Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»
Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Розробка програми конвертації XML документів в HTML-формат засобами XSLT»

за освітньою програмою: «12 Інженерія програмного забезпечення»

зі спеціальності: «121 Інженерія програмного забезпечення»

Виконав: студент групи «ПЗ1912» _____ /Наталя ТКАЧ/
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /доц. Вадим АНДРЮЩЕНКО/
(підпис) (посада, Ім`я ПРІЗВИЩЕ)

Нормоконтролер: _____ /доц. Світлана ВОЛКОВА/
(підпис) (посада, Ім`я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Дніпро – 2023 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»
Department «Computer information technology»

Explanatory Note

to Bachelor's Thesis

on the topic: «Development of a program for converting XML documents into HTML format using XSLT»

in the Speciality: «121 Software engineering»

Done by the student of the group PZ1912: /Natalia TKACH/
Scientific Supervisor: /Vadym ANDRIUSHCHENKO/
Normative controller: /Svitlana VOLKOVA/

Dnipro – 2023

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____/Вадим ГОРЯЧКІН/
(підпис)
Дата_____

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра
студенту Ткач Наталі Сергіївні

1. Тема роботи: «Розробка програми конвертації XML документів в HTML-формат засобами XSLT»
Керівник роботи: Андрющенко Вадим Олександрович, доцент
затверджені наказом № 1209 ст від 07.12.2022
2. Строк подання студентом роботи: 12.06.2023 р.
3. Вихідні дані до роботи: _____.
4. Зміст пояснювальної записки (перелік питань до розробки): реферат, вступ, аналіз сучасного стану предметної області, проектування, тестування та відлагодження, опис програмного продукту, висновки, бібліографічний список
5. Перелік демонстраційного матеріалу:
5.1. доповідь;
5.2. презентація;
5.3. демонстраційне відео.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір та аналіз вимог	15.02.23	10%
2	Зовнішнє проектування	01.03.23	20%
3	Внутрішнє проектування	15.03.23	30%
4	Розробка алгоритмів	01.04.23	40%
5	Розробка програмного забезпечення	01.05.23	65%
6	Тестування програмного забезпечення	20.05.23	85%
7	Подання кваліфікаційної роботи до кафедри	12.06.23	100%
8	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	28.06.23	

Студент _____ _Наталя ТКАЧ_
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ _доц. Вадим АНДРЮЩЕНКО__
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра:

71с., 5 рис., 4 додаток, 43 джерел.

Об'єкт розробки – програма конвертації XML документів в HTML-формат засобами XSLT.

Мета роботи – Метою розробки цього проекту є використання XML для зберігання різного роду даних. Тому, що «чистий» XML не читабельний і не зрозумілий рядовому користувачу. Для підвищення читабельності використовують мову трансформації XSTL, яка перетворює XML в звичний і читабельний HTML.

Ключові слова: трансформація XSTL, HTML, XML, програмне забезпечення.

ЗМІСТ

ВСТУП

Актуальність розробки полягає в актуальності використання XML для зберігання різного роду даних. Основна проблема полягає в тому, що «чистий» XML не читабельний і не зрозумілий рядовому користувачу. Для підвищення читабельності використовують мову трансформації XSTL, яка перетворює XML в звичний і читабельний HTML.

Використання XSTL є досить трудомістким процесом, який вимагає часу. Через це, основною метою роботи є створення програми конвертації XML документів в HTML-формат засобами XSLT.

Предмет дослідження — перетворення XML в HTML.

Об’єкт дослідження — спрощення процесу перетворення XML в HTML з використанням XSTL.

Для досягнення поставленої мети слід виконати наступні завдання:

- провести аналіз поняття прикладного програмного забезпечення;
- провести аналіз мови XSTL;
- провести аналіз варіантів використання;
- спроектувати пакетну будову;
- спроектувати внутрішню будову;
- обрати засоби реалізації;
- створити графічний інтерфейс користувача;
- провести тестування системи.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення

В першу чергу необхідно розглянути поняття програмного забезпечення в цілому, як основу для майбутньої розробки, оскільки розроблятися буде саме прикладне програмне забезпечення.

Прикладна програма (програмна програма, або додаток, або скорочено додаток) — це комп'ютерна програма, призначена для виконання конкретного завдання, відмінного від того, що стосується роботи самого комп'ютера [1], як правило, для використання кінцевими користувачами. [2] Прикладами є текстові процесори, медіаплеєри та бухгалтерське програмне забезпечення. Збірний іменник «прикладне програмне забезпечення» відноситься до всіх програм разом.[3] Інші основні класифікації програмного забезпечення — це системне програмне забезпечення, пов'язане з роботою комп'ютера, та службове програмне забезпечення («утиліти»).

Програми можуть бути в комплекті з комп'ютером і його системним програмним забезпеченням або опубліковані окремо і можуть бути закодовані як пропрієтарні, з відкритим вихідним кодом або проекти.[4] Термін «програма» зазвичай стосується програм для мобільних пристроїв, наприклад телефонів.

Термінологія

Програма – це комп'ютерна програма, яка допомагає людям виконувати певні завдання. Програма може працювати з аудіо, графікою, числами та текстом — або будь-якою комбінацією цих елементів. Його також можна назвати прикладною програмою або прикладним програмним забезпеченням. Деякі програми використовують окрему функцію, наприклад обробку тексту; інші називаються «інтегрованими» та містять кілька програм.

Люди можуть писати програми, які відповідають їхнім конкретним потребам. Ці програми можуть бути аудіо, графікою, комп'ютерною

анімацією та макросами текстового процесора. Це також можуть бути шаблони електронних таблиць, написані користувачем. Фільтри електронної пошти – це програма, створена користувачами, яку люди часто не помічають.

Навколо різниці між прикладним програмним забезпеченням і системами операційної системи точаться суперечки. Одним із ключових прикладів є справа Сполучені Штати проти Microsoft, у якій ключовим питанням було те, чи програмне забезпечення Microsoft Internet Explorer було частиною операційної системи Windows чи окремою частиною прикладного програмного забезпечення. У деяких системах немає різниці між операційною системою та прикладною програмою — наприклад, програмне забезпечення, яке використовується для керування відеомагнітофоном, DVD-програвачем або мікрохвильовою піччю. Це пов'язано з розбіжностями щодо зв'язку між ядром Linux і вбудованими операційними системами.

Наведені вище визначення можуть виключати деякі програми, які можуть існувати на деяких комп'ютерах у великих організаціях. Альтернативне визначення програми див. у розділі Керування портфелем програм.

Метонімія

Коли слово «програма» використовується як прикметник, воно не обмежується лише «прикладним програмним забезпеченням або пов'язаним із ним». Наприклад, такі поняття, як інтерфейси прикладного програмування (API), сервери додатків, віртуалізація додатків, керування життєвим циклом додатків і портативні додатки однаково застосовуються до всіх комп'ютерних програм, а не лише до прикладного програмного забезпечення.

Програми та програми-вбивці

Деякі програми випускаються у версіях для кількох різних платформ; інші працюють лише на одній платформі й тому називаються географічними програмами Microsoft Windows, навчальними програмами Android або іграми Linux. Іноді з'являється нова популярна програма, яка працює лише на одній платформі, що підвищує привабливість цієї платформи. Це відоме як

програма-вбивця або програма-вбивця. Наприклад, VisiCalc був першим сучасним програмним забезпеченням для роботи з електронними таблицями для Apple II і допоміг продати офісам нові персональні комп'ютери. Для BlackBerry це програмне забезпечення електронної пошти.

В останні роки абревіатура «додаток» (введена в 1981 році або раніше [8]) стала популярною для позначення програм для мобільних пристроїв, таких як смартфони та планшети, на відміну від програм для ПК, де ця абревіатура відповідає. Вони зазвичай менші. за розміром. Останнім часом скорочені версії також використовуються в настільних програмах.

Класифікація

Існує багато різних способів класифікації прикладного програмного забезпечення.

З юридичної точки зору програмні додатки в основному класифікують права своїх кінцевих користувачів або передплатників (можливо, із проміжним і багаторівневим рівнями підписки) за допомогою підходу чорної скриньки.

Програмні додатки також класифікуються відповідно до мови програмування, на якій написаний або виконується вихідний код, а також його призначення та результат.

Про право власності та користування

Прикладне програмне забезпечення, як правило, поділяється на дві широкі категорії: програмне забезпечення із закритим вихідним кодом проти програмного забезпечення з відкритим вихідним кодом і програмне забезпечення проти безкоштовного програмного забезпечення або програмного забезпечення за власним бажанням.

Власне програмне забезпечення захищене авторськими правами власності, а ліцензія на програмне забезпечення надає обмежені права на використання. Принцип «відкрито-закрито» стверджує, що програмне забезпечення може бути «відкритим лише для розширення, а не для

модифікації». Такі програми можуть отримувати лише доповнення від третіх сторін.

Безкоштовне програмне забезпечення з відкритим вихідним кодом має запускатися, розповсюджуватися, продаватися чи розширюватися з будь-якою метою, а після відкриття воно має бути змінено або скасовано таким же чином.

Програмне забезпечення FOSS, випущене за безкоштовною ліцензією, може бути безстроковим або безкоштовним. Власник, утримувач або третя сторона, яка виконує будь-які права (авторське право, торговельну марку, патент або юридичне право власності) може мати право додавати винятки, обмеження, терміни або терміни дії до умов ліцензії.

Програмне забезпечення загального користування є невільним FOSS, яке можна запускати, поширювати, змінювати, скасовувати, повторно публікувати або створювати похідні роботи без авторських прав і, отже, відкликання. Її навіть можна продати без передачі комунальної власності іншим структурам. Програмне забезпечення, що є суспільним надбанням, може бути випущено згідно з ліцензією на юридичну відмову від відповідальності (dis), яка забезпечує дотримання цих умов протягом невизначеного періоду (довічно або назавжди).

Мовою кодування

Зі зростанням і майже повсюдним поширенням Інтернету існує важлива відмінність між веб-програмами (написаними з використанням HTML, JavaScript та інших веб-технологій, які зазвичай вимагають підключення до Інтернету та запущеного веб-браузера) та більш традиційними рідними програмами. - якою мовою може користуватися комп’ютер певного типу. У комп’ютерному співтоваристві точаться дебати щодо заміни веб-програм рідними програмами для багатьох цілей, особливо на мобільних пристроях, таких як смартфони та планшети. Популярність веб-програм для деяких цілей справді зростає, але переваги програм роблять їх навряд чи зникнуть найближчим часом, якщо взагалі зникнуть. Крім того, вони можуть доповнювати і навіть інтегруватися один з одним. [9][10][11]

За записом та виїздом

Аплікації також можна вважати горизонтальними або вертикальними. [12][13] Горизонтальні програми є більш популярними та поширеними, тому що вони загального призначення, такі як текстові процесори або бази даних. Вертикальні проекти — це нішеві продукти, розроблені для певного типу промисловості чи підрозділу всередині підприємства чи організації. Пакет інтеграції намагатиметься впоратися з усіма можливими конкретними аспектами, такими як виробництво чи банківський персонал, бухгалтерія чи обслуговування клієнтів.

За платформою

Програми також можна класифікувати за обчислювальними платформами, такими як настільна програма для конкретної операційної системи[16], мережа доставки, наприклад хмарні обчислення та програми Web 2.0, або пристрої доставки, такі як мобільні програми для мобільних пристроїв.

Саму операційну систему можна вважати прикладним програмним забезпеченням для виконання простих завдань обчислення, вимірювання, рендерингу та обробки текстів, які не використовуються для керування обладнанням через інтерфейс командного рядка чи графічний інтерфейс користувача. Це не включає прикладне програмне забезпечення, що входить до складу операційних систем, наприклад програмний калькулятор або текстовий редактор.

Програмне забезпечення інформаційного працівника

- Бухгалтерська програма
- Управління даними
- Контактний менеджер
- Електронна таблиця
- Програмне забезпечення баз даних
- Документація
- Автоматизація документообігу
- Текстовий процесор

- Програмне забезпечення для настільного видавництва
- Програмне забезпечення для створення діаграм
- Програмне забезпечення для презентацій
- Електронна пошта
- Програмне забезпечення для блогів
- Планування ресурсів підприємства
- Фінансове програмне забезпечення
- Банківське програмне забезпечення
- Клірингові системи
- Програмне забезпечення для фінансового обліку
- Фінансове програмне забезпечення
- Управління виїзним обслуговуванням
- Програмне забезпечення для управління персоналом
- Програмне забезпечення для управління проектами
- Програмне забезпечення для календарів
- Програмне забезпечення для планування співробітників
- Програмне забезпечення робочого процесу
- Системи бронювання

  Розважальне програмне забезпечення
- Заставки
- Відео ігри
- Аркадні ігри
- Консольні ігри
- Рухливі ігри
- Ігри для персонального комп'ютера
- Програмне забезпечення ст
- Демо
- 64K вступ

  Програмне забезпечення компілятора

- Інтегроване середовище розробки
- Упорядник
- Посилувач
- Налагоджу вач
- Контроль версій
- Інструмент розробки ігор
- Менеджер ліцензій

Отже, в ході аналізу поняття прикладного програмного забезпечення було виявлено основні характеристики прикладного програмного забезпечення та оглянуто класифікацію прикладних програм. Уся отримана інформація необхідна для розуміння поняття прикладного програмного забезпечення в цілому і суті майбутньої розробки.

1.2 Огляд XML

XML — один з основних факторів будь-якого XSLT-перетворень, оскільки при подібного роду перетвореннях HTML створюється саме на основі XML. З цього виходить необхідність огляду мови XML.

Розширювана мова розмітки (XML) — це простий текстовий формат для представлення структурованої інформації: документи, дані, конфігурація, книги, транзакції, рахунки-фактури та багато іншого. Він був похідним від старішого стандартного формату під назвою SGML (ISO 8879), щоб бути більш придатним для використання в Інтернеті.

Для чого використовується XML?

XML є одним із найбільш широко використовуваних форматів для обміну структурованою інформацією сьогодні: між програмами, між людьми, між комп’ютерами та людьми, як локально, так і через мережі.

Короткий приклад:

```
<part number="1976">
  <name>Windscreen Wiper</name>
  <description>The Windscreen wiper
    automatically removes rain
    from your windscreen, if it
    should happen to splash there.
    It has a rubber <ref part="1977">blade</ref>
    which can be ordered separately
    if you need to replace it.
  </description>
</part>
```

Якщо ви вже знайомі з HTML, ви бачите, що XML дуже схожий. Однак правила синтаксису XML суворі: інструменти XML не оброблятимуть файли, які містять помилки, а натомість видадуть вам повідомлення про помилки, щоб ви їх виправили. Це означає, що майже всі XML-документи можуть бути надійно оброблені комп’ютерним програмним забезпеченням.

Основні відмінності від HTML:

Усі елементи мають бути закриті або позначені як порожні.

Порожні елементи можна закривати як звичайно, <happiness></happiness>, або замість цього можна використовувати спеціальну коротку форму <happiness />.

У HTML вам потрібно взяти значення атрибута в лапки лише за певних обставин (воно містить пробіл або символ, який не дозволяється в імені), але правила важко запам’ятати. У XML значення атрибутів завжди повинні бути взяті в лапки:

<happiness type="joy" />

У HTML є вбудований набір імен елементів (разом з їхніми атрибутами). У XML немає вбудованих імен (хоча імена, що починаються з xml, мають спеціальні значення).

У HTML є список деяких вбудованих імен символів, таких як é для é, але XML цього не має. У XML є лише п’ять вбудованих символьних сутностей: <, >, &, " і ' для <, >, &, " і ' відповідно. Ви можете визначити власні сутності у визначенні типу документа або використовувати будь-який символ Unicode (див. наступний пункт).

У HTML також є посилання на числові символи, наприклад & для &. Ви можете посилатися на будь-який символ Юнікоду, але число є десятковим, тоді як у таблицях Юнікоду число зазвичай у шістнадцятковому вигляді. XML також допускає шістнадцяткові посилання: & наприклад.

XML має ряд переваг перед багатьма іншими форматами. Для будь-якого конкретного сценарію ви могли б придумати кращий формат, але тоді вам доведеться включити витрати на перетворення та обробку вашого формату, а також на навчання та інструмент редагування та пошуку для XML, які зараз дуже популярні. широко доступні. Деякі з переваг XML включають:

Надмірність

Розмітка XML дуже багатослівна. Наприклад, потрібно вказати кожен кінцевий тег, наприклад </description> у прикладі. Це дозволяє комп'ютеру виявляти типові помилки, наприклад неправильне вкладення.

Самоопис

Зручність читання XML (це текстовий формат) і наявність назв елементів і атрибутів у XML означає, що люди, які переглядають документ XML, часто можуть отримати перевагу в розумінні формату

Мережевий ефект і XML Promise

Будь-який XML-документ можна прочитати та обробити будь-яким інструментом XML. Звичайно, деяким XML-інструментам може знадобитися спеціальна XML-розмітка, але сам XML-формат може бути прочитаний будь-яким XML-аналізатором: ви не можете сказати, що цей XML-документ має оброблятися лише таким-то інструментом.

Це означає, що кожен новий XML-документ підвищує цінність кожного іншого XML-документа та кожного XML-інструменту, а кожен новий XML-інструмент підвищує цінність кожного XML-документа, а отже, і будь-якого іншого інструменту. Сьогодні XML є найпоширенішим форматом такого роду в усьому світі.

Приклади

XML сьогодні дуже широко використовується. Він є основою багатьох стандартів, таких як універсальна мова бізнесу (UBL); Universal Plug and Play (UPnP), що використовується для домашньої електроніки; формати обробки текстів, такі як ODF і OOXML; графічні формати, такі як SVG; він використовується для зв'язку з XMLRPC і веб-службами, підтримується безпосередньо мовами комп'ютерного програмування та базами даних, від гігантських серверів до мобільних телефонів.

Якщо ви двічі клацнете піктограму на робочому столі комп'ютера (піктограма цілком може бути намальована за допомогою SVG), велика ймовірність того, що повідомлення XML буде надіслано з одного компонента робочого столу до іншого. Якщо ви везете автомобіль на ремонт, комп'ютер

двигуна надсилає XML до діагностичних систем механіка. Це вік XML: він всюди.

В ході огляду поняття XML було виявлено основні поняття, базові поняття синтаксису та призначення. Вся отримана інформація (особливо синтаксис) необхідні для майбутнього розуміння перетворень.

1.3 Огляд HTML

Фінальним кроком перетворення є HTML-документ. Проте, для створення вірного перетворення необхідно розуміти принципи HTML-розмітки.

HTML (мова розмітки гіпертексту) — це текстовий підхід до опису того, як структуровано вміст у файлі HTML. Ця розмітка повідомляє веб-браузеру, як відображати текст, зображення та інші форми мультимедіа на веб-сторінці.

HTML є офіційною рекомендацією Консорціуму Всесвітньої павутини (W3C), і його зазвичай дотримуються всі основні веб-браузери, включаючи як настільні, так і мобільні веб-браузери. HTML5 — остання версія специфікації.

Як працює HTML

HTML — це текстовий файл, який містить певний синтаксис, файли та правила іменування, які показують комп’ютеру та веб-серверу, що він містить HTML і його слід читати як такий. Застосовуючи ці правила HTML до текстового файлу практично в будь-якому текстовому редакторі, користувач може написати та створити базову веб-сторінку, а потім завантажити її в Інтернет.

Найпростішою умовністю HTML є включення оголошення типу документа на початку текстового файлу. Це завжди стоїть на першому місці в документі, оскільки це фрагмент, який ствердно повідомляє комп’ютеру, що це файл HTML. Заголовок документа зазвичай виглядає так: <!DOCTYPE html>. Це завжди має бути написано таким чином, без будь-якого вмісту в ньому або розбивання його. Будь-який вміст, який передує цій декларації, не розпізнається комп’ютером як HTML.

Типи документів використовуються не лише для HTML, їх можна застосовувати для створення будь-якого документа, який використовує SGML (стандартна узагальнена мова розмітки). SGML — це стандарт для визначення конкретної мови розмітки, яка використовується. HTML є однією з кількох мов розмітки, до яких застосовуються декларації SGML і doctype.

Інша важлива вимога для створення файлу HTML – це збереження його з розширенням .html. У той час як оголошення doctype передає HTML на комп’ютер зсередини файлу, розширення файлу передає HTML на комп’ютер із зовнішнього боку файлу. Маючи обидва, комп’ютер може визначити, що це файл HTML, незалежно від того, читає він файл чи ні. Це стає особливо важливим під час завантаження файлів в Інтернет, оскільки веб-сервер повинен знати, що робити з файлами, перш ніж він зможе надіслати їх на клієнтський комп’ютер для читання внутрішнього вмісту.

Після написання doctype та збереження як файлу HTML користувач може застосувати всі інші синтаксичні інструменти HTML для налаштування веб-сторінки. Після завершення вони, ймовірно, матимуть кілька файлів HTML, які відповідають різним сторінкам веб-сайту. Важливо, щоб користувач завантажував ці файли в тій самій ієрархії, у якій вони їх зберігали, оскільки кожна сторінка посилається на конкретні шляхи файлів інших сторінок, створюючи зв’язки між ними. Завантаження їх у іншому порядку спричинить розрив посилань і втрату сторінок, оскільки вказані шляхи до файлів не відповідатимуть сторінкам.

Основні елементи HTML

Використовуючи HTML, текстовий файл додатково розмічається додатковим текстом, що описує, як має відображатися документ. Щоб зберегти розмітку окремо від фактичного вмісту HTML-файлу, використовується спеціальний синтаксис HTML. Ці спеціальні компоненти відомі як теги HTML. Теги можуть містити пари ім’я-значення, відомі як атрибути, а фрагмент вмісту, укладений у тег, називається елементом HTML.

Елементи HTML завжди мають відкриваючі теги, вміст посередині та закриваючі теги. Атрибути можуть надавати додаткову інформацію про елемент і включаються у відкриваючий тег. Елементи можна описати одним із двох способів:

Елементи рівня блоку починаються з нового рядка в документі та займають власне місце. Приклади цих елементів включають заголовки та теги абзаців.

Вбудовані елементи не починаються з нового рядка документа і займають лише необхідний простір. Ці елементи зазвичай форматують вміст елементів рівня блоку. Приклади вбудованих елементів включають гіперпосилання та теги текстового формату.

Як використовувати та реалізувати HTML

Оскільки HTML повністю заснований на тексті, файл HTML можна редагувати, просто відкривши його в програмі, наприклад Notepad++, Vi або Emacs. Будь-який текстовий редактор можна використовувати для створення або редагування файлу HTML, і якщо його назва має розширення файлу .html, будь-який веб-браузер, як-от Chrome або Firefox, зможе відобразити файл як веб-сторінку .

Для професійних розробників програмного забезпечення існують різноманітні редактори WYSIWYG для розробки веб-сторінок. NetBeans, IntelliJ, Eclipse та Microsoft Visual Studio надають редактори WYSIWYG як плагіни або як стандартні компоненти, що робить його неймовірно простим у використанні та реалізації HTML.

Ці редактори WYSIWYG також надають засоби усунення несправностей HTML, хоча сучасні веб-переглядачі часто містять плагіни веб-розробників, які висвітлюють проблеми зі сторінками HTML, наприклад відсутній закриваючий тег або синтаксис, який не створює правильно сформований HTML.

Chrome і Firefox містять інструменти розробника HTML, які дозволяють негайно переглядати повний файл HTML веб-сторінки, а також можливість редагувати HTML на льоту та негайно вносити зміни в Інтернет-браузер.

1.4 Огляд XSLT

Перш ніж XSLT, ми повинні спочатку дізнатися про XSL. XSL означає розширювану мову таблиць стилів. Це мова стилю для XML так само, як CSS є мовою стилю для HTML.

XSLT означає перетворення XSL. Він використовується для перетворення документів XML в інші формати (наприклад, перетворення XML у HTML).

Що таке XSL

У документах HTML теги попередньо визначені, але в документах XML теги не визначені заздалегідь. Консорціум World Wide Web Consortium (W3C) розробив XSL для розуміння та стилізації XML-документа, який може діяти як мова стилів на основі XML.

XSL-документ визначає, як браузер має відтворювати XML-документ.

Основні частини документа XSL

XSLT: це мова для перетворення XML-документів у різні інші типи документів.

XPath: це мова для навігації в документах XML.

XQuery: це мова для запитів документів XML.

XSL-FO: це мова для форматування документів XML.

XSLT (Extensible Stylesheet Language Transformations) — це мова, спочатку розроблена для перетворення XML-документів в інші XML-документи[1] або інші формати, такі як HTML для веб-сторінок, звичайний текст або об’єкти форматування XSL, які згодом можуть бути перетворені в інші формати, наприклад PDF, PostScript і PNG.[2] Підтримка JSON і перетворення звичайного тексту була додана в пізніших оновленнях специфікації XSLT 1.0.

Станом на серпень 2022 року останньою стабільною версією мови є XSLT 3.0, яка отримала статус рекомендації в червні 2017 року.

Реалізації XSLT 3.0 підтримують Java, .NET, C/C++, Python, PHP і NodeJS. Бібліотека Javascript XSLT 3.0 також може бути розміщена у веб-браузері. Сучасні веб-браузери також включають вбудовану підтримку XSLT 1.0.[3]

Для трансформації документа XSLT вихідний документ не змінюється; скоріше новий документ створюється на основі вмісту існуючого.[4] Як правило, вхідними документами є файли XML, але може бути використано все, з чого процесор може побудувати модель даних XQuery та XPath, наприклад таблиці реляційної бази даних або системи географічної інформації.[1]

Хоча XSLT спочатку був розроблений як мова спеціального призначення для перетворення XML, ця мова є повною за Тьюрінгом, що робить її теоретично здатною до довільних обчислень.[5]

історія

На XSLT впливають функціональні мови[6] та текстові мови зіставлення шаблонів, такі як SNOBOL і AWK. Його прямим попередником є DSSSL, який робив для SGML те, що XSLT робить для XML.[7]

Модель проектування та обробки

На відміну від інших імперативних мов програмування, таких як C, XSLT є декларативним. Він обробляє один або кілька вихідних документів з однією або кількома таблицями стилів XSLT для створення одного або кількох нових документів. Коли вказано правила середовища обробки, результатом обробки є дерево, яке обробляється подібно до виразу XPath. Це пояснюється тим, що правила використовують зіставлення шаблонів для елемента та збігають його форму з функціями, які визначаються обчисленим деревом результатів. Замість переліку конкретної послідовності дій, ці правила просто вказують, як обробляти відповідні елементи, коли вони зустрічаються.

Типова поведінка процесора така. Спочатку процесор будує вихідне дерево з вхідного XML-документа, припускаючи, що таблицю стилів було прочитано та підготовлено. Потім він обробляє кореневий вузол вихідного

дерева, знаходить найкращий відповідний шаблон для цього вузла в таблиці стилів і оцінює вміст шаблону. Інструкції в кожному режимі зазвичай наказують процесору створити вузол у дереві результатів або обробити інші вузли в дереві виводу так само, як і кореневий вузол. Нарешті, результуюче дерево серіалізується як текст XML або HTML.

Типи носіїв

Елемент <output> може додатково приймати атрибут media-type, який дозволяє встановити тип медіа (або тип MIME) для кінцевого виводу, наприклад: <xsl:output output="xml" media-type="application". /xml"/>. Рекомендація XSLT 1.0 рекомендує більш загальні типи атрибутів text/xml і application/xml, оскільки протягом тривалого часу не було зареєстрованого типу носія для XSLT. За цей час text/xsl став стандартом де-факто. У XSLT 1.0 не вказано, як слід використовувати значення типу носія.

З випуском XSLT 2.0 W3C рекомендував зареєструвати медіа-тип MIME application/xslt+xml [21], який пізніше був зареєстрований в Інтернет-агентстві з присвоєння номерів [22].

Робочі чернетки XSLT до 1.0 використовували text/xsl у своїх прикладах вбудовування, і цей тип був реалізований і продовжує просуватися Microsoft в Internet Explorer [23] і MSXML. Він також широко розпізнається в інструкціях обробки таблиці стилів xml іншими браузерами. Таким чином, на практиці користувачі, які бажають керувати перетворенням у браузері за допомогою цієї інструкції обробки, зобов’язані використовувати цей незареєстрований тип носія.[24]

Як працює XSLT

Таблиця стилів XSLT написана у форматі XML. Він використовується для визначення правил перетворення, які будуть застосовані до цільового документа XML. Процесор XSLT приймає таблицю стилів XSLT і застосовує правила перетворення до цільового XML-документа, а потім генерує відформатований документ у формі XML, HTML або текстового формату.

Наприкінці він використовується форматувальником XSLT для створення фактичного виводу та відображення кінцевому користувачеві.

Перевага XSLT

Перелік переваг використання XSLT:

XSLT забезпечує простий спосіб об’єднання XML-даних у презентацію, оскільки він застосовує визначені користувачем перетворення до XML-документа, а результатом може бути HTML, XML або будь-який інший структурований документ.

XSLT надає Xpath для пошуку елементів/атрибутів у документі XML. Таким чином, це зручніший спосіб перегляду XML-документа, ніж традиційний спосіб, за допомогою мови сценаріїв.

XSLT базується на шаблонах. Тому він більш стійкий до змін у документах, ніж низькорівневі DOM і SAX.

Завдяки використанню XML і XSLT сценарій користувальницького інтерфейсу програми виглядатиме чистим і його буде легше підтримувати.

Шаблони XSLT базуються на шаблоні XPath, який є дуже потужним з точки зору продуктивності для обробки XML-документа.

XSLT можна використовувати як мову перевірки, оскільки він використовує підхід зіставлення шаблонів дерева.

Ви можете змінити результат, просто змінивши перетворення у файлах XSL.

1.5 Огляд існуючих рішень

На поточний момент програмних рішень для проведення подібного роду перетворень в автоматичному режимі в вільному доступі не було знайдено. Суть перетворень досліджена максимально детально з точки зору розміток і додаткових атрибутів розмітки, проте, питання систем автоматичного XSTL-перетворення не досліджено зовсім.

Виходячи з усього вищесказаного, можна зробити висновок про високу актуальність розробки системи проведення XSTL-перетворення в автоматичному режимі.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз варіантів використання

Проектування – один з важливих кроків при розробці програми, який дуже часто ігнорується розробниками-початківцями. Зазвичай вони намагаються втримати все у голові або, у кращому разі, записати деякі важливі відомості на аркуші паперу.

Зазвичай, при проектуванні розробники зображують систему графічно, оскільки людині легко розібратися в такому уявленні. Саме тому замість написання громіздких текстів про кожну нагоду майбутньої програми розробники будують різні діаграми для опису своїх систем. Це допомагає їм не забувати, що потрібно реалізувати у програмі, та швидко вводити в курс справи своїх колег.

Що таке діаграма варіантів використання?

В уніфікованій мові моделювання (UML) діаграма варіантів використання може узагальнити деталі користувачів вашої системи (також відомих як актори) та їх взаємодію з системою. Щоб створити його, ви будете використовувати набір спеціалізованих символів і з'єднувачів. Ефективна діаграма варіантів використання може допомогти вашій команді обговорити та представити:

- Сценарії, у яких ваша система або програма взаємодіє з людьми, організаціями або зовнішніми системами
- Цілі, яких ваша система або програма допомагає цим об'єктам (відомим як актори) досягти
- Область вашої системи

Коли застосовувати діаграми варіантів використання

Діаграма варіантів використання не містить багато деталей — наприклад, не очікуйте, що вона моделюватиме порядок виконання кроків. Натомість правильна діаграма варіантів використання відображає

загальнорівневий огляд взаємозв’язків між варіантами використання, акторами та системами. Експерти рекомендують використовувати діаграми варіантів використання, щоб доповнити більш описовий текстовий варіант використання.

UML — це набір інструментів для моделювання, який можна використовувати для створення діаграм. Випадки використання представлені позначеною овальною формою. Фігурки представляють акторів у процесі, а участь актора в системі моделюється лінією між актором і варіантом використання. Щоб зобразити межу системи, намалюйте рамку навколо самого варіанту використання.

Діаграми варіантів використання UML ідеально підходять для:

- Відображення цілей взаємодії системи та користувача
- Визначення та організація функціональних вимог у системі
- Визначення контексту та вимог до системи
- Моделювання основного потоку подій у випадку використання

Компоненти схеми варіантів використання

Щоб відповісти на запитання «Що таке діаграма варіантів використання?» вам потрібно спочатку зрозуміти його будівельні блоки. Загальні компоненти включають:

- Актори: користувачі, які взаємодіють із системою. Актором може бути особа, організація або зовнішня система, яка взаємодіє з вашою програмою чи системою. Вони мають бути зовнішніми об’єктами, які виробляють або споживають дані.
- Система: певна послідовність дій і взаємодій між акторами та системою. Систему також можна назвати сценарієм.
- Цілі: кінцевий результат більшості випадків використання. Успішна діаграма повинна описувати дії та варіанти, використані для досягнення мети.

Символи та позначення на діаграмі випадків використання

Нотація для діаграми варіантів використання досить проста і не містить стільки типів символів, як інші діаграми UML. Ви можете скористатися цим посібником, щоб навчитися малювати діаграму варіантів використання, якщо вам потрібно відновити знання.

- Варіанти використання: овали горизонтальної форми, які представляють різні способи використання, які можуть мати користувачі.
- Актори: фігурки, які представляють людей, які насправді використовують варіанти використання.
- Асоціації: межа між акторами та варіантами використання. У складних діаграмах важливо знати, які актори пов’язані з якими варіантами використання.
- Системні граничні рамки: рамки, які встановлюють область використання системи для випадків використання. Усі нестандартні випадки використання розглядатимуться поза межами цієї системи. Наприклад, у прикладі з бензопилою, наведеному нижче, Психо-вбивця не входить до кола професій.
- Пакети: форма UML, яка дозволяє об’єднувати різні елементи в групи. Подібно до діаграм компонентів, ці групи представлені у вигляді папок файлів.

На рисунку 2.1 зображено діаграму варіантів використання, яка описує можливі дії користувача в системі.

Рисунок 2.1 — Діаграма варіантів використання

2.2 Проектування пакетної будови системи

Проектування пакетної будови системи розуміє під собою створення пакетної структури системи, яка в подальшому при декомпозиції буде перетворена на дигграму класів.

Діаграми пакетів — це структурні діаграми, які використовуються для відображення організації та розташування різних елементів моделі у формі пакетів. Пакет — це група пов’язаних елементів UML, таких як діаграми, документи, класи або навіть інші пакети. Кожен елемент вкладено в пакет, який зображено на схемі як папку з файлами, а потім упорядковано на схемі ієрархічно. Діаграми пакетів найчастіше використовуються для забезпечення візуальної організації багаторівневої архітектури в будь-якому UML-класифікаторі, такому як програмна система.

Переваги пакетної діаграми

Добре розроблена діаграма пакета надає численні переваги тим, хто хоче створити візуалізацію своєї системи або проекту UML.

- Вони забезпечують чітке уявлення про ієрархічну структуру різних елементів UML у даній системі.
- Ці діаграми можуть спростити складні діаграми класів у добре впорядковані візуальні елементи.
- Вони пропонують цінну видимість високого рівня у великомасштабних проектах і системах.
- Діаграми пакетів можна використовувати для візуального пояснення різноманітних проектів і систем.
- Ці візуальні елементи можна легко оновлювати в міру розвитку систем і проектів.

Основні компоненти пакетної діаграми

Склад діаграми упаковки відносно простий. Кожна діаграма містить лише два символи(рис. 2.2):

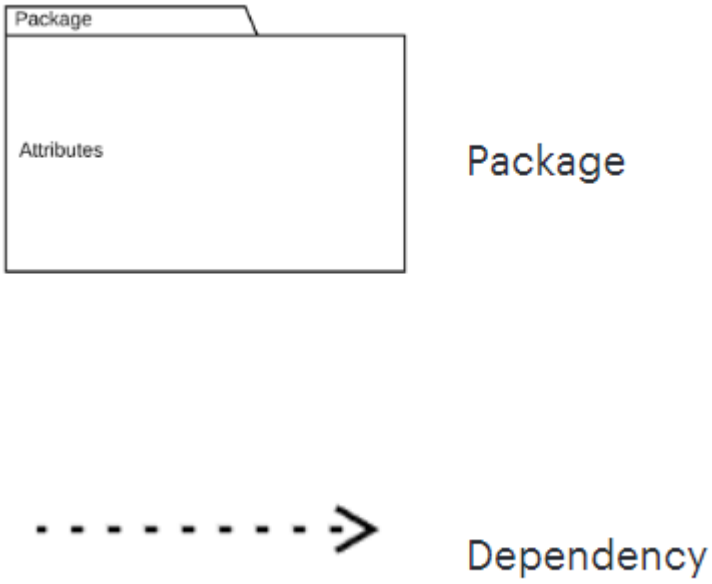

Рисунок 2.2 — Символи діаграми пакетів

Ці символи можна використовувати різними способами для представлення різних ітерацій пакетів, залежностей та інших елементів у системі. Ось основні компоненти, які ви знайдете на діаграмі пакета:

- Пакет: простір імен, який використовується для групування логічно пов’язаних елементів у системі. Кожен елемент, що міститься в пакеті, повинен бути пакетним елементом і мати унікальну назву.
- Packageable element: іменований елемент, який, можливо, належить безпосередньо пакету. Вони можуть включати події, компоненти, варіанти використання та самі пакети. Елементи, що упаковуються, також можуть бути відображені у вигляді прямокутника всередині пакета, позначеного відповідною назвою.
- Залежності: візуальне представлення того, як один елемент (або набір елементів) залежить від іншого або впливає на нього. Залежності поділяються на дві групи: залежності доступу та імпорту. (Додаткову інформацію див. у наступному розділі.)
- Імпорт елемента: спрямований зв’язок між імпортованим простором імен та імпортованим пакетованим елементом. Це використовується для

імпорту вибраних окремих елементів, не вдаючись до імпорту пакета та не роблячи його загальнодоступним у просторі імен.

- Імпорт пакета: спрямований зв’язок між імпортованим простором імен та імпортованим пакетом. Цей тип спрямованого зв’язку додає імена членів імпортованого пакета до його власного простору імен
- Злиття пакетів: спрямований зв’язок, у якому вміст одного пакету розширюється вмістом іншого. По суті, вміст двох пакетів об’єднується для створення нового пакета.

Позначення залежностей на діаграмі пакета

Діаграми пакетів використовуються, зокрема, для зображення залежностей імпорту та доступу між пакетами, класами, компонентами та іншими іменованими елементами у вашій системі. Кожна залежність відображається як з’єднувальна лінія зі стрілкою, яка вказує на тип зв’язку між двома чи більше елементами.

Існує два основних типи залежностей:

Доступ: вказує на те, що один пакет вимагає допомоги з боку функцій іншого пакета.

Залежності також можна розбити на такі категорії:

Використання: трапляється, коли для певного названого елемента потрібен інший для його повного визначення та розгортання. Приклад: клієнт і постачальник.

Абстракція: зв’язує два елементи, що представляють ту саму концепцію на різних рівнях абстракції в системі (зазвичай це стосунки між клієнтом і постачальником).

Розгортання: зображує розгортання артефакту до цілі розгортання.

Використання пакетів з іншими діаграмами UML

Як ми показали раніше в цьому посібнику, пакети — це конструкції UML, які можна використовувати для організації елементів у будь-якому класифікаторі UML у різноманітних діаграмах UML. Схеми пакетів найчастіше використовуються в:

Діаграми варіантів використання: кожен варіант використання зображено як окремий пакет

Діаграми класів: класи організовані в пакети

Пакети також можна використовувати в інших типах моделі UML для організації та впорядкування таких елементів, як класи, об’єкти даних і випадки використання. Поєднуючи структуру пакетної діаграми з іншими діаграмами UML, ви можете спростити будь-який тип моделі, полегшивши її розуміння.

Модельні діаграми

Діаграми пакетів також можна використовувати разом із діаграмами моделей — типом діаграми допоміжної структури UML, яка використовується для зображення логічних, поведінкових або структурних аспектів системи. Навіть прості моделі може бути важко зрозуміти без певної візуальної організації. Використання пакетів може надати користувачам високорівневе представлення моделі з однозначними іменованими посиланнями для кожного з елементів, які вона містить. Крім того, чітко позначені залежності можуть прояснити зв’язки між кожним елементом.

Діаграма пакетів системи зображена на рисунку 2.3.

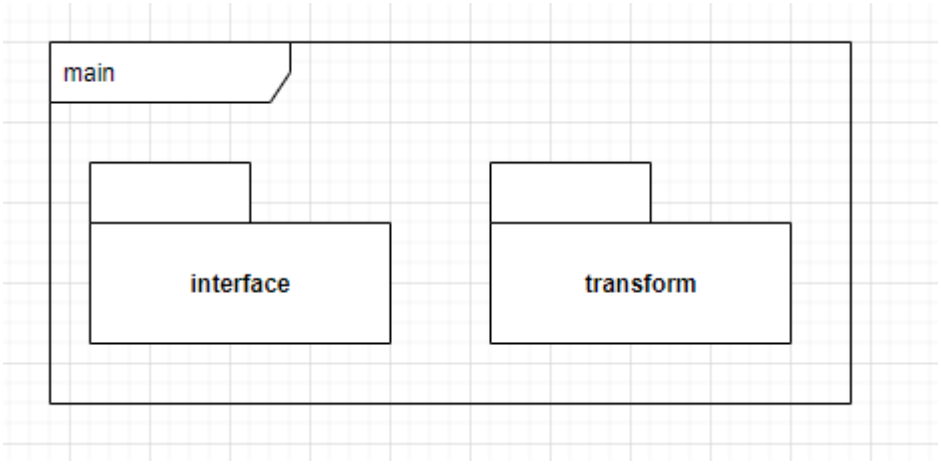

Рисунок 2.3 — Діаграма пакетів

2.3 Проектування внутрішньої будови

В основі будь-якої об'єктно-орієнтованої системи лежить етап проектування структури класів – тому кажуть, що діаграми класів є найпопулярнішими типами UML-діаграми.

Що таке діаграма класів в UML?

Уніфікована мова моделювання (UML) допомагає нам моделювати системи різними способами. Одним із найпоширеніших типів в UML є діаграма класів. Діаграма класів — це тип структурної діаграми, який популярний серед розробників програмного забезпечення для документування архітектури програмного забезпечення. Незалежно від того, чи знайомі ви з UML чи діаграмами класів, наше програмне забезпечення UML розроблено таким чином, щоб воно було простим у використанні.

UML був створений як стандартизована модель для опису методів об'єктно-орієнтованого програмування. Діаграма класів є структурою UML, так само як клас є структурою об’єкта. Різні компоненти діаграми класів можуть представляти фактичний програмований клас, базові об’єкти або взаємодії між класами та об’єктами.

Сама форма класу складається з трьох рядів прямокутників. Верхній рядок містить назву класу, середній рядок містить атрибути класу, а нижній рядок містить методи або операції, доступні для класу. Класи та підкласи згруповані разом, щоб показати статичний зв’язок між кожним об’єктом.

Переваги діаграм класів

Діаграми класів пропонують ряд переваг для будь-якої організації. Використовуйте діаграми класів UML, щоб:

- Проілюструйте моделі даних для інформаційних систем, незалежно від того, наскільки вони прості чи складні.
- Краще зрозуміти загальний огляд схем програми.
- Візуально виражайте будь-які конкретні потреби системи та поширюйте цю інформацію по всьому бізнесу.

- Створюйте детальні діаграми, які висвітлюють будь-який конкретний код, який необхідно запрограмувати та реалізувати в описаній структурі.
- Надайте незалежний від реалізації опис типів, що використовуються в системі, які пізніше передаються між її компонентами.

Базові компоненти діаграми класів

Стандартна діаграма класів складається з трьох розділів:

- Верхня частина: містить назву класу. Цей розділ завжди є обов'язковим, незалежно від того, чи йдеться про класифікатор чи об'єкт.
- Середній розділ: містить атрибути класу. Використовуйте цей розділ, щоб описати якості класу. Це потрібно лише при описі конкретного екземпляра класу.
- Нижній розділ: містить операції класу (методи). Відображається у форматі списку, кожна операція займає окремий рядок. Операції описують, як клас взаємодіє з даними.

Модифікатори доступу учасників

Усі класи мають різні рівні доступу залежно від модифікатора доступу (видимості). Ось рівні доступу з відповідними символами:

- Публічний (+)
- Приватний (-)
- Захищений (#)
- Пакет (~)
- Похідний (/)
- Статичний (підкреслений)

Діапазон учасників

Є дві області для членів: класифікатори та екземпляри.

Класифікатори є статичними членами, тоді як екземпляри є конкретними екземплярами класу. Якщо ви знайомі з базовою теорією ООП, це не є чимось новаторським.

Додаткові компоненти діаграми класів

Залежно від контексту, класи на діаграмі класів можуть представляти основні об’єкти, взаємодії в додатку або класи, які потрібно запрограмувати. Щоб відповісти на запитання "Що таке діаграма класів в UML?" ви повинні спочатку зрозуміти його основний склад.

Класи: шаблон для створення об’єктів і реалізації поведінки в системі. В UML клас представляє об’єкт або набір об’єктів, які мають спільну структуру та поведінку. Вони представлені прямокутником, який містить рядки імені класу, його атрибутів і операцій. Коли ви малюєте клас на діаграмі класів, вам потрібно заповнити лише верхній рядок — інші необов’язкові, якщо ви хочете надати більше деталей.

- Назва: перший рядок у формі класу.
- Атрибути: другий рядок у формі класу. Кожен атрибут класу відображається в окремому рядку.
- Методи: Третій ряд у формі класу. Також відомі як операції, методи відображаються у форматі списку з кожною операцією в окремому рядку.

Сигнали: символи, що представляють односторонній асинхронний зв’язок між активними об’єктами.

Типи даних: Класифікатори, які визначають значення даних. Типи даних можуть моделювати як примітивні типи, так і перерахування.

Пакети: форми, призначені для організації пов’язаних класифікаторів на діаграмі. Вони позначаються великим прямокутником із вкладками.

Інтерфейси: набір сигнатур операцій та/або визначень атрибутів, які визначають узгоджений набір поведінки. Інтерфейси подібні до класів, за винятком того, що клас може мати екземпляр свого типу, і інтерфейс повинен мати принаймні один клас для його реалізації.

Перерахування: представлення типів даних, визначених користувачем. Перелік включає групи ідентифікаторів, які представляють значення переліку.

Об’єкти: екземпляри класу або класів. До діаграми класів можна додавати об’єкти для представлення конкретних або прототипних екземплярів.

Артефакти: елементи моделі, які представляють конкретні сутності в системі програмного забезпечення, такі як документи, бази даних, виконувані файли, компоненти програмного забезпечення тощо.

Взаємодії

Термін «взаємодії» відноситься до різних відносин і зв'язків, які можуть існувати в діаграмах класів і об'єктів. Деякі з найпоширеніших взаємодій включають:

Успадкування: процес набуття дочірнім або підкласом функцій батьківського або суперкласу, також відомий як узагальнення. Його символізує пряма з’єднана лінія із закритою стрілкою, яка вказує на суперклас.

У цьому прикладі об’єкт «Автомобіль» успадковує всі атрибути (швидкість, кількість пасажирів, паливо) і методи (go(), stop(), changeDirection()) батьківського класу («Транспортний засіб»). до конкретних атрибутів (тип моделі, кількість дверей, виробник автомобіля) і методів власного класу (Radio(), windshieldWiper(), ac/heat()). На діаграмі класів успадкування показано за допомогою суцільної лінії з замкнутою порожнистою стрілкою.

Двонаправлена асоціація: зв’язок за умовчанням між двома класами. Обидва класи усвідомлюють один одного і свої стосунки з одним. Ця асоціація представлена прямою лінією між двома класами.

У наведеному вище прикладі клас Car і RoadTrip взаємопов’язані. На одному кінці рядка автомобіль приймає асоціацію "assignedCar" зі значенням кратності 0..1, тому, коли екземпляр RoadTrip існує, він може або мати один екземпляр Car, пов’язаний з ним, або жодного пов’язаного автомобіля. з цим. У цьому випадку потрібен окремий клас Caravan зі значенням кратності 0..*, щоб продемонструвати, що RoadTrip може мати кілька екземплярів Cars, пов’язаних із ним. Оскільки один екземпляр автомобіля може мати кілька

зв’язків «getRoadTrip» — іншими словами, один автомобіль може здійснювати кілька подорожей — значення кратності встановлено на 0..*

Односпрямована асоціація: Трохи менш поширений зв’язок між двома класами. Один клас знає про інший і взаємодіє з ним. Односпрямована асоціація моделюється прямою сполучною лінією, яка вказує відкриту стрілку від відомого класу до відомого класу.

Наприклад, під час вашої подорожі через Арізону ви можете натрапити на радар, де відеокамера фіксує вашу активність за кермом, але ви не дізнаєтесь про це, доки не отримаєте сповіщення поштою. Він не намальований на зображенні, але в цьому випадку значення кратності становитиме 0..* залежно від того, скільки разів ви проїжджаєте повз камеру контролю швидкості.

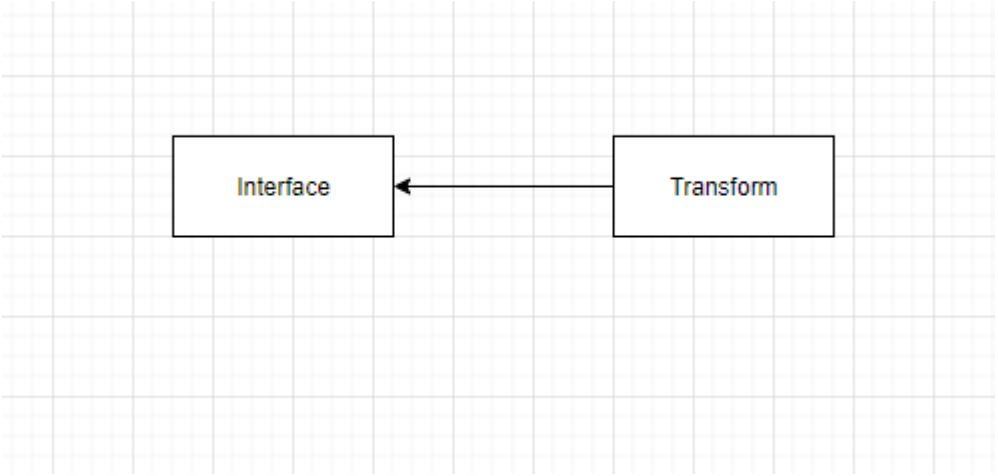

Рисунок 2.4— Діаграма класів

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір засобів розробки

Для створення програмного продукту в першу чергу необхідно розглянути інструменти, які будуть використовуватися при розробці, виявити їх особливості, сильні і слабкі сторони для розуміння можливого функціоналу, який може бути створений.

Python — це інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою, розроблена Гвідо ван Россумом. Спочатку він був випущений у 1991 році. Назва «Python» була розроблена, щоб бути безтурботною та веселою, нагадуючи британській комедійній групі Monty Python. Python, відомий своєю мовою, зручною для початківців, замінив Java як найпоширенішу мову початкового рівня, оскільки він справляється з більшою частиною складності для користувача, дозволяючи новачкам зосередитися на повному розумінні концепцій програмування, а не на найдрібніших деталях.

Python використовується для веб-розробки на стороні сервера, розробки програмного забезпечення, математики та системних сценаріїв, і популярний для швидкої розробки додатків, а також як мова сценаріїв або мова конектора для підключення існуючих компонентів через вбудовані дані високого рівня. структури, динамічні типи та динамічне прив'язування. Завдяки легкому для вивчення синтаксису та акценту на зручності читання Python зменшує витрати на обслуговування програми. Крім того, підтримка Python модулів і пакетів полегшує повторне використання модульних програм і коду. Python — це мова спільноти з відкритим кодом, тому багато незалежних програмістів постійно створюють бібліотеки та функції для неї.

Випадки використання Python

- Створення веб-додатків на сервері

- Створення робочих процесів, які можна використовувати в поєднанні з програмним забезпеченням
- Підключення до систем баз даних
- Читання та редагування файлів
- Виконання складної математики
- Обробка великих даних
- Швидке створення прототипів
- Розробка готового програмного забезпечення

У професійному плані Python чудово підходить для серверної веб-розробки, аналізу даних, штучного інтелекту та наукових обчислень. Розробники також використовують Python для створення інструментів продуктивності, ігор і настільних програм.

Особливості та переваги Python

- Сумісний із різними платформами, включаючи Windows, Mac, Linux, Raspberry Pi та інші
- Використовує простий синтаксис, який можна порівняти з англійською мовою, що дозволяє розробникам використовувати менше рядків, ніж інші мови програмування
- Працює на системі інтерпретатора, яка дозволяє негайно виконувати код, швидко відстежуючи прототипування
- Можна обробляти процедурним, об’єктно-орієнтованим або функціональним способом

Синтаксис Python

- Дещо схожий на англійську мову, з математичним впливом, Python створений для зручності читання
- На відміну від інших мов, які використовують крапку з комою та/або круглі дужки для завершення команди, Python використовує нові рядки для тієї самої функції

– Визначає область (тобто цикли, функції, класи) за допомогою відступів, використовуючи пробіли, а не фігурні дужки (також відомі як фігурні дужки)

Гнучкість Python

Python, мова з динамічною типізацією, є особливо гнучкою, усуває жорсткі правила створення функцій і пропонує більшу гнучкість у вирішенні проблем за допомогою різноманітних методів. Він також дозволяє компілювати та запускати програми аж до проблемної області, оскільки використовує перевірку типу під час виконання, а не під час компіляції.

Менші частини Python

З іншого боку, Python нелегко підтримувати. Одна команда може мати кілька значень залежно від контексту, оскільки Python є мовою з динамічним типом. І підтримувати програму Python, коли вона зростає в розмірі та складності, може бути дедалі складніше, особливо знаходити та виправляти помилки. Користувачам знадобиться досвід для розробки коду або написання модульних тестів, які спрощують обслуговування.

Швидкість — ще одна слабка сторона Python. Його гнучкість, оскільки він динамічно типізований, вимагає значної кількості посилань, щоб отримати правильне визначення, що сповільнює продуктивність. Це можна пом’якшити, використовуючи альтернативну реалізацію Python (наприклад, PyPy).

Python і ШІ

Дослідники ШІ є фанатами Python. Google TensorFlow, а також інші бібліотеки (scikit-learn, Keras) створюють основу для розробки ШІ завдяки зручності та гнучкості, яку він пропонує користувачам Python. Ці бібліотеки та їх доступність є критично важливими, оскільки вони дозволяють розробникам зосередитися на зростанні та розбудові.

3.2 Розробка графічного інтерфейсу

Графічний інтерфейс користувача, розроблений наприкінці 1970-х дослідницькою лабораторією Xerox Palo Alto та комерційно розгорнутий в операційних системах Macintosh від Apple і Windows від Microsoft, був розроблений як відповідь на проблему неефективного використання ранніх текстових інтерфейсів командного рядка. для звичайного користувача.

Графічний інтерфейс користувача стане стандартом дизайну, орієнтованого на користувача, у програмуванні прикладного програмного забезпечення, надаючи користувачам можливість інтуїтивно керувати комп'ютерами та іншими електронними пристроями шляхом прямого маніпулювання графічними значками, такими як кнопки, смуги прокрутки, вікна, вкладки, меню, курсори і вказівний пристрій миші. Багато сучасних графічних інтерфейсів користувача мають сенсорний екран і можливості взаємодії за допомогою голосових команд.

Як працює графічний інтерфейс користувача?

Принципи проектування графічного інтерфейсу користувача відповідають шаблону програмного забезпечення «модель–подання–контролер», який відокремлює внутрішнє представлення інформації від способу, у який інформація подається користувачеві, у результаті чого створюється платформа, на якій користувачі показують, які функції можливі, а не вимагають введення кодів команд. Користувачі взаємодіють з інформацією, маніпулюючи візуальними віджетами, які призначені для реагування відповідно до типу даних, які вони зберігають, і підтримують дії, необхідні для виконання завдання користувача.

Зовнішній вигляд або «шкіра» операційної системи або прикладного програмного забезпечення може бути перероблено за бажанням через природу графічного інтерфейсу користувача, незалежного від функцій програми. Програми зазвичай реалізують власні унікальні елементи відображення графічного інтерфейсу користувача на додаток до елементів графічного інтерфейсу користувача, які вже присутні в існуючій операційній системі.

Типовий графічний інтерфейс користувача також включає стандартні формати для представлення графіки та тексту, що робить можливим обмін даними між програмами, що працюють під стандартним програмним забезпеченням розробки графічного інтерфейсу користувача.

Тестування графічного інтерфейсу користувача відноситься до систематичного процесу генерації тестів для оцінки функціональності системи та її елементів дизайну. Інструменти тестування графічного інтерфейсу користувача, які є ручними або автоматизованими та зазвичай реалізуються сторонніми операторами, доступні за різними ліцензіями та підтримуються різноманітними платформами. Серед популярних прикладів: Tricentis Tosca, Squish GUI Tester, Unified Functional Testing (UFT), Maveryx, Appium і eggPlant Functional.

Графічний інтерфейс користувача складається з одного вікна, яке призначене для вибору файлів (рис. 3.1).

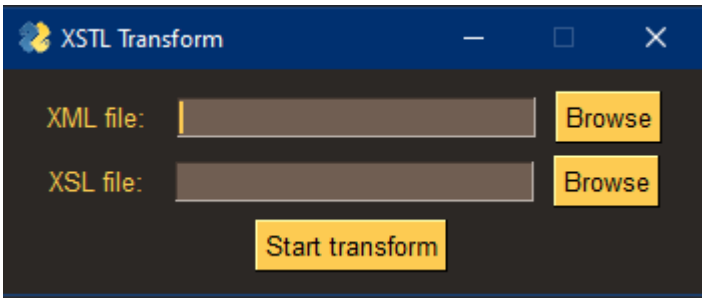

Рисунок 3.1 — Графічний інтерфейс користувача

3.3 Тестування системи

Тестування програмного забезпечення — це метод перевірки того, чи фактичний програмний продукт відповідає очікуваним вимогам, і переконатися, що програмний продукт без дефектів. Він передбачає виконання компонентів програмного забезпечення/системи за допомогою ручних або автоматизованих інструментів для оцінки однієї чи кількох цікавих властивостей. Метою тестування програмного забезпечення є виявлення помилок, прогалин або відсутніх вимог на відміну від фактичних вимог.

Ось приклад XSLT-схеми, яка визначає правила перетворення XML в HTML. В цьому прикладі ми припустимо, що вхідний XML містить список книг, а ми хочемо відобразити їх у вигляді таблиці HTML

```
<!-- books.xml - вхідний XML-документ -->
<library>
  <book>
    <title>Book 1</title>
    <author>Author 1</author>
    <year>2020</year>
  </book>
  <book>
    <title>Book 2</title>
    <author>Author 2</author>
    <year>2021</year>
  </book>
</library>

 <!-- transform.xslt - XSLT-схема для перетворення XML в HTML -->

<xsl:stylesheet
version="1.0"xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" indent="yes"/>
```

```
<xsl:template match="/">
  <html>
    <head>
      <title>Library</title>
    </head>
    <body>
      <table>
        <tr>
          <th>Title</th>
          <th>Author</th>
          <th>Year</th>
        </tr>
        <xsl:apply-templates select="library/book"/>
      </table>
    </body>
  </html>
</xsl:template>

<xsl:template match="book">
  <tr>
    <td>
      <xsl:value-of select="title"/>
    </td>
    <td>
      <xsl:value-of select="author"/>
    </td>
    <td>
      <xsl:value-of select="year"/>
    </td>
```

```
    </tr>
  </xsl:template>
</xsl:stylesheet>
```

В цьому прикладі XSLT-схема має два шаблони:

1. Головний шаблон (**<xsl:template match="/">**) визначає загальну структуру HTML-сторінки. В ньому ми визначаємо заголовок сторінки та таблицю для відображення списку книг.
2. Шаблон для елемента **<book>** (**<xsl:template match="book">**) визначає, як кожен окремий елемент **<book>** повинен бути відображений у рядку таблиці HTML. В цьому шаблоні ми використовуємо **<xsl:value-of>** для виведення значень елементів **<title>**, **<author>** та **<year>**.

В результаті проведення тестування неточностей роботи не було виявлено.

ВИСНОВКИ

Основною метою роботи було створення програми конвертації XML документів в HTML-формат засобами XSLT.

Для досягнення поставленої мети було виконано наступні завдання:

- провести аналіз поняття прикладного програмного забезпечення;
- провести аналіз мови XSTL;
- провести аналіз варіантів використання;
- спроектувати пакетну будову;
- спроектувати внутрішню будову;
- обрати засоби реалізації;
- створити графічний інтерфейс користувача;
- провести тестування системи.

Завдяки чіткому виконанню поставлених на початку роботи завдань в результаті виконання роботи було отримано повнофункціональний додаток конвертації XML документів в HTML-формат засобами XSLT, який протестовано і можна використовувати в реальних умовах.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Transformation". 2012-09-19.
2. "XML Output Method". 2012-09-19.
3. "What is XSLT Used For?". 2018-02-07.
4. "Introduction". XSL Transformations (XSLT) Version 1.0 W3C Recommendation. W3C. 16 November 1999. Retrieved November 7, 2012.
5. XSLT Version 2.0 Is Turing-Complete: A Purely Transformation Based Proof
6. Michael Kay. "What kind of language is XSLT?". IBM. Retrieved July 8, 2016.
7. "A Proposal for XSL". W3C. Retrieved November 7, 2012.
8. "XML and Semantic Web W3C Standards Timeline" (PDF). Archived from the original (PDF) on 2013-04-24. Retrieved 2012-02-04.
9. "XSL Transformations (XSLT)". W3.org. 1999-11-16. Retrieved 2014-07-12.
10. "XSL Transformations (XSLT) Version 1.1". W3.org. 2001-08-24. Retrieved 2014-07-12.
11. "XML Path Language (XPath) 2.0 (Second Edition)". W3.org. 2010-12-14. Retrieved 2014-07-12.
12. "XSL Transformations (XSLT) Version 2.0". W3.org. 2007-01-23. Retrieved 2014-07-12.
13. "XML and Semantic Web W3C Standards Timeline" (PDF). 2012-02-04. Archived from the original (PDF) on 2013-04-24. Retrieved 2012-02-04.
14. "What's New in XSLT 3.0?". w3. Retrieved 6 January 2014.
15. Kay, Michael. "A Streaming XSLT Processor". Balisage: The Markup Conference 2010 Proceedings. Retrieved 15 February 2012.
16. Flatt, Amelie; Langner, Arne; Leps, Olof (2022), "Phase III: Generating Artifacts from the Model", Model-Driven Development of Akoma Ntoso Application Profiles, Cham: Springer International Publishing, pp. 31–37, doi:10.1007/978-3-031-14132-4_5, ISBN 978-3-031-14131-7, retrieved 2023-01-07

17."XSL Transformations (XSLT) Version 2.0 (Second Edition)". www.w3.org. Retrieved 2023-02-07. Example: Multiple Result Documents
18."Discover the Wonders of XSLT: XSLT Quirks". Archived from the original on 2011-07-09. Retrieved 2011-02-11. XSLT is a very specialized language with a distinct declarative flavor.
19.Kay, Michael. "What kind of language is XSLT?". IBM. Retrieved 13 November 2013.
20."Saxonica: XSLT and XQuery". www.saxonica.com. Retrieved 2022-06-29.
21."XSL Transformations (XSLT) Version 2.0". W3C. Retrieved 19 October 2012.
22."Application Media Types". IANA. Retrieved 19 October 2012.
23."XSLT Requirements for Viewing XML in a Browser". Microsoft. Retrieved 19 October 2012.
24.Kay, Michael (2008). XSLT 2.0 and XPath 2.0 Programmer's Reference. Wiley. p. 100. ISBN 978-0-470-19274-0.
25."XSL Transformations (XSLT) Version 1.0: W3C Recommendation – Embedding Stylesheets". W3C. 16 November 1999. Retrieved 20 September 2016.
26."The XSLT C library for GNOME: libxslt". Retrieved 23 November 2012.
27."The XSLT C library for GNOME: The xsltproc tool". Retrieved 23 November 2012.
28."xsltproc man page". Retrieved 23 November 2012.
29."New package: libxslt". Retrieved 23 November 2012.
30."The WebKit Open Source Project - XSLT". Retrieved 2009-10-25.
31."The XML C parser and toolkit of Gnome: Python and bindings". Retrieved 23 November 2012.
32."XML::LibXSLT - Interface to the GNOME libxslt library". CPAN. Retrieved 23 November 2012.
33."libxslt-ruby". Retrieved 23 November 2012.
34."libxml". Retrieved 23 November 2012.

35. "cl-libxml2 High-level wrapper around libxml2 and libxslt libraries".
36. "TclXML". Retrieved 21 May 2013.
37. "libxml++". sourceforge.net. Retrieved 23 November 2012.
38. "Command Line Transformation Utility (msxsl.exe)". Microsoft. Retrieved 22 October 2012.
39. "Saxon-JS". Saxonica. Retrieved 6 September 2018.
40. "Issue 58151: Fails to load xml file on local file system using XMLHttpRequest".
41. Saxon: Anatomy of an XSLT processor - Article describing implementation & optimization details of a popular XSLT processor.
42. Lumley, John; Kay, Michael (June 2015). "Improving Pattern Matching Performance in XSLT". XML London 2015: 9–25. doi:10.14337/XMLLondon15.Lumley01. ISBN 978-0-9926471-2-4.
43. Kay, Michael; Lockett, Debbie (June 2014). "Benchmarking XSLT Performance". XML London 2014: 10–23. doi:10.14337/XMLLondon14.Kay01. ISBN 978-0-9926471-1-7.

ДОДАТКИ

Додаток А – Технічне завдання

ЗАТВЕРДЖЕНО
1116130.01314-01-ЛЗ

ПРОГРАМА КОНВЕРТАЦІЇ XML ДОКУМЕНТІВ В HTML-ФОРМАТ ЗАСОБАМИ XSLT

Технічне завдання

1116130.01314-01

Листів 11

2023

ЗМІСТ

1. ВВЕДЕННЯ

Метою розробки цього проекту є використання XML для зберігання різного роду даних. Тому, що «чистий» XML не читабельний і не зрозумілий рядовому користувачу. Для підвищення читабельності використовують мову трансформації XSTL, яка перетворює XML в звичний і читабельний HTML. Використання XSTL є досить трудомістким процесом, який вимагає часу. Через це, основною метою роботи є створення програми конвертації XML документів в HTML-формат засобами XSLT.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 07.12.22 №1209ст ректора Українського державного університету науки і технологій “Про призначення наукових керівників та затвердження тем бакалаврських робіт” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

Тема дипломної роботи – “Розробка програми конвертації XML документів в HTML-формат засобами XSLT ”. Керівник – Андрющенко В. О.

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення: Продукт що розробляється дозволяє користувачам можливість ефективно та зручно конвертувати XML-документи в HTML-формат засобами XSLT..

Експлуатаційне призначення: Зручне та швидке перетворення XML-документів: Програма надає можливість швидкого та ефективного перетворення XML-документів у HTML-формат без необхідності вручну писати складні коди або скрипти. Призначення полягає у наданні зручного та надійного інструменту для конвертації XML документів в HTML-формат, що допомагає полегшити роботу з XML-даними та забезпечує їх зрозуміле та зручне використання.

4. ВИМОГИ ДО ПРОГРАМИ

4.1. Вимоги до функціональних характеристик

Можливість завантажити вихідний XML-документ з локального файлу або ввести його вручну, отримання HTML-коду в результаті XSLT-трансформації та розробка зручного та інтуїтивно зрозумілого графічного інтерфейсу для взаємодії з користувачем.

4.2. Вимоги до надійності

Вимоги до надійності наступні: забезпечення можливості завантажити вихідний XML-документ з локального файлу або ввести його вручну; збереження структури та візуального представлення XML-документа в HTML-форматі.

4.3. Умови експлуатації

Комп'ютери, на яких запускається програма, повинні мати достатній рівень апаратної потужності для ефективного виконання операцій з конвертації XML в HTML та роботи з графічним інтерфейсом програми.

4.4. Вимоги до складу і параметрів технічних засобів

Технічні засоби повинні мати достатню продуктивність процесора для швидкої обробки XML-документів та виконання XSLT-трансформації

4.5. Вимоги до інформаційної і програмної сумісності

Програма має функціонувати під управлінням ОС Windows 10\11.

Програма написана на мові програмування Python.

4.6. Вимоги до маркування і упаковки

На упаковці або маркувальних матеріалах повинна бути вказана назва програми, версія та інформація про розробника, в документації можуть бути вказані сумісні операційні системи, версії браузерів

4.7. Вимоги до транспортування і зберігання

Упаковка повинна бути витримана з відповідних матеріалів, щоб забезпечити захист програмного забезпечення від пошкоджень під час транспортування та зберігання.

5. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації мають входити:

1. специфікація;
2. текст програми.

Програмна документація повин на відповідати вимогам ДСТУ [1].

6. СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиця 5.1. – Стадії та етапи розробки

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1	Вступ	12.09.22 – 26.10.22	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	27.10.22 – 04.03.23	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	05.03.23 – 31.04.23	
4	Постановка задачі, технічне завдання	01.05.23 – 07.05.23	30%
5	Техніко-економічні показники	08.05.23 – 15.05.23	
6	Розробка інструментальних засобів дослідження	16.05.23 – 21.05.23	
7	Виконання досліджень	22.05.23 – 28.05.23	60%
8	Оформлення тез доповідей	29.05.23 – 01.06.23	
9	Оформлення статті у фаховий журнал	02.06.23 – 06.06.23	
10	Оформлення пояснювальної записки	07.06.23 – 11.06.23	
11	Розробка демонстраційних матеріалів	12.06.23 – 18.06.23	100%
12	Подання кваліфікаційної роботи до кафедри	21.06.23	
13	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	28.06.23	

7. ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Контроль виконання здійснює керівник розробки Андрющенко В. О. Прийом здійснюється уповноваженою комісією.

8. БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

Додаток Б – Специфікація

ЗАТВЕРДЖЕНО
1116130.01314-01-ЛЗ

ПРОГРАМА КОНВЕРТАЦІЇ XML ДОКУМЕНТІВ В HTML-ФОРМАТ ЗАСОБАМИ XSLT

Специфікація

1116130.01314-01

Листів 2

2023

Специфікації

Таблиця 5.2. – Специфікації

Позначення	Найменування	Примітка
	Документація	
1116130.01314-01-ЛЗ	Лист затвердження	
1116130.01314-01	Технічне завдання	
1116130.01314-01-ЛЗ	Лист затвердження	
1116130.01314-01	Специфікація	
1116130.01314-01 12 01-ЛЗ	Лист затвердження	
1116130.01314-01 12 01	Текст програми	

Додаток В – Листи затвердження

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
07.12.2022

ПРОГРАМА КОНВЕРТАЦІЇ XML ДОКУМЕНТІВ В HTML-ФОРМАТ ЗАСОБАМИ XSLT

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01314-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
07.12.22
Керівник розробки
_____ Вадим АНДРЮЩЕНКО
07.12.22
Виконавець
_____ Наталя ТКАЧ
07.12.22
Нормконтролер
_____ Світлана ВОЛКОВА
07.12.22

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
07.12.2022

ПРОГРАМА КОНВЕРТАЦІЇ XML ДОКУМЕНТІВ В HTML-ФОРМАТ ЗАСОБАМИ XSLT

Специфікація
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01314-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
07.12.22
Керівник розробки
_____ Вадим АНДРЮЩЕНКО
07.12.22
Виконавець
_____ Наталя ТКАЧ
07.12.22
Нормконтролер
_____ Світлана ВОЛКОВА
07.12.22

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
07.12.2022

ПРОГРАМА КОНВЕРТАЦІЇ XML ДОКУМЕНТІВ В HTML-ФОРМАТ ЗАСОБАМИ XSLT

Текст програми
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01314-01 12 01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
07.12.22
Керівник розробки
_____ Вадим АНДРЮЩЕНКО
07.12.22
Виконавець
_____ Ткач НАТАЛЯ
07.12.22
Нормконтролер
_____ Світлана ВОЛКОВА
07.12.22

2023

Додаток Г – Текст програми

ЗАТВЕРДЖЕНО
1116130.01314-01 12 01-ЛЗ

ПРОГРАМА КОНВЕРТАЦІЇ XML ДОКУМЕНТІВ В HTML-ФОРМАТ ЗАСОБАМИ XSLT

Текст програми

1116130.01314-01 12 01

Листів 2

2023

Програма

```
//Transform.py
import os.path
import lxml.etree as ET
import webbrowser

def transform_func(xml_path,
xslt_path):
dom = ET.parse('test.xml')
    xslt = ET.parse('test.xsl')
    transform = ET.XSLT(xslt)
    newdom = transform(dom)
    print(ET.tostring(newdom,
pretty_print=True))
    newdom.write("output-doc.html",
pretty_print=True)
    url = "file://" + os.path.realpath
('output-doc.html')
    webbrowser.open(url)

//Interface.py
from PySimpleGUI
 import Window, FileBrowse, In,
Button,
 Column, WINDOW_CLOSED,
theme, Text
from transform import transform_func
theme('DarkAmber')
file_list_column = [
    [
Text("XML file: "),
In(size=(25, 1), enable_events=True,
 key="-FILE-XML-"),
FileBrowse(),
],
[
Text("XSL file: "),
In(size=(25, 1), enable_events=True,
 key="-FILE-XSL-"),
 FileBrowse(),
    ],
 [Button("Start transform",
 key="-START TRANSFORM-")],
]
layout = [
    [
Column(file_list_column,
 element_justification='c')
    ]
]
window = Window("XSTL
Transform",
 layout, element_justification='c')
while True:
event, values = window.read()
 if event == "Exit" or event ==
WINDOW_CLOSED:
        break
 if event == "-START
TRANSFORM-":
  transform_func(values['-FILE-XML-
'],
values['-FILE-XSL-'])

//_init_.py

__version__ = "4.9.2"


def get_include():
    """
    Returns a list of header include
paths (for lxml itself, libxml2
    and libxslt) needed to compile C
code against lxml if it was built
    with statically linked libraries.
    """
    import os
    lxml_path = __path__[0]
    include_path =
os.path.join(lxml_path, 'includes')
    includes = [include_path,
lxml_path]

    for name in
os.listdir(include_path):
        path = os.path.join(include_path,
name)
        if os.path.isdir(path):
```

```
            includes.append(path)

    return includes
```

Шаблони перетворення

```
//xml
<?xml version="1.0"
encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
   <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
   <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
</catalog>
//xslt
<?xml version="1.0"
encoding="UTF-8"?>

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/
XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
   <tr bgcolor="#9acd32">
    <th>Title</th>
    <th>Artist</th>
     <th>Country</th>
    <th>Company</th>
   </tr>
   <xsl:for-each select="catalog/cd">
   <tr>
    <td><xsl:value-of
select="title"/></td>
     <td><xsl:value-of
select="artist"/></td>
      <td><xsl:value-of
select="country"/></td>
      <td><xsl:value-of
select="company"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
//xslt-1

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/
XSL/Transform">
 <xsl:template match="/">
  <html>
   <head>
    <title>Мобільні
телефони</title>
   </head>
   <body>
    <h1>Мобільні телефони
(сортування за ціною)</h1>
     <ul>
      <xsl:apply-templates
select="phones/phone">
```

```
        <xsl:sort select="price" data-type="number"/>
      </xsl:apply-templates>
    </ul>
   </body>
  </html>
 </xsl:template>

 <xsl:template match="phone">
  <li>
   <xsl:value-of select="concat(name, ' - ', price)"/>
  </li>
 </xsl:template>
</xsl:stylesheet>
```