The library documentation

The data capture from a Microsoft library documentation page uses a specific property that is common for these pages, namely the *navigation menu* shown in the left column of the page. It contains *buttons* like the following:

The button shown in bold type font points to the current page. It is indented one level. The interesting part (for us) are the buttons just below pointing to the next level of documentation. Each such button is represented in the HTML source by a `div` element with `class` attribute `toclevel2` followed by an `a` element containing *path* and *text* of the button:

```
<div ... class="toclevel2" ...>
<a ... href="path" ... >text</a></div>
```

The details of an `a` element are described in Appendix A.1.

The function `nextInfo` in Table 10.16 steps an `XmlReader` forward to the next "node of interest". The possible values returned by `nextInfo` indicate:

`ButtonStart`	Found the start `div` element of a button in the sub-menu.
`RefInfo(`*text*`,`*path*`)`	Found some button `<a ...href="`*path*`"...>`*text*`</a>`.
`EndOfFile`	The `XmlReader` has reached the end of the file.

The function `nextInfo` is the essential ingredient in constructing the function:

```
getWEBrefs : string -> (string * string) list
```

that takes a uri as argument and reads through the corresponding web-page source and extracts the list of pairs `(`*title*`,`*uri*`)` corresponding to buttons in the above described sub-menu of the navigation menu. The complete program is found in Appendix A.3. The reader may pay special attention to the following:

- The use of the `HttpUtility.HtmlDecode` function.
- The use of an `Uri` object to convert a *path* in a link to the corresponding absolute *uri*.

```
type infoType = | ButtonStart | RefInfo of string * string
                | EndOfFile;;

let rec nextInfo(r:XmlReader) =
  match r.Read() with
  | false -> EndOfFile
  | _     ->
  match r.NodeType with
  | XmlNodeType.Element ->
    match r.Name with
    | "div" when (r.GetAttribute "class" = "toclevel2")
          -> ButtonStart
    | "a" -> let path = r.GetAttribute "href"
             ignore(r.Read())
             RefInfo(r.Value,path)
    |  _  -> nextInfo r
  |  _    -> nextInfo r ;;
val nextInfo: XmlReader -> infoType
```

Table 10.16 *The* `nextInfo` *function*

10.10 Keyword index example: Putting it all together

The keyword index is generated using the text files:

- `webCat0.txt`
- `keywords.txt`

and the programs:

- `NextLevelRefs`
- `MakeWebCat`
- `IndexGen`

The text file `webCat0.txt` is shown in Table 10.17. It contains two pairs of lines with the title and uri of the two root documentation pages. The text file `keywords.txt` contains titles of documentation web-pages with associated keywords and an extract of the file is shown in Table 10.1.

The keyword index is generates in three steps:

1. Generate the text files `webCat1.txt` and `webCat2.txt` using `NextLevelRefs`.
2. Generate the binary file `webCat.bin` using `MakeWebCat`.
3. Generate the text file `index.html` using `IndexGen`.