XML

Overview

In this project, we learn the usage of XML. XML stands for Extensible Markup Language. There are no rules for tags. It is only used for data transfer and storage unlike html. Further details are provided at the end of the document about the usage of XML (it is also the project requirement)

Objectives of project

- Learn the basics to program in XML
- Understand what a markup language is
- Understand what a metalanguage is
- Know the main differences between XML and JSON

Table of Contents

7. Calendar
8. Git Workflow
9. Technologies used
10. Incidents

Project requirements

You must create an XML document that contains the following structure:
A restaurant that has two menus, one for children and the other for adults. Each menu will be divided into three sections: first, second and dessert. Of these three sections each will have five dishes.

Furthermore, the following questions must be answered;

- What is a markup language?
- What is a metalanguage?
- Differences between XML and JSON
- Define five cases in which XML is used
- What is the relationship between AJAX and XML?

Risk Management

Every project has risks. These risks must be taken into account to improve the workflow of the project. Managing these risks is important for due completion of the project. Unchecked risks could not only lead to hamper of project deliverance but also a badly executed project. The risks hence associated with the project are documented as follows:

Risk	Risk level
Unfamiliarity with XML	Low
Health issues	High
Delivering in time	Medium

Completing all the project requirements	Medium

Task Management

The following are the tasks which needed to be accomplished for the completion of the project. The tasks have been divided according to the project specifics. Each task has been clearly defined and their priority as well as their difficulty and time needed. Difficulty level is explained on a scale of 1 to 5 (5 being the most difficult). Priority is explained on the level of 1 to 5 (again 5 the highest parameter being the most prioritized work).

Task	Priority	Description	Difficulty	Time
Read the description of the project	5	This task involves complete understanding of requirements for the project	1	30 min
Create git repo	5	Creating of git repository for project execution and delivery	1	2 mins
Create files for the project	4	Creating the necessary files, folders and subfolders for this project. This is important especially for this project of PhP unit	1	5 mins
Write into XML file	3	Write the XML file as required	2	10-20 mins
Answer relevant questions	4	Answer all the questions asked in the project requirements	1	10-20mins
Documentation	1	Write the documentation and push it to github	2	30 mins

Defining this part is crucial to the development of the project. It is important to make a good analysis of the situation to organize the project in a good way.

Chronogram

This shows the time-line it took for the project to finish and also the tasks that were achieved during those timeline.

The tasks

Tasks	Tue 1200 hrs	Tue 1300 hrs	Tue 1400 hrs
Read project descriptio-n	X		
Git Repo	X		
Create files	X		
Write into XML files		X	
Answer relevant questions		X	
Documentation			X

Git Workflow

For this project, all commits we'll be pushed directly to the Master branch. All commits will use a descriptive message, so that the admin and other internet user can follow through the processes. For more information go to :
https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Incidents

- None

Technologies used

For this project, we will use the following technologies:

- XML
- GIT

Questions

What is a markup language?

- A markup language is a computer language that uses tags to define elements within a document. It is human-readable, meaning markup files contain standard words, rather than typical programming syntax. While several markup languages exist, the two most popular are HTML and XML.

  HTML is a markup language used for creating web-pages. The contents of each webpage are defined by HTML tags. Basic page tags, such as <head>, <body>, and define sections of the page, while tags such as <table>, <form>, <image>, and <a> define elements within the page. Most elements require a beginning and end tag, with the content placed between the tags. For example, a link to the TechTerms.com home page may use the following HTML code:

  TechTerms.com

  XML is used for storing structured data, rather than formatting information on a page. While HTML documents use predefined tags (like the examples above), XML files use custom tags to define elements. For example, an XML file that stores information about computer models may include the following section:

  <computer>

  <manufacturer>Dell</manufacturer>

  <model>XPS 17</model>

  <components>

  <processor>2.00 GHz Intel Core i7</processor>

<ram>6GB</ram>

<storage>1TB</storage>

</components>

</computer>

XML is called the "Extensible Markup Language" since custom tags can be used to support a wide range of elements. Each XML file is saved in a standard text format, which makes it easy for software programs to parse or read the data. Therefore, XML is a common choice for exporting structured data and for sharing data between multiple programs.

What is a metalanguage ?

- **ML** ("Meta Language") is a general-purpose functional programming language. It has roots in Lisp, and has been characterized as "Lisp with types". ML is a statically-scoped functional programming language like Scheme. It is known for its use of the polymorphic Hindley–Milner type system, which automatically assigns the types of most expressions without requiring explicit type annotations, and ensures type safety – there is a formal proof that a well-typed ML program does not cause runtime type errors. ML provides pattern matching for function arguments, garbage collection, imperative programming, call-by-value and currying. It is used heavily in programming language research and is one of the few languages to be completely specified and verified using formal semantics. Its types and pattern matching make it well-suited and commonly used to operate on other formal languages, such as in compiler writing, automated theorem proving, and formal verification.

Differences between XML and JSON ?

- The fundamental difference, is that XML is a markup language (as it actually says in its name), whereas JSON is a way of representing objects (as also noted in its name).
  With XML (using a certain element vocabulary) you can put:
  `<Document> <Paragraph Align="Center"> Here <Bold>is</Bold> some text. </Paragraph> </Document>`

  This is what makes markup languages so useful for representing documents.

  An object notation like JSON is not as flexible. But this is usually a good thing. When you're representing objects, you simply don't need the extra flexibility. To represent the above example in JSON, you'd actually have to solve some problems manually that XML solves for you.

{ "Paragraphs": [{ "align": "center", "content": ["Here ", { "style" : "bold", "content": ["is"] }, " some text."] }] }

It's not as nice as the XML, and the reason is that we're trying to do markup with an object notation. So we have to invent a way to scatter snippets of plain text around our objects, using "content" arrays that can hold a mixture of strings and nested objects.

On the other hand, if you have a typical hierarchy of objects and you want to represent them in a stream, JSON is better suited to this task than HTML.

{ "firstName": "Homer", "lastName": "Simpson", "relatives": ["Grandpa", "Marge", "The Boy", "Lisa", "I think that's all of them"] }

Here's the logically equivalent XML:

<Person> <FirstName>Homer</FirstName> <LastName>Simpsons</LastName> <Relatives> <Relative>Grandpa</Relative> <Relative>Marge</Relative> <Relative>The Boy</Relative> <Relative>Lisa</Relative> <Relative>I think that's all of them</Relative> </Relatives> </Person>

JSON looks more like the data structures we declare in programming languages. Also it has less redundant repetition of names.

But most importantly of all, it has a defined way of distinguishing between a "record" (items unordered, identified by names) and a "list" (items ordered, identified by position). An object notation is practically useless without such a distinction. And XML has no such distinction! In my XML example <Person> is a record and <Relatives> is a list, but they are not identified as such by the syntax.

Instead, XML has "elements" versus "attributes". This looks like the same kind of distinction, but it's not, because attributes can only have string values. They cannot be nested objects. So I couldn't have applied this idea to <Person>, because I shouldn't have to turn <Relatives> into a single string.

By using an external schema, or extra user-defined attributes, you can formalise a distinction between lists and records in XML. The advantage of JSON is that the low-level syntax has that distinction built into it, so it's very succinct and universal. This means that JSON is more "self describing" by default, which is an important goal of both formats.

So JSON should be the first choice for object notation, where XML's sweet spot is document markup.

Define 5 cases where XML is used?

- XML is good at describing things.
- XML is good for Metadata.
- XML is good for demarcation of scope, bounding information.
- XML, like most ML's is good for varying numbers of fields and objects in a record, or for non-homogenous records. So we don't send a 4K datagram for changes of a few bytes.
- XML is good for exposing data labels, which can actually be a very bad thing in itself. It helps the bad guys learn what is what very quickly.
- XML is not good for bandwidth, nor storage, nor performance.