1. DOM(Document Object Model)

DOM是用与平台和语言无关的方式表示XML文档的官方W3C标准。DOM是以层次结构组织的节点或信息片断的集合。这个层次结构允许开发人员在树中寻找特定信息。分析该结构通常需要加载整个文档和构造层次结构，然后才能做任何工作。由于它是基于信息层次的，因而DOM被认为是 基于树或基于对象的。

优点：

1. 允许应用程序对数据和结构做出更改。
2. 访问是双向的，可以在任何时候在树中上下导航，获取和操作任意部分的数据。

缺点：

1. 通常需要加载整个XML文档来构造层次结构，消耗资源大。
2. SAX(Simple API for XML)

SAX处理的优点非常类似于流媒体的优点。分析能够立即开始，而不是等待所有的数据被处理。而且，由于应用程序只是在读取数据时检查数据，因此不需要将数据[存储](http://www.storworld.com/)在内存中。这对于大型文档来说是个巨大的优点。事实上，应用程序甚至不必解析整个文档；它可以在某个条件得到满足时停止解析。一般来说，SAX还比它的替代者DOM快许多。

优点：

1. 不需要等待所有数据都被处理，分析就能立即开始。
2. 只在读取数据时检查数据，不需要保存在内存中。
3. 可以在某个条件得到满足时停止解析，不必解析整个文档。
4. 效率和性能较高，能解析大于系统内存的文档。

缺点：

1. 需要应用程序自己负责TAG的处理逻辑（例如维护父/子关系等），文档越复杂程序就越复杂。
2. 单向导航，无法定位文档层次，很难同时访问同一文档的不同部分数据，不支持XPath。
3. JDOM(Java-based Document Object Model)

JDOM的目的是成为Java特定文档模型，它简化与XML的交互并且比使用DOM实现更快。

JDOM与DOM主要有两方面不同。首先，JDOM仅使用具体类而不使用接口。这在某些方面简化了API，但是也限制了灵活性。第二，API大量使用了Collections类，简化了那些已经熟悉这些类的Java开发者的使用。

优点：

1. 使用具体类而不是接口，简化了DOM的API。
2. 大量使用了Java集合类，方便了Java开发人员。

缺点：

1. 没有较好的灵活性。
2. 性能较差。
3. DOM4J(Document Objcet Model for Java)

合并了许多超出基本XML文档表示的功能，包括集成的XPath支持、XML Schema支持以及用于大文档或流化文档的基于事件的处理。它还提供了构建文档表示的选项，它通过DOM4J API和标准DOM接口具有并行访问功能。

 为支持所有这些功能，DOM4J使用接口和抽象基本类方法。DOM4J大量使用了API中的Collections类，但是在许多情况下，它还提供一些替代方法以允许更好的性能或更直接的编码方法。直接好处是，虽然DOM4J付出了更复杂的API的代价，但是它提供了比JDOM大得多的灵活性。

优点：

1. 大量使用了Java集合类，方便Java开发人员，同时提供一些提高性能的替代方法。
2. 支持XPath。

缺点：

1. 大量使用了接口，API较为复杂。