記事
Toshihiko Minamoto · 2021年3月25日 6m read

C-CDA 2.1 からインポートトランスフォーメーションを実行する際の前処理サポート

HealthShare HealthConnect と Information Exchange のバージョン 15.03 では、C-CDA 2.1 から SDA へのインポートトランスフォーメーションをサポートしています。
こういったプログラムのビルドは、インストレーションパッケージの `csp/xslt/SDA3` ディレクトリにあります。 インポートトランスフォーメーションの概要は、*Overview of Health Connect* の「CDA Documents and XSL Transforms in HealthShare」をご覧ください。

C-CDA 2.1 サポートに関連するインポート機能の強化の 1 つとして、 **C-CDA ファイルをインポートする前に前処理を実行できる機能が追加されました**。

前処理サポートの導入により、トランスフォーメーションは大幅に簡素化され、その合計処理時間も削減されます。 考えられるユースケースは以下の 2 つです。

- 要素がない、またはその形式が正しくない
- 日付スタンプの変換

(1) ノード値の中の特定の句読点がインポートの実行前に取り除かれる、(2) 特定の状況において、すべての `effectiveTime` 要素にタイムスタンプが取り込まれた `high` 子ノードを持たせる、という 2 点を確実に行う方法として以下のサンプルと手順をご紹介します。 `high`ノードは、インポートされると SDA `ToTime` ノードに変換されます。

プリプロセッサー変換、変更されたトップレベルの変換、および入力ファイルなど全てのサンプルは、この記事に添付してあります。

事前設定

1. お勤め先の組織のドメイン名の管理者の協力を得て、コード拡張に使用するサブドメイン名を取得します。 この手順で使用される例では`extensions.your-healthcare.org`です。
2. 前処理のカスタムモードに使用するプレフィックスと名前を選択します。
   以下のインストラクションの例の中では、プレフィックスに `mo`、モードに `MyMODEName` を使っています。

前処理を行うトランスフォームを作成する

以下の手順で、前処理を行う XSL ファイルを作成します。

1. スタイルシートの要素は以下の形式にします。

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
xmlns:hl7="urn:hl7-org:v3" xmlns:custom_mode_prefix=subdomain
exclude-result-prefixes="hl7 custom_mode_prefix">
```

   事前設定のセクションでお見せした例を使うと、スタイルシートの要素は以下のようになります。

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
xmlns:hl7="urn:hl7-org:v3" xmlns:mo="http://extensions.your-
healthcare.org" exclude-result-prefixes="hl7 mo">
```

2. トップレベルの要素を複製し、すべての属性、要素、テキストノードをコピーするテンプレートを含めます。 どのテンプレートでも、カスタムモードを使用する必要があります。

```
<xsl:template match="/hl7:ClinicalDocument" mode="mo:MyMODEname">
  <hl7:ClinicalDocument>
      <xsl:apply-templates select="@*" mode="mo:MyMODEname"/>
      <xsl:apply-templates select="node()" mode="mo:MyMODEname"/>
  </hl7:ClinicalDocument>
</xsl:template>
```

```
<xsl:template match="node()" mode="mo:MyMODEname" priority="1">
   <xsl:copy>
      <xsl:copy-of select="@*"/>
         <xsl:apply-templates select="node()" mode="mo:MyMODEname"/>
   ?</xsl:copy>
</xsl:template><template match="node()" mode="mo:MyMODEname" priority="1">
</template>
```

3. 更新ごとにテンプレートを作成します。
これらのテンプレートでもカスタムモードを使用する必要があります。 ここでは、テンプレートを 2 つ作成します。ノー
ド値から特定の句読点を取り除くテンプ
レートと、特定の要素にlowとhigh
のタイムスタンプノードがあることをチェックするテンプレートです。
最初のテンプレートは、ノード値から疑問符と感嘆符を削除します。

```
<xsl:template match="text()" mode="mo:MyMODEname" priority="2">
   <xsl:value-of select="translate(., '?!', '')"/>
</xsl:template>
```

2 つ目のテンプレートは、entry.encounter.effectiveTime
要素を検索し、あるはずのhighノードを探します。
もしlow
ノードが存在し、設定もされているが、それに対
応するhigh
ノードが存在ないという場合、プリプロセッサーはlow
ノードのタイムスタンプをコピーし、それと同じタイムスタンプ値を持つhighノードを挿入します。
これは優先度の一番高いテンプレートです。

```
  <xsl:template match="/hl7:ClinicalDocument/hl7:component/hl7:structuredBody/h
l7:component/
       hl7:section[hl7:templateId/@root = '2.16.840.1.113883.10.20.22.2.22']/
       hl7:entry/hl7:encounter/hl7:effectiveTime" mode="mo:MyMODEname" priority
="3">

    <xsl:copy>
      <xsl:copy-of select="@*"/>
```

```
      <xsl:apply-templates select="node()" mode="mo:MyMODEname"/>


      <xsl:if test="string-length(hl7:low/@value) > 0 and not(string-
length(hl7:high/@value) > 0)">
        <xsl:comment>hl7:timestamp was augmented in pre-
processing phase</xsl:comment>
        <xsl:element name="high" namespace="urn:hl7-org:v3">
          <xsl:attribute name="value">
            <xsl:value-of select="hl7:low/@value"/>
          </xsl:attribute>
        </xsl:element>
      </xsl:if>

    </xsl:copy>
  </xsl:template>
```

4. 前処理用の XSL ファイルを ../CDA-Support-Files/Import/ にコピーします。

カスタムトップレベルのトランスフォームを作成する

CCDAv21-to-SDA.xsl か CCDAv21-nonXML-to-SDA.xsl を複製し、以下のように変更します。

1. スタイルシートの要素にはカスタムネームスペースを含めます。

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:mo="http://extensions.your-healthcare.org" xmlns:isc="http://extension-fu
nctions.intersystems.com" xmlns:hl7="urn:hl7-org:v3" xmlns:sdtc="urn:hl7-org:sd
tc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:exsl="http://ex
slt.org/common" exclude-result-prefixes="isc hl7 sdtc xsi exsl">
```

2. 前処理に使用する XSL ファイルを指定するカスタムインポートファイルを参照する include 式を追加します。

```
<xsl:include href="CDA-Support-Files/Import/CCDAPreProcessor.xsl"/>
```

3. 次の行

```
<xsl:variable name="input" select="/hl7:ClinicalDocument"/>
```

を、次のブロックに置き換えます

```
<xsl:variable name="inputRTF">
  <xsl:apply-
templates select="/hl7:ClinicalDocument" mode="custom_mode_prefix:mode" />
</xsl:variable>
<xsl:variable name="input" select="exsl:node-
set($inputRTF)/hl7:ClinicalDocument"/>
```

上の例を使うと、呼び出し元のコードブロックは以下のようになります。

```
<xsl:variable name="inputRTF">
  <xsl:apply-templates select="/hl7:ClinicalDocument" mode="mo:MyMODEname"/>
</xsl:variable>
<xsl:variable name="input" select="exsl:node-
set($inputRTF)/hl7:ClinicalDocument"/>
```

Ensemble のセットアップと C-CDA ファイルのインポート

1. プロダクション環境で、SDA をインポートするビジネスサービスを見つけます。
   このサービスは、Information Exchange では `HS.Gateway.ECR.SDA3XML.FileService`
   と呼ばれており、Edge のプロダクション環境の一部でもあります。
2. そのサービスの **InputXSL**
   という追加設定の値をカスタムトップレベルのトランスフォームのファイル名に設定します。
   添付のコード例のトップレベルのトランザクションには `CCDAv21-to-SDA-PreProc.xsl`
   という名前が付いているため、**InputXSL** の設定値は `CCDAv21-to-SDA-PreProc.xsl` となります。
3. C-CDA ファイルを普段通りにインポートします。

c-cdav21-input-preprocessingsamplecode.zip

#XML #HealthShare

ソースURL:https://jp.community.intersystems.com/post/c-cda-21-%E3%81%8B%E3%82%89%E3%82%A4%E3%83%B3%E3%83%9D%E3%83%BC%E3%83%88%E3%83%88%E3%83%A9%E3%83%B3%E3%82%B9%E3%83%95%E3%82%A9%E3%83%BC%E3%83%A1%E3%83%BC%E3%82%B7%E3%83%A7%E3%83%B3%E3%82%92%E5%AE%9F%E8%A1%8C%E3%81%99%E3%82%8B%E9%9A%9B%E3%81%AE%E5%89%8D%E5%87%A6%E7%90%86%E3%82%B5%E3%83%9D%E3%83%BC%E3%83%88