人文学のためのテキストデータ構造化のチュートリアル

第5章

構造化テキストの作成
――基礎演習 1――

永崎研宣

version 1.0
2026.3.21 作成

本資料は、文部科学省委託事業「人文学・社会科学のＤＸ化に向けた研究開発推進事業」（JPMXP1624）において、学校法人慶應義塾が、大学共同利用機関法人人間文化研究機構から再委託を受けて作成したものです。

1．この演習を始める前に―ソフトウェアとデータの準備

1-1．Oxygen XML Editor のダウンロード

この演習では、Oxygen XML Editor（以下、Oxygen）という商用ソフトウェアの使用を前提としている。Oxygen は基本的には有料だが、豊富なデータ作成機能を備えており、TEI や XML を始める人が最初に扱うものとしては適している。Oxygen は 30 日間の試用ライセンスキーが配布されているので、まずはそれで試してみていただきたい。ソフトウェアのダウンロード及び試用ライセンスキーの入手は以下の URL から可能である。

```
https://www.oxygenxml.com/xml_editor/download_oxygenxml_editor.html
```

1-2．演習用データのダウンロード

この演習で利用するデータは、以下の URL に掲載されている。ここには、筆者が作成したもの以外にも、国立国会図書館及び東北大学附属図書館のデジタルコレクションで公開されているものの再配布を含んでいるので、所蔵・公開機関への感謝とともに、ダウンロードして利用していただきたい。

```
https://www.dhii.jp/dh/tei/textbook/
```

1-3．TEI 古典籍ビューワへのアクセス

この演習では、作成したデータの見え方を確認するために TEI 古典籍ビューワを使用する。そのため、以下の URL にアクセスしてビューワへのファイル読み込みをできるようにしておくこと。なお、このビューワはローカルコンピュータ上で動作させることも可能であり、インターネット接続ができない環境の場合には事前にダウンロードしておくこともできる。

```
https://tei.dhii.jp/teiviewer4eaj
```

2．XML に関する最低限の基本的な知識 ― 演習 1-1

ここでの演習には XML についての最低限の知識が必要になる。具体的な作業の煩雑さについては、Oxygen が非常に丁寧にサポートしてくれるので、それほど煩わされることはないが、なぜ Oxygen がそのように指示をするのか、そもそもなぜマークアップはそのように行わなければならないのか、といったことについての前提知識がないとマークアップの作業もデータ構造の設計も困難である。そこで、ここで知っておくべき XML のごく基本的な事柄について述べておく。

①タグを追記して意味を付与する

ここで言うタグとは、HTML や XML などの様々なマークアップ言語で用いられる、不等号で囲まれた文字列を指しており、その文字列にはタグが付与すべき内容が含まれる。たとえば以下の例では、「テキストデータベース構築の基礎知識」が書名であることを、二つのタグ「< 書名 > と </ 書名 >」で囲むことによって示している。

```
< 書名 > テキストデータベース構築の基礎知識 </ 書名 >
```

XML は、拡張可能マークアップ言語（Extensible Markup Language）という名のとおり、利用者が自由にタグを設定できることに大きな特徴がある。

② XML における開始タグ・終了タグ・空白タグ

XML では 3 種類のタグを用いる。「< 書名 >」のようなタグは開始タグと呼ばれ、単独では機能せず、対となる終了タグ「</ 書名 >」が必要となる。終了タグではタグの名前の前にスラッシュが置かれる。この二つに対して、タグで囲む対象を持たず単独で機能するタグが空白タグである。「< 改行 />」などのように記述し、どこにでも用いることができる。

タグの種類	用例	用法
開始タグ	< 書名 >	終了タグと併用しテキストデータを注釈する
終了タグ	</ 書名 >	開始タグと併用し（以下同文）
空白タグ	< 書名 />	テキストデータを注釈せず、挿入した位置に単独で意味を持たせる

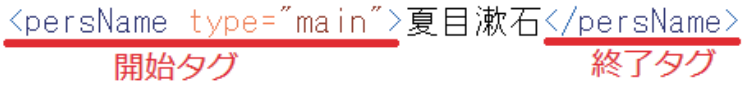
③タグは入れ子構造にできるが、オーバーラップはできない

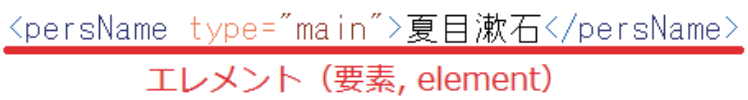
拡張可能で自由な XML における数少ない制約の一つとして、タグは入れ子構造でなければならず、オーバーラップできない、というルールがある。以下に例を示す。

事例 1	< 頁 > < 文章 > テキスト DB…作成する。</ 文章 > < 文章 > 基礎知識…必要である。</ 文章 > </ 頁 >	頁の中に二つの文章が入っている。
事例 2	< 頁 > < 文章 > テキスト DB…作成する。</ 文章 > < 文章 > 基礎知識…必要 </ 頁 > である。</ 文章 >	文章の途中で頁区切りがあったため < 文章 > と < 頁 > がオーバーラップしている。
事例 3	< 頁 /> < 文章 > テキスト DB…作成する。</ 文章 > < 文章 > 基礎知識…必要 < 頁 /> である。</ 文章 >	頁区切りを空白タグで示すことでオーバーラップを回避している。

上の表の事例 2 のように、XML では、始まったタグが終わる前に別のタグが終わってしまうことをオーバーラップと呼んで禁止している。これは、コンピュータの処理能力やデータの整合性確認の実現可能性といったことから定められているものである。ただし、それでも、対象となる資料の性質上、どうしてもオーバーラップをしなければならない場合がある。たとえば、本において一つの文章が始まった後に、その文章が終わる前に頁の区切りが来てしまうのはよくある例である。この場合、上記の事例 3 のように空白タグを利用することでこの制約を回避する。

④エレメント（要素）とは





XML におけるエレメント（要素）とは、タグ及びそれによって囲まれた対象としてのテキストを指す。たとえば、「< 地名 > 南鳥島 </ 地名 > 」と書かれている場合、地名のエレメントは、地名タグ及びタグに囲まれている「南鳥島」の全体を意味することになる。このエレメントを操作することでデータを抽象化して処理できるのが XML の長所の一つである。XML では、このエレメントをツリー構造（木構造）とすることによってデータの操作や処理を効率化している。

⑤アトリビュート（属性）とは

XML において、アトリビュート（属性）は、エレメントの意味を拡張したり制限したり、あるいは他のエレメントと対応づけるなど、様々な役割を担うものである。このアトリビュートも基本的に誰もが自由に作成可能である。ただし、id、もしくは `xml:id` というアトリビュートは特別な意味を持っており、さらに、使える文字に関して一定の制約があるので注意されたい。

⑥ XML におけるサブセットの必要性とスキーマ

XML では誰でも自由にタグや属性を決めることができるため、柔軟性の高さが利便性の高さにつながっているのだが、一方で、各利用者が好き勝手にタグをどんどん設定してしまうと相互運用性が下がってしまい、全体としては利便性が下がってしまう場合がある。そのため、目的を共有するコミュニティのなかでどのようなタグを用いるかのルールを決めることがよくみられる。このルールを XML のサブセットと呼ぶことがある。この観点から言うなら、TEI ガイドラインは人文学全体で共有可能な XML タグの利用方法を定めるルールであり、人文学向けの XML のサブセットであるということもできる。

XML 自体は国際標準規格として広く流通しており、TEI ガイドラインに限らず人文学に関連する他の規格でも用いられている。さらに、人文学に限らなければ、たとえば Microsoft の Word、Excel、PowerPoint なども近年では文書フォーマットとして XML を用いている 。

また、前章で述べたように、このルールの内容をスキーマと呼び、機械可読形式でファイルに記述することができる。このスキーマのファイル形式としては、DTD（拡張子は .dtd）や XML Schema（拡張子は .xsd）、RELAX NG（拡張子は .rng もしくは .rnc）等がある。これを人が手で直接書くことは、近年はあまりみられない。むしろ、人が書いた情報を元に自動的にスキーマファイルを生成するシステムが散見される。TEI でも、スキーマをカスタマイズするための Roma という Web アプリが使えるようになっており、Web ブラウザ上の GUI でスキーマに定義されたタグを追加・削除するなどしてカスタマイズした後、自動的に上記のファイル形式のいずれかでスキーマファイルとして書き出すようになっている。

このスキーマファイルは、XML ファイルを処理する際に、整合性をチェックするために用いられる。その応用的な例として、XML Editor にこれを読み込ませると、XML Editor で XML ファイルを参照している際に、リアルタイムでタグの形式が間違っていないかをチェックし、かつ、任意の入力箇所でどのタグがルール上使用可能かを自動的にチェックしてリストできる。このようなことから、何らかのサブセットに沿ってタグ付けを行いたい場合は必須のものとなる。ただし、この後に使用する Oxygen XML Editor には、最初から TEI のスキーマが組み込まれており、

TEI の編集を開始するとすぐに TEI のスキーマに基づいたサービスが提供される。それについてはこの演習で体験することになる。

⑦整形式の（Well-formed）XML 文書と妥当な（valid）XML 文書

XML 文書は、コンピュータで処理しやすいように一定の書式に沿っていなければならない。それには 2 段階ある。一つの段階は、Well-formed（形式が整った、整形式の）と呼ばれる段階で、これは、タグ等の内容や関係は問わず、ただ、階層構造に沿ってコンピュータが読みこめるようにタグや属性が正しい書式で書かれているかどうか、だけを確認するものである。もう一つの段階は、Valid（妥当な）と呼ばれるものであり、これは Well-formed であることを前提に、さらに、スキーマで定義されたタグや属性の使い方に沿っているかどうかを検証することである。たとえば、TEI に準拠しているかどうかを確認する際には Valid の段階での検証を行うことになる。この区別については、この後の演習のなかで具体的に扱う。

3．基本的なマークアップで XML 文書を作成する― 演習 1-2

学習の成果

この演習では、XML 文書作成に関する以下のような操作の習得を目指す。

- 新しい XML ファイルを作成する
- XML エディタにテキストファイルを挿入する
- 様々な方法で文書の基本的な構造的特徴をマークアップする
- XML 文書にエレメント（要素）とアトリビュート（属性）を付加する
- 形式の整った（well-formed）XML 文書を作成する
- 形式の整った（well-formed）XML 文書を整形したりインデントしたりする

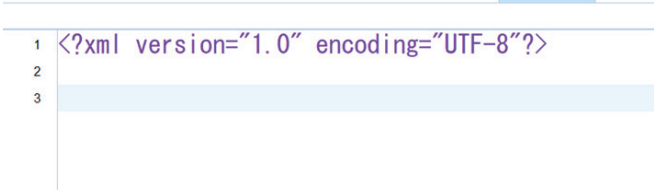
要点

この演習は、夏目漱石の書簡を教材とする Oxygen XML Editor での XML 文書の作成の実習であり、それを通じて文書をマークアップするための様々な手法を体験する。最初に、「新規作成」で開始し、まだマークアップされていないテキストをエディタに挿入して、その後、その文書の構造的なセクションをマークアップすることになる。そして、マークアップした文書の形式が整っているかどうかをチェックする方法と、それを整形してインデントする方法を学ぶ。

① XML ファイルの新規作成を始める

以下の手順で、XML ファイルの新規作成を始めよう：

1. まだ立ち上げていなければ、Oxygen XML Editor（以下、Oxygen）を立ち上げよう。（OSによって、メニューを使ったり、デスクトップのアイコンをダブルクリックしたりすることになる。）
2. もし Oxygen がライセンスキーを聞いてきたら、ライセンスキーのデータの内容をすべてコピーして、ライセンスキー記入欄に貼り付けよう。
3. Oxygen は豆知識の書かれたウェルカム画面を表示するが、その画面は閉じよう。
4. Oxygen でメニューバーの「ファイル」から「新規作成」を選択すると「新規」というダイアログが開き、そこで「新しい文書」の中から「XML 文書」を選択した後に「作成」ボタンをクリックすると、XML 宣言が付された空白の文書が開くはずである。
5. XML 宣言は以下のようになる：




```
<?xml version="1.0" encoding="UTF-8"?>
```

この、エレメントの中に書かれた XML 宣言は、Oxygen を含め、この XML ファイルを処理しようとするすべてのソフトウェアに対して、これが XML ファイルであると示すものである。そして、処理プログラムがどの種類の文字エンコーディングを期待してよいのかということを `@encoding` を用いて伝えている。UTF-8 (Universal Character Set Transformation Format - 8 bit) は、Unicode のエンコーディング方式の一つある。XML 宣言は、開始・終了に山括弧とクエスチョンマークを用いた特別な処理命令の形式をとっているため、終了タグを必要としない。

② `<text>` エレメントを付与する

`<text>` エレメントを用いて、今回のファイルにいくつかの構造を作ってみよう。

1. XML 宣言の次の行に「`<text>`」と入力してみよう。
2. 最後の「>」を入力した時の Oxygen の動作に注目してみよう。Oxygen は入力作業を補助しようとして `</text>` の終了タグを自動的に挿入する。これは、開始タグの `<text>` を入力した場合には、いずれ必ず終了タグ `</text>` を入力しなければならないために、Oxygen が終了タグを自動的に入力してしまうことで、あとで入力する手間を省いてくれているのである。
3. これがどういうタイプのテキストかということはまだ明示していない。ここでは手紙のデー

タを扱うので、@type 属性を追加して、これを「手紙 letter」と分類してみよう。カーソルを少し戻して、開始タグの中の文字列の最後の文字の後ろに持って行こう。そして、スペースキーを押して、「type=”」と入力しよう。ここで、この二重引用符を入力した時に何が起きるかに注目してみよう。Oxygen は再び、閉じる二重引用符を入力することで入力作業を補助してくれる。というのは、アトリビュートの値は常に二つの二重引用符で囲まれるものだからである。

4. 今回のテキストを手紙として分類するために、二つの二重引用符の間に「letter」という文字列を入力しよう。

```
<?xml version="1.0" encoding="UTF-8"?>
<text type="letter"></text>
```

5. 次に、<text> エレメント［つまり、<text> と </text> の間に］<body> エレメントを追加してから、その内側に空白行を追加しよう。スペースと改行は、XML ファイルでは通常、ないものとみなされる（すなわち、空白・改行を表現するなら明示的にタグ等で記述する必要がある）ので、タグ以外のところには、みやすいように改行や空白を入れて整形しても問題ない。
6. 以下のようになっているか、確認されたい。

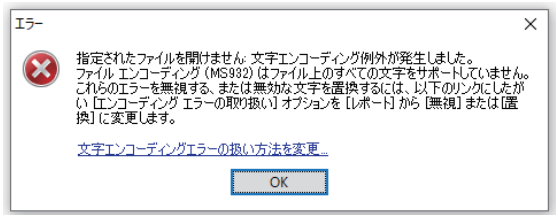
```
<?xml version="1.0" encoding="UTF-8"?>
<text type="letter"><body>

</body></text>
```

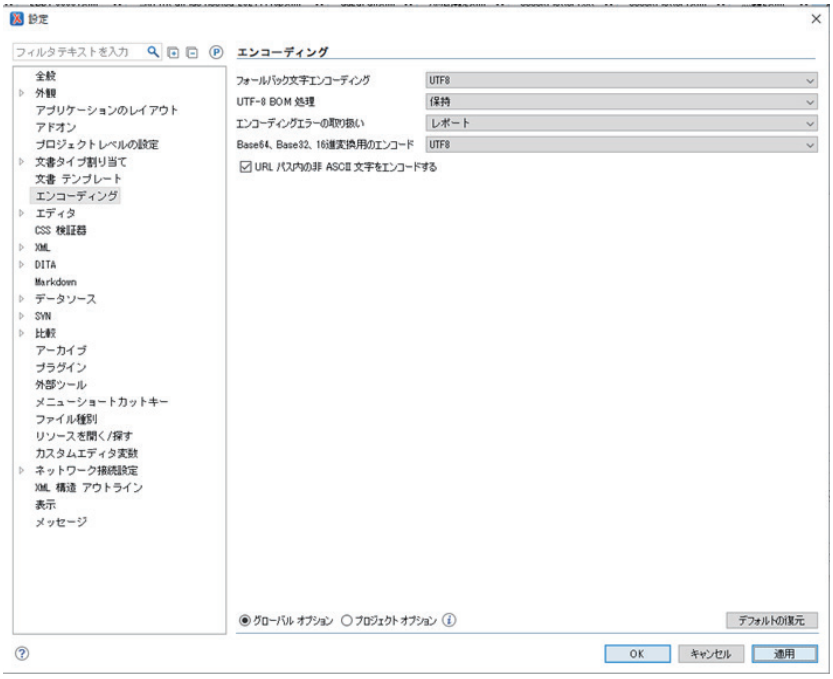
③テキストを挿入する

この演習では、例として、夏目漱石の手紙を用いる。しかし、手紙のテキスト入力から始めると時間がかかってしまうため、すでにテキストの入力をしたファイルが教材データの中に用意されている。この手紙は、1900 年 9 月、英国留学への途上から東京に住む妻、夏目鏡子に宛てたものである。

1. カーソルが <body> と </body> の間にあることを確認してから、Oxygen のメニューバーの「文書（Document）」→「ファイル（File）」→「ファイルを挿入（Insert File）」を選択しよう。
2. ファイル選択のダイアログにて「soseki_letter1.txt」を選択しよう。

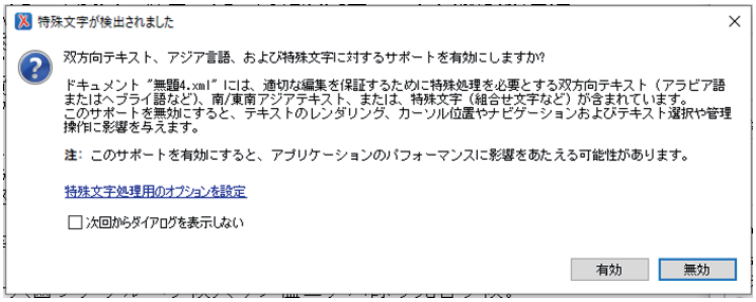
ここで、以下のようなエラーのダイアログが出ることがある。（これが出なければそのまま先に進もう）この場合、「文字エンコーディングエラーの扱い方法を変更」をクリックして、





3. その次の画面で出てくる設定画面で「フォールバック文字エンコーディング」の項目でUTF8 を選択して「適用」をクリックしよう。そして、再度、カーソルが <body> と </body> の間にあることを確認してから、Oxygen のメニューバーの「文書（Document）」→「ファイル（File）」→「ファイルを挿入（Insert File）」を選択して、その後、ファイル選択のダイアログにて「soseki_letter1.txt」を選択しよう。

4. ファイルの挿入ができると、以下のようなダイアログが出ることがある。この場合、必ず「有効」をクリックすること。



5. 作成中の文書の冒頭部分が以下のようになっているか確認されたい。

```
<?xml version="1.0" encoding="UTF-8"?>
<text type="letter"><body>
    [1]
今日ハ九月二十七日ニテ吾等が乗レル船ハ昧爽英領「ペナン」ト申ス港ニ着
キ申候。未明ヨリノ小雨ニ加フルニ出帆時刻ハ午前九時ナレバ遺憾ナガラ上陸ヲ
得ズ。上海ニテハ日本旅館ニ宿泊シ香港ニテモ同朋ノ営業ニ関ル宿屋ニテ
 日本飯の食納ヲナシ候。上海モ香港モ宏大ニテ立派ナル┐ハ到底横濱神戸ノ
比ニハ無之特に香港ノ夜景抔ハ満山ニ+夜光ノ+寶石ヲ無数に鏤メタルガ如クニ候。
又「ピーク」トテ山ノ絶頂迄鉄道車ノ便ヲ假リテ六七+十+度ノ峻坂ヲ上リテ四方ヲ
見渡セバ其景色ノ佳ナル┐実に愉快に候。「シンガポア」ニモ上陸シ馬車ヲ假
リテ植物園博物館及市街を一見致候。茲ニモ日本の旅館アリテ+午+食ヲ認メ
```

6. このテキストでは、タグを使う代わりに独自に定義された文字ベースのマークアップが行われている。たとえば、ページ番号が角括弧内に書かれており、ハイフンが直接に一つの文字を囲んでいるのは削除されたことを示している。また、原本にはない句点を入れて文章を区切っている。
7. この演習の最後の soseki_letter1_1.jpg、soseki_letter1_2.jpg の画像と、このテキストを簡単に比べてみよう。

④手紙の基本的な構造を符号化 (encode) する

1. まず、すぐ上の図を見ていただきたい。一番上の（3 行目 ?）ところで、「[1]」を改ページの <pb/> エレメントに書き換えてみよう。最初に「[1]」を削除してから、その箇所に <pb/> を入力して、Oxygen がどのようにして入力を補助するか見てみよう。なお、<pb/> が、「空（empty）」エレメントであることにも注目しておこう。

   この <pb/> エレメントは頁の始まりを示すものであり、1 ページ目であることを示すために @n 属性と「1」という値を記述することができる。次に「[2]」もあるので、これも同じマークアップに置き換えておこう。ただし、この場合は、コンピュータ <pb/> を数えることで何ページ目か確認できるため、@n 属性による頁番号を記述しなくても問題なく、む

しろ作業の省力化のためにこれを記述しないという選択肢があり得ることにも留意しておきたい。

2. この手紙は 2 つのセクションに分かれている。
   (a)　散文のセクション
   (b)　締めくくり（closer）（挨拶と署名を含む）
   まずは、これらのセクションをマークアップするところから始めることにしよう。
3.「今日ハ九月二十七日ニテ」から、終わりの方の「達スベシト存候。」までを選択して反転表示させよう。
4. そこを反転表示にしたまま、キーボードの「Ctrl」キー（マックの場合はコマンドキー）を押しながら「e」を押す、つまり、ショートカットキーの Ctrl-e を押そう。あるいは、右クリックして、「リファクタリング（Refactoring）」を選び、「タグで囲む（Surround with Tags）」を選んでもよい。ダイアログがポップアップするので、そこに「div」と入力して「OK」をクリックしよう。そうすると、反転表示させたテキストの前に開始タグが置かれ、後ろに終了タグが置かれる。これは既存のテキストにタグ付けをしていく作業において頻繁に使用する機能であるため、ショートカットキーを覚えておくとよいだろう。
5.「金之助」から「鏡どの」までを同様にして今度は `<closer></closer>` で囲もう。
6. ここで、この XML ファイルには div エレメントが一つと closer エレメントが一つあるという状態になっているはずである。

⑤段落と行をマークアップする

次に、段落をマークアップしてみよう。

1.　たくさんのテキストをなるべく効率的にマークアップするために、ここでの散文の領域の場合は、「エレメントで囲む」と「エレメントで区切る」の合わせ技が利用できる。まず、今回のテキストでは、見た目上は 7 つの段落があることを確認しよう。
2.　`<div>` タグを含まないようにしつつ「今日ハ九月二十七日ニテ」から、終わりの方の「達スベシト存候。」までをすべて選択して反転表示させよう。
3.　Ctrl-e か「エレメントで囲む」を用いてこれを段落を表す `<p>` エレメントで囲もう。
4.　ここで一つの `<p>` で囲んだテキストは、見た目上は 7 つの段落なので、7 つの段落に分割してみよう。見た目上の段落の間には空白行があるので、まず、カーソルを「ニ百ニ一モ過タズ感心なㄱニ候。」と「皆御変リナキㄱト存候其」の間の空白行に置いてみよう。そして、alt-shift-d（Alt キーと Shift キーを押しながら「d」キーを押す）(マックでは Ctrl-Opt-D) を用いるか、右クリックのメニューから「リファクタリング（Refactoring）」→「エレメントを分割する（Split Element）」を選ぼう。これにより「`</p><p>`」が入力されて最初の領域が分割され、二つの段落を持つことになるはずである。
5.　他の空白行でも同様にして、最終的にパラグラフが 7 つになるようにタグを入力していこ

う。

⑥散文における改行

これにも何らかの意味を見出す人がいるかもしれないので、この散文の改行をタグで記述してみよう。

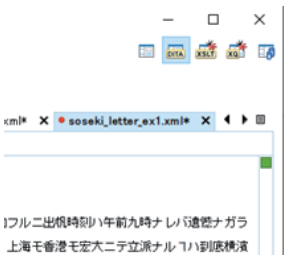
1. `<div><p>` の内側で、改行の記述に用いるエレメント `<lb/>`（line beginning、つまり行頭を意味する）を、「キ申候。未明ヨリノ」の直前に入れてみよう。これは、一度入力したら、あとはコピー&ペーストですぐにできる。もしそのようにすれば、Oxygen は、少し見やすくなるように、行をインデントしてくれるだろう。そうすると、上の方の行は以下のようになっているはずである。

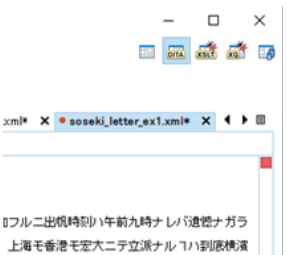
```
1  <?xml version="1.0" encoding="UTF-8"
2  <text type="letter"><body>
3      <pb n="1"/>
4  <div><p>今日ハ九月二十七日ニテ吾等が乗
5  <lb/>キ申候。未明ヨリノ小雨ニ加フルニ
6      <lb/>得ズ。上海ニテハ日本旅館ニ宿
7      <lb/>日本飯の食納ヲナシ候。上海モ
8      <lb/>比ニハ無之特に香港ノ夜景抔ハ
9      <lb/>又「ピーク」トテ山ノ絶頂迄鉄
10     <lb/>見渡セバ其景色ノ佳ナル丁実に
```

2. `<div><p>` 内の `<lb/>` 付与が終わったら、次は、「closer」とマークアップした領域でも、行頭に同じことをしてみよう。

⑦整形式（Well-formed）の XML 文書を整形しインデントする

1. あなたのファイルが「整形式」であることを確認してみよう。Oxygen が幸せの緑の四角形を右上に表示してくれるなら、それが「整形式」であるということになる。もし、怒りの赤い四角形であったなら、その問題を探して誤りを修正した方がよいだろう。（そのファイルの中での問題のある場所は右側の赤いバーによって示されて、赤い下線がつけられ、ウインドウの下部のバーにはそのエラーの理由が英語で表示される。）





（幸せの緑の四角形）　　　　（怒りの赤い四角形）

2. ここで、あなたのファイルを整形してインデントしてみよう。この機能では、その文書の階層構造におけるそれぞれの場所に基づいて空白やインデントの一部やエレメントをきれいに並べ直してくれる。ツールバーから「整形およびインデント（Format and Indent）」というアイコン（いくつかのインデントされた行のようにみえるもの）を選ぶか、メニューバーの「文書」→「ソース」→「整形およびインデント」を選ぼう。
3. ファイルを整形してインデントするという操作は必須ではない。XML の場合は、たとえば、すべてが一つの行になっていてもよい。しかし、整形してインデントしてあった方が他の人にとって読みやすくなることが多い。
4. 「整形及びインデント」をした結果、あなたのファイルは以下のようになっているはずである。：

⑧ここまでの作業を保存する

ここまでの作業結果を保存しよう：

- あなたの作品は整形式になっているか？あなたは幸せの緑の四角形と怒りの赤い四角形のどちらを持っているか？
- 「ファイル（File）」メニューから「保存（Save）」をクリックするか、「保存（Save）」アイコン（古の 3.5 インチフロッピーディスクのように見えます）をクリックしよう。
- 「soseki_leter_ex1.xml」という名前か、もしくは好きな名前をつけてそのファイルを保存しよう。

⑨セルフチェック

・以下の問いに自分で答えることで、この演習の基本的な原理をどれくらい理解しているかチェックしてみよう。
・どうやって Oxygen で新しい XML 文書を作り始めるのか？
・XML 宣言とは何か？
・整形式文書とは何か？
・「タグで囲む」操作はどのようにすると効率的で、どのようにしてそれをすばやく繰り返せるか？
・なぜ「エレメントを分割する」方法が便利な場合があるのか？
・あなたの現在のファイルにおける個々のエレメントとアトリビュートの機能は何か？
・あなたのマークアップを整形してインデントすることの有益な点は何か？

⑩次回にすべきこと

・ あなたの XML ファイルは整形式ではあるが、妥当（valid）ではない。なぜなら特定のスキーマ（たとえば、TEI）に対して妥当性の検証をしていないからである。次回は、TEI 文書の構造と、もっともよく使われるエレメント群が紹介される。もし、今回の演習がはやく終わってしまったなら、オンライン版の TEI ガイドラインを閲覧するとよいだろう。

```
http://www.tei-c.org/release/doc/tei-p5-doc/ja/html/index.html
```

・ 特に、個々のエレメントを参照するページのための「エレメント（Element）」の別表をみるとよいだろう。このファイルで利用した個々のエレメントがどのように定義されているかを見て、検討してみよう。

4．妥当（valid）TEI/XML 文書の作成 ― 演習 1-3

学習の成果

この演習を通じて以下のことを習得できる。

・最小限の妥当な TEI/XML ファイルに必要なエレメントとアトリビュート（属性）を理解する
・TEI/XML ファイルをスキーマと関連づける
・TEI の名前空間を用いる
・最小限の TEI ヘッダとテキストの本文を作成する
・妥当性の検証と形式が整っていること（整形式）の両方をチェックする

要点

この演習では、TEI/XML ファイルの作成と、ここまでに作成してきたものをそれに挿入すること

を体験する。そして、<teiHeader> の要求事項と TEI ファイルの基本的な構造について学ぶことになる。

①新しい XML ファイルを作成して開始する

新しい XML ファイルを演習 1-2 で学んだ手順で作成してみよう。

あらゆる TEI ファイルは <TEI> エレメントか <teiCorpus> エレメントで始まる。多くの場合、<TEI> エレメントを使うことになるだろう。これらのエレメントは、「xmlns」と呼ばれる特別な擬似的な属性（アトリビュート）を持つが、これは、一定のルールに基づいたエレメントのまとまりである「名前空間」を示すものである。これは、（上書きされない限り）その内部にあるあらゆるエレメントによって引き継がれる。これによって我々は、他のスキーマのものではなく、ほかならぬ TEI のスキーマにおける（たとえば）<title> エレメントを扱っているということを確認できるようになっている。では、以下の手順で <TEI> エレメントと名前空間を記述してみよう。

1. <TEI> エレメントを追記してから、TEI の名前空間 (namespace) である「http://www.tei-c.org/ns/1.0」を入力しよう。（つまり、「xmlns="http://www.tei-c.org/ns/1.0"」と入力する）。そして、開始タグと終了タグの間にいくつかの空白を入れよう。そうするとあなたのファイルは次のようになっているはずである。

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">



</TEI>
```

2. Oxygen に何が起きているかということと、Oxygen がどのようにして入力作業を補助するかに注目しよう。そして、作業中のファイルが今、幸せの緑の四角形ではなく怒りの赤い四角形になっていることにも着目しよう！そのファイルは整形式（well-formed）か？（そのはずである）。それではなぜこれは赤いのだろうか？
3. もし怒りの赤い四角形が表示されているなら、それは現在使用している Oxygen が TEI のスキーマを組み込んでいるからである。つまりこの場合、TEI の名前空間の入った <TEI> タグで始まっている XML ファイルは、自動的に、Oxygen に組み込み済みの TEI スキーマに関連づけられるのである。そしてここでは、TEI スキーマと照合して、そこに書かれたタグ群の中に、TEI では必須のタグ、<teiHeader> を入れていないことについて不満を言っ

ているのである。<teiHeader> は、当該電子文書についての説明を記述するためのエレメントであり、書誌情報や目録情報、メタデータから、そのファイルがどのように方針で作成されたかということまで、様々な情報を記述することができるようになっている。他人が作成したテキストデータを再利用しようとする場合、そういった情報は欠かせないものであり、それゆえに、すべての妥当な TEI ファイルはこの <teiHeader> を持っていなければならないと TEI スキーマで定義されている。

② <teiHeader> を追加する

ここで我々は、<TEI> エレメントの内側に、<teiHeader> エレメントを追加しなければならない。

1. カーソルを <TEI> 開始タグと終了タグの間に持っていって、<teiHeader> エレメントを入力しよう。Oxygen が </teiHeader> 終了タグを入力してくれることに注意しよう。そしてさらに、Oxygen に正しいオプションが設定されていたならば、Oxygen は、TEI スキーマを理解して、特定の内容が <teiHeader> の内側に必要とされると知っているために、Oxygen は自動的にそのマークアップを提供してくれる。（もしそうでなければ、それを手で入力しなければならないことになるので大変手間がかかることになってしまう。）これがうまくいっていれば、以下のようになるはずである。:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <TEI xmlns="http://www.tei-c.org/ns/1.0">
3  <teiHeader>
4      <fileDesc>
5          <titleStmt>
6              <title></title>
7          </titleStmt>
8          <publicationStmt></publicationStmt>
9          <sourceDesc></sourceDesc>
10     </fileDesc>
11 </teiHeader>
12
13 </TEI>
14
```

2. 手元のファイルはまだ幸せの緑の四角形ではなく怒りの赤い四角形になっていることに注意しよう。これは、いくつかのマークアップをしたにも関わらず、まだいくつかの必要なエレメントが足りないためである。
3. まず、<title> 開始タグと終了タグの間に「漱石筆鏡子宛て書簡」というタイトルを一つ追加しよう。この <titleStmt> の内側で許容される入力できる他のエレメントとしては、<author> (夏目漱石) がある。ここでは、<author> 夏目漱石 </author> と入力しておこう。
4. それから、自分自身の仕事を記録しておくために、<titleStmt> 以下に、責任の所在を示すための汎用的なエレメントである <respStmt> を（自分の名前で <name> エレメントと、

「TEI 符号化」等と入れた <resp> エレメントとともに）記述することができる。

```
<respStmt>
  <name> 自分の名前 </name>
  <resp> TEI 符号化 </resp>
</respStmt>
```

4. そして、段落 <p> を <publicationStmt> の内側に追加して、このファイルが何に関するものかということを記録するテキストを書いてください。テキストは「TEI の演習」などといったものでよいでしょう。
5. <sourceDesc> の内側に、<p> エレメントを入力し、その内側に「このファイルは、夏目鏡子への手紙のデジタル画像から作成された。」というテキストを入力しておこう。さらに詳しく説明を加えるために、「漱石筆鏡子宛て書簡」というタイトルを <ref> エレメントで囲み、さらにこの <ref> に「https://touda.tohoku.ac.jp/collection/database/library/public/10030010003081」という値を持つ @target を付与すると良い。これは、その URL でこのテキストデータの元になった画像が公開されているからである。
6. 作業中のファイルは、現在、次のようになっているはずである。:

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
    <teiHeader>
        <fileDesc>
            <titleStmt>
                <title>漱石筆鏡子宛て書簡</title>
                <author>夏目漱石</author>
            </titleStmt>
            <publicationStmt><p>TEIの演習</p></publicationStmt>
            <sourceDesc>
                <p>このファイルは、
                    <ref target="https://touda.tohoku.ac.jp/collection/database/library/10030010003081">
                        漱石筆鏡子宛て書簡</ref>のデジタル画像から作成された。
                </p>
            </sourceDesc>
        </fileDesc>
    </teiHeader>
```

7. たとえ <teiHeader> で必要とされる点をすべて満たしたとしても、このファイルは全体としてはまだ妥当（valid）ではないということに注意しよう。この状態で画面の下部のバーを見ると、以下のようなエラーメッセージが出ている。

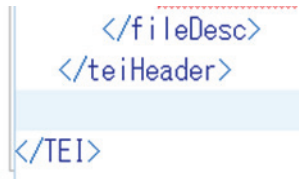
```
element "TEI" incomplete; expected element "TEI", "facsimile", "fsdDecl", "sourceDoc", "standOff" or "text"
テキスト グリッド 作者
```

8. これは、TEI エレメント内に必要なエレメントがまだ足りず、ここに列挙されたいずれかのエレメントが必要である、というメッセージである。TEI では、ヘッダ（<teiHeader>）以外に何らかのコンテンツを含むタグを必要とする。それぞれ、<facsimile> はデジタル画像（の情報）、<fsdDecl> は素性構造、<sourceDoc> は文書の見たままをタグで表現するテキスト、<standOff> は Linked Data などの本文に関係するが本文中には書き込まないデータ、<text> はテキストの論理構造をタグで表現するテキスト、を含むことになっている。では次に、この問題を解決しよう。

③ <text> を追加する

すべての TEI ファイルは、<titleStmt> と <publicationStmt>、<sourceDesc> を含む <fileDesc> を伴う <teiHeader> に加えて、ヘッダに続くものとして、少なくとも、<sourceDoc> か、<facsimile> か、<text>、<standOff> 等の内容を含むエレメントを必須としている。今回の場合には、我々は、先ほど作ったタグ付きテキストを入力することによって、<text> エレメントを追加しようとしている。そのためには :

1. すきまをあけるために、</teiHeader> 終了タグの次の行に一つ空白行を付け加えよう。



2. カーソルを空白行に置いたまま、メニューバーの「文書（Document）」メニューで、「ファイル」→「ファイルの挿入」を選択しよう。これが「文書」メニューであることに注意しよう！もし最初の演習が終わっていれば、その時に保存したファイルを選ぼう。もし終わっていなかったら、「soseki_letter_ex1_finished.xml」というファイルが教材データの中にあるのでそれを選択しよう。これは演習 1-2 を完了させた状態のものである。

   なお、このときも、「双方向テキスト、アジア言語、および特殊文字に対するサポートを有効にしますか ?」というダイアログが出ることがあるが、その場合は必ず「有効」ボタンをクリックすること。
3. これを追加したらすぐに、Oxygen は、我々のファイルが妥当ではないと考えるだろう。なぜなら、XML 宣言が <text> エレメントの直前に書かれているからである。そこで、この冗長な XML 宣言を削除しよう。
4. 作業中の文書は、今、妥当な (valid) はずである。そして、幸せの緑の四角形が右上の端に表示されているはずである。もしそうでなければ、表示されているエラーメッセージをみて、問題の解決を試みよう。

④日本語 TEI スキーマを関連づける

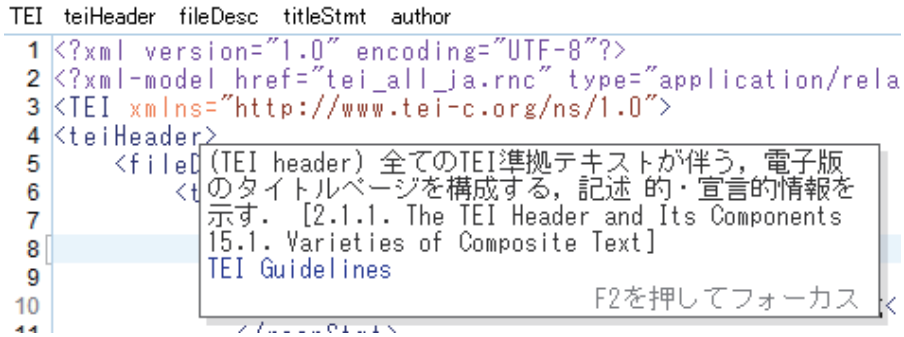
TEI における素晴らしいことの一つは、様々な言語でエレメントや属性の説明を入手することができるという点である。これは Google のようなものに支援されているのではなく、ボランティアによるものであり、アップデートされるのに時間がかかることがある。日本語版は、鶴見大学の大矢一志氏による翻訳が基本となっており、近年新たに追加されたエレメントや属性については TEI 協会東アジア／日本語分科会がボランティアとして取り組んでいる。

ここまでは、Oxygen に搭載された標準の TEI の枠組みを用い、TEI の名前空間において `<TEI>` エレメントではじまる文書で作業をしてきたが、エレメントや属性の説明は英語であった。これは、日本語 TEI スキーマを組み込むことで日本語で表示できるようになる。以下の手順を試してみよう。

1. 今回作業中の文書を読み込んで表示した状態で、「文書」メニューから「スキーマ」というサブメニューを選び、「スキーマの割当て」を選ぶ。
2. 配布されたフォルダを開いて、「tei_all_ja.rnc」というファイル（これが日本語版の TEI スキーマファイル）を選択する。
3. これで、ファイルの 2 行目あたりに、以下のように、この日本語スキーマファイルを指すタグが入力される。（ただし、href の値のファイルパスについては以下の例とは異なっている場合があるので注意されたい。）

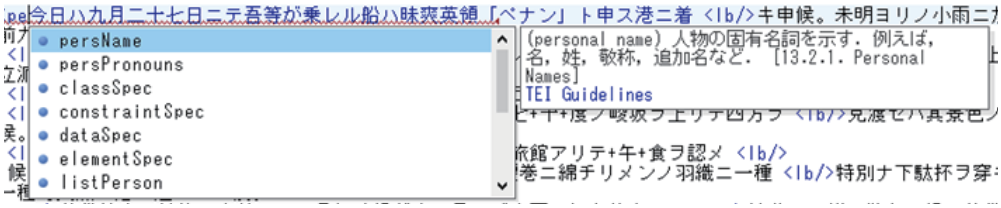
```
TEI  teiHeader  fileDesc  titleStmt  author
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-model href="tei_all_ja.rnc" type="application/relax-ng-compact-syntax"?>
3 <TEI xmlns="http://www.tei-c.org/ns/1.0">
```

4. このようになった上で、右上に緑の四角形が表示されているなら、マウスカーソルを TEI のタグにあててみよう。以下は `<teiHeader>` にあててみた例だが、このように、日本語でタグの解説が表示される。



さらに、ここですでに気づいている人もいるだろうが、TEI スキーマが割り当てられた状態でタグを入力しようとして任意の箇所で「<」を入力すると、その箇所で使用可能

なタグのリストとその解説が表示される。日本語スキーマを割り当てると、その解説も日本語で表示される。たとえば以下のようになる。



このように、TEI でタグ付けをする場合に Oxygen のような XML エディタを使うと、タグの階層や名称を正確に暗記しなくても、スキーマに書かれたルールや解説を読み取ってこのように適宜使えるもののみを候補として提示してくれる。タグを入力する際も、タグの名称を入力せずともこの候補リストから選択するだけで入力できる。TEI に限らず XML のタグ付けは一見すると大変そうに見えることもあるが、このように、適切な道具立てがあれば、あまり人の手を動かすことなく進めていくことができる。

⑤マークアップを改良する

1．日付・地名をマークアップ

最初の行に日付が書かれている。この「九月二十七日」を、日付を表現する <date> タグで囲もう。そして、属性 @when を付与し、ISO8601 に準拠した日付情報を値として記述してよう。これは、本文からは判断できないが、夏目漱石に関わる周辺情報から明治 33 年（1900 年）であることがわかる。そうすると日付は以下のようにして「1900-09-27」と記述することになる。

```
<p>今日ハ<date when="1900-09-27">九月二十七日</date>ニテ吾等ガ
```

このようにして書いておくと、本文からは得られない情報である日付情報を、本文の文章を改編することなく、機械可読な形で記述し、コンピュータで処理できるようになる。

同じ行には地名「ペナン」も登場するので、これも地名を表すエレメント <placeName> でタグ付けしておこう。この場合、以下のようになる。

```
ﾙ味爽英領「<placeName>ペナン</placeName>」ト申ス
```

他にも様々な地名が登場しているので、それぞれ、<placeName> タグを付けてみよう。なお、直前と同じタグで囲みたい場合は、Ctrl-e でいちいちタグ名を入力せずとも Ctrl-/ だけでタグ付けできるのでこれも試してみよう。

2. <add> と <del> を追加する

1. <add> エレメントは何かが追加されたものである場合に注記する。これは、文章が意味をなさないと判断された時に、著者によって直接に追加されたちょっとした文字列や、あとから写字生が追記したものに関して使うことができる。<del> エレメントは同じようなものだが、逆に削除を記すもので、いわゆるミセケチのことである。
2. この手紙の中にはいくつかの削除があり、削除された単語の直前と直後をハイフンで注記することでそれを記述している。たとえば以下のようになっている。:

```
- ノ如く -
-・-
```

これらのそれぞれを、ハイフンの代わりに <del> を使って注記しよう。たとえば、最後のものは以下のようになる。

```
投ゲル<del>・</del>貨幣ヲ
```

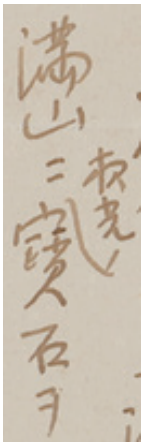
3. この手紙には追記が複数ある。これは以下のように「+」で囲んでいる。

```
+ 夜光ノ +
+ 十 +
…
```

これを、「+」の代わりに <add> を使って注記しよう。

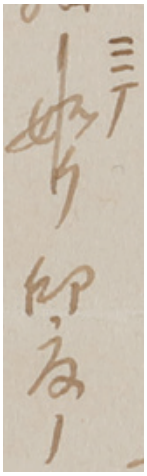
```
二<add>夜光ノ</add>寶石ヲ
```

4. 追記（<add>）に関しては、「どこから追記したか」という情報を記述すると後で役立つことがある。たとえば以下の例では行の右側から追記をしている。これは @place 属性の right という値で記述できる。これと同様にして、書簡画像を見ながら、追記の位置関係が明白なものは @place 属性を付与しよう。



```
満山ニ<add place="right">夜光ノ</add>寶石ヲ
```

5. 手書き資料の場合、何かを見せるように消してからそれを書き直したことがみて明らかな場合がある。たとえば以下のような場合である。これは、<subst> というタグで <del> と <add> を囲むことで、両者の関係を記述できる。



```
<subst>
    <del>ノ如ク</del>
    <add place="right">ニテ</add>
</subst>
印度ノ
```

改行や空白が何度も入っているが、すでにみてきたように、タグで明示的に改行や空白を入れない限り、本文に該当するテキストに改行や空白を入れても改行や空白としては見做されないので、このように、見やすい形に整えて編集作業ができる。

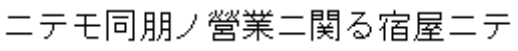
3．<closer> を改良する

1. 手紙の最後の <closer> には署名と宛名がある。署名には <signed>、このような場合の宛名には <salute> を使う。それぞれのタグを付してみよう。
2. <closer> は今、次のようになっているはずである。:

```
<closer><lb/> <signed>金之助</signed> <lb/> <salute>鏡どの</salute></closer>
```

4．明らかな <sic> 間違い </sic> をタグ付けする

この手紙には、ところどころ、漱石が書き間違えたかもしれないと思われる箇所がある。以下の箇所の「同朋」はおそらく「同胞」と書きたかったところではないかと想定される。



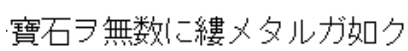
この場合、朋は誤りなので <sic> でタグ付けする。それだけでもよいのだが、修正したテキストも記述しておきたい時は <corr> で記述した上で両者を <choice> で囲む。

```
ニテモ同
<choice>
    <sic>朋</sic>
    <corr>胞</corr>
</choice>
ノ営業ニ関る宿屋ニテ
```

このように記述しておくと、漱石の書き癖を分析したい場合には <sic> を、漱石が書こうとした内容を分析したい場合は <corr> を読み取ってテキストとして用いればよい、ということになる。

ここで記述した sic/corr のそれぞれの表記について、表示を切り替えたり検索をしたりするためのもっとも簡単な方法は、冒頭でアクセスしておくことを指示した TEI 古典籍ビューワに読み込ませることである。それにより、sic/corr はボタンでテキストが切り替えられ、ブラウザ上でそれぞれの表記のテキストを検索することも可能である。

同様に、以下の例も修正してみよう。ここでは「鏤」と書くべきところを糸偏で書いてしまい「縷」になっている。



次に、以下の箇所の「留守中ノ」を検討してみよう。

留守中ノ寸法ノ合ウ様縫直シ

ここは、「留守中ニ」あるいは「留守中」とすべきところであると思われるが、一意に修正するのは難しそうである。この場合、(1) <corr> と <choice> は書かずに <sic> のみにするか、(2) 修正を一つだけ担当者の責任で書き込むか、(3) 複数の想定し得る案を書き込むか、あるいは、(4) 複数の担当者の間で意見が分かれた場合には追記するか、ということになる。あるエレメントの判断について責任者を明示するためには @resp 属性を用いる。それぞれの例を以下に挙げてみよう。

1. <corr> と <choice> は書かずに <sic> のみにするか。

```
留守中
<sic>ノ</sic>
寸法ノ合ウ様
```

2. 修正一つのみを担当者の責任で書き込み。
   「担当者 1」を @resp 属性で記述。

```
留守中
<choice>
    <sic>ノ</sic>
    <corr resp="担当者1"></corr>
</choice>
寸法ノ合ウ様
```

3. 複数の想定し得る案を書き込み。
   複数の <corr> を <choice> の下位に置く。ノを削除する場合とニに置き換える場合。

```
留守中<choice>
    <sic>ノ</sic>
    <corr></corr>
    <corr>ニ</corr>
</choice> 寸法ノ合ウ様
```

4. 複数の担当者の間で意見が分かれた場合。
   @resp でそれぞれの担当者を記述。

```
留守中
<choice>
    <sic>ノ</sic>
    <corr resp="担当者1"></corr>
    <corr resp="担当者1">ニ</corr>
</choice>
寸法ノ合ウ様
```

このように、TEI では、複数の解釈を共存させることができるため、状況によってはこれを活用するのも一つの選択肢である。

5．記述の正規化

記述を正規化する際にも、<choice> を利用できる。この場合は、元の資料の記述をなるべくそのまま記述したものを <orig> として、何らかの正規化された記述を <reg> とする。今回の書簡では、合略仮名「ㇳ（コト）」が多く使われているが、このテキストを検索するときに「ㇳ」を入力しないと「コト」の検索ができないというのはやや不便である。一方、「コト」と記述している場合もあるかもしれないので、結局「ㇳ」も「コト」も両方検索しなければならない。このような不便を避けるために、テキスト入力の際に「コト」に正規化してしまうことも多かっただろう。しかしながら、漱石の文字の選び方が何らかの研究の材料になるかもしれないと考えた場合には、「ㇳ」がそのまま記述されている方がむしろ有用である。このような、記述における多様性の有用性と検索における利便性の相克は、日本語に限らず多くの言語で常に問題になるところである。そこで、TEI ガイドラインでは、元の記述と正規化された記述とを併記しつつ、いずれかを選べるようにする記述方法を提供しているのである。

ここまでに習得した Oxygen のタグ入力方法や検索置換機能などを活かして、この書簡の中の「ㇳ」を以下のように記述してみよう。

```
<choice>
    <orig>ㇳ</orig>
    <reg>コト</reg>
</choice>
```

これに加えて、一箇所だけだが、合略仮名「𠦂（トモ）」も用いられているので、これも同様にタグ付けしてみよう。

なお、このように、正規化の仕方が一意に決まっているものについては、テキストの方でこのように対策をしなくとも、テキストは元資料のままに入力しておいて、検索の際に「ㇳ」と「コト」を同時に検索してしまうこともできる。そのような検索の仕方に対応した検索システムもい

くつか存在する。ただ、この場合はこの書簡のなかでは一意に決定できているが、検索対象のテキストが膨大な量になったときに対応できるのかどうか、別の正規化ルールと共存できるのか、といったことを考慮するなら、上記のように併記しておくことには一定のメリットがあると考えられる。

ここでも、TEI 古典籍ビューワに読み込ませてみよう。そうすると、orig/reg の選択肢が表示され、どちらかを切り替えて表示できるようになるはずである。検索も、ブラウザ上ではあるが、それぞれにあわせてできるようになっている。

6．縦書きスタイルの記述

この書簡は縦書き形式で書かれている。テキスト処理においては縦書きだろうと横書きだろうと線形であれば同じことだが、表示に際しては縦書きと横書きではかなり印象が異なり、そこから読み手が受ける情報にも異なる面があるかもしれない。あるいは、大規模なテキスト処理において、縦書きかどうかの情報が与えられていると、そこから何らかの傾向を発見できるかもしれない。そこで、縦書きであることを記述しておこう。

縦書きであることはテキスト処理においては見た目の問題として扱われるので、`@style` 属性を用いる。TEI ガイドラインにおいて、`@style` 属性は、スタイルシート言語である CSS (Cascading Style Sheet) に準拠することになっているので、CSS での縦書き右左表記である「writing-mode: vertical-rl」を `<body>` の `@style` の値として与えておこう。

```
<body  style="writing-mode: vertical-rl;">
```

⑥今回の作品を保存する

今回作った作品を保存しよう。

・自分の作品を自動的に整形してインデントしたか？
・自分の作品は形式が整っているか？幸せの緑の四角か怒りの赤い四角を持っているか？
・「ファイル」メニューから、「保存」を選ぶか、あるいは Save アイコンをクリックしよう。
・「soseki_letter_ex2.xml」という名前か、あるいは好きな名前でファイルを保存しよう。

⑦セルフチェック

以下の質問に答えることで、この演習の中核的な原理のいくつかを理解をしているかどうかチェックしよう。

・最小限の妥当な TEI XML 文書のためにどのエレメントと属性を必要としているか？
・`<teiHeader>` のどの三つの部分がすべての TEI 準拠の文書で必須とされているか？
・どこでこれらのエレメントと属性は許可されているか？
・あなたが利用したそれぞれのエレメントと属性の機能は何か？（もしあやふやなら、確認し

よう！）

・なぜこれらのエレメントと属性が TEI/XML で必須だと思うのか？

⑧さらに先へ

この演習を通じて経験したことは、XML を編集し妥当な TEI ファイルを作ることである。もしはやく終わってしまったら、TEI ガイドラインのオンライン版を眺めてみるとよいだろう。

```
http://www.tei-c.org/release/doc/tei-p5-doc/ja/html/index.html
```

・ 特に、今回利用したエレメントがどのように定義されているかを確認するために、TEI で定義されているすべてのエレメントをアルファベット順に並べている「Appendix C Elements」から、リンクされている個々のエレメントの解説ページを確認してみよう。

・ 他にどんなエレメントが <text> エレメントの中で許容されているだろうか？それを確認するために、<text> エレメントの解説ページの「May contain（下位）」の項目を見てみよう。あなたなら、それらを何のために使うだろうか？

・TEI に準拠したテキストの構造を自分で設計するためには、大まかなイメージを持つために、デフォルトのテキスト構造の章 や、すべての TEI 文書で利用可能なエレメント を読んでおくとよいだろう。これらの頁は、日本語訳も公開されており、本書の教材頁からリンクされているので参照されたい。