都是保存在属性中。属性是绑定在标签上，且属性可以是任意类型的数据。***OCAF*** 提供许多直接可以使用的属性如：整数、实数、轴、平面。也有用于拓朴、功能、可视化的属性。每种类型的属性由 ***GUID*** 来标识。这样做的好处就是所有类型的属性都以相同的方式处理。可以创建新的实例，访问、绑定到标签和从标签上删除等。

1．访问标签的属性

使用函数 ***TDF_Label***::***FindAttribute*** 来访问标签的属性。如下例所示，

Example

```
if(current.FindAttribute
        (TDataStd_Integer::GetID(),INT))
{
    // the attribute is found
}
else {
    // the attribute is not found
}
```

2．使用 ***GUID*** 来标识属性 ***Identifying an attribute using a GUID***

可以创建一个属性对象并得到其 ***GUID***。如下例所示，创建了一个整数属性，通过方法 ***ID*** 来得到 ***GUID***。

Example

```
Handle(TDataStd_Integer) INT = new TDataStd_Integer();
Standard_GUID guid = INT->ID();
```

3． 将属性绑定到标签 ***Attaching an attribute to a label***

使用函数 ***TDF_Label***::***Add*** 来将属性绑定到标签。重复绑定相同 ***GUID*** 的属性到一个标签会出现错误。***TDF_Attribute***::***Label*** 可以得到绑定属性的标签。如下所示：

Example

```
current.Add (INT); // INT is now attached to current
current.Add (INT); // causes failure
TDF_Label attach = INT->Label();
```

4. 测试标签绑定状态 ***Testing the attachment to a label***

可以使用函数 ***TDF_Attribute*::*IsA*** 来检验属性是否已经绑定到标签上，函数的参数是属性的 ***GUID***。在下例所示，是检测当前标签是否有整数属性，然后得出这个标签属性的数量。函数 ***TDF_Tool*::*HasAttribute*** 用来检测标签是否绑定的有属性，函数 ***TDF_Tool*::*NbAttributes*** 返回标签绑定属性的数量。

Example

```
// Testing of attribute attachment
//
if (current.IsA(TDataStd_Integer::GetID())) {
    // the label has an Integer attribute attached
}
if (current.HasAttribute()) {
    // the label has at least one attribute attached
    Standard_Integer nbatt = current.NbAttributes();
    // the label has nbatt attributes attached
}
```

5. **删除标签的属性** ***Removing an attribute from a label***

若要将属性从标签中删除，可以使用 ***TDF_Label*::*Forget***，函数参数为属性的 ***GUID***。若要删除标签所有属性，使用函数 ***TDF_Label*::*ForgetAll***。

Example

```
current.Forget(TDataStd_Integer::GetID());
// integer attribute is now not attached to current label
current.ForgetAll();
// current has now 0 attributes attached
```

6. 特定属性的创建 ***Specific attribute creation***

见《***Application Framework User's Guide***》。

五、**示例程序** ***Sample Code***

```
//-----------------------------------------------------------
// Copyright (c) 2012 eryar All Rights Reserved.
//
//    File    : Main.cpp
//    Author  : eryar@163.com
//    Date    : 2012-11-4 21:25
//    Version : 0.1v
//
// Description : OpenCASCADE Application Framework sample
code.
//
```

```
//=====================================================

#include <iostream>
using namespace std;

#include <TDF_Tool.hxx>
#include <TDF_ChildIterator.hxx>
#include <TDataStd_Integer.hxx>
#include <TDocStd_Document.hxx>

// Use Toolkit: TKLCAF
#pragma comment(lib, "TKernel.lib")
#pragma comment(lib, "TKLCAF.lib")

int main(int argc, char* argv[])
{
    TCollection_AsciiString entry;

    Handle_TDocStd_Document myDF    = new
TDocStd_Document("myDocument");

    // Main label and root label of the data framework.
    TDF_Label   mainLabel   = myDF->Main();
    TDF_Label   root        = mainLabel.Root();

    cout<<"Main label :";
    mainLabel.EntryDump(cout);

    cout<<endl<<"Root label :";
    root.EntryDump(cout);

    // Create a label with tag 10 at Root.
    TDF_Label   myLabel    = root.FindChild(10);

    cout<<endl<<"Entry of the new label :";
    myLabel.EntryDump(cout);

    // Retrieving child labels.
    cout<<endl<<"Retrieving child labels: "<<endl;
    for (TDF_ChildIterator it(root); it.More(); it.Next())
    {
        it.Value().EntryDump(cout);
        cout<<endl;
    }
```

```
    // Attaching an attribute to a label.
    Handle_TDataStd_Integer INT = new TDataStd_Integer;
    myLabel.AddAttribute(INT);

    // Testing of attribute attachment.
    if (myLabel.IsAttribute(INT->GetID()))
    {
        cout<<"The attribute is attached to the label."<<endl;
    }
    else
    {
        cout<<"The attribute is not attached to the
label."<<endl;
    }

    // Removing an attribute from a label.
    myLabel.ForgetAttribute(INT->GetID());

    // Testing of attribute attachment.
    if (myLabel.IsAttribute(INT->GetID()))
    {
        cout<<"The attribute is attached to the label."<<endl;
    }
    else
    {
        cout<<"The attribute is not attached to the
label."<<endl;
    }

    return 0;
}
```