XML & JSON

XML

SAX解析

1. 创建XML解析对象

```
 NSURL *fileURL = [[NSBundle mainBundle] URLForResource:@"bookstore"
withExtension:@"xml"];
 NSXMLParser *xmlParser = [[NSXMLParser alloc]
initWithContentsOfURL:fileURL];
```

2. 设置XML解析对象的代理
3. 开始解析

```
 BOOL flag = [xmlParser parse];
 if (!flag) {
     NSLog(@"xmlParser parse error.");
 }
```

NSXMLParser对象代理方法：

- (void)parserDidStartDocument:(NSXMLParser *)parser

> *当开始解析XML文档的时候，调用这个方法。通常在这个方法里，`创建存储模型对象的数组`*

- (void)parser:(NSXMLParser *)parser didStartElement:(NSString *)elementName namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName attributes:(NSDictionary *)attributeDict

> *当开始解析，遇到元素的开始标签的时候，调用这个方法。通常在这个方法里，`创建模型对象或解析标签中的属性保存在模型对象中`*

- (void)parser:(NSXMLParser *)parser foundCharacters:(NSString *)string

> *当解析到xml标签的文本内容的时候，调用这个方法，通常在这里先`暂存解析到的文本内容`*

- (void)parser:(NSXMLParser *)parser didEndElement:(NSString *)elementName namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName

> *当解析xml内容遇到结束标签的时候，调用这个方法。通常在这个方法里，`需要将模型对象保存入数组中或把标签对应的文本内容解析出来，保存在模型对象中（KVC）`*

- (void)parserDidEndDocument:(NSXMLParser *)parser

> *当整个xml文档全部解析结束的时候，该方法会被调用*

- (void)parser:(NSXMLParser *)parser parseErrorOccurred:(NSError *)parseError

> *当遇到解析错误时，该方法会被调用*

DOM解析 (以GDataXMLNode为例)

使用:

添加头文件搜索路径： `Build Settings->Header Search Paths->"/usr/include/libxml2"`

用非ARC的方式编译GDataXMLNode.m文件： `-fno-objc-arc`

链接libxml2.dylib该动态库： `Build Phases->Link Binary With Libraries->"libxml2.dylib"`

```
/*
 * <节点> (GDataXMLNode)
 * 根据 DOM，XML 文档中的每个成分都是一个节点。
 * DOM 是这样规定的：
 * 整个文档是一个文档节点
 * 每个 XML 标签是一个元素节点
 * 包含在 XML 元素中的文本是文本节点
 * 每一个 XML 属性是一个属性节点
 * 注释属于注释节点
 */
```

1. 根据XML文件创建NSData对象

```
 NSURL *fileURL = [[NSBundle mainBundle] URLForResource:@"bookstore"
withExtension:@"xml"];
 NSData *data = [NSData dataWithContentsOfURL:fileURL];
```

2. 根据NSData对象创建GDataXMLDocument对象（即DOM模型对象），该对象在内存中是以树形结构存储的

```
 GDataXMLDocument *doc = [[GDataXMLDocument alloc] initWithData:data
options:0 error:nil];
```

3. 通过DOM模型对象，取出XML文件的根元素

```
 GDataXMLElement *rootElement = [doc rootElement];
```

4. 由于是树形结构，所以，可以根据子元素的名字使用根元素获取到所有子元素，并类推获得子元素的子元素

```
 for (GDataXMLElement *element in elements) {
     // 创建图书对象
     YMBook *book = [[YMBook alloc] init];

     // 根据属性名字，解析book元素的属性值
     book.category = [[element attributeForName:kCategory]
stringValue];


     // 解析book的子元素包含的文本内容及其子元素的属性
     GDataXMLElement *titleElement = [element elementsForName:kTitle]
[0];
     book.title = [titleElement stringValue];
     book.lang = [[titleElement attributeForName:kLanguage]
stringValue];

     GDataXMLElement *authorElement = [element
elementsForName:kAuthor][0];
     book.author = [authorElement stringValue];

     GDataXMLElement *yearElement = [element elementsForName:kYear]
[0];
     book.year = [yearElement stringValue];

     GDataXMLElement *priceElement = [element elementsForName:kPrice]
[0];
     book.price = [priceElement stringValue];

     [_bookStore addObject:book];
 }
```

JSON

从结构上看，所有的数据（data）最终都可以分解成三种类型：

- 第一种类型是标量（scalar），也就是一个单独的字符串（string）或数字（numbers），比如"北京"这个单独的词
- 第二种类型是序列（sequence），也就是若干个相关的数据按照一定顺序并列在一起，又叫做数组（array）或列表（List），比如"北京，上海"
- 第三种类型是映射（mapping），也就是一个名/值对（Name/value），即数据有一个名称，还有一个与之相对应的值，这又称作散列（hash）或字典（dictionary），比如"首都：北京"

Douglas Crockford发明了JSON，Json的规定非常简单：

1. 并列的数据之间用逗号（", "）分隔
2. 映射用冒号（": "）表示

3. 并列数据的集合（数组）用方括号("[]")表示
4. 映射的集合（对象）用大括号（"{}"）表示

比如，下面这句话：

> *"北京市的面积为16800平方公里，常住人口1600万人。上海市的面积为6400平方公里，常住人口1800万。"*

写成JSON格式就是这样：

```
[
   {"城市":"北京","面积":16800,"人口":1600},
   {"城市":"上海","面积":6400,"人口":1800}
]
```

如果事先知道数据的结构，上面的写法还可以进一步简化：

```
[
   ["北京",16800,1600],
   ["上海",6400,1800]
]
```

`rfc-4627`

`www.json.org/json-zh.htm`

`www.sojson.com`

NSJSONSerialization

```
    + (id)JSONObjectWithData:(NSData *)data
                     options:(NSJSONReadingOptions)opt
                       error:(NSError **)error

    + (NSData *)dataWithJSONObject:(id)obj
                           options:(NSJSONWritingOptions)opt
                             error:(NSError **)error
```

JSONKit

使用： `-fno-objc-arc`

序列化: `NSArray` `NSDictionary` `NSString`

```
- JSONData
- JSONDataWithOptions:includeQuotes:error:
-
JSONDataWithOptions:serializeUnsupportedClassesUsingDelegate:selector:error:
- JSONDataWithOptions:serializeUnsupportedClassesUsingBlock:error:

- JSONString
- JSONStringWithOptions:includeQuotes:error:
-
JSONStringWithOptions:serializeUnsupportedClassesUsingDelegate:selector:erro
r:
- JSONStringWithOptions:serializeUnsupportedClassesUsingBlock:error:
```

反序列化: `NSData` `NSString`

```
- objectFromJSONData
- objectFromJSONDataWithParseOptions:
- objectFromJSONDataWithParseOptions:error:
- mutableObjectFromJSONData
- mutableObjectFromJSONDataWithParseOptions:
- mutableObjectFromJSONDataWithParseOptions:error:

- objectFromJSONString
- objectFromJSONStringWithParseOptions:
- objectFromJSONStringWithParseOptions:error:
- mutableObjectFromJSONString
- mutableObjectFromJSONStringWithParseOptions:
- mutableObjectFromJSONStringWithParseOptions:error:
```