MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY

Institute of Computer and Information Technologies
Department of software engineering

ADMIT TO DEFENSE
Head of Department
______________Сяофена
"___" ____________ 2023

DEGREE PROJECT
(EXPLANATORY NOTE)

GRADUATE OF EDUCATIONAL AND QUALIFICATION LEVEL "BACHELOR"

Topic: "Service-oriented tools for checking the identity of XML documents"

Виконавець: Лю Сяофена

Керівник: к.ф.-м.н., доцент Оленін Михайло Вікторович

Нормоконтролер: к.т.н., доцент Радішевський Микола Федорович

Kyiv 2023

NATIONAL AVIATION UNIVERSITY

Faculty cybersecurity and software engineering
Department Software Engineering
Degree of education master
Speciality 121 Software engineering
Education-professional program Software engineering

APPROVED
Head of Department
______________Сяофена
"___" ____________ 2023

TASK
for the completion of the student's diploma project
Лю Сяофена

1. Project topic: Service-oriented tools for checking the identity of XML documents .

approved by the rector's order dated 20 .10.2023 No. /art .

2. Project implementation period: from 09/01/2023 to 31/12/2023.

3. Source data for the project: recommendations of the international consortium "W3C" regarding the standardization and canonization of XML documents. Research on the method of determining the equivalence of XML documents.

4. Contents of the explanatory note:
   1. Determine the XML file identification category.
   2. Method for Identifying and Classifying Design Documents.
   3. Develop tools for classifying XML file identifiers.

5. List of mandatory graphic material:
   1. Scheme of the tool's algorithm .
   2. Diagram of options for using the tool.
   3. Diagram of states of the tool .

6. Calendar plan-schedule

№	Task	Deadline	Performance note
1.	Familiarization with the statement of the problem and the study of literature Writing 1 section, presentation to the supervisor	14.10.2023-31.10.2023	
2.	Preprint of section 1 and auxiliary pages (draft) - title, task, schedule, abstract, list of abbreviations, content, introduction, source list. First standard control.	15.10.2023-22.10.2023	
3.	Writing 2 section, presentation to the supervisor	22.10.2023-01.11.2023	
4.	Writing 3 section, presentation to the supervisor	01.11.2023-14.11.2023	
5.	General editing and printing of an explanatory note, graphic material	14.11.2023-20.11.2023	
6.	Passing standard control	20.11.2023-26.11.2023	
7.	Development of the text of the report. Creating of graphic material for presentation	26.11.2023-27.11.2023	
8.	Get feedback from the supervisor, reviews.	27.11.2023-13.12.2023	
9.	Preparation of materials for transmission to the secretary of the DEC (software, GM, CD-R with electronic copies of software, GM, presentations, supervisors review, review, certificate of progress, 2 folders, 2 envelopes)	13.12.2023-19.12.2023	
10.	Graduation project presentation	19.12.2023-31.12.2023	

Date of issue of the assignment 05/09/2023 p.

Supervisor: Ph.D.-M.Sc. Associate Professor Olenin M.V.

The task was accepted by:

ABSTRACT

Explanatory note to the thesis " Service-oriented tools for checking the identity of XML documents "

The object of the research: – to consider the methodology of checking the identity of XML documents using the examples of the created software, which will be programmed using the tools provided by me.

The goal of the work is to create distinguish XML file identifiers for classification.

Research methods – use of software reverse engineering methods.

Type of development: object-oriented approach.

Hardware and software – PC with Windows 11 or Windows 10 operating system, an environment for object-oriented programming –VS Code. The use of artificial intelligence methodology is impossible without an Internet connection.

Application development was done in Node.js runtime environment Intended tool development assumption

The results of the work can be used in various fields in the development of XML or in programs for creating and checking existing XML files.

SOFTWARE IMPROVEMENT, CODE GENERATION, DOCUMENTATION CREATION, VULNERABILITY FINDING, REVERSE ENGINEERING, ADDING SOFTWARE MODULES.

目录

LIST ACRONYMS AND ABBREVIATIONS

XML – Extensible Markup Language;

SGML – Standard General Markup Language;

HTML – Hypertext markup language;

W3C – World Wide Web Consortium;

DTD – Document Type Definition ;

XSL – eXtensible Stylesheet Language ;

CSS – Cascading Style Sheets;

XLL – eXtensibleLink Language;

URL – uniform resource locator;

DOM – Document Object Model;

IU – User Interface;

NPM – Node Package Manager;

CHAPTER 1

Analysis based on inspection of XML document identification methods. Main XML features

1.1 Background

With the rapid development of network technology, a large amount of information expands and gathers, how to quickly and efficiently obtain effective knowledge from a large amount of information becomes more and more important in people's production and life. XML(extension markup language), HTML (Hypertext markup language) and other new generation of electronic document description language described documents have gradually replaced the original plain text format documents, XML because of its flexibility, simplicity, easy to expand and other characteristics, has become the network application "such as digital library, e-commerce, etc. data representation and exchange standards.

XML(Extensible Markup Language) is a semi-structured data description language designed by the World Wide Web Consortium W3C(WordWide Web Consortium), which is an important branch of SGML(Standard General Markup Language) specifically for We applications. SGML is a universal language used to describe documents with tags that existed long before the invention of the Web. Because of its huge size, it is difficult to learn and use, so people put forward HTML language to make up for the shortcomings of SGML. With the growing and deepening of Web applications, HTML has become a problem in practical applications. As a result, the W3C recommends a simplified version of SGML-XML. Compared with HTML, XML is independent of machine platforms, providers and programming languages, making it a bridge between different systems, different databases and different languages. Therefore, XML gives powerful capabilities and flexibility to Web-based data mining technology. At the same time, it is easy to realize the integration of heterogeneous data, easy to transmit and exchange data, which makes the query and search of heterogeneous database more simple.

At present, the Internet has formed a huge data warehouse composed of XML format data, which contains a wealth of information, so the mining of XML documents has become one of the best ways to quickly and effectively obtain information from the Internet. XML document mining, as the name suggests, is data mining for the data represented by XM documents. Data Mining refers to the non-trivial process of obtaining effective, novel, potentially useful, and ultimately understandable patterns from large amounts of data. The broad view of data mining is: data mining is the process of "mining" interesting knowledge from a large number of data stored in the database data warehouse or other information base. Data mining, also known as Knowledge Discovery in Database (KDD), is also regarded as a basic step in the database knowledge discovery process. There are four main tasks in data mining: predictive modeling, cluster analysis, association analysis, and anomaly detection. Among them, predictive modeling involves two types of predictive modeling tasks: classification for predicting discrete target variables and regression for predicting continuous target variables. Therefore, classification task is an important task in data mining. Therefore, XML document classification, as an important part of XML document mining, has gradually become the focus of research and discussion by scholars at home and abroad.

1.2 Research Objectives

1.2.1 XML formatting

The XML file formatting function allows users to format the content of the input XML file beautifully. Its purpose is to make the content of the XML file clearer and easier to read, and users can easily understand and edit the XML file. The formatted form makes the XML render more structured and readable by setting appropriate indentation and merging text content. If there is an error in the formatting operation, the user will get the appropriate feedback to indicate that the formatting failed.

1.2.2 XML automatic check

The automatic XML file verification function is to automatically verify the XML file that the user needs to verify after importing it into its functional interface, helping the user to determine whether it is a valid XML document. If there is a syntax error in the XML file content, the user will see the corresponding error message, and help the user to mark the location of the syntax error; If the content of the XML file is syntactically correct and valid, the user will see a syntactically correct XML file message that the functional interface passes to the client. This is very important for ensuring the correctness and integrity of the XML.

1.2.3 XML file import

XML file import requires a series of methods, the user can choose from the local computer to validate the XML file, and then import it into the component. This is useful for working with existing XML documents for subsequent formatting, validation, or other operations.

1.2.4 XML file export

The XML file export function enables users to save the contents of validated XML files to their local computer for future use or sharing with others. The user can select the location and name of the file in the file dialog box. This is useful for saving the state of an XML document after editing or processing it.

1.3 Introduction to XML Language

XML (eXtensible Markup Language) stands for Extensible Markup Language. XML text is a kind of semi-structured data. Semi-structured data is a kind of data between strictly defined structured data (such as relational database data) and unstructured data (such as pictures, sounds, videos, etc.). Different from other data, semi-structured data has the following characteristics:

(1) Pattern information is implicit. Different from the data in relational databases, although semi-structured also has a certain structure, this structure is implicit in the data and generally does not have a special schema definition.

(2) The structure of semi-structured text is irregular. In semi-structured text, the elements that make up a data collection may be heterogeneous.

(3) The data in semi-structured text does not have strict type constraints. Semi-structured text generally does not use predefined patterns, the data is irregular in structure, and there are no strict constraints on data types.

Currently, semi-structured data mainly comes from the following sources:

(1) HTML, XML, SGML and other files on the Internet. These files do not have strict schema requirements;

(2) Medical records in hospitals and online emails. In the literature retrieval system of libraries, there is a large amount of semi-structured data in the form of XML;

(3) When integrating heterogeneous databases, because various databases are involved, the data patterns are not unified. This also requires the use of large amounts of semi-structured text. The most commonly used type of semi-structured text is XML text.

In 1995, XML text began to take shape and was filed with the World Wide Web Consortium (W3C). In February 1998, XML officially became a W3C standard, called XML1.0. Since XML was proposed, it has increasingly become a standard for network data exchange. XML uses a standard, simple, self-describing way to encode text and data, so that XML text can be easily exchanged between different hardware, operating systems, and applications without requiring too much effort. human interference. In fact, XML was designed to transmit and carry data information. Data in XML text is contained in text strings.

The XML specification specifies the syntax rules that XML text must follow: how to define tags, how to define elements, how to define attributes of elements, etc. XML text has strong flexibility. XML developers or users in different fields can define elements according to their own needs to meet their own needs, so it has good scalability.

In addition, although XML text has great flexibility, it also has many strict specifications. Such as how to define the label, the format the label should have, etc. This provides a unified standard for parsing XML text and provides convenience for writing programs that parse and operate XML text. There are two types of languages that are often used together with XML text: Document Type Definition (DTD) or XML Schema, and eXtensible Stylesheet Language (XSL). Among them, DTD or XMLSchema specifies the logical structure that XML files should have. XML text that conforms to the provisions of DTD or XMLSchema is called well-formed XML text.

1.4 XML Technology Overview

1.4.1 The emergence and development of XML

XML (eXtensible Markup Language) is an important branch of SGML(Standard General Markup Language) designed by the World Wide WEB Consortium (W3C) specifically for Web applications. In general, XML is a Meta-markup Language that provides a format for describing structured data. In more detail, XML is a language similar to HTML that is designed to describe data. XML provides an independent way to run programs to share data. It is a new standard language used to automatically describe information. It enables computer communication to extend the function of the

INTERNET from information transmission to a variety of other human activities. XML consists of thousands of rules that can be used to create markup languages and process all newly created markup languages in a simple program called an analyzer, just as HTML provided a display for the first computer user to read INTERNET documents, XML created an Esperanto that anyone can read and write. XML solves two WEB problems that HTML cannot solve, namely, the problem of fast INTERNET development and slow access speed, and the problem of having a lot of information available, but it is difficult to find the part of information you need. XML can increase the structure and semantic information, so that computers and servers can immediately process a variety of forms of information, so the use of XML's extended function can not only download a lot of information from the WEB server, but also greatly reduce the network traffic.

The TAG in XML is not predefined, and the user must customize the required tag. XML is a language that can describe itself. XML uses the Document TypeDefinition (DTD) to display this data,XSL(eXtensible Style Sheet Language) is a mechanism to describe how these documents are displayed, and it is XML's style sheet description language. Older than CSS(Cascading Style Sheets) for HTML, XSL consists of two parts: a way to transform XML documents, and a way to format XML documents. XLL(eXtensibleLink Language) is an XML linking language that provides links in XML, similar to HTML but more powerful, using XLL to link in multiple directions and the links can exist at the object level rather than just the page level. Because XML can tag more information, it makes it easy for users to find the information they need. With XML, WEB designers can not only create text and graphics, but also build multi-level, interdependent systems of document type definitions, data trees, metadata, hyperlink structures, and style sheets.

1.4.2 Key features of XML

It is the characteristics of XML that determine its excellent performance. XML as a markup language has many characteristics:

1. Easy.

XML consists of rules that can be used to create markup languages and process all newly created markup languages in a simple program, often called a parser. XML can create an Esperanto that anyone can read and write. This ability to create Esperanto is called the unity function, for example, XML creates tags that always appear in pairs, and relies on a new coding standard called the Unified code.

2. Open to the public

XML is SGML There is a lot of mature software available on the market to help write, manage, etc. Open standard XML is based on proven standard technologies and optimized for the web. Many of the industry's leading companies work side by side with the W3C working group to help ensure interactive operations support for developers, authors, and users on a variety of systems and browsers. And improving XML. XML interpreter can use programming methods to load an XML document, when the document is loaded, users can obtain and manipulate the information of the entire document through the XML file object model, which speeds up the network running speed.

3. Efficient and scalable

Support the reuse of document fragments, users can invent and use their own labels, but also share with others, extensibility is large, can define an unlimited set of annotations in XML. XML provides a schema for labeling structured data. An XML component can declare the data associated with it as retail price, sales tax, title, quantity, or any other data element. As many organizations around the world adopt the XML standard, more features will emerge that will allow once locked data to be passed over a cable in any way and rendered in a browser or passed on to other applications for further processing. XML provides an independent application method to share data. With DTDS, different groups of people can use a common DTD to exchange data. Your application can use the standard DTD to validate the data you receive, or it can use a DTD to validate your own data.

4. Internationalization

The standard is international and supports most languages in the world. This stems from a new coding standard that relies on its unified code, which supports all the world's mixed texts written in major languages. For most number processing in HTML, a document is usually written in a particular language, be it English or Ukrainian or Arabic, and if the user's software can't read the characters in that particular language, then he can't use the document, but software that can read XML can smoothly process any combination of characters in these different languages. Therefore, XML can exchange information not only between different computer systems, but also across national and cultural boundaries.

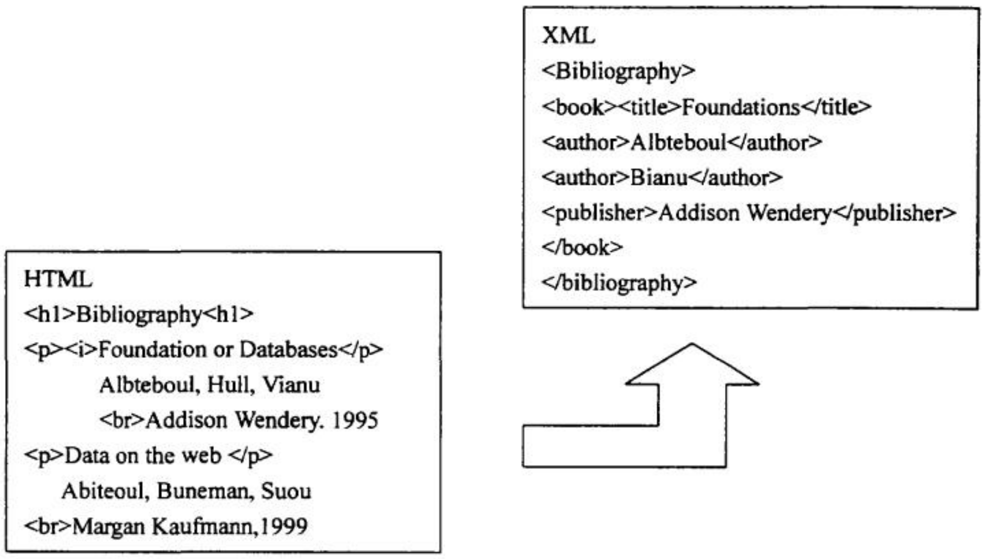
1.4.3 Description of XML and comparison with HTML

The Web-based Internet consists of four technical elements, namely HTTP, HTML, URL (uniform resource locator) and browser. HTML is easy to learn, which is convenient for non-computer professionals to create their own multimedia homepage with hypertext features. So that the Web home page and every ordinary person is closely connected, thus creating a very rich and colorful Internet world, so many people believe that HTML is the main basis of Web technology. On the other hand, the simplicity of HTML language makes web applications rapidly spread, and when people want to further apply the Web to new fields such as e-commerce, medical care and insurance, they find that it is the original "advantages" that restrict the expansion of web applications. This is because the current Web technology built on HTML has two fatal weaknesses:

HTML only describes how the information is displayed, not the content itself. In other words, HTML is a "display description" language that says only how graphics, text, and buttons should be laid out on the home page of the web. It says nothing about the properties of the information itself.

A lot of work that can be completed in the client has to be handled by the web server, which greatly increases the burden of the network and reduces the efficiency of the network operation. Extensible markup Language (XML) is proposed to solve the above two defects. XML is actually an important branch of SGML, the standard general-purpose markup language. It contains a set of basic rules by which anyone can create a markup language that meets the needs of their specific application domain, and the markup language thus created is no longer a description of the display of information, but a certain property of the information itself (such as product specifications in a shopping order, price performance indicators, and delivery methods).

Like HTML, XML uses a set of elements as markup, but unlike HTML, XML is no longer a pure markup language, but a definition language. XML can label the information content itself in a file with certain attributes in explicit terms and nested structures, and such attribute tags can be arbitrarily defined by the user. In other words, XML can be used to define its own markup language, thus breaking HTML's constraint of only a fixed set of tags, that is, XML can define an infinite number of tags to describe any property of information in the Web. For example, in HTML, the product name, product price, and performance indicators are not identified in the file, and the computer will not be able to identify such information attributes from the file, so there is no further processing of such attributes (such as classification, retrieval, or some specific processing). On the other hand, XML can label the Product name, Product price, and Performance indicators with "product name", "product price", and "performance", and use the basic rules of XML to express the above information attributes explicitly. In this way, the computer can easily identify these attributes, so that it can further make a variety of different processing as needed. Obviously, the proposal of XML and the formulation of relevant standards and the development of related technologies supporting XML (such as XML language interpreters, including a new generation of browsers, etc.) will greatly promote the development of Web applications to deeper and wider fields.



```
HTML
<h1>Bibliography<h1>
<p><i>Foundation or Databases</p>
        Albteboul, Hull, Vianu
        <br>Addison Wendery. 1995
<p>Data on the web </p>
    Abiteoul, Buneman, Suou
<br>Margan Kaufmann,1999
```

```
XML
<Bibliography>
<book><title>Foundations</title>
<author>Albteboul</author>
<author>Bianu</author>
<publisher>Addison Wendery</publisher>
</book>
</bibliography>
```

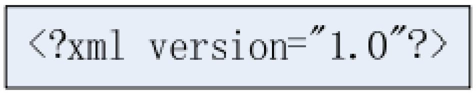
Fig. 1.1. “Graph comparing XML and HTML”

1.5 The process of XML validation

1.5.1 A well-structured check

This stage is used to perform a syntactic check of the document, checking against a series of syntax specifications to determine that the document is well-constructed.

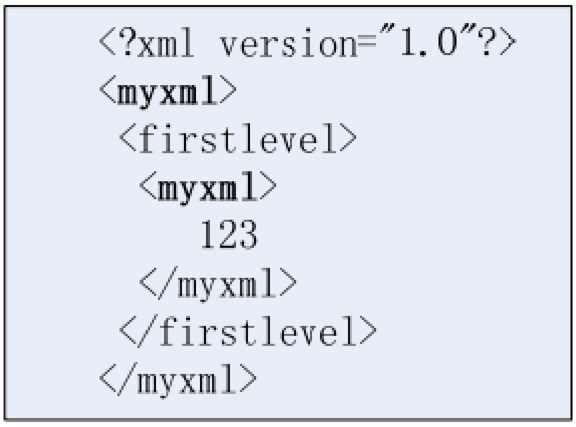
XML documents usually start with an XML declaration, although this is not required. But it is good practice to include a declaration because it tells the application or person that a piece of XML content will follow. It can also provide the processor with additional information about the document, such as its character encoding type. The most concise form of the declaration is shown in Figure 1.2



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

Fig. 1.2 “The XML version element”

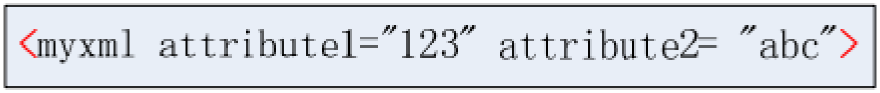
XML documents have one and only one root element. It surrounds all other elements in the document, and its name must be unique in the document. The example shown in Figure 1.3 is incorrect because it has nested elements with the same name as the root element.



```
<?xml version="1.0"?>
<myxml>
 <firstlevel>
  <myxml>
     123
  </myxml>
 </firstlevel>
</myxml>
```

Fig. 1.3 “XML with duplicate root element”

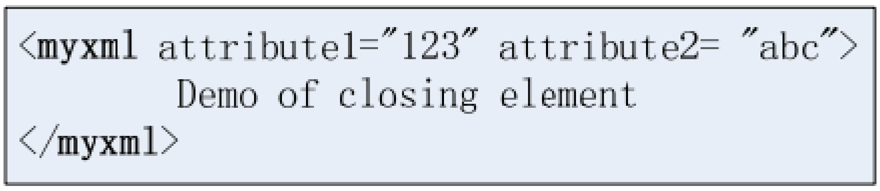
Elements of XML must be closed. In the example shown in Figure 1.4, both the left - and right-closed tags are highlighted in red. Open elements are not allowed in a well-formed XML document.



```
<myxml attribute1="123" attribute2= "abc">
```

Fig. 1.4 “XML instance illustrating open and close tags”

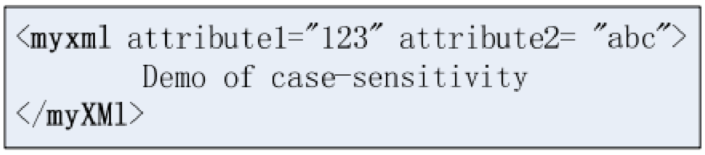
The start label must have a corresponding end label. In the example shown in Figure 1.5, the opening and closing tags are highlighted in bold. You can see that the end tag simply precedes the element name with a slash "/" instead of repeating the attribute declaration that appears in the start tag.



```
<myxml attribute1="123" attribute2= "abc">
        Demo of closing element
</myxml>
```

Fig. 1.5 “XML instance illustrating closing of an element”

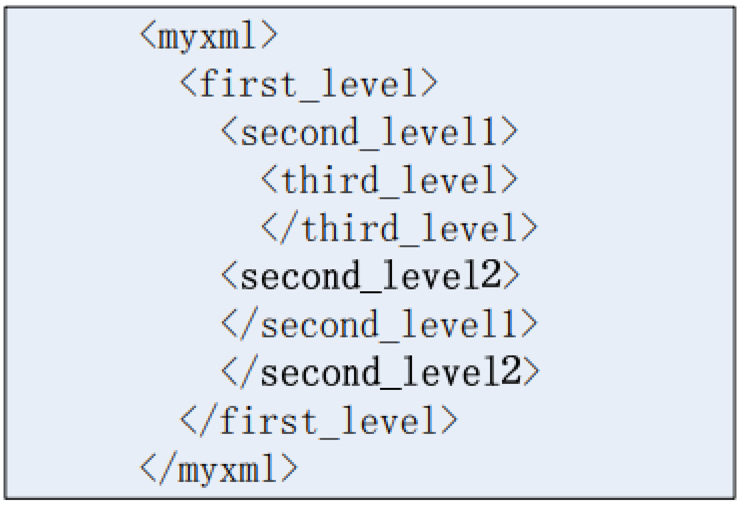
XML elements are case-sensitive. The XML element in Figure 1.6 starts and ends with the same name but inconsistent case. This is unacceptable for XML validators.



```
<myxml attribute1="123" attribute2= "abc">
        Demo of case-sensitivity
</myXMl>
```

Fig. 1.6 “XML instance illustrating case-sensitivity of element names”

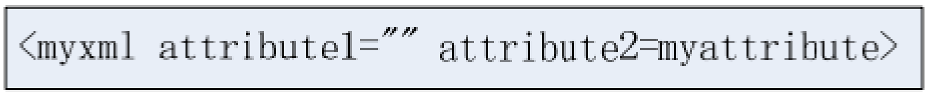
XML elements must be nested correctly. The example in Figure 1.7 depicts three layers of nested elements. Notice that there are two elements in the second layer, and although each has a corresponding end label, there is a nesting disorder. The correct </second_level1> end tag should appear before the <second_level2> start tag.



```
<myxml>
  <first_level>
    <second_level1>
      <third_level>
      </third_level>
    <second_level2>
    </second_level1>
    </second_level2>
  </first_level>
</myxml>
```

Fig. 1.7 “XML instance illustrating nesting of elements”

Attribute values for XML must be written within quotation marks. If an attribute exists, it must have a value, even if it is null. In the example shown in Figure 1.8, although attribute 2 has an assignment, it is invalid because it is not quoted. On the other hand, even though the value of attribute 1 is an empty string, it is valid.



```
<myxml attribute1="" attribute2=myattribute>
```

Fig. 1.8 “XML instance illustrating quoting of attributes”

Element names must follow the following XML naming convention:

- The first character can be a letter or underscore, not a number or other punctuation mark.
- After the first character, numbers and punctuation can be used.
- Cannot contain Spaces.

- cannot contain the ":" character, because the colon is a reserved character.
- "xml" in any combination of case and case cannot be used as the first three characters of the name.
- The element name must be immediately followed by the "<" character with no Spaces between.

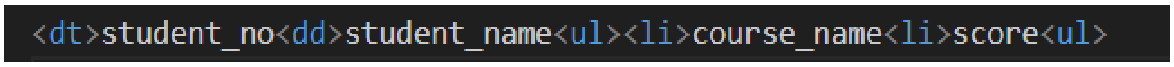
The use of some special characters requires entity references. The W3C specifies that there are five predefined entity references in XML. For example, if you put the character "<" directly inside an element, it will cause an error because the parser will treat it as the beginning of a new element. To avoid this error, you need to replace it with the corresponding entity "<".

For an empty element, such as <br></br>, where no content exists in the start and end labels, it can be abbreviated to a single label <br/>.

1.6 XML documents,document structures and DOM trees

1.6.1 XML documents

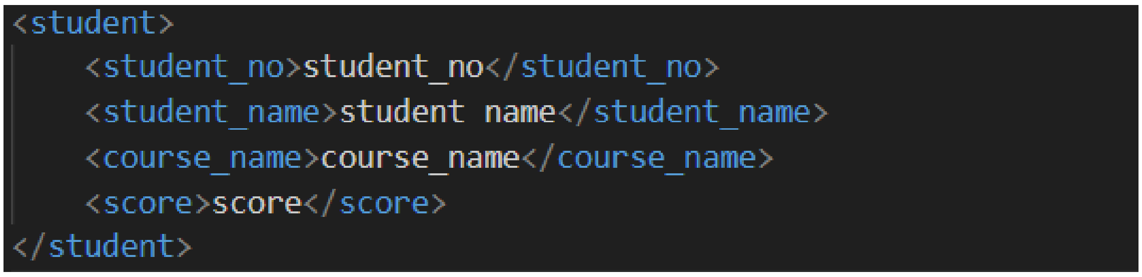
XML documents are made up of elements, which are defined by tags, each of which has a name. Tags come in two forms: the start tag (denoted by the tag name) and the end tag (denoted by/tag name) The start tag and the end tag generally come in pairs, and the content between them is the element. Elements can contain both text content and other elements: elements can only be nested, not overlapped. Some start tags contain attributes that can be treated as child elements of this element, and the value of the attribute can be treated as the text content of the child elements. An XML document has only one root element, which contains everything in the document and represents the entire document, while the other elements represent a component of the document. XML is a meta-markup language, the so-called "meta-markup", that is, users can define tags according to their own needs. For example, to describe a book, you can define tags <book>, <author>, and so on. Any name that satisfies XML naming conventions can be defined as a tag. HTML is a predefined markup language that can only be described using defined tags, such as <html><body>, etc. For example, to describe a student, using HTML can be represented as follows:



```
<dt>student_no<dd>student_name<ul><li>course_name<li>score<ul>
```

Fig. 1.9 “HTML example”

If described in XML, it can be expressed as follows:



```
<student>
    <student_no>student_no</student_no>
    <student_name>student name</student_name>
    <course_name>course_name</course_name>
    <score>score</score>
</student>
```

Fig. 1.10 “XML example”

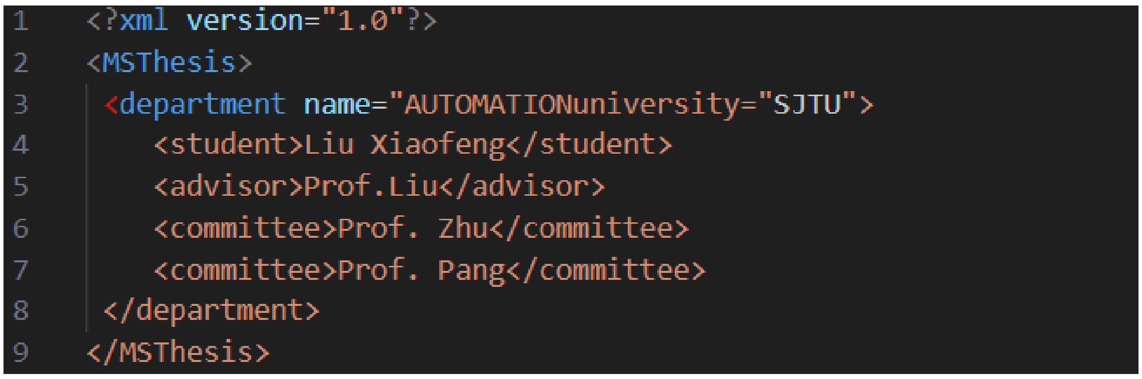
It is easy to see from the examples above that XML documents are semantic and structured, whereas HTML is ambiguous. Just looking at the markup of the XML document, we can infer that this is probably describing a student, but looking at the markup of the HTML alone, we can't find this information. This is an important indication that XML is more flexible than HTML.

XML documents are made up of DTDS and XML text. Both DTDS and DTDS in SGML are used to define tags and rules for their use. Put simply, DTDS are about how XML text should be organized. For example, a DTD may indicate that a <student> must have a child tag <student no> and <student name>, and that <course name> and <socre> may or may not have one. However, not all XML text has its corresponding DTD, which makes the classification of XML documents complicated. This paper is based on this.

1.6.2 Document structures

An XML document generally consists of four parts: an XML declaration, an XML element, processing instructions, and comments. An XML declaration is used to declare that the document is an XML document; XML element is the basic unit of XML file content, an element contains a start tag, an end tag and the data content between the tags, the element can also be nested elements, to achieve circular nesting, the outermost element is called the root element, an XML document can only have one root element; Processing instructions are some imperative statements contained in XML documents to tell XML to process some information or perform certain actions. Comments are character data in an XML file that is used for interpretation and is not processed by the XML processor.

XML is a markup language with strict syntax requirements and strict syntax restrictions. A well-formed XML document is an XML document that satisfies all syntax restrictions. Here is an example of a well-formed XML document.



```
<?xml version="1.0"?>
<MSThesis>
 <department name="AUTOMATIONuniversity="SJTU">
    <student>Liu Xiaofeng</student>
    <advisor>Prof.Liu</advisor>
    <committee>Prof. Zhu</committee>
    <committee>Prof. Pang</committee>
 </department>
</MSThesis>
```

Fig. 1.11 “XML structure diagram”

1.6.3 DOM tree

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It provides a structured representation of documents that can change the content and presentation of documents. DOM connects web pages to scripts and other programming languages. Script developers can manipulate, manipulate, and create dynamic XML elements (tags) through document object properties, methods, and events. Just as the tags on a web page are nested in layers, with the outermost layer being <HTML>, so is the document object model, but it is usually understood as the shape of a tree, which is often referred to as the DOM tree. The root is the window or document object, which corresponds to the outer perimeter of the outermost label and represents the entire document. Below the root of the tree (the tree is usually drawn upside down, as in a genealogy or family tree, where the root is the only common ancestor) are objects at the child level, which also have their own child objects, and all objects except the root object have their own parent object and the child objects of the same object are brothers. For example, the XML document describing students above, its root node is <student>, it has four sub-nodes <student no><student name>, <course>, <score> are the sub-objects of the root node, the four sub-nodes are brothers to each other.

The W3C has developed a set of standards for the document object model and is working on more. Modern browsers support some of these standards, and there are several techniques available to parse XML into documents into DOM tree models.

1.7 XML key terms and concept

To understand the purpose and function of XML, it is necessary to have a preliminary understanding of the parts that make up a document.

1. XML declaration --< xml version="1.0"? >

Usually appearing in the first line of the document, it defines the version number (1.0) of the XML and the character encoding used (optional), the default being ISO-8859-1 = Latin-1/ Western European character set. This declaration is required in all XML documents.

2.Root element -- The topmost element in the document, which is the parent of all other elements in the document. The root node is unique within a particular XML document. In the above example, the root element is <MSThsis>.

3. Children and parents (nested elements) -- The elements inside the tag are called children, and the elements outside are called parents. This parent-child structure is called the nested element hierarchy. In the example above, the department element is a child of the MSThesis element and a parent of the student, advisor, and committee child elements. A child element without children is called a leaf-level child.

4. Stats -- When defining an element, it is sometimes necessary to specify some unique attribute values associated with it. Each attribute has a name and an associated value. Attributes of the same element cannot be repeated. In the example above, the element department has two attributes, name and university, whose values are AUTOMATION and SJTU.

5. content -- The part between an element's opening tag and its closing tag is called the element's content, which can be text, child elements, or both. The content of a leaf child contains only one value. In the above example, the department element is part of the content of the root element MSThesis. The contents of the three leaf sublevels, student, advisor, and committee, are their respective values.

These terms and concepts will be used frequently in the following discussion, and they are also basic requirements for XML document development at any level of complexity.

1.8 Application of XML

In general, XML applications fall into four categories:

1. When the customer needs to interact with different data source;

the data may come from different databases, all of which have their own different complex formats. But customers interact with these databases in only one standard language, XML. Because of its customizability and extensibility, XML is sufficient to express various types of data. After receiving the data, the customer can process it or transfer it between different databases. In such applications, XML solves the problem of a uniform interface to data. However, unlike other data transfer standards, XML does not define the specific specification of the data in the data file, but attaches tags to the data to express the logical structure and meaning of the data. This makes XML a specification that programs can automatically understand.

2. Used to distribute a large number of computing loads on the client;

Customers can choose and craft different applications to process the data according to their needs, and the server only needs to send out the same XML file. Using XML puts the initiative of processing the data in the hands of the client, and all the server does is encapsulate the data in the XML file as fully and accurately as possible. The self-explanatory nature of XML enables the client to understand the logical structure and meaning of the data while receiving the data, which makes the widespread and general distributed computing possible.

3. It is used to present the same data to different users in different faces;

This application can also be seen in the above example. It is similar to the same script, but we can use TV series, movies, plays, cartoons and other forms of expression. This application will pave the way for the development of personalized and stylized web user interfaces.

4. Applies to network proxy;

Some customers acquire data not for direct use but to organize their own databases as needed. For example, the Ministry of Education establishes a huge question bank, takes out a number of questions in the question bank during the exam, and then packages the test paper into an XML file, which passes through a filter in each school, filters out all the answers and sends them to each examinee, and the unfiltered content can be directly sent to the teacher. With just a few small programs, the same XMI file can be turned into multiple files and sent to different users.

1.9 Research significance

Computer technology has gradually become an indispensable part of People's Daily life and work. Millions of people are using computers to generate and distribute information all the time. In the information generation stage, people mainly use Microsoft Office Word and other word processing software to edit documents. Almost everyone who can use a computer can use these software, and it is easy to write beautiful and elegant documents. It is estimated that more than 40 billion documents already exist, and billions more are added every year.

In the stage of information release, WEB technology develops rapidly, and has a tendency to replace paper, television and other media. People spend time almost every day browsing WEB pages to get information. XML is more and more widely used, and its flexibility and convenience are better than HTML, helping developers to provide a faster way in their work, and the focus of the generation stage of code is how easy to write, how easy to read and how to write correctly. Therefore, a problem arises: how to write XML documents correctly and conveniently, quickly and efficiently.

Developers write a lot of data transfer documents and look at a lot of XML documents every day. Developers often need to extract information from XML documents quickly and efficiently. However, there are few tools for verifying and formatting XML documents, and the compatibility and function of these tools are not very good. If there is a simple way for ordinary users to use the tool quickly without too much knowledge of the use of the tool, then it will be welcomed by users.

1.10 The relevance of XML development

Human society has entered the information age, and information technology has penetrated into every aspect of people's lives. Information technology has greatly accelerated the spread of knowledge, and people's ability to acquire knowledge has also been greatly improved. People are exposed to information all the time, consciously or unconsciously. Especially in the past decade or so, with the widespread application of the Internet, the data on the Internet has grown rapidly. According to recent statistics, the number of global web servers has exceeded 1.7 billion, and the web pages involved are basically in the form of static text. Most of the written information we come into contact with daily exists in the form of printed or electronic text. Today, the development of information technology is getting faster and faster, and the degree of electronic information is getting higher and higher. Under such a development situation, the current number of electronic texts is incalculable, which makes the effective management of these texts particularly important. In 1998, W3C launched the XML (eXtensible Markup Language) data format standard. Due to the semi-structured data characteristics of XML text, XML text has been widely used in fields such as data exchange and data storage. In recent years, databases specifically designed to store XML text data and indexing technologies related to XML have also begun to receive widespread attention from researchers. An increasing amount of data on the Internet is in the form of XML text. People are exposed directly or indirectly to more and more data in the form of XML text. XML text has become one of the most important texts in the information society.

Faced with such large-scale data, users are often prone to a state of "data explosion but lack of knowledge". Because what users really need is only a small part of the data, and users are not interested in the vast majority of other data. Or the knowledge contained in large amounts of data cannot be discovered manually. Faced with such a huge amount of data, how to discover valuable knowledge or information is a very difficult problem. It is against this background that data mining developed rapidly, and its emergence solved this problem to a certain extent. Data mining is the process of finding useful knowledge hidden in a large amount of noisy, incomplete, and fuzzy practical application data, and this knowledge is often unknown to people in advance . However, traditional data mining mainly focuses on structured relational database data or transaction database data, while there is little research on semi-structured XML data. In order to mine these XML data, the data must first be managed effectively. How to effectively manage these text data in the form of XML is a major problem in the current fields of information science and computer science. In traditional text management methods, text management work is completed manually by experts in specific fields or managers, which requires a lot of time and energy and is very inefficient. Moreover, as the number of texts further increases, the effect of this manual management method will become worse and worse, making it difficult to effectively meet the needs of actual management work. Therefore, a new and effective method for efficient text management is urgently needed. As a result, technologies such as text filtering and automatic text classification emerged. They can free users from tedious text management, make text management more systematic, and improve the efficiency of text management. Therefore, automatic text classification has received widespread attention since it was first proposed, and research on it has also made certain progress.

Conclusion

From these parts, we understand the development background of XML, the application of XML, the basic characteristics of XML, and the development prospect of XML; The design of XML verification tool needs four functions, such as XML file import, XML file export, XML file verification, XML file formatting.

The generation and improvement of XML document validation shows great potential in the development process. This combination of XML automatic validation techniques provides the benefits of code understanding, refactoring, and document generation. However, challenges related to the quality and reliability of calibration data need to be addressed. XML verification module is to promote the intelligent automation development programming technology, to promote the developer's high efficiency, high quality work.

CHAPTER 2
Requirements for the code generation system and improvements to the web page module

1.11 Web Design Requirements

1. Home/Overview:

Displays the main functions of the XML file verification tool, namely, XML file import, XML file export, XML file verification, and XML file formatting.

Provides fast file import and file export to quickly read and extract data.

2. XML file import function:

Allow the user to find the file that needs XML verification in the local file of the computer, open the file, and the file will be automatically imported into the XML verification tool.

3. XML document verification function:

The tool verifies the content of the file, automatically identifies the location of the syntax error, and tells the specific error location in text form.

4. XML document formatting function:

The tool can format the content of the imported XML document and display the content of the document in hierarchical arrangement, which helps users to read the content of the file.

5. XML document export function:

Users can extract the content of the verified XML document and download it to the local computer.

6.Content display area:

After the XML file to be verified is imported to the tool, the document content is displayed in the content display area.

7.Result display area:

After the XML file is verified in the tool, the verification result is displayed in the result pane.

8.Mobile adaptation:

Ensure that web tools are mobile friendly and provide a good mobile user experience to improve user efficiency.

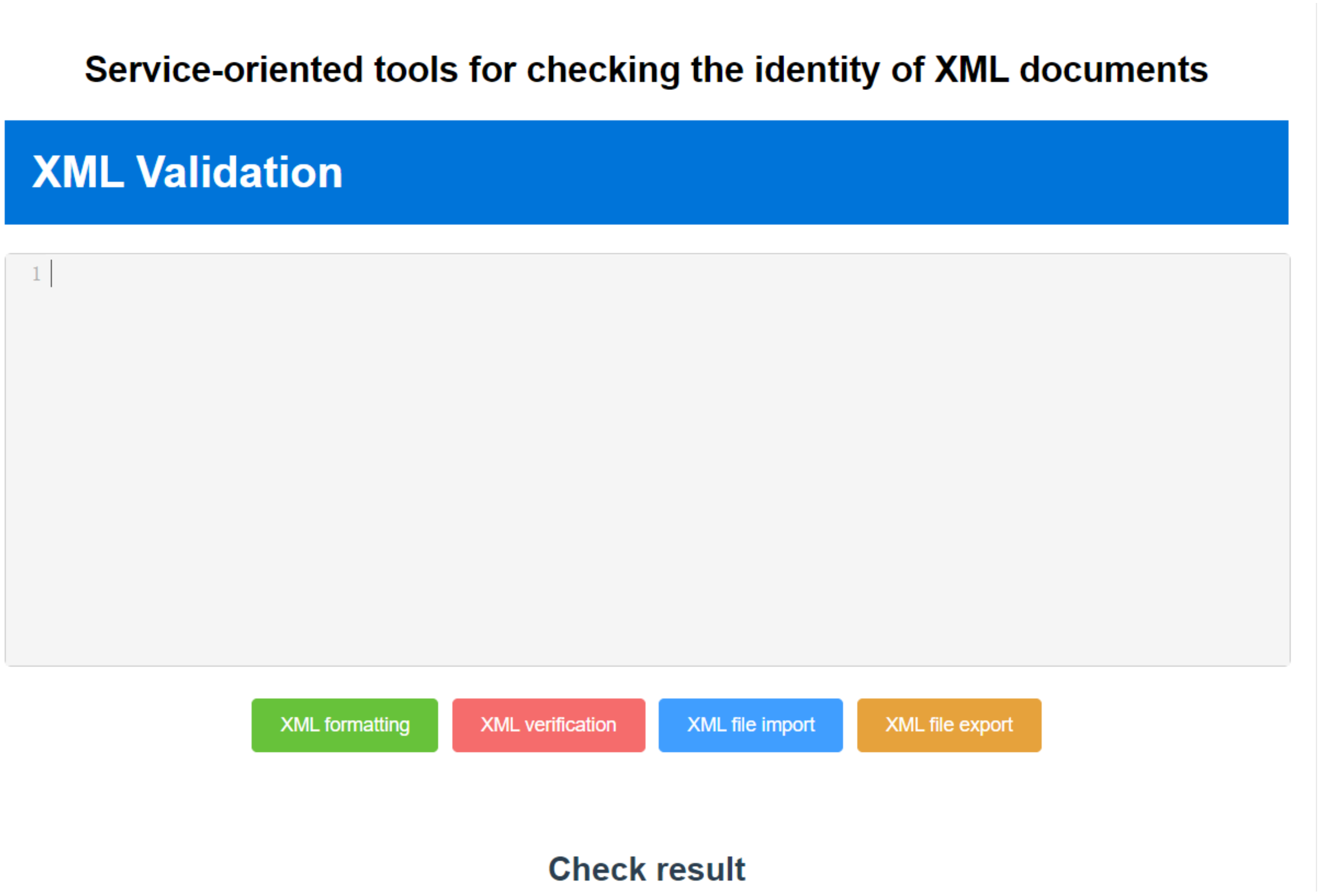


Fig. 2.1 “XML file verification tool detailed introduction and requirements”

1.12 Capabilities for frontend part of web app applications

1.12.1 Functional requirement

User pages:

1. Use reasonable colors, fonts and ICONS to improve the readability and appeal of the page, ensure that the page layout is reasonable, easy to navigate and understand, and provide clear identity and navigation elements.

2. Ensure a good user experience across devices and screen sizes, and use techniques such as media queries to ensure page layouts are adapted to different screen sizes.
3. Provide intuitive navigation system, users can easily find the required information, reduce the learning cost of users to use the tool, so that users can quickly master the method of using the tool.
4. Implement easy-to-understand document content verification results, including file verification and error handling. Provide real-time feedback to ensure users are aware of the validation status of their input files.
5. Effectively present and display file verification results, including verification success, verification failure, etc. Supports formatting of file contents to improve the readability of files.
6. Ensure compatibility in major browsers, including Chrome, Firefox, Safari, Edge, etc. Test and handle differences between browsers.
7. Optimize front-end performance, including page loading speed and resource utilization, ensure that there are no problems such as stutters in the process of user utility tools, and provide users with efficient tools.
8. Do unit testing, integration testing, and user interface testing to ensure front-end code quality, and consider using automated testing tools to improve testing efficiency.

System main:

1. Create a page layout, including the XML file content display area, XML file verification result area, and XML file verification function, to ensure a user-friendly interface.
2. Design your pages with HTML and CSS, ensuring responsive design to suit different devices.
3. The result of XML document verification is displayed on the page, whether it succeeds or fails, and where the syntax error of the verification file is located.

4. Use intuitive navigation and clear function buttons to give users insight into specific areas.
5. Provides the main language interface, can meet the needs of most users.
6. Make sure the app can be adapted to different devices and screen sizes, including mobile, tablet and desktop devices.
7. Provides the format of XML file, you can read XML file more clearly, convenient for users to use.

User side:

1. Consider user file verification requirements, such as XML file verification and XML file formatting requirements.
2. Implement a formatting function that allows users to easily read and modify the information in their XML files.
3. XML file input: Front-end applications should allow users to input XML files and verify their XML file contents.
4. Error handling: The front-end application should properly handle errors and provide the user with information about the specific location of the XML file syntax error.

1.12.2 Non-functional project requirements

1. Non-functional requirements represent all aspects of the quality of the software system, which refers to those aspects related to the operation and performance of the system, such as the loading speed of the website interface, the availability of the website, the security level of the website, the maintainability of the website, etc., without designing specific functional behavior requirements. Incorrect support for non-functional methods can lead to unsatisfactory system operation for users, so these non-functional project requirements are critical to ensuring that the system meets user needs and expectations in all aspects.
2. Performance: Front-end applications should have high quality performance requirements.

3. Scalability: The front-end application should have a good program architecture, such as module addition (page addition), the scalability of design changes.
4. Reliability: The front-end application should be able to maintain stable operation for a long time, minimizing failures and errors.
5. Recoverability: The system should be able to quickly return to normal operation in the event of a failure.
6. Compliance: Ensure that the system is designed and operated in accordance with relevant regulations and standards.
7. User interface: The front-end application should have a well-designed interactive and dynamic U1 to get a good user experience.
8. Usability: Front-end applications should be well designed with soft interfaces.

1.13 Detailed Description

1.13.1 Capabilities for frontend part of web app applications:

User pages:

1. There should be an XML file input button on the main page, through which the user can import the XML file that needs to be verified in the computer host into the tool, and the XML file content will be displayed in the content display area.
2. There should be an XML file verification button on the main page, through which the user can verify the imported XML file. The verification result will be displayed in the result display area of the main page, and the parts with syntax errors will be automatically marked in red.
3. There should be an XML file formatting button on the main page, the user can format the imported XML file through it, the formatted XML file content will be hierarchical hierarchical arrangement, which is helpful for users to read and write.
4. There should be an XML file download button on the main page, through which the user can download the verified XML file to the local.

System main:

1. Since the application does not require a complex architecture, it is advisable to use the easiest programming language for web application development, which is called Javascript, in this way it will be possible to implement a website quickly and efficiently.
2. Using the front-end Vue framework in combination with the Javascript programming language allows for the creation of dynamic, scalable Web applications with an architecture that facilitates further development, which will provide more options for changing the product after the initial release.
3. Design the overall architecture of the project and decide whether you need to create a single Vue component or multiple components and the relationships between them. Consider how to manage your application's state, data flow, and event handling.
4. Developing the front-end part of the application using AI to generate code will have the potential to speed up the software development process, due to the fact that developers will eliminate the part of developing a complete architecture from scratch and will only gradually supplement the XML document validation part of the generated code.

User side:

1. User interface: A user-friendly interface and easy to understand, it is possible for users to easily use the tool through the display of the home page and obtain the necessary information through simple interaction.
2. Data display: Structured display of data, such as the data of the XML document uploaded from the local computer is displayed in the content display area of the home page, which facilitates users to view the content of the imported file, and provides a clear understanding and comfortable interaction after completing a certain amount of work in the home page.
3. Data entry: Entering and updating data allows users to operate in the system for extended periods of time without receiving errors or delayed responses.
4. Error handling: Providing a correct error description or displaying the location information of a syntax error under loading will make it possible for the user to understand that he continues to do everything correctly. When the verification process

is complete, the result display area will be green, and the user will know the structure and progress of the verification.

5. Data export: After verifying a series of documents, users can download the documents to a local computer.

1.13.2 Requirements for frontend part of web app applications

1. User experience: The front end should provide a good user experience, including an easy-to-use interface, clear function buttons, and a responsive design to provide a consistent experience across a variety of devices and screen sizes.
2. Performance: The front end should optimize performance as much as possible to ensure page loading speed and response time.
3. Compatibility: The front end should support different browsers and operating systems to ensure compatibility and stability. So you can use modern CSS and JavaScript frameworks to help achieve this, and take advantage of the Vue progressive framework, which is suitable for building modern user interfaces.
4. Scalability: The front end should have good scalability in order to adapt to the growth and changes of the application. Modular design, reusable components, and front-end frameworks can be used to help achieve this goal. The tool's VUE architecture not only adopts the best build principles and rules, but also ensures that the system is well scalable. This means that the system has the flexibility to support, change, and remove certain parts of the software without introducing problems.
5. Code quality: The front-end part should follow good coding practices, including following coding specifications, using version control, testing code and debugging tools to ensure the quality and maintainability of the code.

Interface:

Since the application is a service tool for the user, the interface should be clear and quickly understandable to the user. In order to make the system as comfortable as possible, there is a need to reduce: the amount of information in the form of buttons and information bars, as well as excessive colored sections on the page. Its creation process is as follows:

- Correct part and function button area.
- On the navigation panel, right above is the name of the tool.

- In the middle, a blank box with the file contents of the XML file that the user imported locally.
- The four color buttons below the content box are function buttons, including XML file formatting, XML file verification, XML file import and XML file export functions from left to right.
- There should be a dynamically generated result display area below the function button, showing the result of the user imported XML file verification.

Design:

Web application design has many advantages that make it the solution of choice in many areas. Web applications can be accessed cross-platform, and by accessing Web applications through a browser, users can use various devices such as computers, tablets, and mobile phones without installing additional software to achieve cross-platform access. Not only that, Web applications are relatively easy to update and maintain because changes are made only on the server side and users do not have to manually update the application, which reduces maintenance costs. Users can access Web applications directly through the browser, without the cumbersome installation process, improving user convenience.

With Web applications, developers can update the content of the application in real time, ensuring that users always have access to the latest version and data. All data is stored on remote servers, making data backup, recovery, and security management more centralized and convenient. Web applications allow multiple users to collaborate simultaneously, share information and resources, and help teams work together. Users do not need to manually upgrade the application because all updates are done on the server side, thus reducing the burden on users. With simple Web links, Web applications can easily integrate with other applications or services for more functionality and scalability. Web applications can provide real-time interaction, such as online chat, collaborative editing and other functions to enhance user experience. Because Web applications are based on the Internet, users can access applications from anywhere in the world, enabling global coverage and service.

It can be done with the JavaScript programming language and an architecture called Vue, which will increase responsiveness and

Dynamic applications, i.e. web pages will respond to the user's actions and change immediately according to his needs. In addition, for greater necessity, it will be possible to use the Element UI, a CSS framework that provides front-end solutions that have been developed, etc.

Conclusion

In this section, I focus on explaining the key design requirements for the web application I want to develop. I selected and investigated all aspects of the WEB in advance, divided the main part of the program into modules such as content display area, function button area and result display area, and finally released the display of the final product.

Through online and offline research, I have learned that WEB page development is a complex task involving design, coding, and implementation. When conducting the analysis, we need to consider several aspects, including user experience, functional requirements, technology choices, and so on. In terms of user experience, a successful WEB page should be able to provide a clean interface design and have good response speed. Through reasonable layout and intuitive operation, users can easily find the required information and complete various interactive actions. We also need to pay attention to page load times and compatibility issues to ensure that different devices and browsers can display properly. In terms of functional requirements, we need to determine the functional modules to be implemented according to the project requirements, and carry out detailed planning and design.

When analyzing WEB page development, it is necessary to fully consider factors such as user experience, functional requirements and technical choices, and flexibly use relevant knowledge and tools to achieve the expected goals. Only through in-depth analysis and comprehensive thinking can users be provided with high-quality XML document verification tools that meet their expectations..

CHAPTER 2
The design idea of XML file verification tool

1.14 Development idea

1.14.1 Define project objectives

First, define the goals and requirements of the project. In this project, the goal is to create an XML document validation tool that allows users to enter, Format and validate XML content, and download and export XML files after the function is finished.

1.14.2 Select technology stack

Determine the technology stack required for the XML file validation tool project, including a front-end framework (such as Vue), a text editor library (such as CodeMirror), a CSS framework (such as Bootstrap or Element UI), and any other libraries and tools that may be needed.

1.14.3 Conceptual user interface

Design the user interface, including page layout, component layout, buttons, input fields, text editors, etc. Ensure that the user interface is intuitive, easy to use, and meets the needs of the project.

1.14.4 Project structure

Design the overall structure of the project. Decide whether you want to create a single Vue component or multiple components, and the relationships between them. Consider how to manage your application's state, data flow, and event handling.

1.14.5 Design function module

Implement the main functions of the XML editor, including entering, formatting, and validating XML content. This may require the use of an external library, such as xml-formatter, to accomplish some tasks; Implement the main functions of the XML editor, including entering, formatting, and validating XML content.

1.15 Development framework

1.15.1 Vue.js

Vue is a popular JavaScript framework for building user interfaces. It is a progressive framework that can be gradually applied to existing projects, or it can be used as a complete development tool to create new single-page applications.

Vue.js has the following features:

1. Lightweight and easy to learn: Vue.js is a lightweight framework that is easy to use and suitable for beginners to get started quickly.
2. Reactive data binding: Vue.js provides reactive data binding capabilities that make it easy to implement data-driven views. When the data changes, the view is automatically updated, and vice versa.
3. Componentized system: Vue.js adopts the idea of componentization, which can split a complex interface into multiple components, and realize a complex interface by combining different components.
4. Flexible template syntax: Vue.js template syntax is very flexible, supporting expressions, conditional statements, loops, etc., you can easily build dynamic interfaces.
5. Instructions and bindings: Vue.js provides a rich syntax of instructions and bindings that make it easy to manipulate the DOM, control data flow, and more.
6. Plug-in system: Vue.js provides a powerful plug-in system, you can install plug-ins to extend the functionality of the framework.
7. Strong scalability: Vue.js is very scalable and can be customized and expanded according to actual needs.

8. Cross-platform and cross-browser compatibility: Vue.js supports multiple platforms and browsers, including desktop, mobile, etc., making it easy to achieve cross-platform and cross-browser development.

Vue.js is a powerful, easy to learn and use JavaScript framework for building all types of front-end applications. Vue can be integrated with other libraries and technologies, such as Vue Router and Vuex, to build more complex applications. It also supports server-side rendering, making applications better for search engine optimization and performance optimization.

Overall, Vue is a flexible, efficient, and easy-to-use JavaScript framework for building modern user interfaces.

1.15.2 element-ui

Element-UI is a set of component libraries based on Vue.js framework, which provides rich UI components, including buttons, tables, dialogs, forms, menus, navigation, etc., which can help developers quickly build beautiful and functional web pages or applications. Element-UI is designed to be simple and easy to use, while providing rich documentation and examples.

Element-UI has the following features:

1. Easy to use: Element-UI provides an easy-to-use API that allows developers to quickly get started with component libraries.
2. Rich components: Element-UI provides a large number of components, including forms, navigation, notifications, boxes, buttons, tables, prompt boxes, etc., which can meet the needs of most developers.
3. Responsive Design: Element-UI supports responsive layout, which automatically adjusts the layout and style of components according to different screen sizes and device types.
4. Good compatibility: Element-UI supports a variety of browsers and operating systems, including Chrome, Firefox, Safari, Edge, etc., and has good support for mobile devices.

5. Active community: Element-UI has a large community where developers can ask for help, share experiences, discuss techniques, and more.
6. Customization: Element-UI supports customization, which can customize the style and behavior of components according to the needs of developers.
7. Detailed documentation: Element-UI documentation is very detailed, providing a wealth of examples and instructions to help developers better understand and use the component library.

The Element UI can be used flexibly, allowing you to bring in the components you need, or you can bring them in as a whole. Its seamless integration with Vue.js allows it to be combined and interact by way of components to quickly build complex user interfaces.

Overall, the Element UI is a powerful, easy-to-use library of front-end UI components for building modern Web applications. It reduces development time, increases development efficiency, and provides a consistent and aesthetically pleasing user interface.Development environment and tools.

1.15.3 Node.js

Node.js is a JavaScript runtime environment based on the Chrome V8 engine for the development of server-side and web applications. It provides an event-driven, non-blocking I/O model that can handle a large number of concurrent connections, making it easier to build high-performance networking applications.

Here are some of the features of Node.js:

1. Asynchronous non-blocking I/O model: Node.js adopts event-driven, non-blocking I/O model, which can effectively handle a large number of concurrent requests and improve the throughput and response speed of the system.
2. Single thread: Node.js adopts a single thread model and realizes efficient request processing through the event loop mechanism. Although single-threading may not be as good as multi-threading for compute-intensive tasks, it is effective in network I/O intensive scenarios.

3. Lightweight and efficient: Because it runs on a V8 engine, Node.js has a small memory footprint and a fast startup time. In addition, it uses an event-based architecture that avoids the overhead of thread creation and destruction in the traditional server model.
4. Modularized development: Node.js supports CommonJS module specifications, making it easy for developers to organize code in a modular way and reuse existing modules. This helps improve the maintainability and reusability of your code.
5. Rich ecosystem: Node.js has a large package manager (npm) and open source community, offering numerous third-party modules and tools to quickly build various types of applications.
6. Cross-platform: Node.js can run on multiple operating systems, including Windows, MacOS, and Linux.

Node.js is widely used to build Web servers, real-time communication applications, microservice architectures, command line tools, and more. It can also be used with front-end frameworks (such as Vue.js, React) to share the same set of JavaScript code for full-stack development.

Overall, Node.js is an efficient, lightweight, and scalable JavaScript runtime environment for building high-performance web applications and server-side applications. It has advantages in terms of development efficiency and performance, and has a strong ecosystem of open source modules.

1.15.4 Npm

npm (Node Package Manager) is a JavaScript community driven package manager for installing, managing, and running various packages and modules in Node.js programs. It is one of the core components of Node.js and helps developers easily extend JavaScript applications with different libraries and tools. With npm, developers can easily install third-party libraries and tools to use in their applications. It supports installing packages from the public npm registry, as well as from the local file system. In addition, npm provides a number of useful command-line tools such as npm install, npm update, npm uninstall, etc., for managing dependencies of packages and modules. In addition to being used to install packages and modules, npm also provides many other features such as dependency checking, version control, development tools, and more.

Here are some of the key features and functions of npm:

1. Package management tool: npm allows developers to easily install, update, and remove JavaScript packages. These packages can include libraries, frameworks, tools, and applications.
2. Packages and modules: The core concept of npm is package. Each package contains one or more JavaScript modules, as well as a package.json file that contains information about the package, such as name, version, author, dependencies, and so on.
3. Dependency management: npm can effectively manage project dependencies. With the package.json file, you can specify the various dependencies required by your project, and then run the npm install command to install these dependencies.
4. Global commands: npm also allows you to install global command line tools so that you can run them in your terminal. These tools are commonly used to perform a variety of development tasks.
5. Version control: npm uses Semantic Versioning specifications to manage package versions. This gives developers a clear idea of whether the new version has backwards-compatible changes.
6. Registry: npm uses a central registry to store and distribute packages. Developers can publish their own packages to the npm registry or download dependency packages from it.
7. Script execution: In package.json, you can define custom scripts to simplify common tasks in your project, such as build, test, and deploy. You can then use the npm run command to execute the scripts.

8. Ecosystem: Since npm is part of the Node.js ecosystem, it integrates very well with a large number of other tools and frameworks such as Webpack, Babel, React, etc.

Overall, npm is an important part of the JavaScript ecosystem, providing developers with a powerful tool to manage the dependencies of JavaScript packages and projects, making development easier and more efficient.

1.15.5 Development tool—Visual Studio Code

Visual Studio Code is a lightweight but powerful code editor developed by Microsoft that supports multiple programming languages, including but not limited to JavaScript, TypeScript, HTML, CSS, Python, C#, Java, and more. Visual Studio Code offers many useful tools and plug-ins such as code highlighting, auto-completion, debugging, Git integration, and more, making it an ideal choice for developers and designers.

Here are some of the main features and functions of Visual Studio Code:

1. Cross-platform: Visual Studio Code runs on Windows, macOS, and Linux operating systems, so developers can maintain a consistent development experience across different platforms.
2. Lightweight: VS Code is a lightweight editor that starts quickly and takes up less system resources, which makes it the tool of choice for many developers, especially for developing Web applications and other lightweight projects.
3. Strong extension support: VS Code offers a wide range of customization and functional extension options through Extensions. Developers can install extensions to add support for a variety of programming languages, frameworks, tools, and technologies to meet their specific needs.
4. Intelligent Code completion: VS Code has a powerful code completion function, supports a variety of programming languages, and intelligent prompts according to the context and the syntax of the code base.

5. Built-in debugger: VS Code integrates a debugger to support multiple programming languages. This allows developers to easily debug their applications, step through code and view variables and stack traces.
6. Version Control integration: VS Code supports various version control systems, such as Git, providing an intuitive user interface that makes it easy to manage and commit code changes.
7. Terminal integration: VS Code has a built-in terminal window that allows developers to execute commands in the editor without switching to an external terminal.
8. Task automation: VS Code supports task automation by configuring tasks to automate common build, test, and deployment operations.
9. Integrated terminal: VS Code has an integrated terminal window built in so that developers can run commands directly in the editor without switching to an external terminal.
10. Rich community support: Visual Studio Code has a large and active community, which means that there are plenty of extensions, themes, and resources available to developers to meet their different needs.

In summary, Visual Studio Code is a powerful and flexible integrated development environment for multiple programming languages and project types, with an excellent extended ecosystem and community support, and is the tool of choice for many developers and teams.

1.16 Core function design

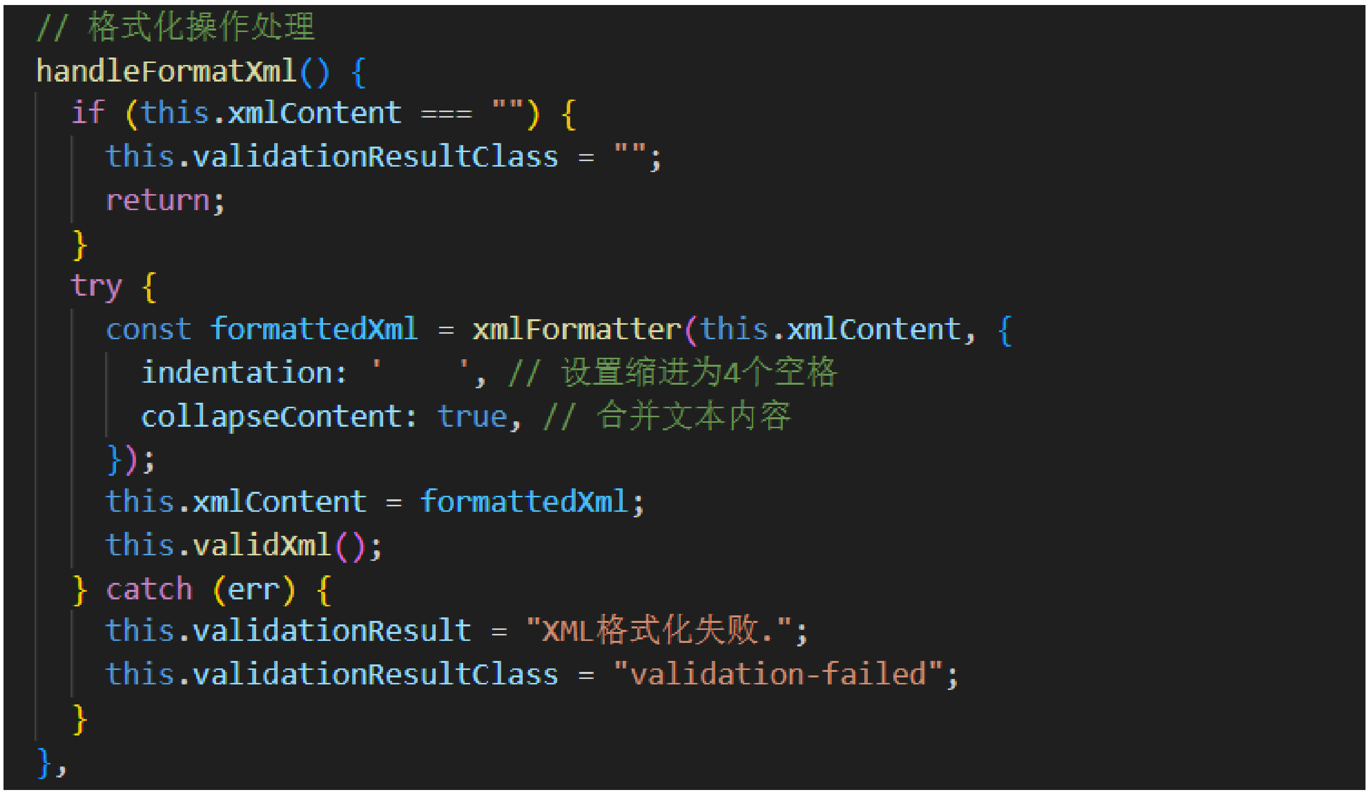
1.16.1 XML content formatting

The handleFormatXml method is used to format the XML content. Inside the method, it uses the external library XMl-formatter to format the XML content. Here is a detailed explanation of the feature:

1. User trigger: The handleFormatXml method is called when the user clicks the button with the "XML Formatting "label.
2. Input check: First, the method checks whether xmlContent is empty. If xmlContent is empty, it does not perform a formatting operation, but simply returns.
3. Formatting process: If xmlContent is not empty, it attempts to format the XML content. Formatting is performed by the xml-formatter library. In doing so, the library uses the specified options (indentation, text content merging, and so on) to beautify the XML content.
4. Update XML content: Once XML content is formatted, xmlContent is updated to the formatted content.
5. Validation: Any exceptions will be caught and handled during the formatting operation. If formatting fails, validationResult will be set to "XML formatting failed." and validationResultClass will be set to "validation-failed" to display the appropriate message and style on the user interface.
6. Finished: Finally, the method ends and the user can see the formatted XML content or error message on the interface.

This feature allows the user to beautifully format the input XML content for easier understanding and editing. By setting appropriate indents and merging text content, XML can be rendered more structured and readable. If there is an error in the formatting operation, the user will get the appropriate feedback to indicate that the formatting failed.

The following is the core code：



```
// 格式化操作处理
handleFormatXml() {
  if (this.xmlContent === "") {
    this.validationResultClass = "";
    return;
  }
  try {
    const formattedXml = xmlFormatter(this.xmlContent, {
      indentation: '    ', // 设置缩进为4个空格
      collapseContent: true, // 合并文本内容
    });
    this.xmlContent = formattedXml;
    this.validXml();
  } catch (err) {
    this.validationResult = "XML格式化失败.";
    this.validationResultClass = "validation-failed";
  }
},
```

Fig. 2.1. “XML file content formatting code”

1.16.2 XML automatic check

The handleValidateXml method is used to verify XML content. Here is a detailed explanation of this feature:

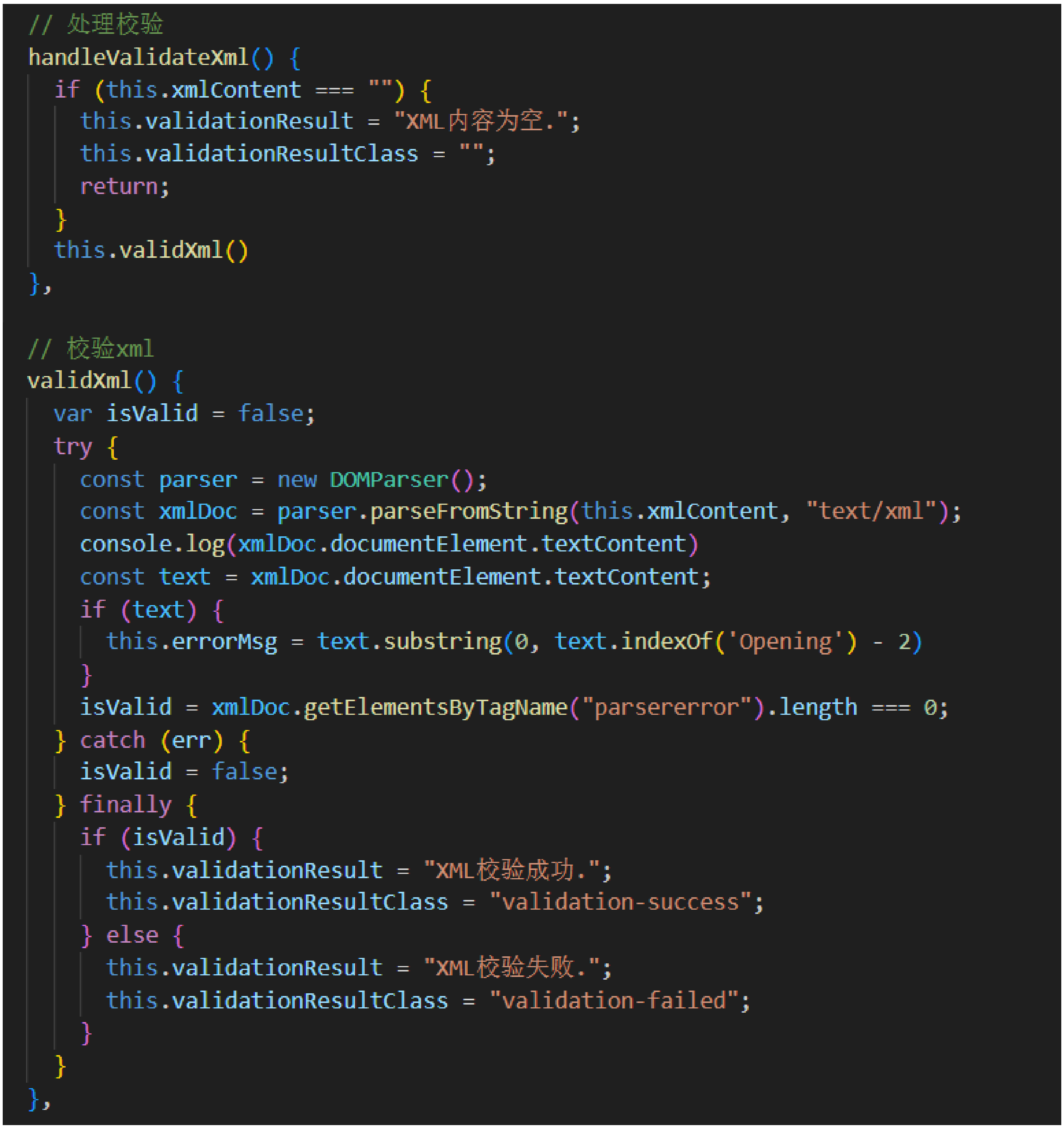
1. User trigger: The handleValidateXml method is called when the user clicks the button with the "XML validation "label.
2. Input check: First, the method checks whether xmlContent is empty. If xmlContent is empty, it sets the validationResult to "XML content is empty." and clears any previous validation results. After that, it returns without further validation.
3. Validation process: If xmlContent is not empty, it attempts to validate the XML content.
4. Using DOMParser: During the validation process, it creates an instance of DOMParser and parses the XML content into a DOM document using the parseFromString method of that instance. This allows it to check the validity of the XML.
5. Check the verification result: Method checks whether there is an element named "parsererror". If such an element is present, the XML has a syntax

error. In this case, it sets validationResult to "XML validation failed." and validationResultClass to "Validation-failed" to display the appropriate message and style on the user interface.

6. Extract error message: If there is a syntax error, the method also attempts to extract the error message from the error message and store it in the errorMsg variable for display.
7. Verification success: If the validation passes without syntax errors, it sets the validationResult to "XML validation successful." and the validationResultClass to "Validation-Success" to display a successful message and style on the user interface.
8. Finished: The method is complete and returns. The verification result will be displayed on the user interface.

This feature allows users to validate the XML content they enter to determine if it is a valid XML document. If the XML has syntax errors, the user will see the appropriate error message, and if the XML is valid, the user will see a success message. This is useful for ensuring the correctness and integrity of the XML.

The following is the core code：



```javascript
// 处理校验
handleValidateXml() {
  if (this.xmlContent === "") {
    this.validationResult = "XML内容为空.";
    this.validationResultClass = "";
    return;
  }
  this.validXml()
},

// 校验xml
validXml() {
  var isValid = false;
  try {
    const parser = new DOMParser();
    const xmlDoc = parser.parseFromString(this.xmlContent, "text/xml");
    console.log(xmlDoc.documentElement.textContent)
    const text = xmlDoc.documentElement.textContent;
    if (text) {
      this.errorMsg = text.substring(0, text.indexOf('Opening') - 2)
    }
    isValid = xmlDoc.getElementsByTagName("parsererror").length === 0;
  } catch (err) {
    isValid = false;
  } finally {
    if (isValid) {
      this.validationResult = "XML校验成功.";
      this.validationResultClass = "validation-success";
    } else {
      this.validationResult = "XML校验失败.";
      this.validationResultClass = "validation-failed";
    }
  }
},
```

Fig. 2.2 “XML file automatically verifies code”

1.16.3 XML file import

The file import function is implemented through the following methods: openFileInput, handleFileChange, and handleSelectFile. This set of methods allows the user to import XML content from a file and display it in the interface.

1. openFileInput method:User triggered:

   The openFileInput method is called when the user clicks the Select File button.

Action: This method actually triggers a click event on a file input element (input type="file") to open the file selection dialog so that the user can select the XML file to import.

2. handleFileChange method:User trigger:

The handleFileChange method is automatically called when the user selects a file and changes the value of the input element in the file.

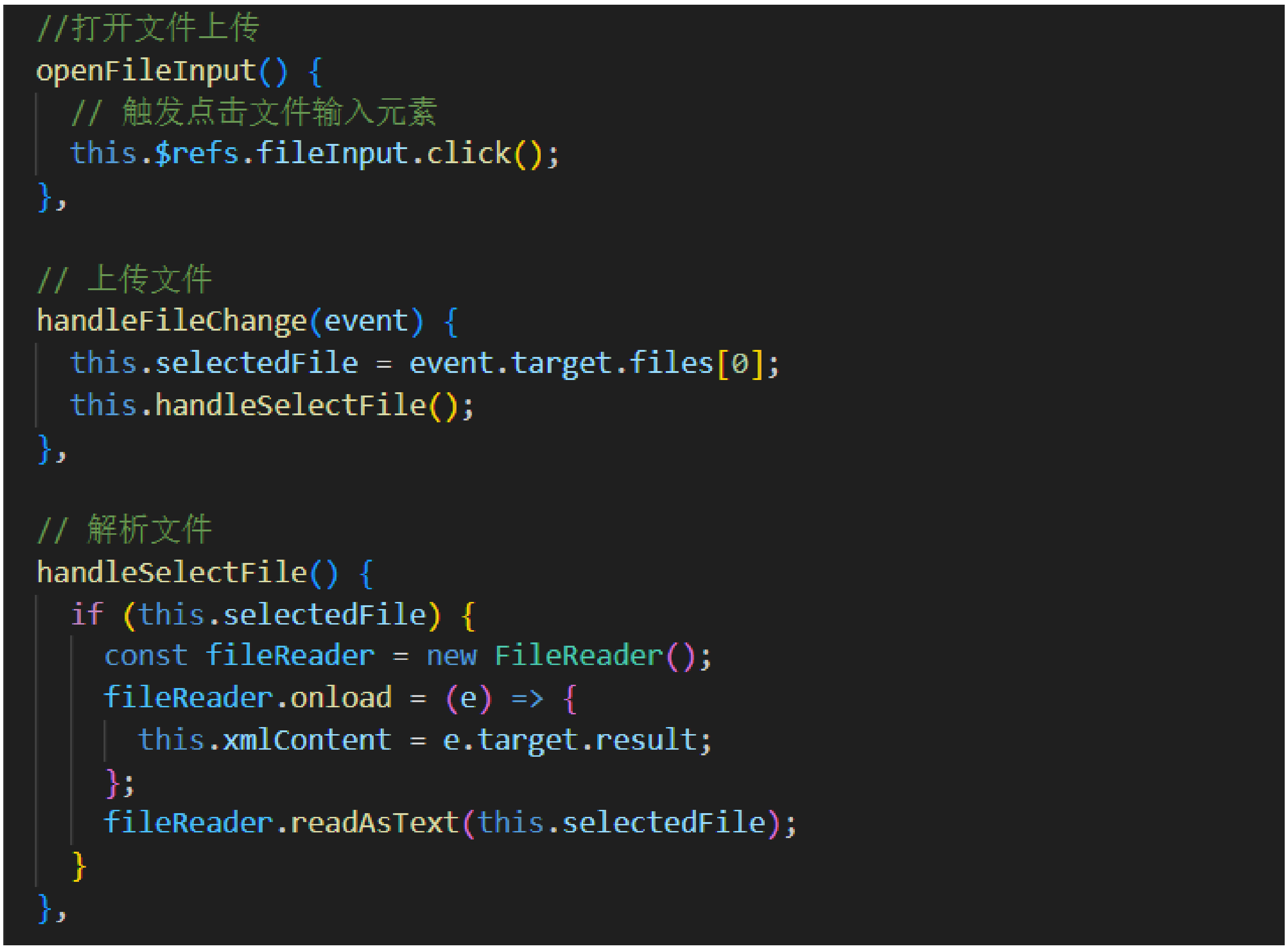
Action: this method takes the file selected by the user and stores it in the this.selectedFile variable. It then calls the handleSelectFile method.

3. handleSelectFile method:

Action: This method uses the FileReader object to read the file selected by the user. It sets up an event handler to listen for file loading events, and when the file loads successfully, the event handler reads the contents from the file and stores them in the this.xmlContent variable. In this way, the content of the file selected by the user is imported into the component and can be displayed on the interface.

Through this series of methods, the user can select an XML file from the local computer and import its contents into the component. This is useful for working with existing XML documents for subsequent formatting, validation, or other operations.

The following is the core code：



```
//打开文件上传
openFileInput() {
  // 触发点击文件输入元素
  this.$refs.fileInput.click();
},

// 上传文件
handleFileChange(event) {
  this.selectedFile = event.target.files[0];
  this.handleSelectFile();
},

// 解析文件
handleSelectFile() {
  if (this.selectedFile) {
    const fileReader = new FileReader();
    fileReader.onload = (e) => {
      this.xmlContent = e.target.result;
    };
    fileReader.readAsText(this.selectedFile);
  }
},
```

Fig. 2.3 “XML file upload code”

1.16.4 XML file export

The file export function is implemented through the handleExportXml method. This method allows the user to export the current XML content as a downloadable file. Here is a detailed explanation of this feature:

1. User trigger: When the user clicks the "XML Export "button, the handleExportXml method is called.

2. Input check: First, the method checks whether xmlContent is empty. If xmlContent is empty, it does not perform the export operation, but simply returns.
3. Export file: If xmlContent is not empty, it performs the file export operation. This is done through the following steps:

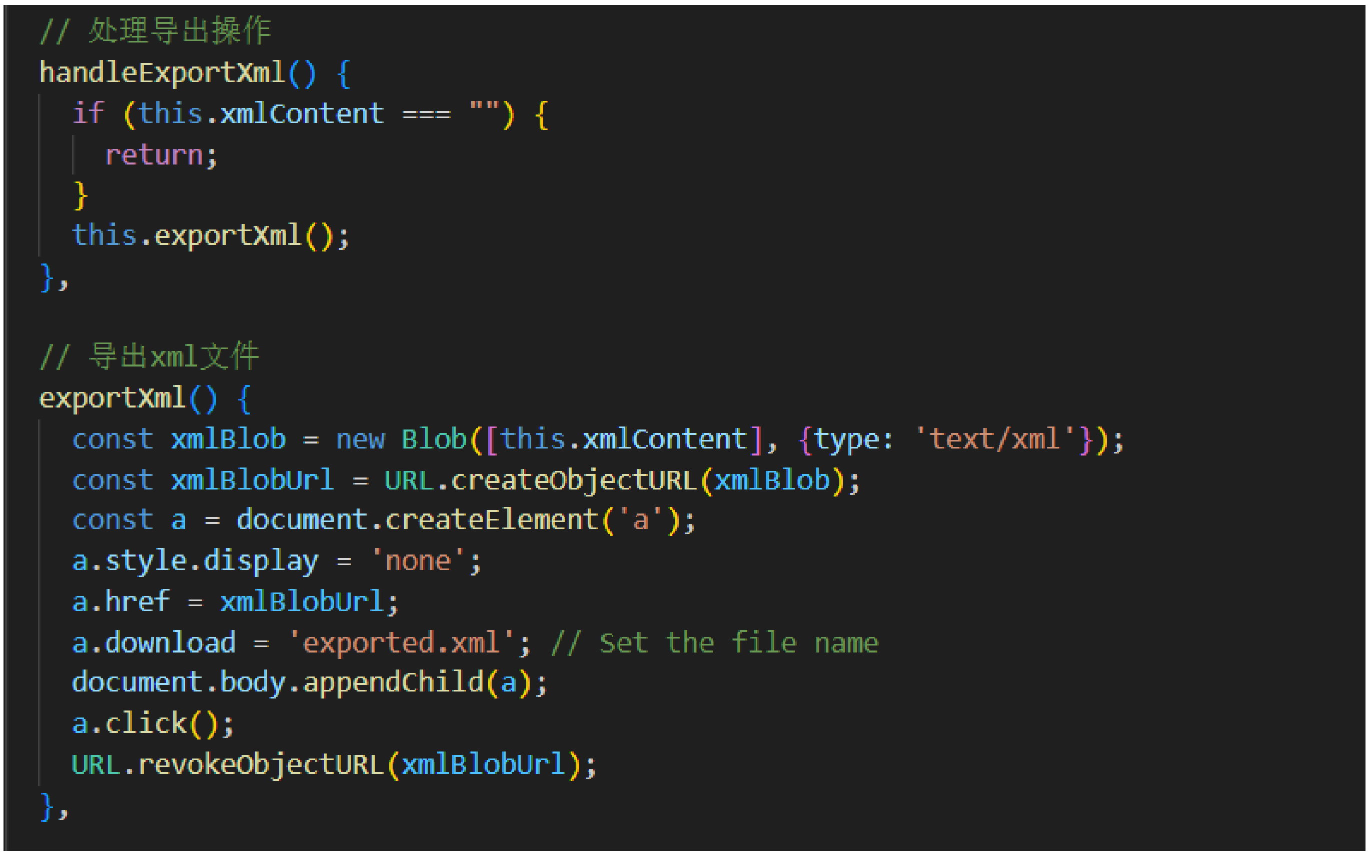
- Creates a Blob object that contains the XML content to be exported.

- Use url.createObjecturl to create a temporary URL to download the file after the user clicks the download link.
- Create an element to simulate the user clicking the download link.
- Set the properties of the download link, including the URL of the link and the name of the download file.
- Add the download link to the page.
- The user's click action is simulated by calling a.click() to trigger the download.
- Finally, use url.revokeObjecturl to free the temporary URL to ensure that the resource is freed.

4. Done: After the method is finished, the user will be able to download the XML file by clicking the download link. If the XML content is empty, no file will be downloaded.

This feature enables users to save the current XML content to their local computer for future use or sharing with others. The user can select the location and name of the file in the file dialog box. This is useful for saving the state of an XML document after editing or processing it.

The following is the core code：



```
// 处理导出操作
handleExportXml() {
  if (this.xmlContent === "") {
    return;
  }
  this.exportXml();
},

// 导出xml文件
exportXml() {
  const xmlBlob = new Blob([this.xmlContent], {type: 'text/xml'});
  const xmlBlobUrl = URL.createObjectURL(xmlBlob);
  const a = document.createElement('a');
  a.style.display = 'none';
  a.href = xmlBlobUrl;
  a.download = 'exported.xml'; // Set the file name
  document.body.appendChild(a);
  a.click();
  URL.revokeObjectURL(xmlBlobUrl);
},
```

Fig. 2.4 “XML file export code”

1.17 Characteristic highlights

1.17.1 Codemirror

codemirror is a popular text editor library commonly used to create code editors and text editors in Web applications. It offers a wealth of features, including syntax highlighting, autocomplete, indentation, collapsing blocks of code, multiple themes and modes, and more. In the provided Vue.js component code, codemirror is used to create a text editor for users to enter and edit XML content. Here is an introduction to the main features of the codemirror component:

1. Syntax highlighting: codemirror provides syntax highlighting for a variety of programming languages and file types. In the provided code, an XML schema is used, which allows elements, attributes, and text in an XML document to be highlighted in different colors, making it easier to read and edit.

2. Code folding: codemirror allows users to fold blocks of code, making it easier to browse and edit large documents. This is useful for working with longer XML documents or documents that contain deeply nested elements.
3. Themes: codemirror allows you to select different themes to change the look of the editor. In the provided code, the "base16-light" theme is used, which gives the editor a bright look.
4. Custom options: You can configure various options for codemirror, including indent size, autofocus, whether to display line numbers, whether to enable autocomplete, and more. In the provided code, some custom configuration is done through the cmOptions object.
5. Rich event support: codemirror provides a variety of events so that you can respond to user input and editing actions. In the provided code, for example, the v-model is used to bind the editor's content bidirectional, and then an event handler is used to respond to user actions such as button clicks.
6. For Vue.js: The provided code is a Vue.js component that integrates codemirror into the Vue.js application. By using it as a Vue.js component, you can more easily manage its state and communicate with the rest of the application.

In summary, codemirror is a feature-rich text editor library that is ideal for implementing text editing capabilities in Web applications, especially those that need to support multiple programming languages and file types. In the provided example, it is used to create an XML document editor so that users can enter, format, and validate XML content.

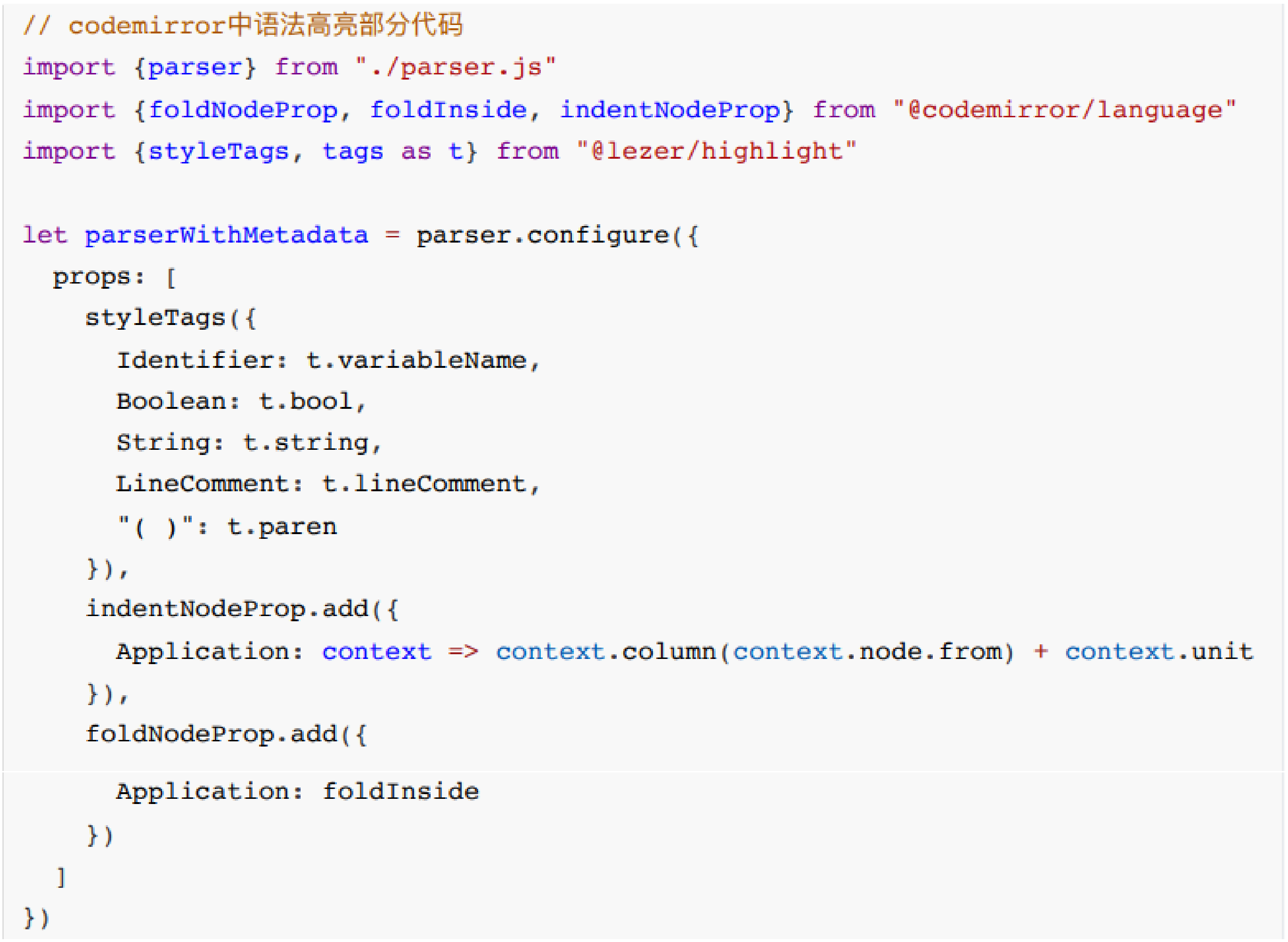
1.17.2 Label highlighting

Syntax highlighting is a text editor feature used to highlight different code elements in different colors or styles based on the syntax structure of the code to improve the readability and ease of use of the code. This is useful for programming and editing various programming languages, markup languages, and configuration files. Among the provided Vue.js components, the codemirror library is used to provide syntax highlighting of XML content.

Here are some key aspects of syntax highlighting:

1. Code structure Highlighting: Syntax highlighting highlights the structure of code by classifying different parts of the code and applying different colors or styles to them. For example, in XML, elements, attributes, comments, and text content are often displayed in different colors.
2. Readability: Syntax highlighting makes code easier to understand because it clearly shows the structure of the code and the relationships between different elements. This helps programmers identify and correct syntax errors more easily.
3. Reduce errors: Syntax highlighting helps reduce errors by highlighting different code elements. Programmers can more easily identify missing tags, attributes, or other syntax problems.
4. Editing support: Syntax highlighting also provides contextual information about code editing, such as autocompletion and code prompts. This further increases developer productivity.
5. Multi-language support: Syntax highlighting can be customized according to the programming language or file type of the code. This means that different syntax highlighting rules can be used for different types of files.

In the provided Vue.js component, codemirror uses XML schema to implement XML syntax highlighting. This makes labels, attributes, comments, and text content in XML documents appear in different colors, making XML documents easier to understand and edit. Syntax highlighting is standard on many text editors and integrated development environments and helps you write and maintain various types of code and documentation.



```
// codemirror中语法高亮部分代码
import {parser} from "./parser.js"
import {foldNodeProp, foldInside, indentNodeProp} from "@codemirror/language"
import {styleTags, tags as t} from "@lezer/highlight"

let parserWithMetadata = parser.configure({
  props: [
    styleTags({
      Identifier: t.variableName,
      Boolean: t.bool,
      String: t.string,
      LineComment: t.lineComment,
      "( )": t.paren
    }),
    indentNodeProp.add({
      Application: context => context.column(context.node.from) + context.unit
    }),
    foldNodeProp.add({
      Application: foldInside
    })
  ]
})
```

Fig. 2.5 “Syntax highlighted portions of code in codemirror”

Conclusion

This section reviews the basic requirements for the design of the web application that will be developed. All aspects of the program are studied, the main part of the program is decomposed into program modules, and the final product form is obtained.

Describe the development environment and tools required in the software development process.

In the development of XML verification tool, the four core functions include XML file formatting, XML file verification, XML file import, XML file export design core code.

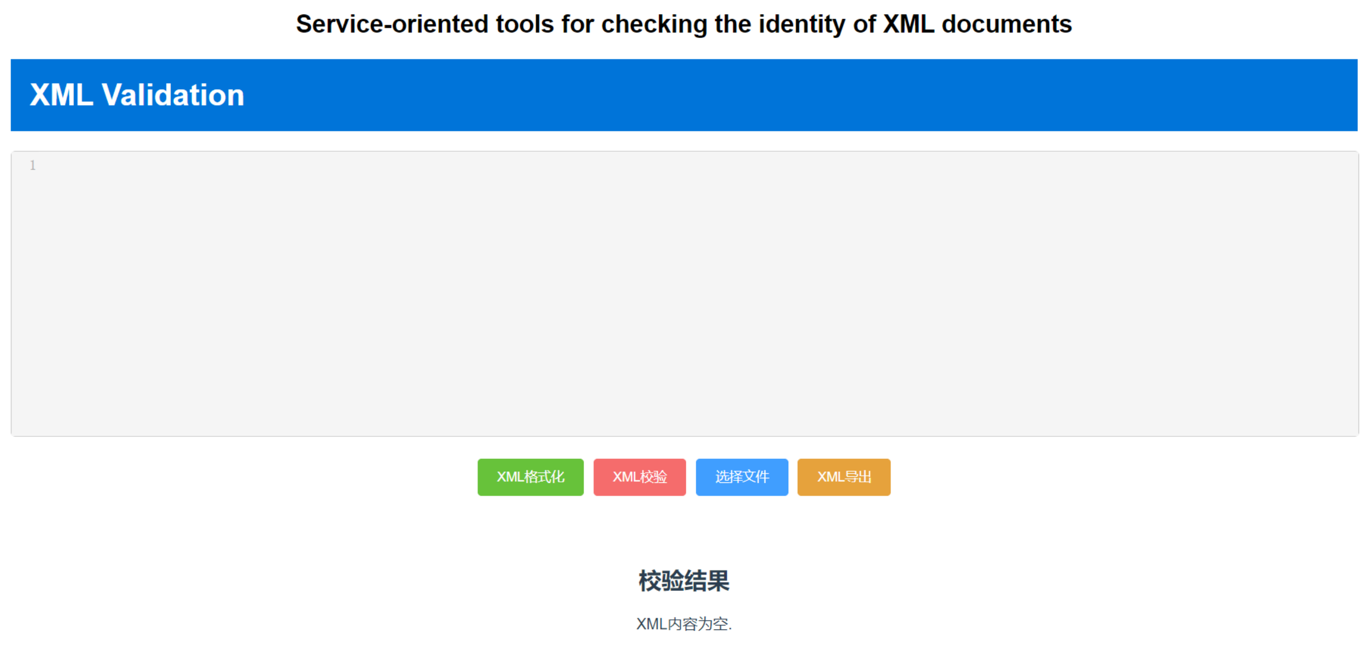
On the basis of the previous research, the functional requirements of the specific system are described.

Each requirement is detailed for comprehensive information.

CHAPTER 3
The Design function realization

1.18 IU interface design

1.18.1 XML validation tool initial interface


Service-oriented tools for checking the identity of XML documents

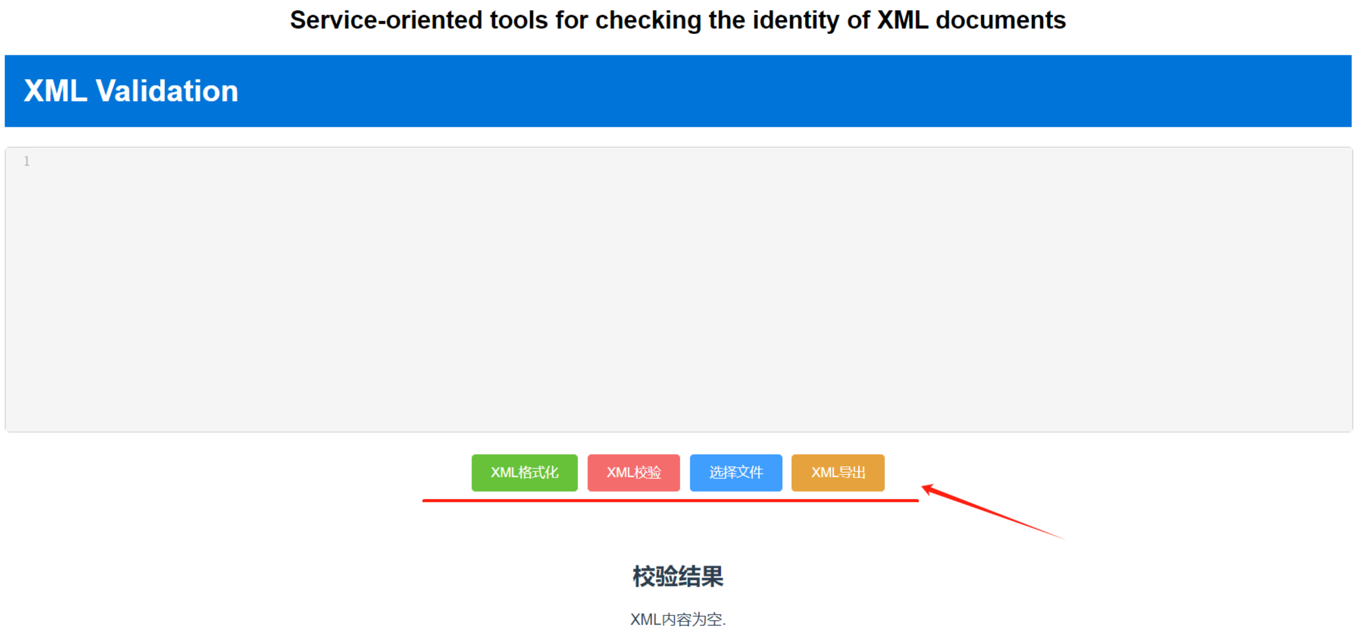
XML Validation

校验结果

XML内容为空.

Fig. 3.1 “XML validation tool initial interface”

1.18.2 XML validation tool interface function details


Service-oriented tools for checking the identity of XML documents

XML Validation

校验结果

XML内容为空.

Fig. 3.2 “XML validation tool interface function detail”

Figure 3.2 shows the buttons for the four functions implemented in this design.

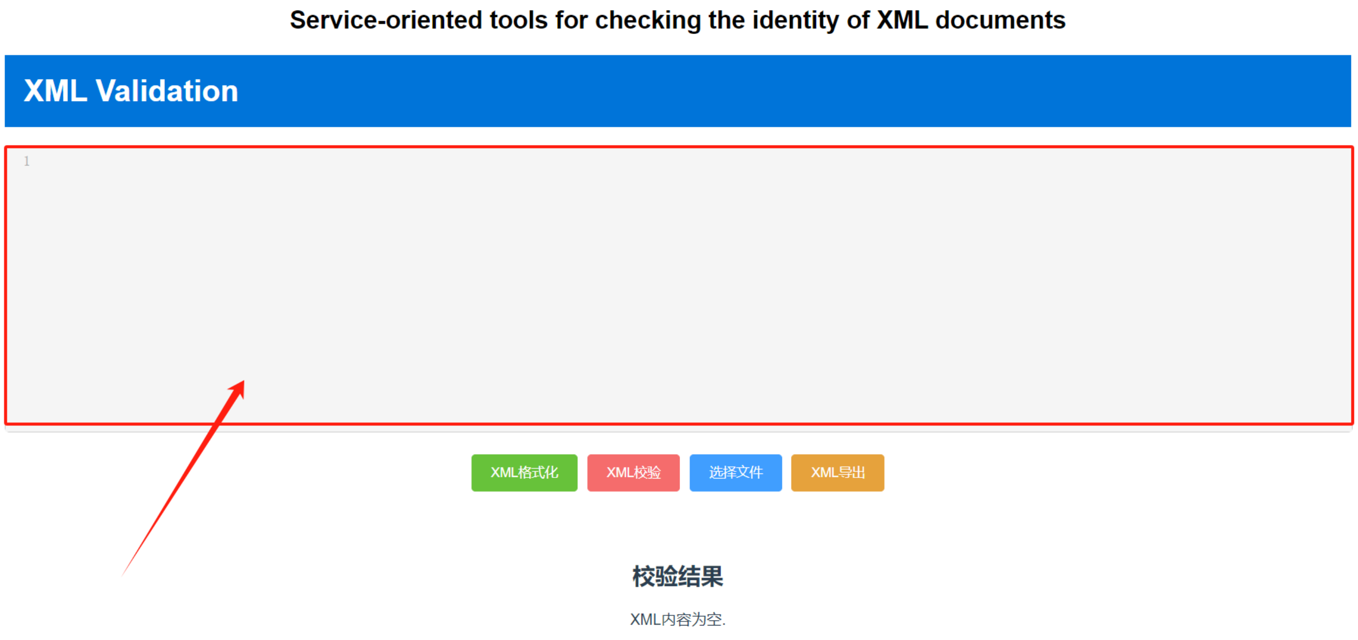
Green button: XML formatting;

Red button: XML verification;

Blue button: File import;

Yellow button: XML file export;

1.18.3 XML document content display section


Service-oriented tools for checking the identity of XML documents

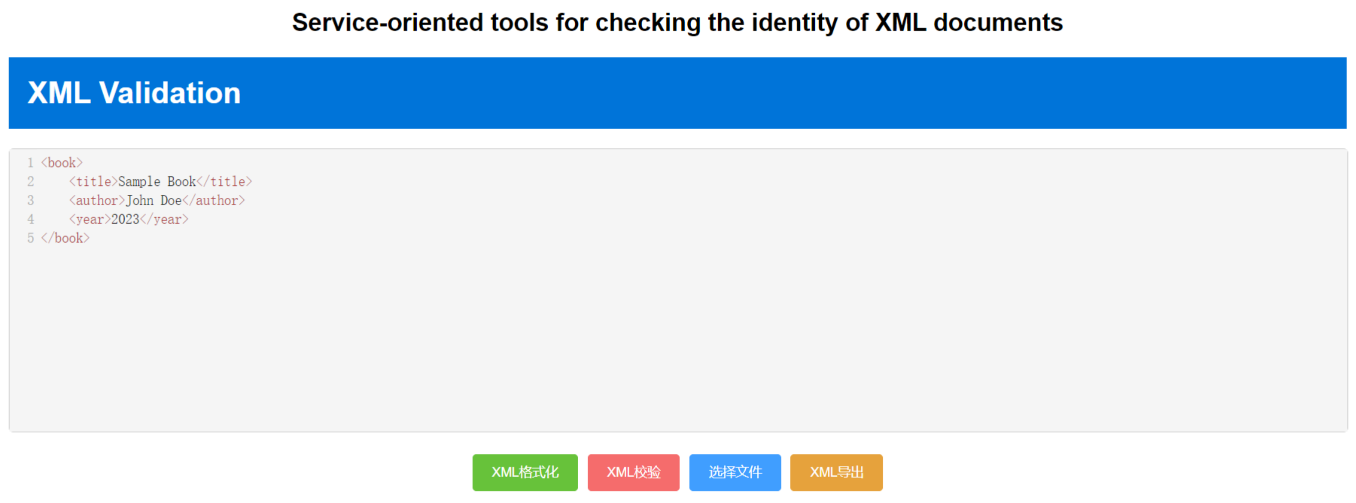
XML Validation

校验结果

XML内容为空.

Fig. 3.3 “XML document content display section”

Fig 3.3 the winning red box part is through the blue button (import) file will be imported after the show to mark the red part of the contents of the documents. Fig 3.4 below is an example.

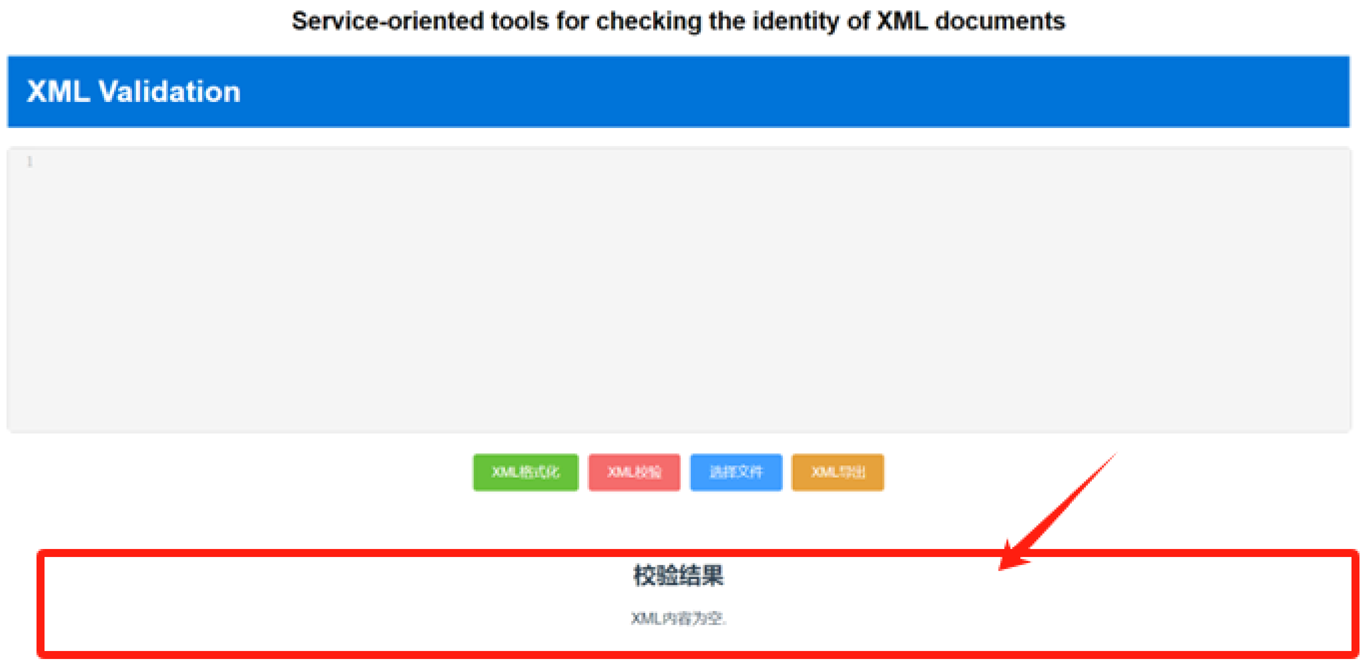

Service-oriented tools for checking the identity of XML documents

XML Validation

```
<book>
    <title>Sample Book</title>
    <author>John Doe</author>
    <year>2023</year>
</book>
```

Fig. 3.4 “Example diagram of the content display section of an XML document”

1.18.4 XML file verification results display part


Service-oriented tools for checking the identity of XML documents

XML Validation

校验结果

XML内容为空.

Fig. 3.5 “XML file verification results display part”

The red part of Fig 3.5 shows the automatic verification process of the file content after the file is imported, and the verification result is displayed in the red part. When the result of the check is correct, the result text is "XML check success" and its background will turn green. When the checksum result is an error, the result text reads "XML checksum failure and incorrect location in document content" and its background turns red.

1.19 Module function realization

1.19.1 File import function implementation

In the file import function, the user selects the blue button to open the local file on the computer in the first step, finds the file that needs XML verification and clicks the file in the second step, and clicks Open in the third step. Then the content of the file will be displayed in the content display area of the tool for the user to view.

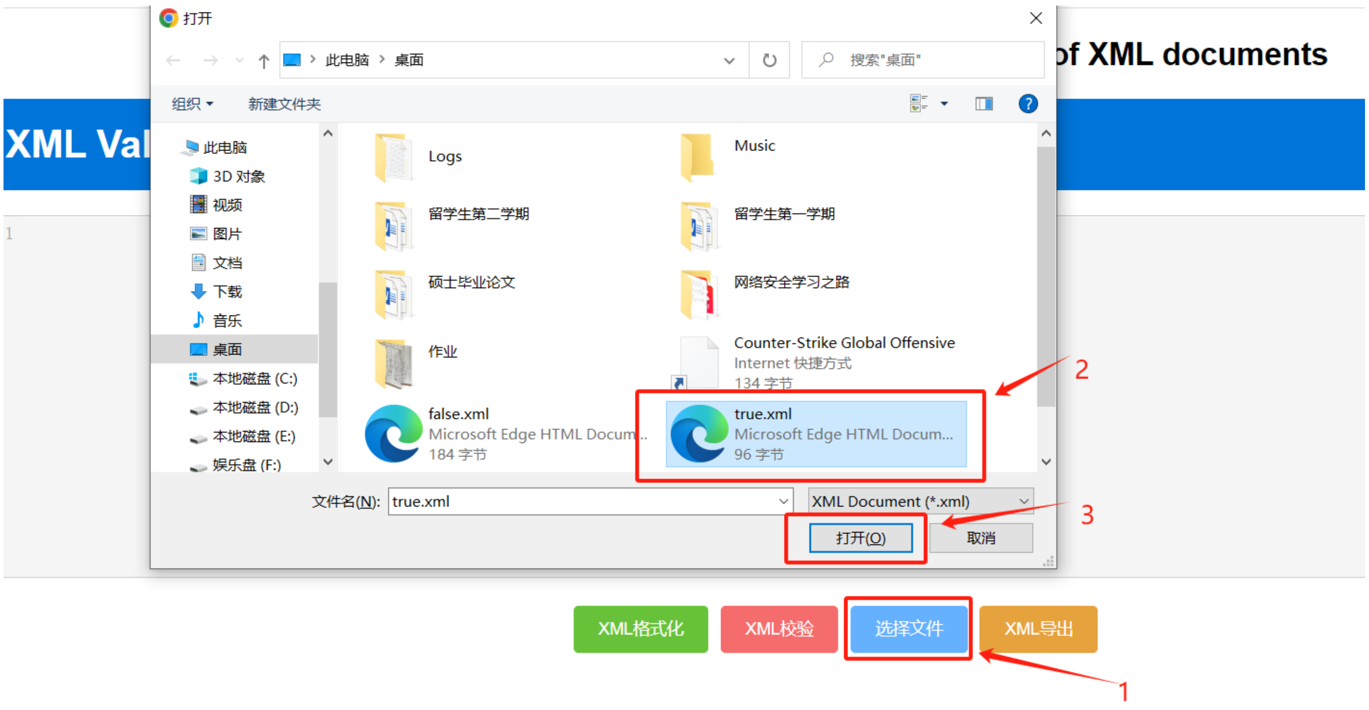
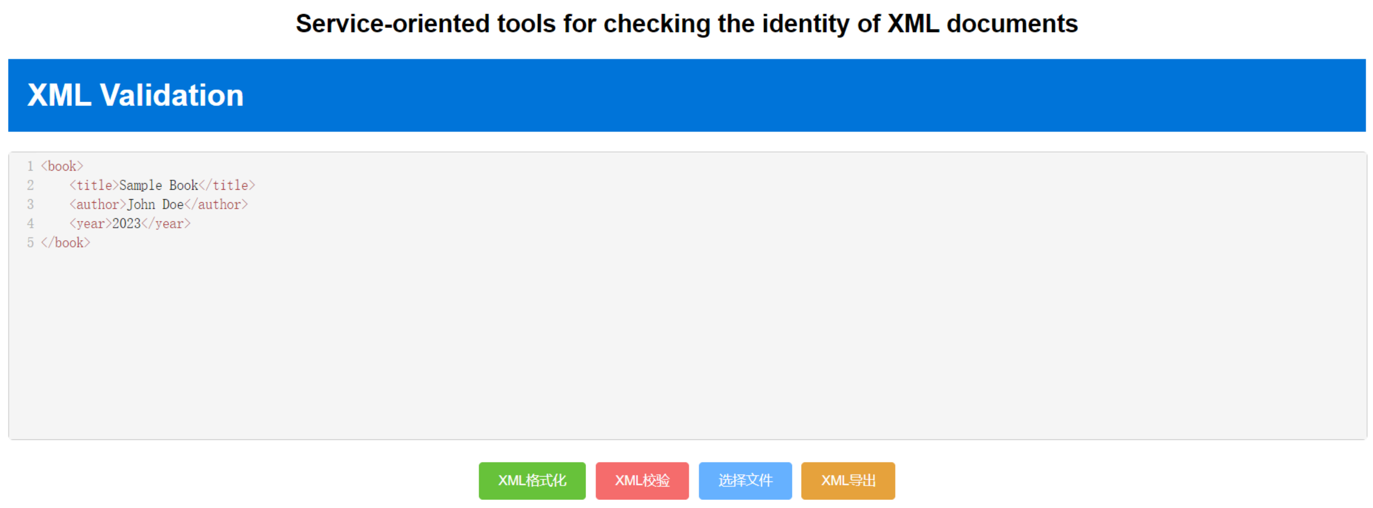


Fig. 3.6 “XML file import process diagram”


Service-oriented tools for checking the identity of XML documents

XML Validation

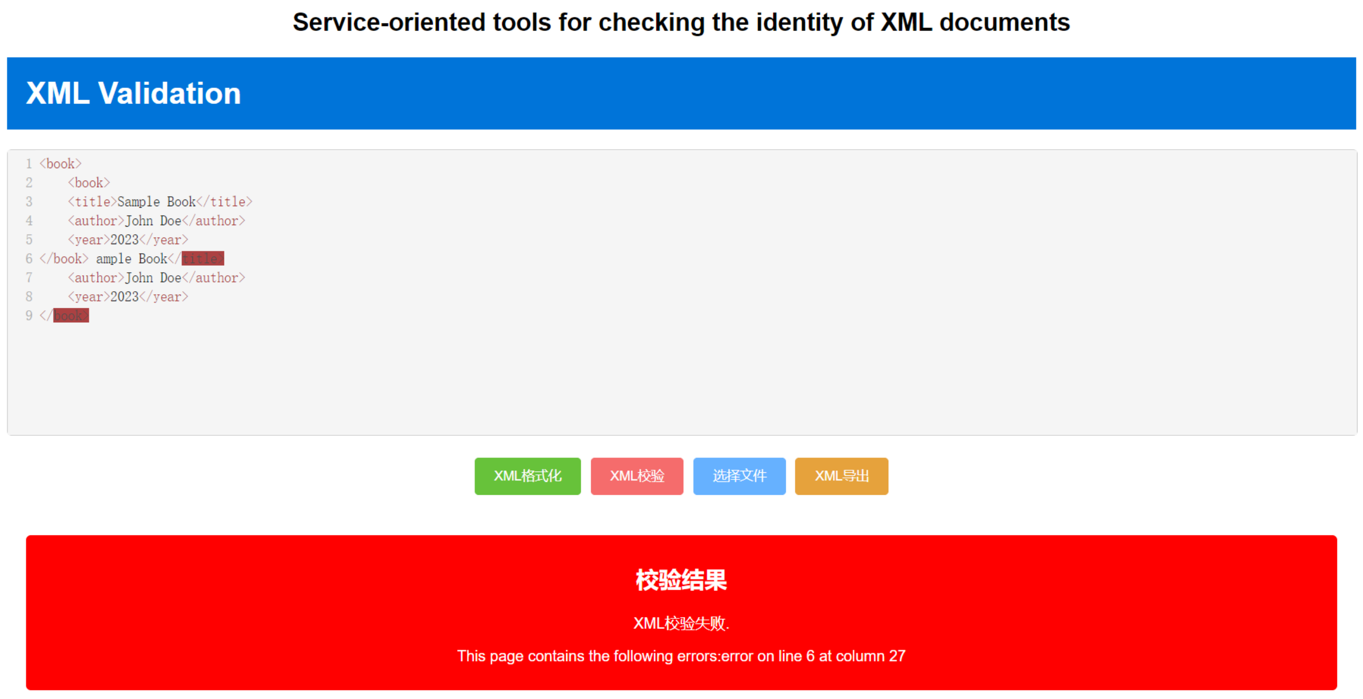
```
<book>
    <title>Sample Book</title>
    <author>John Doe</author>
    <year>2023</year>
</book>
```

Fig. 3.7 “XML file import content display diagram”

1.19.2 XML file verification function

XML file verification function after the file is imported, the file verification first checks the content of the document to check whether the file is empty. If it is empty, the tool will display "XML content is empty". If it is not empty, it will try to verify the XML content, parse the XML file into a DOM document, and check its validity.

If the XML file has syntax errors, in this case, the tool result will display "XML verification failed" and the syntax error information will be extracted, marked in red, and the detailed location of the syntax error will be displayed in the tool result area.


Service-oriented tools for checking the identity of XML documents

XML Validation

```
1 <book>
2     <book>
3     <title>Sample Book</title>
4     <author>John Doe</author>
5     <year>2023</year>
6 </book> ample Book</title>
7     <author>John Doe</author>
8     <year>2023</year>
9 </book>
```

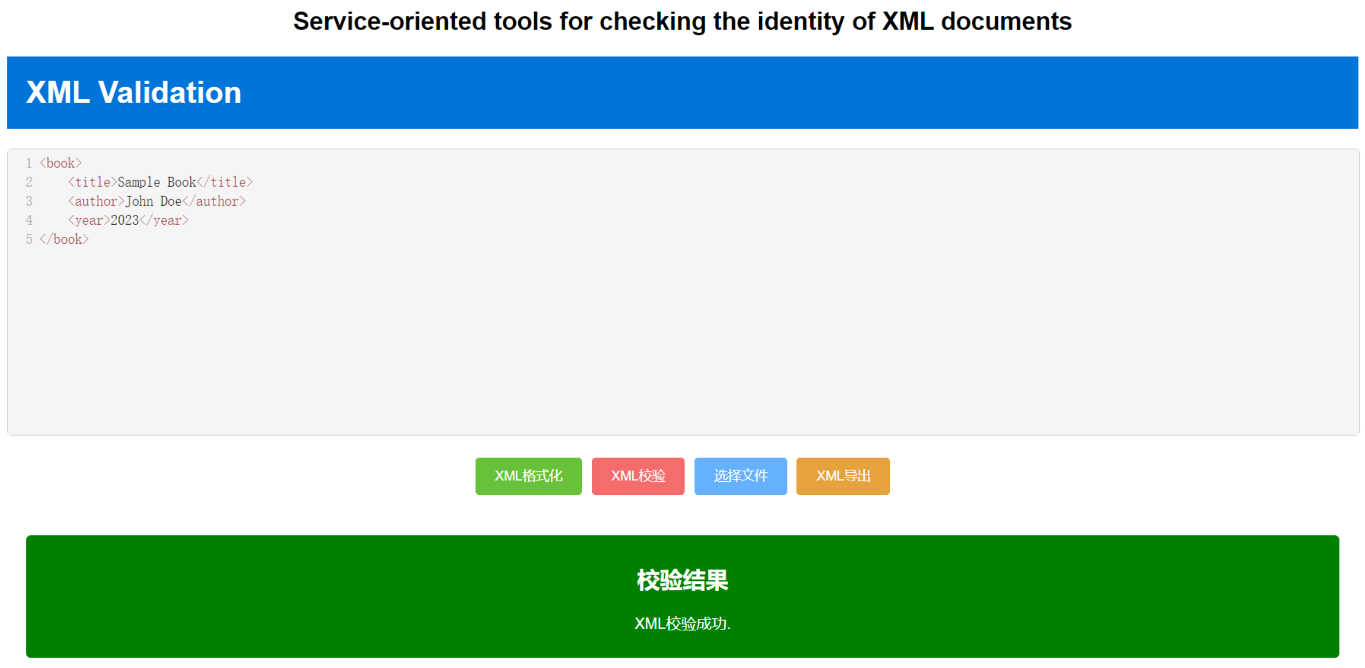
校验结果

XML校验失败.

This page contains the following errors:error on line 6 at column 27

Fig. 3.8 “The XML file exists mistakes in grammar”

If the XML file is free of syntax errors, the result area of the tool displays "XML verification success" and the background of the result area turns green.


Service-oriented tools for checking the identity of XML documents

XML Validation

```
1 <book>
2     <title>Sample Book</title>
3     <author>John Doe</author>
4     <year>2023</year>
5 </book>
```

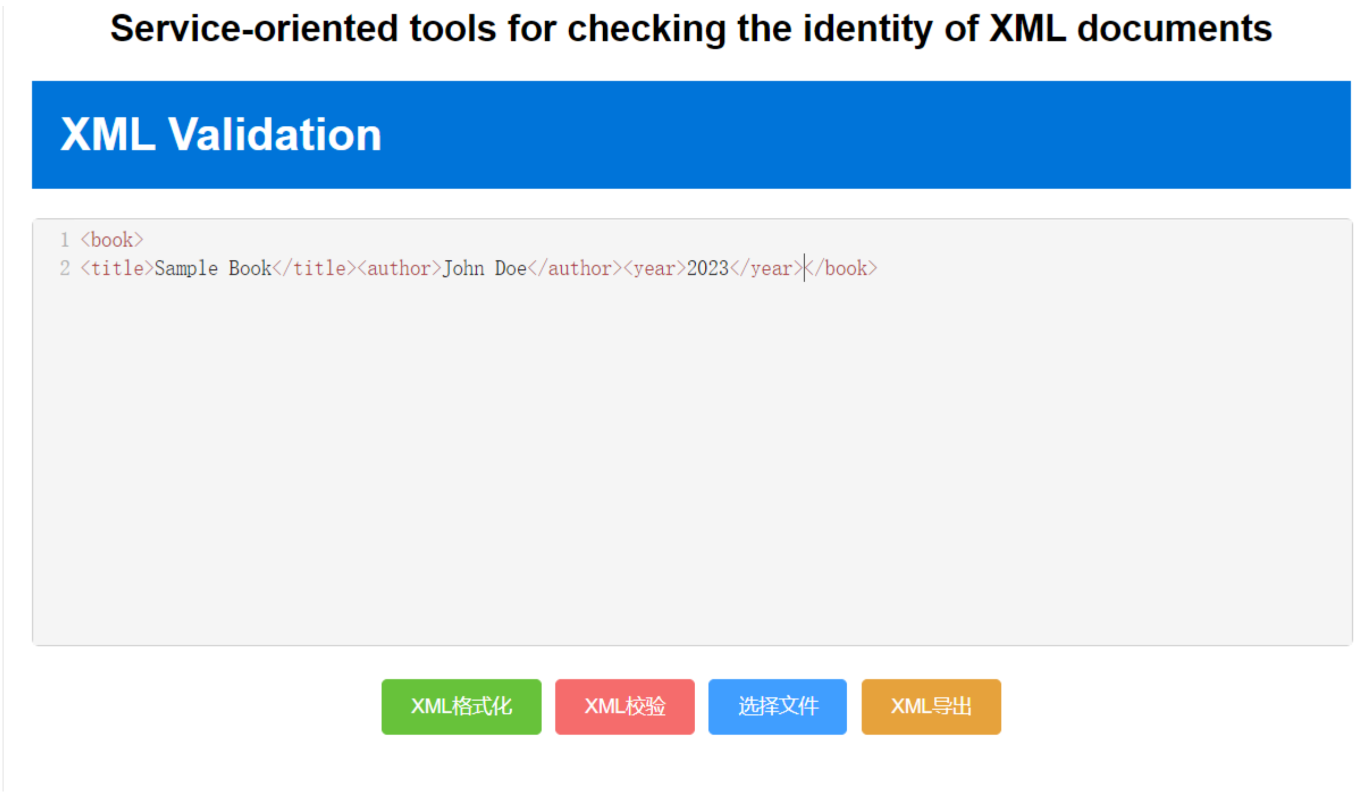
校验结果

XML校验成功.

Fig. 3.9 “The XML file not exists mistakes in grammar”

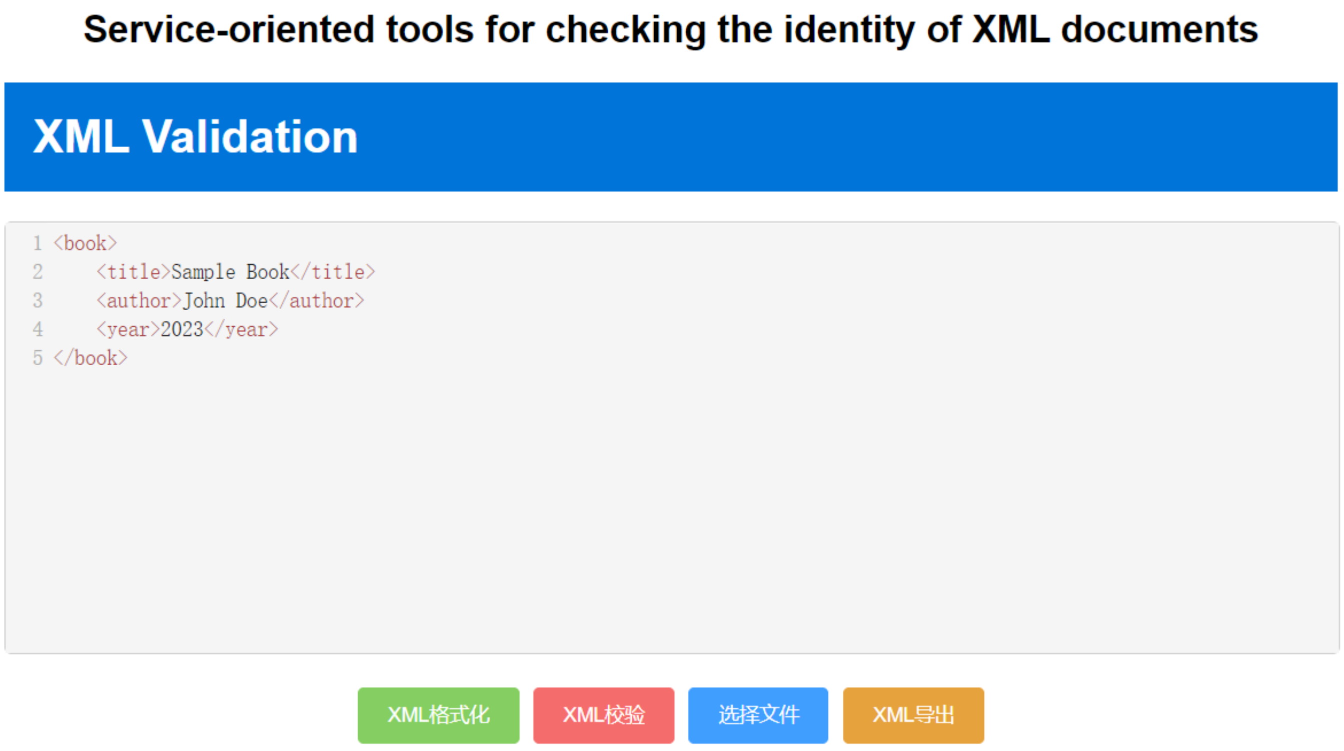
1.19.3 XML file formatting function

XML file formatting function is implemented, the user clicks the green button formatting method is called, first determine whether the XML file content is empty, if it is empty, it does not need to perform formatting operations, but directly return. If the XML file is not empty, the file is formatted. The formatted file is easier to understand and edit.


Service-oriented tools for checking the identity of XML documents
XML Validation
XML格式化
XML校验
选择文件
XML导出

```
<book>
<title>Sample Book</title><author>John Doe</author><year>2023</year></book>
```

Fig. 3.10 “Example diagram of XML file before formatting”


Service-oriented tools for checking the identity of XML documents
XML Validation
XML格式化
XML校验
选择文件
XML导出

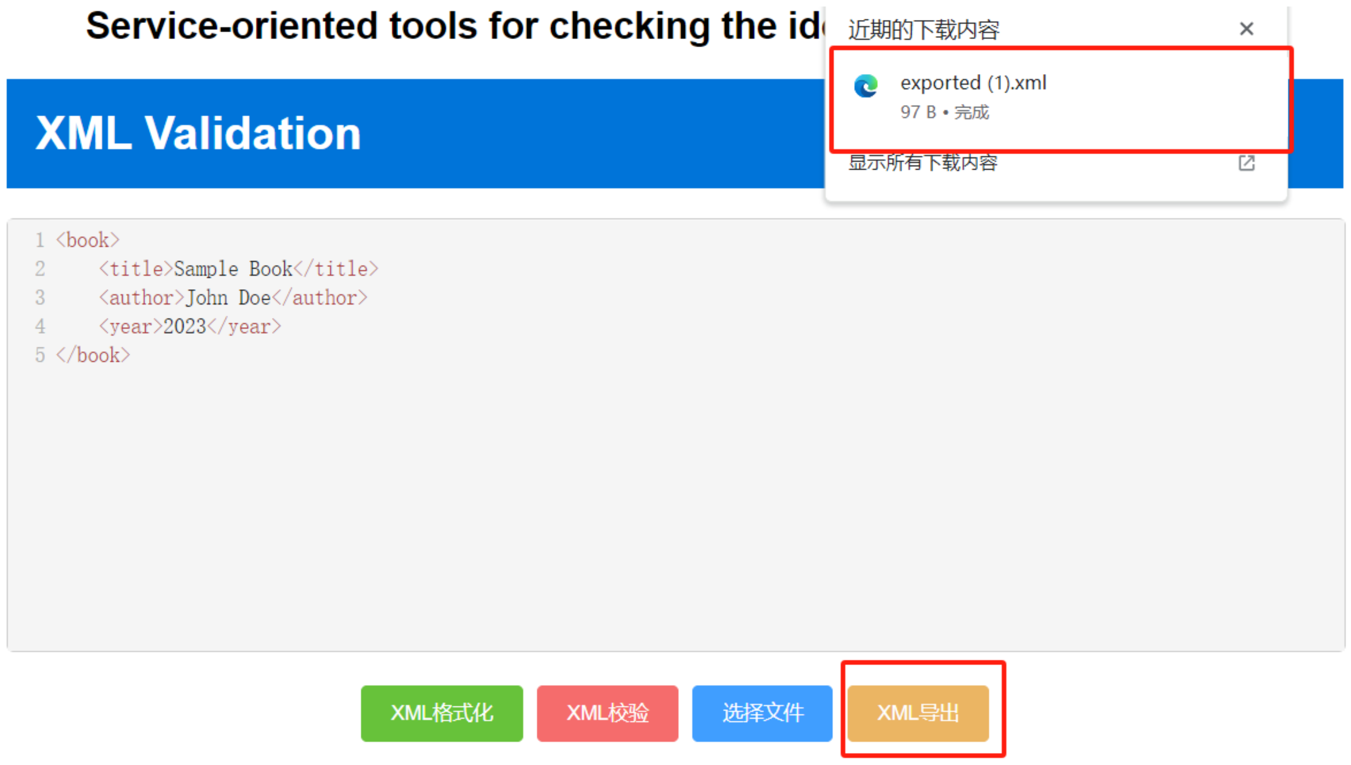
```
<book>
    <title>Sample Book</title>
    <author>John Doe</author>
    <year>2023</year>
</book>
```

Fig. 3.11 “Example diagram of the formatted XML file”

.

1.19.4 XML file export function

XML file export function: The user clicks the yellow button to export the file to the local computer. First, check whether the content of the XML file is empty. If it is empty, the export operation will not be performed. If the content of the XML file is not empty, export it.



```
<book>
    <title>Sample Book</title>
    <author>John Doe</author>
    <year>2023</year>
</book>
```

Fig. 3.12 “XML file export example diagram”

Conclusion

This chapter introduces the implementation and testing of XML document verification tool. Users can import files that need to be verified quickly and easily, and format XML files for verification. Finally, users can export and download the processed files to the computer.

CONCLUSIONS

Since its inception, XML has grown tremendously in networking and data applications. XML is also supported by all the major companies and enterprises in the world. With the continuous expansion of XML application field, the research on XML calibration has become a hot topic in network research, and any efficiency improvement and reform of XML calibration process is very meaningful. In this regard, the work of this topic is summarized as follows:

Firstly, the paper deeply understands the XML verification technology, and investigates the present, application significance and future development direction of XML research in the emerging stage. Secondly, the system function module design, XML file verification function implementation, Web application IU interface design, which is the core of all work; Finally, all the functions of the modules are realized by using JavaScript programming language, the simulation test is completed, and the expected goal is achieved.

Through the study of this project, I understand that to develop a complete project, there must be a clear goal and idea, and the necessary summary and demonstration at each stage.

In the realization part of this paper, based on the introduction of the system development and operation environment, the key technology of the system implementation is introduced, and the implementation of XML document formatting module, XML document verification module, XML document import module, XML document export module. Through the development of these modules, a WEB publishing system for XML document verification is formed, which meets the needs of users.

Through the writing of this paper, I have a better understanding of JavaScript language and XML-related technologies. I always thought XML was just a simple markup language, but now I have discovered the depth of XML. But after all, time is limited, so the knowledge of JavaScript and XML is not deep enough. In the future study, I will work harder to improve my learning efficiency and learning quantity.

LIST OF REFERENCES

1. Roberto J. Bayardo, Daniel Gruhl, Vanja Josifovski, Jussi Myllymaki., « An evaluation of binary xml encoding optimizations for fast stream based xml processing » , 2004. –17c
2. Fang Liu,Xiejun Xiao « The cornerstone of XML application: XML parsing technology » Computer engineering and design, 2005. –10c.
3. hui Zhao., « Design and implementation of an XML parser based on DOM » Shanxi University, 2005.
4. Kai Tang, « Automatic classification of XML files based on content and hierarchical structure » Computer engineering and applications 2007. –168c.
5. Serialization identification of schema-based XML document elements,[Eletronic resource] – Mode of access: https://xueshu.baidu.com/usercenter/paper/show?paperid=9cc0c270f787678a12a563758d4d8a1c&site=xueshu_se
6. X.Wang, M.Ester, W.Qian, «A feature vector based Classification for XML Dissemination »Database Systems for Advanced Applications,2010,298-409.
7. Juan Chen, « Research on safe and fast XML parsing technology »Xidian University,2017.
8. Jan van Lunteren, Ton Engbersen, Joe Bostian, Bill Carey, Chris Larsson, «Xml accelerator engine »in The First International Workshop on High Performance XML Processing,2004.
9. XML Tutorial, [Eletronic resource] – Mode of access: http://www.w3schools.com/xml/
10. Yongping Ma, «Research and application of XML document conversion technology »,Zhejiang University,2007.
11. Fang Li, «Research on XML transformation technology »,Tianjin Polytechnic University,2008.

12. Yuan J,Xu D,Bao H, «An efficient XML documents classification method based on structure and keywords frequency», Journal of Computer Research & Development,2006.
13. Ammari F T, Lu J, Abur-rous M, «Intelligent XML Tag Classification Techniques for XML Encryption Improvement», SocialCom/PASSAT，2011.

APPENDIX A.

Listing of the app source code

```
<template>
  <div style="margin-top: 0" class="xml-validation">
    <h1 class="top">Service-oriented tools for checking the identity of XML
documents</h1>
    <h1 class="title">XML Validation</h1>
    <el-form ref="xmlValidationForm" class="xml-form">
      <el-row :gutter="20">
        <el-col :span="24">
          <el-form-item>
            <codemirror
                class="code-mirror"
                v-model="xmlContent"
                :options="cmOptions"
            ></codemirror>
          </el-form-item>
          <el-form-item class="form-buttons">
            <el-button type="success" @click="handleFormatXml">XML formatting</el-button>
            <el-button type="danger" @click="handleValidateXml">XML verification</el-
button>
            <el-button type="primary" @click="openFileInput">XML file import</el-button>
            <el-button type="warning" @click="handleExportXml">XML file export</el-
button>
          </el-form-item>
        </el-col>
        <el-col :span="24">
        </el-col>
      </el-row>
      <el-row v-if="validationResult" :class="validationResultClass" class="validation-
result">
        <el-col :span="24">
          <h2>Check result</h2>
          <p>{{ validationResult }}</p>
          <p>{{ errorMsg }}</p>
        </el-col>
      </el-row>
    </el-form>
    <input type="file" ref="fileInput" @change="handleFileChange" accept=".xml"
style="display: none;">
  </div>
</template>

<script>
import xmlFormatter from 'xml-formatter';
import { codemirror } from 'vue-codemirror'
import "codemirror/theme/base16-light.css";            // 这里引入的是主题样式，根据设置的
theme 的主题引入，一定要引入！！
require("codemirror/mode/xml/xml");                    // 这里引入的模式的js，根据设置的
mode 引入，一定要引入！！

export default {
  components: {
    codemirror
  },
  data() {
    return {
      xmlContent: "",
      validationResult: "",
      validationResultClass: "",
      errorMsg: "",
      cmOptions: {
        disabled: false,
        tabSize: 2,
        autofocus: true,
        mode: 'xml',
        theme: 'base16-light',
        lineNumbers: true,
        line: true,
      }
    };
  },

  watch: {
    xmlContent: {
      handler: "handleValidateXml",
      immediate: true,
    },
  },
  methods: {

    // 格式化操作处理
    handleFormatXml() {
      if (this.xmlContent === "") {
        this.validationResultClass = "";
        return;
      }
      try {
        const formattedXml = xmlFormatter(this.xmlContent, {
          indentation: '    ', // 设置缩进为4 个空格
          collapseContent: true, // 合并文本内容
        });
        this.xmlContent = formattedXml;
        this.validXml();
      } catch (err) {
        this.validationResult = "XML 格式化失败.";
        this.validationResultClass = "validation-failed";
      }
    },

    // 处理校验
    handleValidateXml() {
      if (this.xmlContent === "") {
        this.validationResult = "XML 内容为空.";
        this.validationResultClass = "";
        return;
      }
      this.validXml()
    },

    // 校验 xml
    validXml() {
      var isValid = false;
      try {
        const parser = new DOMParser();
        const xmlDoc = parser.parseFromString(this.xmlContent, "text/xml");
        console.log(xmlDoc.documentElement.textContent)
        const text = xmlDoc.documentElement.textContent;
        if (text) {
          this.errorMsg = text.substring(0, text.indexOf('Opening') - 2)
        }
        isValid = xmlDoc.getElementsByTagName("parsererror").length === 0;
      } catch (err) {
        isValid = false;
      } finally {
        if (isValid) {
          this.validationResult = "XML 校验成功.";
          this.validationResultClass = "validation-success";
        } else {
          this.validationResult = "XML 校验失败.";
          this.validationResultClass = "validation-failed";
        }
      }
    },

    // 处理导出操作
    handleExportXml() {
      if (this.xmlContent === "") {
        return;
      }
      this.exportXml();
    },

    // 导出 xml 文件
    exportXml() {
      const xmlBlob = new Blob([this.xmlContent], {type: 'text/xml'});
      const xmlBlobUrl = URL.createObjectURL(xmlBlob);
      const a = document.createElement('a');
```

```
      a.style.display = 'none';
      a.href = xmlBlobUrl;
      a.download = 'exported.xml'; // Set the file name
      document.body.appendChild(a);
      a.click();
      URL.revokeObjectURL(xmlBlobUrl);
    },

    //打开文件上传
    openFileInput() {
      // 触发点击文件输入元素
      this.$refs.fileInput.click();
    },

    // 上传文件
    handleFileChange(event) {
      this.selectedFile = event.target.files[0];
      this.handleSelectFile();
    },

    // 解析文件
    handleSelectFile() {
      if (this.selectedFile) {
        const fileReader = new FileReader();
        fileReader.onload = (e) => {
          this.xmlContent = e.target.result;
        };
        fileReader.readAsText(this.selectedFile);
      }
    },


  },
};
</script>

<style scoped>
.xml-validation {
  margin: 20px;
  padding: 20px;
  background-color: #fff;
  border: 1px solid #e6e6e6;
  border-radius: 5px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.2);
}

.top {
  text-align: center;
  font-size: 26px;
  color: #000;
}
```

```css
.title {
  text-align: left;
  background-color: #0074d9;
  color: #fff;
  padding: 20px 20px;
}

.xml-form {
  text-align: center;
}
.code-mirror {
  text-align: start;
  line-height: 20px;
  font-size: 15px;
  width: 100%;
  margin-right: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.xml-input {
  width: 100%;
  padding: 20px 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.form-buttons {
  display: flex;
  justify-content: center;
}

.validation-result {
  text-align: center;
  margin: 20px;
  padding: 10px;
  border-radius: 5px;
}

.validation-failed {
  background-color: red;
  color: white;
}

.validation-success {
  background-color: green;
  color: white;
}
</style>
```