UNICODE AND REGULAR EXPRESSIONS

Unicode と正規表現

4D デベロッパカンファレンス 2010 配付資料

目次

Unicode と正規表現

概要

4D v11 SQL より, 4D はキャラクターセットに Unicode, エンコーディングに UTF-16, 文字列の対照(コレーション)とインデックスの構築に ICU を採用しています。いずれもたいへん広大なトピックですが, このセッションでは, 4D で Unicode を扱う際に知っておくべきたいせつな情報を中心に取り上げます。

非 Unicode

このドキュメントでは, Unicode 以前の互換性エンコーディングを都合上, 4D エンコーディングと呼びます。4D エンコーディングは, 日本語環境で起動した 4D の場合, Windows 版の Shift_JIS(コードページ 932)とほぼ同じです。Unicode と 4D エンコーディングの相互変換には, 4D Extensions フォルダに収められた特殊なファイル「japanese.uni」が使用されています。

4D エンコーディングは, 最終的に廃止される予定ですが, 非 Unicode モードをサポートするために残されています。たとえば非 Unicode モードのプラグイン, 非 Unicode モードのコンポーネントとの文字列の受け渡しに使用されています。また, 4D のデザインモードにも一部 4D エンコーディングが使用されている箇所があります。

4D のデザインモードで 4D エンコーディングが使用されている箇所

4D v11 は, 互換性に関係する理由により, 一部 Unicode ではない箇所があります。はじめにこのことを意識しておくことはたいせつです。

各種エディタ

Unicode ではない箇所の例として, フォームエディタ, メソッドエディタ, ラベルエディタ, デバッガなどの各種エディタが挙げられます。いずれの場合も, データは Unicode ですが, エディタが Unicode ではない, という点に留意してください。たとえば, メソッドエディタの場合, 変数の内容は Unicode ですが, スタティックテキストとして入力できる文字列は 4D エンコーディングに限られます。フォームエディタの場合, XIFF や変数の内容は Unicode ですが, フォームエディタやプロパティリストに入力できる文字列は 4D エンコーディングに限られます。フォームやメソッドに Unicode のスタティックテキストを使用したい場合, XLIFF を使用するのが標準的な方法です。

なお, v12 では, メソッドエディタの入力エリアが Unicode になります。ただし, 互換性の理由から, メソッドの内部保存形式は 4D エンコーディングとなります。

インポートダイアログ

4D v11 のインポートダイアログは, 4D エンコーディングを使用しています。なお, v12 では, インポートダイアログが Unicode になり, またより標準的なテキストファイルによるデータ交換フォーマットとして, SQL インポートがサポートされるようになります。

非 Unicode モードを使用するべきではない理由

4D v11 SQL にアップグレードされたデータベースは, デフォルトのステータスが非 Unicode モードです。非 Unicode モードは, 段階的な移行を助けるためのステップであり, できるだけ早い段階で使用をやめることが推奨されています。非 Unicode モードは最終的に廃止される予定です。

非 Unicode モードを使用するべきではない理由はいくつか挙げられます。

はじめに, 非 Unicode モードは速度面で非常に不利です。4D のデータベースエンジン, SQL サーバー, 文字列コマンドは, すべて内部的には Unicode で動作しています。非 Unicode モードは, 毎回の演算で Unicode

から 4D エンコーディング，およびその逆の変換を実施することによって実現しているので，かなりのパフォーマンスダウンを覚悟しなければなりません。

次に，非 Unicode モードはエンコーディングの変換を実施する必要があるので，文字列データが失われる可能性があります。とりわけ Unicode から 4D エンコーディングへの変換の際，変換できない文字列は'?'に置換され，元のデータが分からなくなってしまうので問題です。

さらに，非 Unicode モードでは文字列のサイズに 32,000 バイトという制限があります。これは XML など，おおきなテキストデータを扱う際に致命的です。また，正規表現コマンドも単純な ASCII 文字列以外には適用することができません。

最後に，アプリケーションには単一の Unicode コレーションルールしか存在しない以上，文字列の照合は飽くまで Unicode を基準に計算されるのであり，たとえ非 Unicode モードであっても厳密な意味で 4D 2004 以前と同じ動作にはならないことが挙げられます。つまり，非 Unicode モードであっても，Unicode を意識する場面があるのであれば，全面的に Unicode モードへ移行したほうが合理的であるということです。

Unicode モードへの移行手順

Unicode モードへ移行するには，前持って影響を受ける処理内容およびその影響の性質を把握することがたいせつです。

コマンドの振る舞い

4D エンコーディングと Unicode では，キャラクターセットおよびエンコーディングが異なります。したがって，Char，Character code (Ascii) コマンドより返される値が違います。それで文字コードに基づくロジックは見直さなければなりません。文字コードに基づくロジックの典型例に，半角全角の判別が挙げられます。

4D エンコーディングと Unicode では，文字列の長さの単位が異なります。したがって Length，Position，Get highlight コマンドより返される値が違います。また，Substring，HIGHLIGHT TEXT などに渡すべき値が違います。したがって文字列をバイト単位で扱うロジックは見直さなければなりません。バイト単位で扱うロジックの典型例に，改行位置の計算が挙げられます。

4D エンコーディングと Unicode では，文字列の対照が異なります。つまり，文字列の比較に使用するルールが違います。Unicode では，使用する言語（ロケール）ごとに対照テーブルのデフォルトがあり，さらに数段階の設定ができるようになっているので，このルールを一言で述べることは困難です。影響がおおきい分野として，長音記号の扱い，濁音半濁音の扱いが挙げられます。

最後に，Unicode モードでは，外部との対話に UTF-8 エンコーディングがデフォルトで選択されています。他のエンコーディングを使用したい場合，明示的に USE CHARACTER SET コマンドを使用してください。

正規表現を積極的に活用する

Unicode は，コレーション，サロゲートペア，正規化（NFC，NFD，NFKC，NFKD）など，さまざまなコンセプトが関係している複雑な仕様です。これを Position や Character code など，従来の文字列コマンドだけで処理することには限界があります。4D v11 に正規表現コマンド Match regex が追加されたのは，パターンマッチングができればもっと便利だから，というよりは，これがなければ Unicode を完全に扱うことがほぼ不可能だからです。正規表現の使用は，必須であると考えてください。逆に，正規表現を使用すれば，前述したような複雑な仕様の多くは ICU により適切にハンドリングされるということです。

知っておくと便利なこと

Unicode はコードポイントが基本単位であり，内容はテキストであることが前提です。したがってピクチャのようなバイナリデータを収納したり，バイト配列のようにデータを文脈に関係なく扱ったりすることはできません。その目的には BLOB やピクチャが存在します。

もっとも, Unicode の冒頭 128 コードポイントが ASCII と同じ, また冒頭 256 コードポイントが ISO-8859-1 と同じであることは知っておくと便利です。このことはつまり, エンコーディングに ISO-8859-1 を指定する限り, TEXT//BLOB 変換でデータが失われることはないことを意味します。

また, 4D v11 ではコンポーネントメソッドを作成する場合, テキスト型の引数および返り値はホストとコンポーネントの Unicode モードが一致しなければ, ホストをコンパイルすることができませんが, テキスト型をポインタ経由で渡せば, この制限を回避することができ, したがって Unicode モードのホストから非 Unicode モードのコンポーネントを呼び出すことができます。これは管理された環境で非 Unicode モードを使用したい場合に有効なテクニックです。

リンク

4D のドキュメントに加え, Unicode および ICU の公式ドキュメントをいつでも参照できるようにしておくと便利です。

Unicode
http://unicode.org/

ICU
http://site.icu-project.org/