# NewsML解説と NewsML-Toolkit紹介

XMLコンソーシアムWeek
2002年6月13日

**XMLコンソーシアム**
**応用技術部会　NewsML-WG**
**日本電気株式会社　新田 一樹**

# NewsML-Toolkitの紹介

# NewsML-Toolkitとは？(1)

- ロイター通信社が無償で提供しているNewsML用ライブラリ
  - NewsMLデータを読み込む
  - 必要なデータを取得する
  - NewsMLデータを変更する(ver2.0より)
- 期待される効果
  - プログラマーの作業効率化
  - バグ作りこみの回避
- 2種類のAPIライブラリ
  - LLT (Low Level Toolkit)
  - ALT(Application Level Toolkit)

NewsMLを操作するのに特化した、プログラミングツール！

保証の問題、システムの作り方などの要件による

今回はLLTを中心に解説

# NewsML-Toolkitとは？(2)

- NewsML用プログラミングAPI
  - DOM(Document Object Model)をベースとするAPI(Application Program Interface)
  - DOMプログラミングに比べてNewsMLアクセスが容易
  - NewsMLの構造定義に則したデータアクセスを提供
- プラットフォームはJava
  - サポートしている言語はJavaのみ (2002年5月)
- バージョン
  - ver 0.1alpha : 2000年12月1日 Reutersのサイトにて公開
  - ver 1.0 : 2001年5月30日 SourceForgeにて公開
  - ver 1.1beta : 2001年11月25日 SourceForgeにて公開
  - ver 1.1 : 2001年12月12日 SourceForgeにて公開
  - (ALT ver 1.0 beta: 2001年12月20日 SourceForgeにて公開)
  - ver 2.0 : 2002年4月21日 SourceForgeにて公開
  - (ALT ver 1.0 final: 2002年3月14日 SourceForgeにて公開)

# NewsML-Toolkitの機能概要

- NewsML-DOMツリーの生成
  - NewsMLファイルをもとに、NewsMLツリーを簡単に生成します
  - デフォルトはXercesパーサ(http://xml.apache.org/xerces2-j/)
  - 別のパーサを利用することも可能です (独自の拡張が必要)
- データ取得処理のインターフェースと実装
  - NewsMLの要素に対応したノードオブジェクト
  - 外部リソースのDefaultVocabularyの取得機能
  - 要素の文字列表現を取得機能 (ver1.1)
- コンフォーマンスチェック (Ver1.1)
  - DateAndTimeのフォーマットの妥当性
  - FormalNameの妥当性
  - DefaultVocabularyForの評価
  - etc...
- デモ・アプリケーション
  - NewsML-Explorer

# NewsML-Toolkitの配布について

- **提供社(者)**
  - Reuters (http://www.reuters.co.uk/)
    - **通信社。**NewsML**発案社。**
  - WAVO (http://www.wavo.com/)
    - **ソフトウェア会社。**XMLNews**関連の開発など。詳細不明。**
  - David Megginson (http://www.megginson.com/)
    - SAX**を考案・開発**
    - NITF**の初期バージョンをもとに**XMLNews**を開発**
    - W3C**の**"XML Information Set Working Group"**の元議長**
- **入手先**
  - SourceForge
    - http://newsml-toolkit.sourceforge.net/

# NewsML-Toolkitに必要な環境

- for NewsML-Toolkit Ver 1.0
  - JDK1.2以降
  - NewsML-Toolkit ver1.0 (http://newsml-toolkit.sourceforge.net/)
  - DOM Level2 対応のXMLパーサ (Xerces(http://xml.apache.org/xerces-j/))
- for NewsML-Toolkit Ver 1.1 - Ver 2.0
  - JDK1.2以降
  - NewsML-Toolkit ver1.1 (http://newsml-toolkit.sourceforge.net/)
  - DOM Level2 対応のXMLパーサ (Xerces(http://xml.apache.org/xerces-j/))
  - SAXPath (http://saxpath.sourceforge.net/) – event-based XPath parsing
  - Jaxen (http://jaxen.sourceforge.net/) - Java XPath Engine based on SAXPath
  - Gnu Regular Expression library (http://www.cacas.org/~wes/java/)
  - for unittest: JUnit (http://www.junit.org/)
  - for building: Ant (http://jakarta.apache.org/ant/)

# NewsML-Toolkit インストール方法

- **共通**
  - **必要なファイルをダウンロードする**
  - **環境変数CLASSPATHに次に示すJarファイルを定義する**
- Ver 1.0
  - xerces.jar
  - newsml-toolkit.jar
- Ver 1.1 or Ver 2.0
  - xerces.jar
  - newsml-toolkit.jar, newsml-conformance.jar, newsml-normalize.jar (only newsml-toolkit.jar with ver2.0)
  - saxpath.jar
  - jaxen-full.jar ( don’t use Final Version! not work... )
  - junit.jar
  - gnu-regexp-1.1.4.jar
  - ant.jar

# プログラミング概説(1)

## ドキュメントオブジェクトの生成(DOM)

```
File file = new File( “kiji-sample.xml” );
DocumentBuilderFactory factory =
                                    DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.parse( file );
```

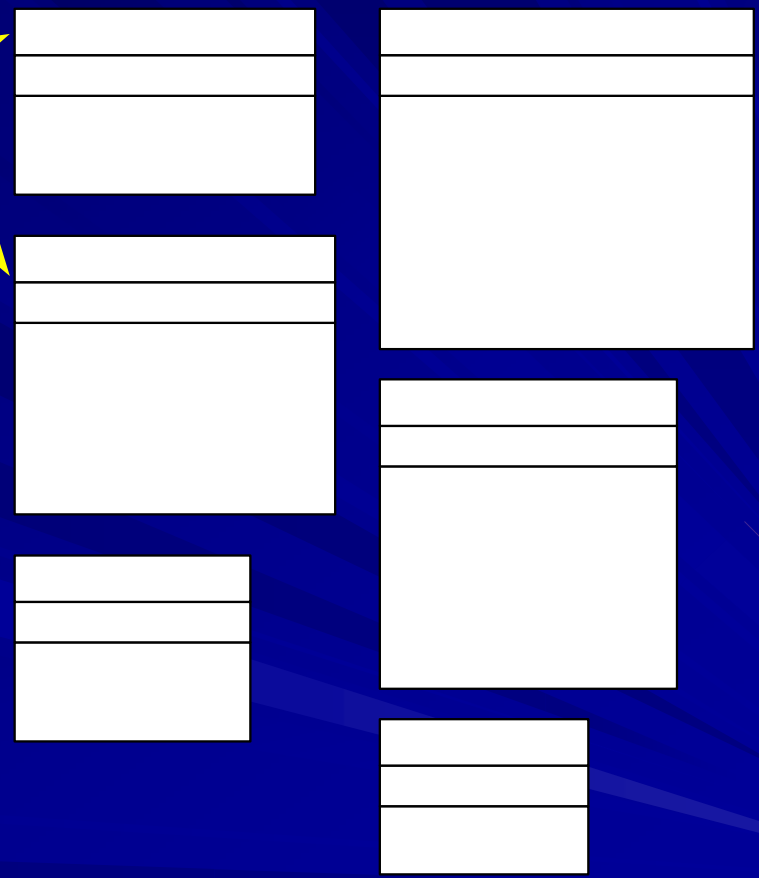
## ドキュメントオブジェクトの生成(NTK)

```
DOMNewsMLFactory factory = new DOMNewsMLFactory();
NewsML newsML = factory.createNewsML( “kiji-sample.xml” );
```

# プログラミング概説(3)

## NewsMLの要素に対応したクラス

```
<NewsComponent Duid="NC0001"
    xml:lang="ja-JP" Essential="no" EquivalentsList="no">
  <Comment>関連写真があります</Comment>
  <NewsLines>
    <HeadLine>鈴木2得点</HeadLine>
    <SubHeadLine>日本準決勝進出</SubHeadLine>
    <ByLine>新聞太郎</ByLine>
    <DateLine>2001年6月2日、新潟県、新潟スタジアム</DateLine>
    <CreditLine>Pressnet</CreditLine>
    <CopyrightLine xml:lang="en">
            NSK all rights reserved.</CopyrightLine>
    <RightsLine>
            2次使用禁止、新聞紙面用メディア使用禁止</RightsLine>
    <SeriesLine>コンフェデ杯特集</SeriesLine>
    <KeywordLine>コンフェデ杯</KeywordLine>
    <KeywordLine>サッカー</KeywordLine>
    <NewsLine>
      <NewsLineType FormalName="Karimidashi" />
      <NewsLineText>○△サンプル</NewsLineText>
    </NewsLine>
  </NewsLines>
  .....
```



このクラス図に表記したメソッド名はホンの一部です

# プログラミング概説(4)

## 要素の選択と要素値・属性値の取得

```
<Metadata>
  <MetadataType FormalName="NskAreaInformation"/>
  <Property FormalName="NskRelevantArea">
    <Property FormalName="NskCountry" Value="JPN"/>
    <Property FormalName="NskJpnAreaCode" Value="15201"/>
  </Property>
  <Property FormalName="NskRelevantArea">
    <Property FormalName="NskCountry" Value="KOR"/>
  </Property>
  <Property FormalName="NskRelevantArea">
    <Property FormalName="NskCountry" Value="CMR"/>
  </Property>
  <Property FormalName="NskOriginatedArea">
    <Property FormalName="NskCountry" Value="JPN"/>
    <Property FormalName="NskJpnAreaCode" Value="15201"/>
    <Property FormalName="NskLocation" Value="新潟スタジアム"/>
  </Property>
</Metadata>
```

新聞協会が定めた地域情報表現のための Metadata

NewsML / NewsItem / NewsComponent / Metadata

# プログラミング概説(5)

## DOMプログラミング

```
// エレメントのタグ名でノードリストを取得する
NodeList loList =
doc.getElementsByTagName("Property");

// 取得したノードリストは、階層を問わず、Property要素すべてのリストを
// 持っている。一覧表示してみる。
for ( int i = 0 ; i < loList.getLength() ; i++ ) {
    Element loElm = (Element)( loList.item(i) );
    System.out.println( "Property[" + i + "]:
FormalName = " +
        loElm.getAttribute("FormalName") + " :
Value = " +
        loElm.getAttribute("Value") );
}
```

```
Property[0]: FormalName = NskRelevantArea : Value =
Property[1]: FormalName = NskCountry : Value = JPN
Property[2]: FormalName = NskJpnAreaCode : Value = 15201
Property[3]: FormalName = NskRelevantArea : Value =
Property[4]: FormalName = NskCountry : Value = KOR
Property[5]: FormalName = NskRelevantArea : Value =
Property[6]: FormalName = NskCountry : Value = CMR
Property[7]: FormalName = NskOriginatedArea : Value =
Property[8]: FormalName = NskCountry : Value = JPN
Property[9]: FormalName = NskJpnAreaCode : Value = 15201
Property[10]: FormalName = NskLocation : Value = 新潟スタジアム
```

## NTKプログラミング

```
NewsMLSession session =
newsML.getSession();

// Property要素をXPath表現で取得します。
// ほしいのはすべてのProperty要素なので、//ではじめます。
BaseNode[] baseNodes =
session.getNodesByXPath( (BaseNode)aoNewsML, "//Property" );

// 取得したProperty要素を一覧表示してみます
Property prop = null;
for ( int i = 0 ; i < baseNodes.length; i++ ) {
    prop = (Property)baseNodes[i];
    System.out.println("Property[" + i + "]:
FormalName = " +
        prop.getName() + " : Value = " +
prop.getValue() );
}
```

# プログラミング概説(2)

## XPath表現の対応(ver1.1以降)

```
BaseNode baseNode =
        session.getNodeByXPath( newsML,
"//Property[@FormalName=¥"NskOriginatedArea¥"]/Property[@FormalName=¥"NskJpnAreaCode¥"]“ );
```

```
<Metadata>
  <MetadataType FormalName="NskAreaInformation"/>
  <Property FormalName="NskRelevantArea">
    <Property FormalName="NskCountry" Value="JPN"/>
    <Property FormalName="NskJpnAreaCode" Value="15201"/>
  </Property>
  <Property FormalName="NskRelevantArea">
    <Property FormalName="NskCountry" Value="KOR"/>
  </Property>
  <Property FormalName="NskRelevantArea">
    <Property FormalName="NskCountry" Value="CMR"/>
  </Property>
  <Property FormalName="NskOriginatedArea">
    <Property FormalName="NskCountry" Value="JPN"/>
    <Property FormalName="NskJpnAreaCode" Value="15201"/>
    <Property FormalName="NskLocation" Value="新潟スタジアム"/>
  </Property>
</Metadata>
```

意味：関連場所

地域コード

意味：発生場所

地域コード

同じ地域コードでも、親Propertyの値で意味が違う！

# コンフォーマンスチェック

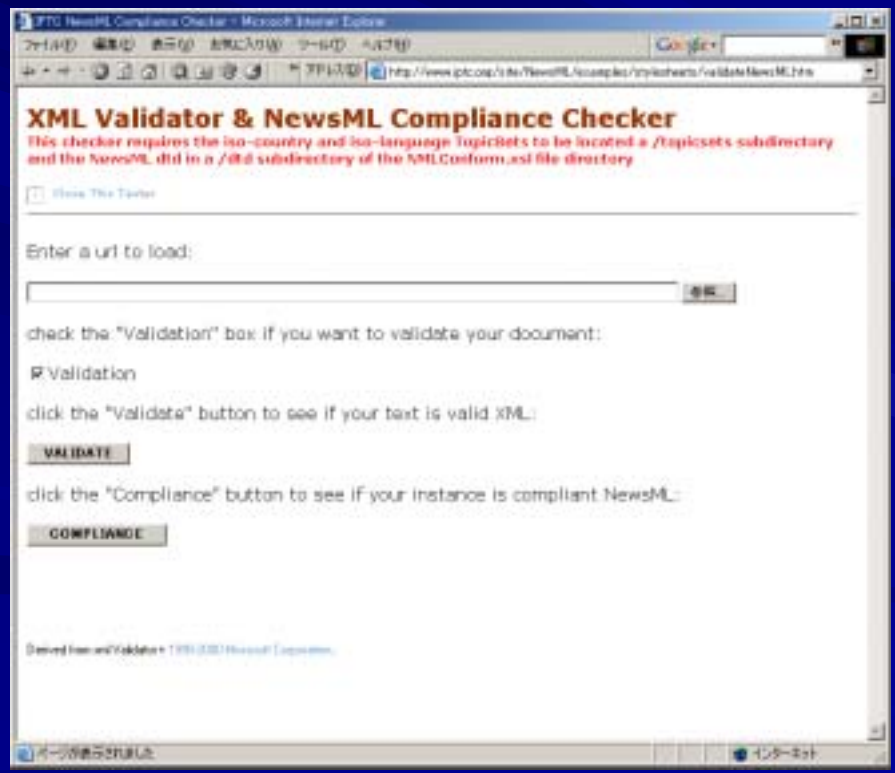
- NewsMLとしての仕様をチェックするクラス
  - NewsMLの機能仕様書(ver1.0)に従ったノードのチェッククラス
  - クラスをインスタンス化してrun(...)メソッドにチェックしたいノードのBaseNodeオブジェクトを渡して実行する。エラー時は例外をスロー。
  - ただしバグもある（DefaultVocabularyForのチェック、DateAndTimeのチェックなど）
- 次のようなコンフォーマンスチェッククラスがある
  - 日付チェック
    - ISO-8601形式に適合しているか？　日付として間違った日付ではないか？
  - NewsIdentifier
    - PublicIdentifierが他のエレメント（NewsItemIdなど）をと整合しているか？
  - NewsComponent
    - Roleがあるか？ / BasisForChoiceにマッチするメンバーがあるか？
- コンフォーマンスチェックを管理するクラス
  - NewsMLTestManagerクラスにより、複数のコンフォーマンスチェックの実行、レポート機能を提供

CatalogTest
ContentItemTest
DateTimeTest
DefaultVocabularyForTest
EuidTest
FormalNameTest
NewsComponentTest
NewsIdentifierTest
NewsItemTest
NewsLineTest
PatternTest
PropertyTest
RefTest
ResourceTest
RevisionIdTest
RevisionStatusTest
TopicSetTest
TopicUseTest
XMLLangTest

# (参考) NewsML-Checker

- XSLTを使ったNewsMLコンプライアンスチェッカー
  - IPTCより提供
  - XMLの妥当性検証だけでなく、NewsMLの仕様に適合しているかどうかをチェックする

http://www.iptc.org/site/NewsML/NewsMLConformance.zip



# NewsML-Toolkitを利用するメリット

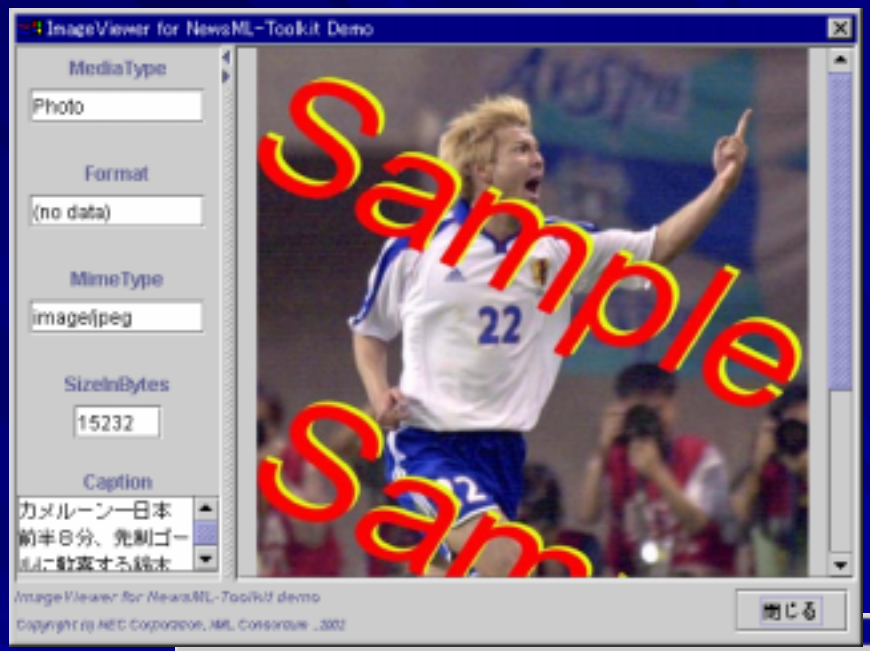
- **オープンソース**
  - フリー (GNU Lesser General Public License)
  - 再配布可能
- **煩雑なDOMを隠蔽するラッパー**
  - DOMプログラミングをより簡単にするためのメソッドを実装
- **NewsMLの思想に基づいたクラス設計**
  - NewsMLの構造と要素・属性の関係をクラス化
  - HeadLineGroup, SubjectCodeItemなど、NewsMLの潜在的なクラスの抽象化

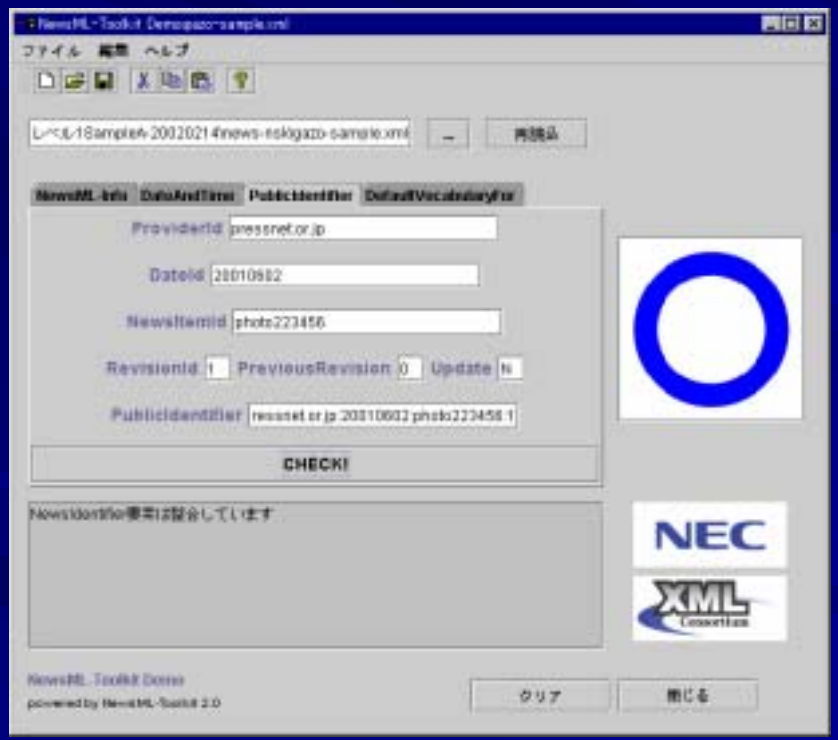
# NewsML-Toolkitの課題

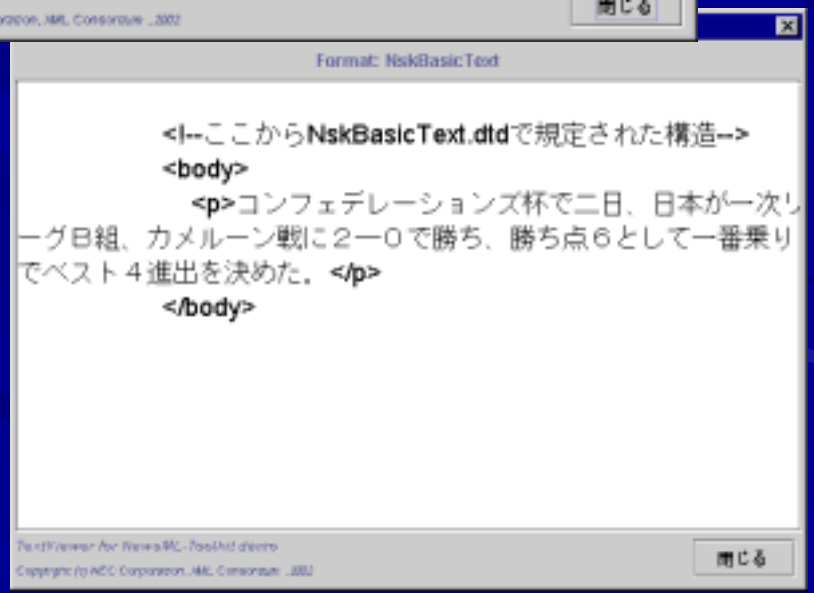
- **処理速度**
  - DOMベースの処理なので処理速度に限界あり
  - 大量のドキュメントのバッチ処理には向かない
- **ノードの更新処理**
  - ver1.1では、更新処理は未実装
  - ver2.0では、ノードレベルでの更新・追加処理実装（コンフォーマンスは維持されるとは限らない）
- **DTDの変更への対応**
  - NewsMLv1.0.dtdに即したクラス設計になっているが、DTDが変更になった場合はクラスの構成が変わるため、変更に対応していく必要がある

# Toolkitデモンストレーション

NewsML-Toolkitを使ったデモンストレーション用プログラムを作成しました。簡単なコンフォーマンスチェックも実装してみました。



# Webサービス向けNewsML解説

# Webサービス実験におけるNewsML-WGの役割

- **新聞社/通信社とのNewsMLデータ借用交渉**
  - XMLコンソーシアムに参加している社と交渉
- **Webサービスで利用できるように**
  - NewsMLの説明
  - WebサービスWGに対して、実装方式の提案
  - 実現するためにNewsMLを一部修正
- **その他**
  - 画像借用のため、画像に透かしを入れる

# デモ用コンテンツと提供元紹介

- デモのためのニュースコンテンツを提供していただいた社（団体）※順不同
  - 読売新聞社様
    - 新幹線NewsMLも提供
  - 共同通信社様
  - 毎日新聞社様
  - Javaコンソーシアム様
    - Javaコンソーシアムで集めたニュース素材をNewsML-WGでNewsML変換を行ったもの

**デモ用データのため正式な各社のサービス用フォーマットとは必ずしも同じではありません**

# Webサービスで利用しやすくするための工夫(1)

## NewsService要素によるニュースの判別

```
<NewsEnvelope>
  <SentFrom>
    <Party Scheme="NskTiffServiceId" FormalName="YOMIURI"/>
  </SentFrom>
  <DateAndTime>20020329T033300+0900</DateAndTime>
  <NewsService Scheme="XmlConsNewsService"
               FormalName="XMLConsYomiuri"/>
  <NewsProduct Scheme="NskNewsProduct"
               FormalName="NskNewsML:1"/>
</NewsEnvelope>
```

| コンテンツ種別 | NewsService名 |
|---|---|
| 読売コンテンツ | XMLConsYomiuri |
| 共同ニュース | NewsPack |
| 毎日コンテンツ | MainichiJapanese |
| 電光ニュース | XMLConsDenko |
| Javaコンソーシアム | XMLConsJavaCon |

# Webサービスで利用しやすくするための工夫(2)

## MediaType要素によるコンテンツ種別の判断

```
<NewsComponent Duid="NC-C-01">
  <Role FormalName="Main"/>
  <ContentItem>
    <MediaType Scheme="IptcMediaTypes" FormalName="Text"/>
    <Format Scheme="NskFormats" FormalName="NskBasicText"/>
    <MimeType Scheme="IptcMimeTypes" FormalName="text/xml"/>
    <DataContent>
      <body>
        <p>  東京大学の卒業式が....
      </body>
    </DataContent>
  </ContentItem>
</NewsComponent>
```

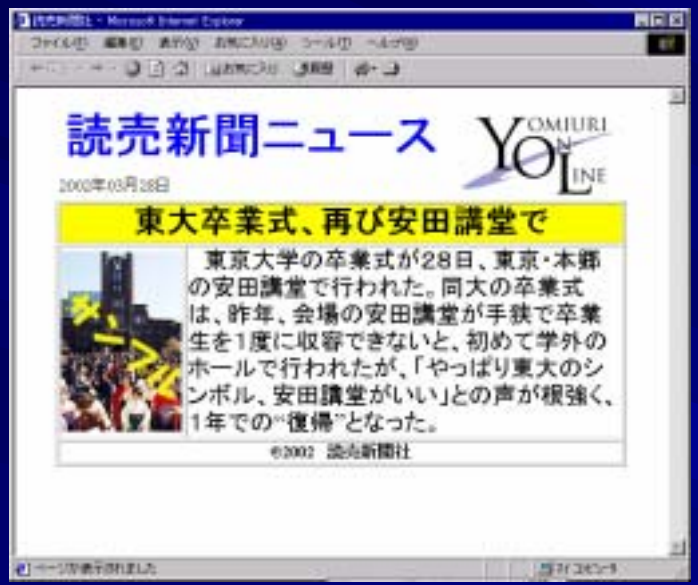
```
<NewsComponent Duid="NC-C-02">
  <Role FormalName="Supplementary"/>
  <ContentItem Href="./20020329h14.jpg">
    <MediaType Scheme="IptcMediaTypes" FormalName="Photo"/>
    <MimeType Scheme="IptcMimeTypes" FormalName="image/jpeg"/>
  </ContentItem>
</NewsComponent>
```

Topicset.iptc-mediatype.xml

- Text
- Graphic
- Photo
- Audio
- Video
- ComplexData

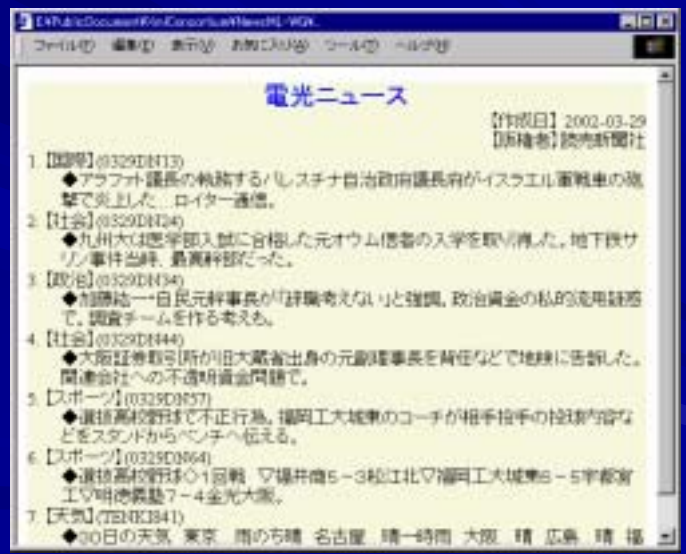
# XSLTスタイルシートの提供

- 提供したコンテンツの表示用スタイルシート




読売コンテンツ表示例

新幹線NewsML表示例



# 検索モデルと対応するNewsML要素(1)

- SubjectCodeでの検索
  - IPTCが定めたコンテンツの内容に関する分類コードで、topicset.iptc-subjectcode.xmlというトピックセットで定義されている
  - SubjectCodeは8桁の数字で表現


- 世界共通コード
  - IPTCのみが定義することができる
  - ユーザによるTopicSetの拡張は許されていない
  - SubjectQualifier要素によりさらに細かい分類情報を付加可能

(例)

```
<DescriptiveMetadata>
  <SubjectCode>
    <!--スポーツ-->
    <Subject FormalName="15000000"/>
    <!-- サッカー -->
    <SubjectMatter FormalName="15054000"/>
    <!-- プロフェッショナル -->
    <SubjectQualifier
                  FormalName="15000010"/>
  </SubjectCode>
</DescriptiveMetadata>
```

# 検索モデルと対応するNewsML要素(2)

## 日付による検索

- NewsMLにはさまざまな日付情報を定義できる

| | |
|---|---|
| NewsML/NewsEnvelope/DateAndTime | NewsMLデータの配信日時。ISO-8601形式 |
| NewsML/NewsItem/Identification/DateId | 日付識別子（ProviderId要素で表すドメインネームが存在した日付けである必要がある）。その版が作られた日付とは限らない。YYYYMMDD形式 |
| NewsML/NewsItem/Identification/DateLabel | 日付を表すラベル。ユーザ指向。形式は問わない |
| NewsML/NewsItem/NewsComponent/DateLine | ニュースの作成日および場所の自然言語記述 |
| NewsML/NewsItem/NewsComponent/RightsMetadata/CopyrightDate | 著作権日付の自然言語記述 |
| NewsML/NewsItem/NewsComponent/RightsMetadata/UsageRights/StartDate<br>NewsML/NewsItem/NewsComponent/RightsMetadata/UsageRights/EndDate | 著作権の期間を表す。自然言語記述 |
| NewsML/NewsItem/NewsManagement/FirstCreated | 最初の版が作成された日時。ISO-8601形式。 |
| NewsML/NewsItem/NewsManagement/ThisRevisionCreated | その版が作成された日時。ISO-8601形式。 |

対象にしやすい

# 検索モデルと対応するNewsML要素(3)

- ニュースの提供元／作成者による検索
  - 作成者名などによる部分一致検索
    - ByLine要素は作成者に関する情報を格納する要素。
    - 人間向けに自然言語記述形式。
  - より厳密な検索
    - Provider要素、Creator要素はParty要素により提供社、作成者を指定
    - Party要素ではFormalName属性でTopicSetで定義されている値を使う必要があるので記述形式によらない厳密な検索が可能

```
<NewsLines>
   <HeadLine>鈴木2得点
</HeadLine>
   <SubHeadLine>日本準決勝進出
</SubHeadLine>
   <ByLine>新聞太郎</ByLine>
   <DateLine>２００１年６月２日、新潟
県、新潟スタジアム</DateLine>
```

```
<AdministrativeMetadata>
   <Provider>
      <Party FormalName="NSK"/>
   </Provider>
   <Creator>
      <Party FormalName="NSK"/>
   </Creator>
</AdministrativeMetadata>
```

topicset.nsk-party.xml

topicset.iptc-provider.xml

# 検索モデルと対応するNewsML要素(4)

## コンテンツ種別検索

```
<ContentItem>
   <MediaType
FormalName="Text"/>
   <Format
Scheme="NskFormats"
FormalName="NskBasicTe
xt"/>
   <MimeType
FormalName="text/xml"/>

.....
```

TopicSet（IPTC作成や日本新聞協会が作成している）で定義されているものの一部

| MediaType |
| --- |
| Text |
| Graphic |
| Photo |
| Audio |
| Video |
| ComplexData |
| NskBasicText |

| Format |
| --- |
| IIM |
| MPEG |
| EPS Adobe Illustrator v9.0 Compatible |
| SVG |
| PDF |
| XHTML |
| GIF89a interlaced |

| MimeType |
| --- |
| text/vnd.IPTC.NITF |
| application/x-binhex |
| application/x-latex |
| audio/mp3 |
| image/bmp |
| text/html |
| text/xml |

# 検索モデルと対応するNewsML要素(5)

- DataContent内全文検索
  - DataContent要素はDTDで*ANY*と宣言されている
    - <!ELEMENT DataContent ANY>
  - プレーンテキスト or XML or CSV etc.....
  - 全文検索対象とする
    - 今後、データの表現方法が標準化されればそのデータ用の検索方法を考える…
      - NITF

# Webサービス向けNewsMLとは？

- **標準的な表現方法で**
  - NewsMLには多種多様な要素が定義されており、その定義も仕様書に規定されている
  - 要素の意味を正しく解釈して正しいNewsMLにするべき
  - 複雑な構造はなるべく避けるべき
- **メタ情報はなるべくつける**
  - 公開したNewsMLデータをなるべくサービスでヒットさせるには、多くの情報を格納しておくべき
  - 権利情報は今後さらにシビアに！
  - データのメンテナンスも重要
- **コンテンツデータは標準的なものを**
  - あらゆる社のNewsMLをかき集めるサービスを考えた場合、そのコンテンツデータそのもののフォーマットも標準的なものを使うべき
    - PDF、XHTML、NITF、…

# NewsMLの今後の課題

- セキュリティコントロール
  - 公開情報の信頼性確保
- コンテンツデータの標準化
  - NITFがIPTC推奨になっているが日本ではこれから新聞協会で検討する方向
- 外字、ルビ
  - 漢字圏、特に日本の新聞では重要な課題

# ありがとうございました

次はWebサービスの
プレゼンテーションです

付録

# NewsML-Toolkitプログラミング

## DOMプログラミングとの比較

# 目次

- **ドキュメント・オブジェクトの生成**
  - DOMプログラミングおよびNewsMLプログラミングで必須となるドキュメント・オブジェクト（ツリー・オブジェクト）の生成方法について説明します。
- **要素の選択と属性値の取得**
  - 要素について、その選択方法と要素値・属性値の取り出し方を説明します。
- **Duidによる要素の取得**
  - Duidを使った要素の取得について説明します。
- **複雑な問い合わせ**
  - 複雑な入れ子構造になったPropertyエレメントを使って、データの取得方法を説明します。
- **ノードの文字列表現**
  - DOMノードの文字列化方法およびToolkit1.1での文字列化機能について紹介します。
- **コンフォーマンス**
  - NewsML-Toolkit(ver1.1)のコンフォーマンス・チェックについて紹介します。

紹介するコードはすべて抜粋です。実際のコードは別紙をご覧ください。

# ドキュメント・オブジェクトの生成(DOM)

```
// クラスのインポート
import java.io.File;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;

// ファイルオブジェクトを作成する
File sourceFile = new File("some-newsml-file.xml");

// ファクトリの生成（バリデーションあり）
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
factory.setValidating( true );

// ドキュメント・ビルダーの生成
DocumentBuilder builder = factory.newDocumentBuilder();

// ファイルのパース
Document doc = builder.parse(sourceFile);
```

DOMツリーの管理オブジェクト取得

# ドキュメント・オブジェクトの生成(Toolkit)

```
// クラスのインポート
import org.newsml.toolkit.dom.XercesDOMFactory;
import org.newsml.toolkit.dom.DOMNewsMLFactory;
import org.newsml.toolkit.NewsML;

// Xercesを使ったDOMファクトリの生成（バリデーションあり）
XercesDOMFactory factory = new XercesDOMFactory();
factory.setValidation( true );

// NewsMLファクトリの生成
DOMNewsMLFactory newsmlFactory = new DOMNewsMLFactory( factory );

// NewsMLツリーの生成
NewsML newsml =
        newsmlFactory.createNewsML( "some-newsml-file.xml" );
```

# 目次

- **ドキュメント･オブジェクトの生成**
  - DOMプログラミングおよびNewsMLプログラミングで必須となるドキュメント･オブジェクト（ツリー･オブジェクト）の生成方法について説明します。
- **要素の選択と属性値の取得**
  - 要素について、その選択方法と要素値･属性値の取り出し方を説明します。
- **Duidによる要素の取得**
  - Duidを使った要素の取得について説明します。
- **複雑な問い合わせ**
  - 複雑な入れ子構造になったPropertyエレメントを使って、データの取得方法を説明します。
- **ノードの文字列表現**
  - DOMノードの文字列化方法およびToolkit1.1での文字列化機能について紹介します。
- **コンフォーマンス**
  - NewsML-Toolkit(ver1.1)のコンフォーマンス･チェックについて紹介します。

紹介するコードはすべて抜粋です。実際のコードは別紙をご覧ください。

# 要素と属性値の取得(DOM)

```
// エレメントのタグ名でノードリストを取得する
NodeList loList = aoDoc.getElementsByTagName("Property");

// 取得したノードリストは、階層を問わず、Property要素すべてのリストを
// 持っている。一覧表示してみる。
for ( int i = 0 ; i < loList.getLength() ; i++ )
{
    Element loElm = (Element)(loList.item(i));
    System.out.println("Property[" + i +
        "]: FormalName = " + loElm.getAttribute("FormalName") +
            " : Value = " + loElm.getAttribute("Value")
        );
}
```

```
<Property FormalName="NskCountry"
Value="JPN" />
```

```
Property[0]: FormalName = NskRelevantArea : Value =
Property[1]: FormalName = NskCountry : Value = JPN
Property[2]: FormalName = NskJpnAreaCode : Value = 15201
Property[3]: FormalName = NskRelevantArea : Value =
Property[4]: FormalName = NskCountry : Value = KOR
Property[5]: FormalName = NskRelevantArea : Value =
Property[6]: FormalName = NskCountry : Value = CMR
Property[7]: FormalName = NskOriginatedArea : Value =
Property[8]: FormalName = NskCountry : Value = JPN
Property[9]: FormalName = NskJpnAreaCode : Value = 15201a
Property[10]: FormalName = NskLocation : Value = 新潟スタジアム
```

# 要素と属性値の取得(Toolkit1.1)

```
// NewsMLセッションオブジェクトを取得します。
NewsMLSession session = aoNewsML.getSession();

// Property要素をXPath表現で取得します。
// ほしいのはすべてのProperty要素なので、//ではじめます。
BaseNode[] baseNodes =
        session.getNodesByXPath( (BaseNode)aoNewsML,
            "//Property" );

// 取得したProperty要素を一覧表示してみます
Property prop = null;
for ( int i = 0 ; i < baseNodes.length; i++ )
{
        prop = (Property)baseNodes[i];
        System.out.println("Property[" + i +
            "]: FormalName = " + prop.getName() +
                " : Value = " + prop.getValue()
            );
}
```

```
Property[0]: FormalName = NskRelevantArea : Value = null
Property[1]: FormalName = NskRelevantArea : Value = null
Property[2]: FormalName = NskRelevantArea : Value = null
Property[3]: FormalName = NskOriginatedArea : Value = null
Property[4]: FormalName = NskCountry : Value = JPN
Property[5]: FormalName = NskJpnAreaCode : Value = 15201
Property[6]: FormalName = NskCountry : Value = KOR
Property[7]: FormalName = NskCountry : Value = CMR
Property[8]: FormalName = NskCountry : Value = JPN
Property[9]: FormalName = NskJpnAreaCode : Value = 15201a
Property[10]: FormalName = NskLocation : Value = 新潟スタジアム
```

# 要素と属性値の取得(Toolkit1.0)

```
// Toolkit1.0の場合の処理例。
// エラー処理は無視しています

// NewsItemオブジェクトを取得します
NewsItem newsItem = aoNewsML.getNewsItem(0);
// NewsComponentオブジェクトを取得
NewsComponent newsComponent = newsItem.getRootNewsComponent();
// Metadataオブジェクトを取得
Metadata[] metas = newsComponent.getMetadata();
for ( int i = 0 ; i < metas.length; i++ )
{
    // Propertyオブジェクトを取得
    Property[] props = metas[i].getProperty();
    for ( int j = 0 ; j < props.length; j++ )
    {
        // Property要素のうちFormalNameオブジェクトを取得し（getName()）
        // FormalNameオブジェクトの値を取得します（2個目のgetName()）
        String formalName = props[j].getName().getName();
        String value      = props[j].getValue();
        System.out.println("Property[" + i +
        "]: FormalName = " + formalName +
            " : Value = " + value
        );
    }
}
```

```
<NewsItem>
   ....
   <NewsComponent>
      ....
      <Metadata>
         <MetadataType
FormalName....>
         <Property>
            <Property>
            <Property>
         </Property>
      </Metadata>
      ....
```

# 目次

- **ドキュメント・オブジェクトの生成**
  - DOMプログラミングおよびNewsMLプログラミングで必須となるドキュメント・オブジェクト（ツリー・オブジェクト）の生成方法について説明します。
- **要素の選択と属性値の取得**
  - 要素について、その選択方法と要素値・属性値の取り出し方を説明します。
- **Duidによる要素の取得**
  - Duidを使った要素の取得について説明します。
- **複雑な問い合わせ**
  - 複雑な入れ子構造になったPropertyエレメントを使って、データの取得方法を説明します。
- **ノードの文字列表現**
  - DOMノードの文字列化方法およびToolkit1.1での文字列化機能について紹介します。
- **コンフォーマンス**
  - NewsML-Toolkit(ver1.1)のコンフォーマンス・チェックについて紹介します。

紹介するコードはすべて抜粋です。実際のコードは別紙をご覧ください。

# Duidによる要素の取得(DOM)

```
// Duid属性が"NC0001"である要素のタグ名を取得
String duid = "NC0001";
Element loElm = aoDoc.getElementById( duid );
System.out.println( "ID=¥""+ duid +"¥"" + "のタグ名：" +
      loElm.getTagName() );
```

```
<NewsComponent Duid="NC0001"/>
```

```
ID="NC0001"のタグ名：NewsComponent
```

# Duidによる要素の取得(Toolkit1.1)

```
// Duid属性が"NC0001"である要素のタグ名を取得
String duid = "NC0001";

// NewsMLセッションオブジェクトの取得
NewsMLSession session = aoNewsML.getSession();
BaseNode baseNode = session.getNodeByDuid( duid );

System.out.println( "ID=¥"" + duid + "¥"" + "のタグ名：" +
    baseNode.getXMLName() );
```

ID="NC0001"のタグ名：NewsComponent

# 目次

- **ドキュメント･オブジェクトの生成**
  - DOMプログラミングおよびNewsMLプログラミングで必須となるドキュメント･オブジェクト（ツリー･オブジェクト）の生成方法について説明します。
- **要素の選択と属性値の取得**
  - 要素について、その選択方法と要素値･属性値の取り出し方を説明します。
- **Duidによる要素の取得**
  - Duidを使った要素の取得について説明します。
- **複雑な問い合わせ**
  - 複雑な入れ子構造になったPropertyエレメントを使って、データの取得方法を説明します。
- **ノードの文字列表現**
  - DOMノードの文字列化方法およびToolkit1.1での文字列化機能について紹介します。
- **コンフォーマンス**
  - NewsML-Toolkit(ver1.1)のコンフォーマンス･チェックについて紹介します。

紹介するコードはすべて抜粋です。実際のコードは別紙をご覧ください。

# 複雑な問い合わせ(DOM)

```
/**
 * 特定のPropertyエレメントを取得する方法を実装してみます。
 * たとえば次のPropertyエレメントを取得します。
 * ・FormalName属性が"NskOriginatedArea"のProperty要素の子供
 * ・その子供Propertyのうち、FormalNameが"NskJpnAreaCode"のValue属性
 * XPath表現で書くと
 * Property[@FormalName="NskOriginatedArea"]/
 *   Property[@FormalName="NskJpnAreaCode"]/@Value
 * です。
 */
  // エレメントのタグ名でノードリストを取得する
  NodeList loList = aoDoc.getElementsByTagName("Property");

  // 今回、FormalName="NskOriginatedArea"のProperty要素の、
  // FormalName="NskJpnAreaCode"のPropertyサブ要素を取得するので、まず最初に
  // NskJpnAreaCodeのProperty要素を取得します。そのあと、その親Property要素の
  // FormalName属性がNskOriginatedAreaかどうかを判定します。
  for ( int i = 0; i < loList.getLength(); i++ )
  {
      Element loElm = (Element)loList.item(i);
      String lsForm = loElm.getAttribute("FormalName");
      if ( lsForm.equals("NskJpnAreaCode") )
      {
          Element loParent = (Element)loElm.getParentNode();
          if ( loParent.getAttribute("FormalName").equals("NskOriginatedArea") )
          {
              System.out.println("Ans = " + loElm.getAttribute("Value") );
          }
      }
  }
```

```
<Metadata>
  <MetadataType FormalName="NskAreaInformation"/>
  <Property FormalName="NskRelevantArea">
    <Property FormalName="NskCountry" Value="JPN"/>
    <Property FormalName="NskJpnAreaCode" Value="15201"/>
  </Property>
  <Property FormalName="NskRelevantArea">
    <Property FormalName="NskCountry" Value="KOR"/>
  </Property>
  <Property FormalName="NskRelevantArea">
    <Property FormalName="NskCountry" Value="CMR"/>
  </Property>
  <Property FormalName="NskOriginatedArea">
    <Property FormalName="NskCountry" Value="JPN"/>
    <Property FormalName="NskJpnAreaCode" Value="15201a"/>
    <Property FormalName="NskLocation" Value="新潟スタジアム"/>
  </Property>
</Metadata>
```

```
Ans = 15201a
```

# 複雑な問い合わせ(Toolkit)

```
/**
 * 特定のPropertyエレメントを取得する方法を実装してみます。
 * たとえば次のPropertyエレメントを取得します。
 * ・FormalName属性が"NskOriginatedArea"のProperty要素の子供
 * ・その子供Propertyのうち、FormalNameが"NskJpnAreaCode"のValue属性
 * XPath表現で書くと
 * Property[@FormalName="NskOriginatedArea"]/Property[@FormalName="NskJpnAreaCode"]/@Value
 * です。
 */
NewsMLSession session = aoNewsML.getSession();

BaseNode baseNode =
    session.getNodeByXPath(aoNewsML,
        "//Property[@FormalName=¥"NskOriginatedArea¥"]/Property[@FormalName=¥"NskJpnAreaCode¥"]");

System.out.println("Ans = " + ((Property)baseNode).getValue() );
```

```
<Metadata>
  <MetadataType FormalName="NskAreaInformation"/>
  <Property FormalName="NskRelevantArea">
    <Property FormalName="NskCountry" Value="JPN"/>
    <Property FormalName="NskJpnAreaCode" Value="15201"/>
  </Property>
  <Property FormalName="NskRelevantArea">
    <Property FormalName="NskCountry" Value="KOR"/>
  </Property>
  <Property FormalName="NskRelevantArea">
    <Property FormalName="NskCountry" Value="CMR"/>
  </Property>
  <Property FormalName="NskOriginatedArea">
    <Property FormalName="NskCountry" Value="JPN"/>
    <Property FormalName="NskJpnAreaCode" Value="15201a"/>
    <Property FormalName="NskLocation" Value="新潟スタジアム"/>
  </Property>
</Metadata>
```

Ans = 15201a

# 目次

- **ドキュメント･オブジェクトの生成**
  - DOMプログラミングおよびNewsMLプログラミングで必須となるドキュメント･オブジェクト（ツリー･オブジェクト）の生成方法について説明します。
- **要素の選択と属性値の取得**
  - 要素について、その選択方法と要素値･属性値の取り出し方を説明します。
- **Duidによる要素の取得**
  - Duidを使った要素の取得について説明します。
- **複雑な問い合わせ**
  - 複雑な入れ子構造になったPropertyエレメントを使って、データの取得方法を説明します。
- **ノードの文字列表現**
  - DOMノードの文字列化方法およびToolkit1.1での文字列化機能について紹介します。
- **コンフォーマンス**
  - NewsML-Toolkit(ver1.1)のコンフォーマンス･チェックについて紹介します。

紹介するコードはすべて抜粋です。実際のコードは別紙をご覧ください。

# ノードの文字列表現(DOM)

```
// エレメントのタグ名でノードリストを取得する
NodeList loList = aoDoc.getElementsByTagName("DataContent");

// kiji-sample0926.xmlには、DataContent要素はひとつしかないので、
// Elementオブジェクトをitem(0)からキャストする。
Element loElm = (Element)loList.item(0);

// DataContent要素内のデータをXMLとして取得する。
StringBuffer lzBuff = new StringBuffer();
makeString( loElm, lzBuff );    // 文字列化処理
System.out.println( new String(lzBuff) );
```

※ makeString( Node, StringBuffer )は独自メソッド（別紙参照）

```
<!--ここからNskBasicText.dtdで規定された構造-->
<body>
    <p>コンフェデレーションズ杯で二日、日本が一次リーグB組、カメルーン戦に2－0で勝ち、
勝ち点6として一番乗りでベスト4進出を決めた。</p>
</body>
```

# ノードの文字列表現(Toolkit)

```
// NewsMLセッションの取得
NewsMLSession session = aoNewsML.getSession();

// XPathによるBaseNodeの取得
BaseNode baseNode = session.getNodeByXPath(aoNewsML,
    "NewsItem/NewsComponent/ContentItem/DataContent");

// DataContentオブジェクトの取得
DataContent content = (DataContent)baseNode;

// プレーンテキストだけを取り出す
System.out.println("=====Output Plain Text=====");
System.out.println( content.getText() );

// XML表現で取り出す（ただし日本語が化ける）
System.out.println("=====Output XML Text=====");
String data = content.getXMLString();
System.out.println( data );
//System.out.println( new String( data.getBytes( "ISO8859_1" ), "MS932" ) );
```

```
=====Output Plain Text=====
コンフェデレーションズ杯で二日、日本が一次リーグＢ組、カメルーン戦に２―０で勝ち、勝ち
点６として一番乗りでベスト４進出を決めた。
=====Output XML Text=====
<body>
<p>&#12467;&#12531;&#12501;&#12455;&#12487;&#12524;&#12540;&#12471;&#12519;&#12531;&#1
2474;&#26479;&#12391;&#20108;&#26085;&#12289;&#26085;&#26412;&#12364;&#19968;&#27425;&#12522;&#12540;&#12464;&
#65314;&#32068;&#12289;&#12459;&#12513;&#12523;&#12540;&#12531;&#25126;&#12395;&#65298;&#8213;&#65296;&#12391;
&#21213;&#12385;&#12289;&#21213;&#12385;&#28857;&#65302;&#12392;&#12375;&#12390;&#19968;&#30058;&#20055;&#1242
6;&#12391;&#12505;&#12473;&#12488;&#65300;&#36914;&#20986;&#12434;&#27770;&#12417;&#12383;&#12290;</p>
</body>
```

# 目次

- **ドキュメント･オブジェクトの生成**
  - DOMプログラミングおよびNewsMLプログラミングで必須となるドキュメント･オブジェクト（ツリー･オブジェクト）の生成方法について説明します。
- **要素の選択と属性値の取得**
  - 要素について、その選択方法と要素値･属性値の取り出し方を説明します。
- **Duidによる要素の取得**
  - Duidを使った要素の取得について説明します。
- **複雑な問い合わせ**
  - 複雑な入れ子構造になったPropertyエレメントを使って、データの取得方法を説明します。
- **ノードの文字列表現**
  - DOMノードの文字列化方法およびToolkit1.1での文字列化機能について紹介します。
- **コンフォーマンス**
  - NewsML-Toolkit(ver1.1)のコンフォーマンス･チェックについて紹介します。

紹介するコードはすべて抜粋です。実際のコードは別紙をご覧ください。

# コンフォーマンス(only Toolkit1.1)

## 例) NewsEnvelope/DateAndTime要素のチェック

DateAndTime = "20010630T2300+0900"

```
try {
    NewsMLSession session = aoNewsML.getSession();
    BaseNode baseNode = session.getNodeByXPath( aoNewsML, "NewsEnvelope" );

    // ISO-8601フォーマットをチェックするクラスのインスタンス化
    DateTimeTest test = new DateTimeTest();
    IdText dateandtime = ((NewsEnvelope)baseNode).getDateAndTime();
    // NewsEnvelope/DateAndTimeの値のチェック
    System.out.println("DateAndTime = ¥"" + dateandtime.toString()  + "¥"");
    // DateAndTimeノードがあっているかどうかをチェックする。
    // 間違っている場合、NewsMLException例外をスローする
    test.run( (BaseNode)dateandtime, false );
}
catch ( NewsMLException exp )
{
    System.err.println("NewsMLException occurred!:" + exp.getMessage());
}
```

```
DateAndTime = "20010630T2300+0900"
NewsMLException occurred!:Date does not follow NewsML ISO 8601 subset: 20010630T2300+0900
```

# パーサについて

- **デフォルトはXerces**
  - Apache-Jakartaプロジェクトの Xercesを想定した XercesDOMFactoryクラスが提供されている
- **それ以外のパーサを使いたいとき**
  - DOMFactoryインターフェースをインプリメントしてユーザ独自のファクトリクラスを作成する

```
import org.newsml.toolkit.dom.DOMFactory;

public class JaxpDOMFactory implements DOMfactory {
   // コンストラクタ
   public JaxpDOMFactory() {....   }

   // createDOM(Reader, String)インターフェースの実装
   public org.w3c.dom.Document createDOM(
      Reader reader, String baseUrl )  { ....  }
   ...
}
```