

船舶海洋情報学

05. XMLによるデータの表現



XML (Extensible Markup Language)

情報を保管、ラベル付け、構造化、または保護するための「コンテナ」のようなもの
異なるシステムが相互にコミュニケーションするための手段・基盤

- 1) データはXML文書(テキスト)で表現される
- 2) **タグ**によって情報は「**要素**(element)」に分割され、構造化・意味付けされる(**マークアップ**)
- 3) **要素の役割・順序・包含関係・位置・参照関係を決められたルールに従って表現**
- 4) XML文書で表されたデータが、決められたルールに則った書式かどうか判別
- 5) XMLの書式のルールから、そのデータの読み書きを行うプログラムを作成

なぜXMLが必要か？

- ・ データは貴重な「資産」
- ・ データの価値は、保管の仕方や、必要な情報へのアクセス性によって左右される

【旧式のデータ表現】 例: CSV形式

53628, トクホコーラ, 165

データの配列順に意味付けがなされている
データ受け取り側のプログラムには別途
配列順と意味付けの情報が必要

膨大なデータがあったとしても、配列順と意味付けの
情報が失われたら利用できない

タグで囲ま
れた部分が
1つの要素

XML形式

```
<商品>  
  <コード>53628 </コード>  
  <品名>トクホコーラ </品名>  
  <価格> 165 </価格>  
</商品>
```

要素として表現すると、要素の
順番や数、包含関係などを詳細に表現可能

要素(**element**)の**タグ**が自由に設定でき、タグに意味付けが
されるので**データ内容の把握が容易**。

この例は要素の**属性(attribute)**として表現することも可能

```
<商品 name="トクホコーラ" code="53628" price="165">  
</商品>
```

要素「商品」が属性name, code, price を持つ場合、
それぞれの値を指示すればよい。
ただし、**属性では順番や数、包含関係などは表現できない**

前回の復習: HTML (HyperText Markup Language)

- ・ Webページを記述するための言語(データ形式)
- ・ 「タグ」を用いて表示する要素やブラウザに対する命令を意味付け
- ・ テキストファイルとして作成し、拡張子を「.html」とする

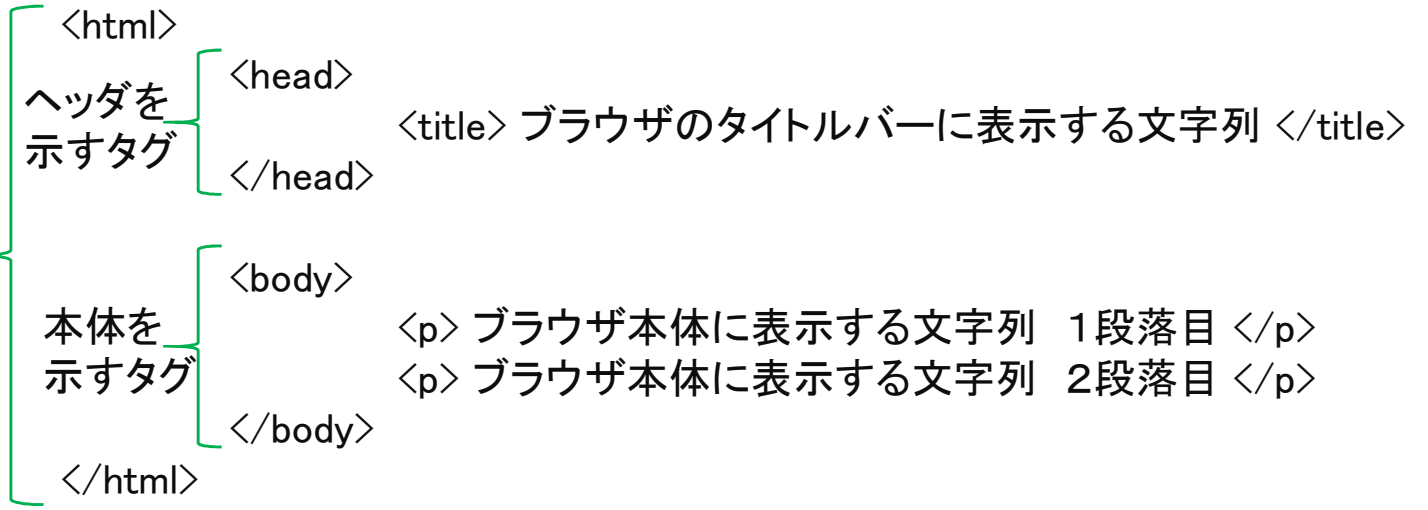
XMLより古く、設計思想が異なるなどの理由で文法が不完全

HTMLファイルの構成

文書型宣言: HTMLのバージョン等を示す省略可

```
<!DOCTYPE HTML PUBLIC “./W3C//DTD HTML 4.01 Transitional//EN”  
HTTP://www.w3.org/TR/html4/loose.dtd>
```

HTML文書の始まりと終わりを示すタグ

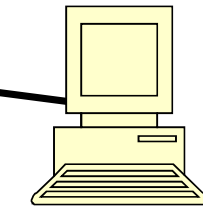
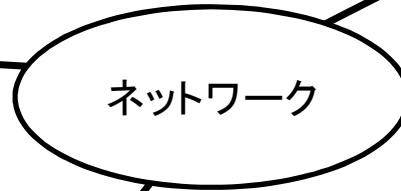


XMLはシステム間のデータ交換に用いられる

IEEE1888 (Ubiquitous Green Community Control Network)
商業施設やオフィスなどの電力・施設管理を目的としたIoT通信規格
2015年3月 ISO/IECの国際標準としても承認 (ISO/IEC/IEEE 18880)

ISO16425 (船内LAN装備指針)では、
機器同士でやりとりする文字列をXML形式にすることを推奨

ネットワークI/O機器



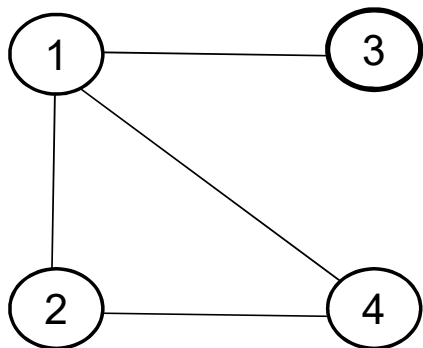
ストレージ

船級協会共通構造規則 (CSR BC & OT) に対応した
規則計算ソフトウェア PrimeShip-HULL へ入力する船殻データ形式
= XML形式 各社異なるCADから変換して出力

「海事産業における製品情報の高度利用の
ための情報共有基盤「SPEEDS」」→3DデータをXMLで

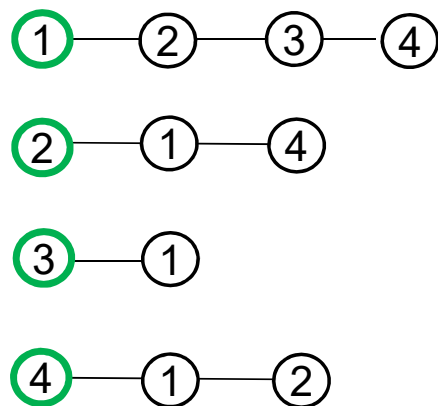
複雑なデータをXMLで表してみる

グラフ構造



	①	②	③	④
①	0	1	1	1
②	1	0	0	1
③	1	0	0	0
④	1	1	0	0

行列によるノードの接続表現



リストによるノードの接続表現

- 1) データ構造は、変更や追加などが容易な表現とする
- 2) データを処理するアルゴリズムとデータ構造は表裏一体であるので、アルゴリズムに適した構造とすべき

```
<graph>
  <node pid="1">
    <link destination="2"/>
    <link destination="3"/>
    <link destination="4"/>
  </node>
  <node pid="2">
    <link destination="1"/>
    <link destination="4"/>
  </node>
  <node pid="3">
    <link destination="1"/>
  </node>
  <node pid="4">
    <link destination="1"/>
    <link destination="2"/>
  </node>
</graph>
```

XMLによるノードの接続表現の例
他にもっと良い表現方法があるかも

DTDによるXML文書の型定義・品質管理(1)

DTD (Document Type Definition: 文書型定義)

記述できる要素のリストと内容、それぞれの要素に記述できる要素を定義

グラフの例) graph.dtd

```
<!ELEMENT graph (node*)>
<!ELEMENT node (link*)>
<!ATTLIST node pid CDATA #REQUIRED >
<!ELEMENT link EMPTY>
<!ATTLIST link destination CDATA #REQUIRED>
```

Graph要素はnode要素を0以上の複数個含む

node要素は属性としてpidを有し、文字列で表す

link要素には他の要素を何も含まない

link要素は属性としてdestinationを有し、文字列で表す

「文字列」を意味する

グラフの例におけるgraph.dtdに従ったXML文書

```
<?xml version="1.0"?>
<!DOCTYPE doc SYSTEM "graph.dtd">

<graph>
  <node pid="1">
    <link destination="2"/>
    <link destination="3"/>
    <link destination="4"/>
  </node>
  <node pid="2">
    <link destination="1"/>
    <link destination="4"/>
  </node>
  <node pid="3">
    <link destination="1"/>
  </node>
  <node pid="4">
    <link destination="1"/>
    <link destination="2"/>
  </node>
</graph>
```

DTDによるXML文書定義は、

- ・DTDが読みにくい、
- ・DTDでは要素の構造や依存関係を表すがデータの型についての定義がない

などの理由で次に説明する「XMLスキーマ」による定義が主流

スキーマによるXML文書の型定義・品質管理(2)

W3C XML Schema による文書型定義

記述できる要素のリストと内容、**データ型**、それぞれの要素に記述できる要素を定義

グラフの例) graph.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="graph">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="node" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="link" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:attribute name="destination" type="xs:int" use="required"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="pid" type="xs:int" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

SchemaそのものもXMLで記述される

node要素は0以上の複数個存在

グラフの例におけるgraph.xsdに従ったXML文書

```
<graph>
  <node pid="1">
    <link destination="2"/>
    <link destination="3"/>
    <link destination="4"/>
  </node>
  <node pid="2">
    <link destination="1"/>
    <link destination="4"/>
  </node>
  <node pid="3">
    <link destination="1"/>
  </node>
  <node pid="4">
    <link destination="1"/>
    <link destination="2"/>
  </node>
</graph>
```

Schemaの文法の詳細は専門書などを参照

XMLスキーマからJavaのクラスを生成できる

W3C XML記述できる要素のリストと内容、**データ型**、それぞれの要素に記述できる要素を定義しているので、そのままjavaのクラスを生成可能

JAXB(Java Architecture for XML Binding)アノテーションと呼ばれる特殊な形式で表現される JAXB とは、XML と Java オブジェクトを相互変換するための API 仕様のこと。Java SE6 からは標準ライブラリに組み込まれているので、特に jar を追加することなく使える。

例) JDK (java Development Kit) をインストールして、コマンドプロンプトにて
> xjc speeds_3.xsd

を実行するとXMLスキーマファイル speeds_3.xsd から Javaのクラスファイルを生成

XMLファイルを読み書きするためのライブラリも用意されている

XMLスキーマからPythonのクラスを生成できる

PyXB (“pixbee”) は XMLSchema で定義されたデータ構造に対応するクラスのための Python ソースコードを生成する純粋な Python パッケージ

XMLの例: SVG (Scalable Vector Graphics)

図形をXMLで表現

ビットマップのような表現ではなく、**座標や数式によって表現**



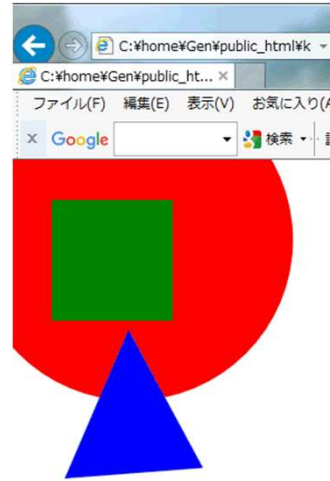
ビットマップ



SVG

<https://ferret-plus.com/7089> より画像を引用

XMLの要素が形を、
各要素の属性として色や座標を表す



```
<svg xmlns="http://www.w3.org/2000/svg" >  
  <desc> Three shapes </desc>  
  <circle fill="red" cx="3cm" cy="2cm" r="4cm" />  
  <rect fill="green" x="1cm" y="1cm" width="3cm" height="3cm" />  
  <polygon fill="blue" points="110,160,50,300,180,290"/>  
</svg>
```



```
<svg xmlns="http://www.w3.org/2000/svg"  
  style="margin-left: calc(50% - 0.5em);"  
  viewBox="0 0 1024 1024"  
  width="1024" height="1024">  
  <g transform="scale(0.03125)">  
    <path d="M 128 128 h 768 v 768 h -768 Z" />  
  </g>  
</svg>
```

XMLの例: MathML (Mathematical Markup Language)

複雑な数式の意味をXMLで表現

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
<math>
  <mrow> <mi>x</mi> <mo>=</mo>
  <mfrac>
    <mrow>
      <mrow> <mo>-</mo> <mi>b</mi>
      </mrow> <mo>&PlusMinus;</mo>
      <msqrt>
        <mrow>
          <msup> <mi>b</mi> <mn>2</mn> </msup> <mo>-</mo>
          <mrow>
            <mn>4</mn> <mo>&InvisibleTimes;</mo> <mi>a</mi> <mo>&InvisibleTimes;</mo> <mi>c</mi>
          </mrow>
        </mrow>
      </msqrt>
    </mrow>
    <mrow> <mn>2</mn> <mo>&InvisibleTimes;</mo> <mi>a</mi>
  </mfrac>
</mrow>
</math>
```

スタイルシート:XMLデータの表示を指示

XML + XSL (Extensible Stylesheet Language)

XMLに格納されているデータを抽出し、決められた書式で表示する

【身近な応用例】 XMLでデータを記述しておき、HTMLファイルをブラウザで表示する場合にスキーマを利用して整形表示

XMLデータ
Building.xml

XSLTスタイルシート
Building1-2.xsl

```
<?xml version="1.0" encoding="Shift_JIS"?>
<?xml-stylesheet type="text/xsl" href="building1-2.xsl" ?>
<buildings title="東京情報大の建物一覧">

<building id="1">
  <name>本館棟</name>
  <floor>7</floor>
  <comment>講義室、演習室、事務局、ゼミ室</comment>
</building>

<building id="2">
  <name>体育館棟</name>
  <floor>2</floor>
  <comment>アリーナ、トレーニングルーム</comment>
</building>

<building id="3">
  <name>フロンティア共同研究センター</name>
  <floor>3</floor>
  <comment>「国際共同研究の拠点」</comment>
</building>

</buildings>
```

```
<?xml version="1.0" encoding="Shift_JIS"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />
<xsl:template match="/" >
  <html>
  <head>
  <title>
    <xsl:value-of select="buildings/@title" />
  </title>
  </head>
  <h1><xsl:value-of select="buildings/@title" /></h1>
  <table border="1">
    <tr> <th>No.</th> <th>名前</th> <th>階数</th> <th>特徴</th> </tr>
    <xsl:apply-templates select="buildings" />
  </table>
  </html>
</xsl:template>
<xsl:template match="buildings">
  <xsl:for-each select="building">
    <tr>
      <td><xsl:value-of select="@id" /></td>
      <td><xsl:value-of select="name" /></td>
      <td><xsl:value-of select="floor" /></td>
      <td><xsl:value-of select="comment" /></td>
    </tr>
  </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

XMLの要素を抽出して表示

XMLデータ Building.xml をブラウザへ表示した結果

東京情報大の建物一覧

No.	名前	階数	特徴
1	本館棟	7	講義室、演習室、事務局、ゼミ室
2	体育館棟	2	アリーナ、トレーニングルーム
3	フロンティア共同研究センター	3	国際共同研究の拠点

XMLの暗号化 (XML Encryption)

- ・ W3Cに提案されているXMLの暗号化技術で、ツリー構造XML文章のうち、ある要素以下を全て暗号化する仕組み。
ルートタグを暗号化すればXML文章全体の暗号化も可能。またデータ保護項目の優先度付けを行い、特定の要素のみ暗号化することも可能

W3Cの「XML Encryption Syntax and Processing」に紹介されているXMLエレメント暗号の例

```
<?xml version='1.0'?>
```

```
<PaymentInfo xmlns='http://example.org/paymentv2'>
```

```
<Name>John Smith</Name>
```

```
<CreditCard Limit='5,000' Currency='USD'>
```

```
<Number>4019 2445 0277 5567</Number>
```

```
<Issuer>Example Bank</Issuer>
```

```
<Expiration>04/02</Expiration>
```

```
</CreditCard>
```

```
</PaymentInfo>
```

```
<?xml version='1.0'?>
```

```
<PaymentInfo xmlns='http://example.org/paymentv2'>
```

```
<Name>John Smith</Name>
```

```
<EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'  
xmlns='http://www.w3.org/2001/04/xmlenc#'>
```

```
<CipherData>
```

```
<CipherValue>A23B45C56</CipherValue>
```

```
</CipherData>
```

```
</EncryptedData>
```

```
</PaymentInfo>
```

公開鍵暗号を用いて
クレジットカード情報を暗号化

まとめ

- (1) **XML**とは: 情報を保管、ラベル付け、構造化、または保護するための「コンテナ」**タグ**を用いて情報を意味付け・構造化(**マークアップ**)
- (2) XMLの構成: **要素**(element)と**属性**(attribute)
「要素」は要素間の包含関係・依存関係などを表現可能
「属性」は要素に付随する値のみ表現(包含関係や依存関係などは表現できない)
- (3) XML文書の定義:
DTD: 要素間の関係を定義／データの型は文字列か数字の区別のみ
XML scheme: データ型も詳細に定義／スキーマからJavaやPythonのクラス生成可能
- (4) XML + **XSL** スタイルシート: XMLからデータを抽出して**整形表示**
- (5) XML文書の暗号化: **XML encryption**

レポート課題:

前回の課題の自分のホームページをイメージしたロゴをSVGで作成し、ブラウザで表示して確認せよ。少なくとも3種類以上の図形を組み合わせて構成すること。XMLの理解が目的なのでSVG作成ツールの使用を避け、テキストファイルを編集してがんばること。(XMLコードの参考程度に使うならよい)

【提出方法】上記のsvgファイルを下記の課題提出用フォルダへ、課題の番号と提出者が分かるようにファイル名を以下のようにしてアップロードせよ

第5回2TE19xxxZ名前.svg

<https://share.iii.kyushu-u.ac.jp/public/hROwAAqIPi5ATI4BUXJtIJeJbMLJzVszfitNI89GHcMK>

上記フォルダへのリンクは下記ホームページから

<http://sysplan.nams.kyushu-u.ac.jp/gen/edu/NavallInformationProcessing/2019/index.html>