Events vs. Trees

There are two major types of XML (or SGML) APIs:

Tree-based APIs
: These map an XML document into an internal tree structure, then allow an application to navigate that tree. The Document Object Model (DOM) working group at the World-Wide Web Consortium (W3C) maintains a recommended tree-based API for XML and HTML documents, and there are many such APIs from other sources.

Event-based APIs
: An **event-based API**, on the other hand, reports parsing events (such as the start and end of elements) directly to the application through callbacks, and does not usually build an internal tree. The application implements handlers to deal with the different events, much like handling events in a graphical user interface. SAX is the best known example of such an API.

Tree-based APIs are useful for a wide range of applications, but they normally put a great strain on system resources, especially if the document is large. Furthermore, many applications need to build their own strongly typed data structures rather than using a generic tree corresponding to an XML document. It is inefficient to build a tree of parse nodes, only to map it onto a new data structure and then discard the original.

In both of those cases, an event-based API provides a simpler, lower-level access to an XML document: you can parse documents much larger than your available system memory, and you can construct your own data structures using your callback event handlers.

Consider, for example, the following task:

> *Locate the record element containing the word "Ottawa."*

If your XML document were 20MB large (or even just 2MB), it would be very inefficient to construct and traverse an in-memory parse tree just to locate this one piece of contextual information; an event-based interface would allow you to find it in a single pass using very little memory.

To understand how an event-based API can work, consider the following sample document:

```
<?xml version="1.0"?>
<doc>
<para>Hello, world!</para>
</doc>
```

An event-based interface will break the structure of this document down into a series of linear events, such as these:

```
start document
start element: doc
start element: para
characters: Hello, world!
end element: para
end element: doc
end document
```

An application handles these events just as it would handle events from a graphical user interface: there is no need to cache the entire document in memory or secondary storage.

Finally, it is important to remember that it is possible to construct a parse tree using an event-based API, and it is possible to use an event-based API to traverse an in-memory tree. Have fun!