ALL ABOUT XSLT

XSLT のすべて

4D デベロッパカンファレンス 2010 配付資料

XSLT を積極的に使用するべき理由

XML の処理を標準的な方法で

4D で XML が使用される場面は，増える一方です。ストラクチャ定義，定数定義，バックアップ設定，キーボードショートカット設定，アプリケーションビルド設定，検査ログファイル，パスファイル（4DLINK），ローカライズリソース（XLIFF），マクロ，クエリプラン・クエリパスなど，4D 内部で多用されているだけでなく，Web サービス（WSDL，SOAP エンベロープ）や SVG など，外部とのインタフェースにも XML がしばしば用いられます。

XML は，構造化された標準テキストファイルであり，広く普及している汎用的な技術です。プラットフォーム独立であり，高い互換性を有しています。冗長であるともいわれますが，XML 文書は比較的容易に JSON，CSV，バイナリデータなど他の形式に変換することができます。

XML は 4D 2003 で限定的に実装され，4D 2004 で大幅に API が拡充されました。4D v11 SQL では，アプリケーション本体が Unicode となったことにより，本格的に XML を処理できるプラットフォームが整いました。加えてベクトルピクチャーの表現に SVG 形式が選択されたことは，いまや XML が 4D アプリケーションに欠かせないものになったことを意味します。

さまざまな API の特徴を把握する

現在，4D で XML を処理する手段は，3 タイプに大別されます。DOM，SAX そして PROCESS HTML TAGS です。これら異なる API は，それぞれに特徴があり，その特性を把握することがたいせつです。

DOM は，ランダムアクセスであり，オブジェクトモデルをオンメモリで展開するため，処理が高速であるといわれてきました。一方の SAX がシーケンシャルアクセスであり，ドキュメントサイズに制約されないのと対照的です。しかし，両者の違いはそれだけではありません。DOM は，4D の Web サービスコマンドと連動し，SOAP ヘッダのカスタマイズに使用できます。また，SVG ピクチャの制御に使用できるのも DOM のおおきな特徴です。反対に SAX は，XML のノードベースの API であり，プロセッシングインストラクション，コメント，ドキョメントタイプなど，あらゆるノードタイプに対するアクセスがあります。その点，DOM はノードではなく，属性または要素の値がアクセスの対象なので，XML に対する完全なコントロールはありません。

PROCESS HTML TAGS は，XML コマンドではありませんが，テンプレートによるテキストの生成ができるので，ダイナミックな XML の出力に幅広く利用されてきました。簡単な条件分岐を組み込むことができ，4D の変数・メソッド・コマンドをコールバックできるのも強みのひとつです。

XSLT の長所

XSLT は XML を処理し，HTML，（元とは違うタイプの）XML，あるいはその他の標準テキストに変換するための標準的な言語であり，非常に強力です。4D の XSLT は Apache の xalan-c ライブラリを使用しており，XSLT 1.0 をフルサポートしています。DOM や SAX のようにノードや要素・属性といった単一のオブジェクトではなく，ノードセットをクエリすることができるので，並び替え，グループ化など，高度なデータ処理をすることができます。DOM コマンドならば何重もの階層的ループを要する処理が，XSLT ならばわずか数行で済むことも珍しくありません。DOM コマンドはソース XML の構造が事前に分からない場合，あるいは後で変わるかもしれない場合，コードを最適化することが困難ですが，XSLT はそうしたケースにも柔軟に対応することができます。最後に XSLT は 4D からは完全に独立した技術であり，豊富な情報資源と人材にサポートされているスタンダートな技術です。

XSLT を活用する

W3C

XSLT は World Wide Web Consortium で仕様が公開されているので, XSLT, xPath, XML のドキュメントをブックマークしておくことは有益です。もし XSLT で SVG を扱うのであれば, SVG のドキュメントもブックマークしておくと良いでしょう。

http://www.w3.org/TR/xslt
http://www.w3.org/TR/xpath/
http://www.w3.org/XML/
http://www.w3.org/TR/SVG11/

サンプル

XSLT のサンプルは, インターネットや参考書などから入手することもできますが, 4D アプリケーション自体も XSLT を使用しているので, そのプログラムを研究するという方法も有効です。GRAPH コマンド, ストラクチャ定義の HTML 書き出し, メンテナンスアンドセキュリティセンターのログファイル出力などは, いずれも XML データを XSLT で加工することにより, HTML や SVG などのドキュメントを作成しています。

コマンド

4D で使用する XSLT 関連コマンドは APPLY XSLT TRANSOFRMATION および SET XSLT PARAMETER のふたつだけです。実際のロジックは, BLOB または外部ドキュメントから読み込まれることになります。後者の場合, プログラムの本質的なソースコードが 4D ストラクチャの外部に標準テキストファイルとして置かれることになるので, バージョン管理やメンテナンスの点でも非常に便利です。

テスト

XSLT は,実行直前にコンパイルされる言語なので, インタプリタ言語のように試行錯誤しながら編集できない不便さがあります。また記述したプログラムに問題があった場合, たいていは解析の段階でエラーになるか, 何の結果も返されない場合がほとんどなので, 原因の特定が難しい場合も少なくありません。このセッションでは, 4D v11 のマクロ機能を活用して統合開発環境を拡張し, メソッドエディタとペーストボード使用して XSLT を手早く編集する方法を紹介します。XSLT の動作を確認しながら開発することができ, コードの骨組みができた後は, 一般的な XML エディタなどでプログラムを仕上げると良いでしょう。

応用編

XSLT のコマンド・関数はそれほど多くありません。それらの組み合わせ, および XSLT に XSLT を重ねるなどの創意工夫によって, ほんとうに強力な XSLT が出来上がります。とりわけ下記のコマンドは, 用法次第では非常に有用なものになるポテンシャルを秘めています。

<xsl:copy-of />	XML 断片の挿入, XML 断片の書き出し。
<xsl:comment />	コメントの出力。これで 4D タグを挿入すれば PROCESS HTML TAGS に続ける。
document()	ルックアップテーブルや外部ドキュメントの参照。(ドキュメントのみ。BLOB 不可)
str:encode-uri()	URI のパーセントエンコーディング。(拡張関数)
dyn:evaluate()	xPath を引数で受け取るなど。(拡張関数)
exsl4D:format-number()	16 進数変換など。(隠しコマンド)