テキストデータと
バイナリデータの使い方

この機能は、Release10.0 以降で利用可能です。

データベースのテキストデータおよびバイナリデータへのアクセスは、ほとんどコーディングをすることなく、データベースの他のデータ型へのアクセスと同じくらい容易に実現することができます。このドキュメントでは、テキストデータとバイナリデータに対して NX^{J} アプリケーションからどのようにアクセスするかを説明します。

テキストデータへのアクセス

データベースの他のキャラクタデータへのアクセスと同様、データベース（例えば、Oracle の CLOB や LONG フィールド）のテキストデータにアクセスすることができます。テキストデータは、どんな NX^{J} フィールドコントロールに対してもターゲットフィールドとして割り当てることができます。通常、これは複数行を含むテキストをサポートする Text Area コントロールになります。一旦、コントロールにフィールドを関連付ければ、データの取り出しや編集、他のデータベースフィールドと同様の検索を実行することができます。

データベースフィールドに対し、NX^{J} プログラミング言語スクリプトからのアクセスのみを必要とするのであれば、`NullableTextVariable` クラスを使用して、スクリプトに変数を宣言することができます。フィールドコントロールと `NullableTextVariable` は、新しいプロパティである `mimeType` を持っています。このプロパティは、フィールドの内容がどのようにブラウザに表示されるかを管理するために使用されます。（次のセクションを参照してください。）

URL としてテキストフィールドの内容を表示する

テキストフィールドコントロールが、URL（例えば、それが HTML ソースを含んでいる場合）としてアクセスしたいデータを含んでいる場合、Link と Inline Flame コントロールは、まるでそれが Web アプリケーションのドキュメントであるかの

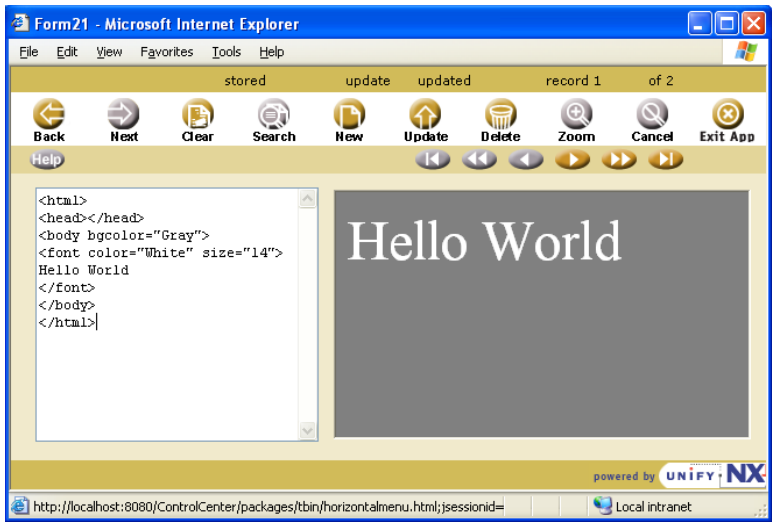
ように、テキストデータを取り出すための新しい構文をサポートします。Link コントロールの Href プロパティと Inline Frame コントロールの Source プロパティは、以下の構文を使用して設定することができます。

```
=&[<data view name>.]<variable name>[;<mime type>]
```

変数がフォーム上のデータビューのメンバであるときのみ、データビュー名は必要とされます。mime type （例：‘text/html’）は、オプションです。

ユーザがリンクをクリックすると、この構文を使用するリンクは新しいウィンドウの URL として、テキストフィールドの内容を表示します。内容が変更されるときは常に、インラインフレームはその URL としてテキストフィールドの内容をロードします。

下図は、テキストフィールドの内容を示している Text Area コントロールと、そのソースとして同じ内容を示しているインラインフレームのサンプルです。



注 – 変数が Null の場合、エラーページが表示されるかもしれません。あるいは、ブラウザが内容をロードしようとしたときに、空のドキュメントをダウンロードするかもしれません。

‘=&’ 構文は、java.lang.String または java.util.StringBuffer 型変数のためにも動作します。

バイナリデータへのアクセス

バイナリデータは、一般的にフィールドコントロールに関連付けられません。代わりに、通常は NullableBinaryVariable クラスを使用してスクリプト内で変数を宣言します。テキスト変数クラスのように、このクラスも mimeType プロパティを持っています。

バイナリデータの使用が実際に効果を発揮するのは、URL リファレンスとして使用されるときです。テキストデータのように、リンクとインラインフレームは、同じ ‘=&’ 構文を使用する URL として、バイナリ変数を示すことができます。バイナリデータは、ブラウザの通常の方法でブラウザによって開かれるか、ダウンロードされるでしょう。例えば、Microsoft Word ドキュメントがバイナリフィールドに格納される場合、ブラウザは Microsoft Word を使用してデータを開きます。

また、フィールドがイメージファイル（例えば、jpeg, png 等）を含む場合、イメージ、イメージボタン、Link コントロールの Source プロパティとして、バイナリ変数を指定するために ‘=&’ 構文を使用することができます。例えば、イメージとしてフォームに表示したい ICON という名前のデータベースフィールドがあり、そこにすべてのイメージが jpeg として格納される場合は、フォームにイメージコントロールが追加され、以下の構文が Source プロパティに設定されます。

```
=&ICON;image/jpeg
```

フォームスクリプトでは、変数は以下のように宣言されます。

```
multi_valued NullableBinaryVariable ICON;
```

注 – 変数が、Null または有効でないイメージの場合、‘image missing’ アイコンが表示されます。

‘=&’ 構文は、byte[] 型変数のためにも動作します。

テキストとバイナリデータのアップロード

File Chooser コントロールは、データベースへの保存やコードでの処理のために、アプリケーションにテキストデータとバイナリデータをアップロードするための機能を実現するために使用することができます。

フォームに File Chooser コントロールを追加するには、最初にツールバーの File Chooser コントロールをクリックします。次にフォーム上の File Chooser コントロールを配置したい場所をクリックします。File Chooser コントロールは、テキストフィールドと “Browse” とタイトル付けられたボタンで表示されます。フォームには必要なだけ File Chooser コントロールを追加することができます。

File Chooser コントロールには、このコントロールに固有の 3 つのプロパティがあります。

- Storage Variable プロパティは、アップロードしたファイルを格納したい NX^{J} 変数を指定するために使用されます。変数がデータビューのメンバである場合、名前に `<data view name>.<variable name>` としてデータビューから完全指定しなければなりません。
- Allowed Mime Types プロパティは、ユーザがアプリケーションへのアップロードを許可するファイルのタイプを制限します。ドロップダウンにリストされた mime type の 1 つを選択するか、またはセミコロンで区切ることで複数の mime type を入力することができます。
- Size Limit プロパティは、ユーザがアプリケーションへのアップロードを許可するファイルの最大のサイズを制限します。受け取りたい最大のファイルサイズをこのプロパティに設定します。この数は、HTTP プロトコルを処理するうえで若干のオーバヘッドを概算するため、指定した数値より若干大きめのファイルをアップロードできるかもしれませんが、多くても 1024 バイトを超えることはできません。

これらのプロパティはそれぞれ、NX^{J} プログラミング言語スクリプトから変更することもできます。詳細については、NX^{J} プログラミング言語 Javadoc を参照してください。

作業方法

File Chooser コントロールがフォーム上にある場合、以下の表 1 に含まれるコマンドのどれかが呼び出されると、File Chooser コントロールのテキストフィールドに入力された文字列によって指定されたファイルは、アプリケーションにアップ

ロードされます。（フォームが Find モードの場合、File Chooser コントロールは利用できないことに注意してください。）ファイルは、その後 Storage Variable プロパティによって指定される変数に保存されます。レコードが更新されるとき、変数の値はデータベースに保存されます。

表 1　　ファイルアップロードを開始するコマンド

Add/Update	Previous Record	Next Form
Delete Record	Previous Set	Zoom
Clear To Find	First Record	Cancel Zoom
Next Record	Last Record	All Developer-Defined Commands
Next Set	Previous Form	

アップロード処理のカスタマイズ

ここまでに記述してきたステップでは、NX^J プログラミング言語コードを書くことなく、アップロードとデータベースへのファイルの保存を実現してきました。アップロードの動作をカスタマイズする必要がある場合、File Chooser コントロールは ON UPLOAD と WHEN UPLOAD COMPLETE の 2 つコードセクションを追加することができます。

ON UPLOAD コードセクションは、Storage Variable プロパティを設定する代わりに使用することができます。このコードセクションが定義される場合、Storage Variable プロパティは無視され、ユーザ自身でファイルの格納を処理しなければなりません。このコードセクションには、4 つの引数が渡されます。最初の引数は、ファイルを読み込む java.io.InputStream オブジェクトです。第二の引数は、アップロードするファイルの mime type を含む String です。第三の引数は、アップロードするファイルのサイズを含む int です。そして第四の引数は、アップロードするファイルの文字コードを含む String です。

実行システムがアップロードを完了した後、WHEN UPLOAD COMPLETE コードセクションは呼び出されます。セクションには、java.lang.String としてアップロードされたファイルの mime type が渡されます。ここでは、アップロードされたファイルを確認するためや、別のフィールド変数に mime type を保存する、といった処理をコードに追加することができます。

(以降の問題の処理の例を参照してください。) ここでエラーを見つけた場合、rejectOperation() を呼び出すことにより、フォーム上の操作の処理を中止することができます。

以下のサンプルコードでは、読み込んだファイルをディスクに保存し、ファイルの保存場所を別のフィールド変数に保存しています。

```
ON UPLOAD (
       java.io.InputStream        iStream,
       String                             contentType,
       int                                contentLength,
       String                             characterEncoding
       )
{
       // getUniqueFileName is declared elsewhere
       File storLoc = new File("c:/tmp/" + getUniqueFileName() );
       BufferedOuputStream oStream = new BufferedOutputStream(
              new FileOutputStream( storLoc ) );
       int nextByte = 0;
       while ( ( nextByte = iStream.read() ) >=0 )
       {
              oStream.write(nextByte);
       }
       oStream.close();

       // Put the file name in another variable
       FILENAME = storLoc.getAbsolutePath();

       // Save the mime type
       MIMETYPE = contentType;
}
```

問題の処理

ユーザが、File Chooser コントロールのテキストフィールドにファイル名を入力ミスした場合、ブラウザは、空のファイル (length - 0) を "application/octet-stream" mime type でアップロードしようとします。これは 2 つの方法でデータベースに挿入されることを防ぐことができます。

1 つ目の方法として、Allowed Mime Type プロパティに "application/octet-stream" を含まないように設定することができます。これは、ファイルのアップロードを拒否して、ファイル名に入力ミスの可能性があることを示すエラーメッセージをユーザに表示します。

2 つ目の方法は、WHEN UPLOAD COMPLETE コードセクションを実装することです。このコードセクションでは、contentType が “application/octet-stream” と等しくかつ、バイナリ変数の length が 0 であるかを確認します。その場合、ユーザにメッセージを表示することができ、フォーム上の操作の処理を中止するために、rejectOperation() を呼び出します。コードは、以下のようになるでしょう。

```
WHEN UPLOAD COMPLETE (
       String contentType
       )
{
       if ( contentType.equals(“application/octet-stream”)
              && myBinVariable.getByteArray().length == 0 )
       {
              session.displayToMessageBox(
                     “Please check the spelling of the file. “
                     + “And try again.” );
              rejectOperation();
       }
}
```