BdbInterface

BdbInterface 是 igem 项目中，对 bdbxml 的 C++ api 进行自身需要进行封装的接口类。目的是方便使用 bdbxml 进行数据管理。

一 dbxml 说明：

我们的数据库源文件分三块管理，分为 species，module, reaction 三块。其中每种物质或者反应都用单独的文件来进行存储。 在 dbxml 数据库中，用三个 container 来管理上述的三个部分，单独的文件作为 doc 由 container 进行组织。

xml 整个文档作为一个结点存在，结点由子结点组成，这种组织方式明显是递归组合的性质。我们使用 xml 最关心的是结点的 element 和 attribute 两部分内容。查找给定结点、查找给定结点的 element 值、查找给定结点的 attribute 值、修改给定结点的 element 和 attribute 值都是必须的。

对 xml 中 node 进行定位是很关键的部分，dbxml 使用 xquery 来进行查询操作，使用 xquery，指定 container、指定 document（这个不是必须的）、指定查询条件，就可以得到想要的 node 的内容。查询的语句和 sql 的 select 语句类似，但是由于 xml 的非结构化，查询结果表示比较灵活。dbxml 定义了 xmlResults 和 xmlValue 作为结果返回的类，通过 xmlValue 类，很容易将 node 转换成我们想要的格式，字符串、整型等基础类型。

xquery 使用 xpath 来定位结点，因为 xpath 本身的巧妙、高效和成熟的使用，所以在 BdbInterface 中，也使用 xpath 类似的字符串对 node 进行定位，而且 xpath 本身是比较简单，能够很快入门并且功能完全足够我们使用。下文的接口定义中，都会有一个参数 node_path。

二、BdbInterface 接口需求场景分析

1、查找 species、reaction 和 module 数据库中的任意结点的非结点元素值。

可以容易看到，这种最简单的应用场景是在文件中找到像如下结点的 element 值。

<fw_prom_eff>0.0</fw_prom_eff> 我们要能够找到结点的 0.0 进行返回。

2、查找数据库中结点的所有相同结点名字的子结点的元素值。

这种情况如下所示：

```
<link>
        <index>dmr_cIlam156_prot</index>
        <index>bid_pcIlam189_cIlam156_2</index>
</link>
```

要得到 link 结点所有 index 的内容，所以要使用一个字符串数组进行存储。

3、查找数据库中任意结点的属性值。

最基本的应用，查找到 index 结点的 id 属性的 value cIlam156_prot 字符串。

```
        <index id="cIlam156_prot"/>
```

4、查找数据库中结点的所有相同结点名字的子结点的相同属性值。

```
<listOfChains>
            <chain><module id="cIlam156" type="protein" class="biobrick"
                    bsite="cIlam2"/></chain>
            <chain><module id="cIlam156" type="protein" class="biobrick"
                    bsite="cIlam2"/></chain>
</listOfChains>
```

比如查找出 listOfChains 的所有子结点 chain 的 module 子结点的 id 值，返回该数组。

5、返回一个完全的结点

考虑到有些结点，比如 reaction 中的数学表达式，它可以直接被复制到 sbml 文件中去，所以没有必要对其进行处理，将整个结点作为字符串返回。

6、设置一个结点的非结点 element

<fw_prom_eff>0.0</fw_prom_eff>

设置 fw_prom_eff 为 1.0.

如果遇到下面的这种情况：

```
<link>
        <index>dmr_cIlam156_prot</index>
        <index>bid_pcIlam189_cIlam156_2</index>
</link>
```

设置 link 的 index 内容，因为 link 有两个 index 结点，可以在 xpath 中添加其他的限制条件来定位 index 结点，比如从位置上看是第一个还是第二个子结点，如果有属性值，可以在 xpath 语句中添加来区分。

7、设置一个结点的属性值

<index id="cIlam156_prot"/>

设置 idex 的 id 为其它的字符串。

如果有多个结点名字一样，同 6，使用 xpath 添加限制条件来区别对待。

三、数据结构定义：

数据库枚举类型：

```
Enum container_index
{
    MODULE = 0,
    SPECIES,
    REACTION
};
```

接口返回值枚举类型：

```
enum BdRetVal
{
    no_err,
    xml_exception,
    no_container
};
```

接口说明：

1

bdbXMLInterface::bdbXMLInterface()

说明：bdbXMLInterface 的构造函数。

BdbXMLInterface 设计了 3 个 container，分别为 species、module、reaction 命名的。这是考虑到我们以后会有很多 xml 的数据文件，所以我们应该支持大批量的文件导入操作，暂时根据 xml 所在的文件夹名字来确定 xml 存入哪个 container 中。如果是 species/路径下的文件，在调用下面说明的 add_file 的时候，该文件被加入 species 的 container 中。

2

BdRetValue add_files(const string& pathname, const string& docname)

说明： 返回给定数据库中给定种类的结点值的所有列表。

输出参数： res，返回结果值

输入参数： pathname 我们数据库路径下的 xml 文件路径。为了支持整个文件夹 xml 文件的导入，一定要包含在 species、module、reaction 文件夹下。比如/home/jkdirac/abc/database/species/IPTG.xml 或者/home/jkdirac/abc/module/cell/E_coli.xml 都是可以的路径。

doc 文档名字，E_coli.xml 的 document 文档可以是 E_coli

3

BdRetValue get_node_element(container_index container_type, const string *doc, const string *node_path, vector<string> &res);

说明： 返回给定数据库中给定种类的结点值的所有列表。

输出参数： res，返回结果值

输入参数： contianer_type 选择数据库种类

doc 文档名字，如果该指针为空，就在整个数据库中进行查询

node_path 查询的结点的 xpath 路径(除去文档和数据库部分)，如果该指针为空，返回整个文档内容。

使用举例 1：

如果是要查询 species 的 cIlam156_dimer.xml 的 link 的 index 所有内容，调用方式 get_node_element(SPECIES, “cIlam156_dimer”, “/species/link/index”, res);

如果函数返回值没有提示错误，res 就包含了字符串 dmr_cIlam156_prot 和 bid_pcIlam189_cIlam156_2

4

BdRetValue get_node_attr(container_index container_type, const string *doc, const string *node_path, vector<string> &res);

说明： 返回给定数据库中给定种类的结点的属性值列表。

输出参数： res，返回结果值

输入参数： contianer_type 选择数据库种类

doc 文档名字，如果该指针为空，就在整个数据库中进行查询

node_path 查询的结点的 xpath 路径(除去文档和数据库部分)，如果该指针为空，返回整个文档内容。

使用举例 1：如果是要查询 species 的 cIlam156_dimer.xml 的 listOfBsites 的 bsite 所有 lable 值，调用方式 get_node_element(SPECIES, “cIlam156_dimer”, “/species/listOfBsites/bsite/@label”, res);

如果函数返回值没有提示错误，res 就包含了字符串 cIlam2，如果有多个 bsite,那么 res 会包含所有的 bsite 的 label。

使用举例 2：

如果要查询 species 的 pcIlam189cilam156_2.xml 中 listOfChains 中 bsite 属性为 pcIlamcIlam 的 module 的 id 值。调用方式为 get_node_element(SPECIES, “pcIlam189cIlam156_2”, “/species/structure/listOfChains/chain/module[@bsite=\”pcIlamcIlam\”]/@id”, res); res 会返回 pcIlam189_v1。

5

BdRetValue get_node(container_index container_type, const string *doc, const string *node_path, string *res);

说明： 返回给定数据库中给定种类的结点所有内容。

输出参数： res，返回结果值

输入参数： contianer_type 选择数据库种类

doc 文档名字，如果该指针为空，就在整个数据库中进行查询

node_path 查询的结点的 xpath 路径(除去文档和数据库部分)，如果该指针为空，返回整个文档内容。

使用举例 1：

如果要查询 species 的 pcIlam189cilam156_2.xml 中 listOfChains 的 chain 结点，调用方式 get_node(SPECIES, “pcIlam189cIlam156_2”, “/species/structure/listOfChains/chain”, res); res 字符串将会是如下值：

```
<chain>
    <module id="X__pcIlam189_dna" type="dna" class="substituent"/>
    <module id="pcIlam189_v1" type="dna" class="virtual_biobrick" bsite="pcIlamcIlam"/>
    <module id="pcIlam189_dna__Y" type="dna" class="substituent"/>
</chain>
```

6

BdRetValue set_node_element(container_index container_type, const string *doc, const string *node_path, const string &set_value);

使用举例 1：

如果要修改 species 的 pcIlam189cilam156_2.xml 中 link 的 index 结点内容为 “abc”，调用方式 string value(“abc”);
set_node_element (SPECIES, “pcIlam189cIlam156_2”, “/species/link/index”, value);

7

BdRetValue set_node_attr(container_index container_type, const string*doc, const string *node_path, const string &set_value);

使用举例 1：

如果要修改 species 的 pcIlam189cilam156_2.xml 中 species 的 id 属性内容为 “abc”，调用方式 string value(“abc”);
set_node_attr(SPECIES, “pcIlam189cIlam156_2”, “/species/@id”, value);

替换功能实现比较繁琐，如果没有必要 6 7 两个函数就不提供了