Софийски университет „Св. Кл. Охридски”

Факултет по математика и информатика

Специалност
„Софтуерно инженерство”

Курсов проект

XML технологии за семантичен уеб

Зимен семестър, 2023/2024 год.

Тема №68:
„Каталог на игрални филми“

Автори:

Атанас Ников,ФN№ 3MI0600006

Светлозар Стефанов,ФN№ 4MI0600030

Ръководители:

доц. д-р Александър Димов

гл. ас. д-р Явор Данков

януари, 2024

София

Съдържание

1 Въведение

Темата, избрана от нашия екип, е с нестихваща популярност от самата зора на кинематографското изкуство. В наши дни киното е едно достъпно и модерно изкуство, което вълнува хора от всякакви възрасти и култури. Нашият екип не прави изключение - като големи почитатели на големия екран, ние решихме да представим своя прочит за филмова колекция, избирайки най-любимите си продукции. Подбраните филми в колекцията са не просто резултат на нашата субективна оценка - всички те са световно признати филми, присъстващи в престижни класации на IMDb с изключително високи потребителски оценки и спечелили множество награди за кино.

Този каталог изпълнява простата, но и трудна задача да препоръчаме съвкупност от изключително добре направени филми с разнообразни жанрове и сюжети, които всеки трябва да гледа. Посредством възможностите, предлагани от XML и съпътстващите езика технологии, можем да предоставим добра репрезентация на тази колекция от филми:

- Организация на данните във филмовата колекция

  Цялата информация, изграждаща нашия каталог, може с помощта на XML да се подреди в удобна йерархична структура, която ни помага да следим позициите на всички елементи, техните взаимоотношения, и да разширяваме и редактираме данните по лесен начин.

- Валидация на информацията

  Можем да валидираме постъпилата в каталога информация и самата структура на документа чрез DTD (Document Type Definition).

- Трансформация на данните и стилизация

  XSL (Extensible Stylesheet Language) в комбинация с XSL-FO предлага възможност за интерпретиране на информацията в XML документа към други файлови формати, което допринася за достигане до крайния резултат - завършен PDF документ. Нещо повече - благодарение на XSLT имаме възможността да придадем подобаващ облик на агрегираната информация в каталога и да я представим по лесен за възприемане от потребителска гледна точка начин.

Настоящият документ проследява стъпка по стъпка направата на курсовия проект и описва в детайли всички важни за филмовия каталог характеристики.

2 Анализ на решението

2.1 Работен процес

Началото на работния процес бе поставено с избора на конкретни заглавия, които ще бъдат представени. След известни дискусии относно личните си предпочитания, ние проверихме щателно информацията за популярните игрални филми, позовавайки се на надеждни източници като една от най-големите бази данни за филми и сериали - ***IMDb*** (Internet Movie Database). Проучихме за филми, които имат високи оценки от потребителите на платформата, както и същевременно добро представяне в контекста на професионалната филмова критика - световноизвестни журнали, вестници и издателства, а също, разбира се, са и носители на уважавани филмови награди като Оскарите и Златните глобуси. В частност, прегледахме престижни колекции от филми като например бележитата класация на IMDb на 250-те най-добри филма за всички времена. Подбрахме такива заглавия, които не са насочени към конкретна тематика, а напротив - имат разнообразни жанрове, идеи, начини и места на заснемане, за да покажем богатството, което предлага тази тема, и как може тя да се разгърне.

След като източникът на информацията бе изяснен, последва създаването на първоначалната йерархична структура в ***XML*** документа. Този процес бе осъществен постепенно, с прилагане на различни експериментални опити, докато не се получи добре подреден XML файл, който е достатъчно информативен и същевременно не е наводнен с излишна информация. Имахме множество дискусии, като всеки подкрепяше мнението си с подходящи мотиви и впоследствие избрахме най-рационалните решения за структурното оформление на документа. Избрахме да използваме ***UTF-8*** encoding за гъвкавост при избора на символи за графично представяне на текста. Придържахме се към кратки и ясни имена и уместни идентификатори на компонентите в документа, към които лесно можем да реферираме.

Създадохме хранилище за проекта в системата за контрол на версиите Git, което позволи лесно да се внасят промени и същевременно да се пази информацията за предходните версии на проекта, които служеха за резервно копие при наличие на проблем по време на разработката. Така имахме и възможността да работим по проекта на различни физически устройства без да срещнем никакви технически проблеми.

Използвахме различни среди за разработка (IntelliJ IDEA, Visual Studio Code, Oxygen XML Editor), всяка от които предоставя богат инструментариум за гъвкаво редактиране на XML-базирани езици за маркиране и поддръжка на съпътстващите ги файлови формати. Съответно разполагахме с професионални приставки за

форматиране на XML код за правилна и четима подредба на информацията във файловете на проекта.

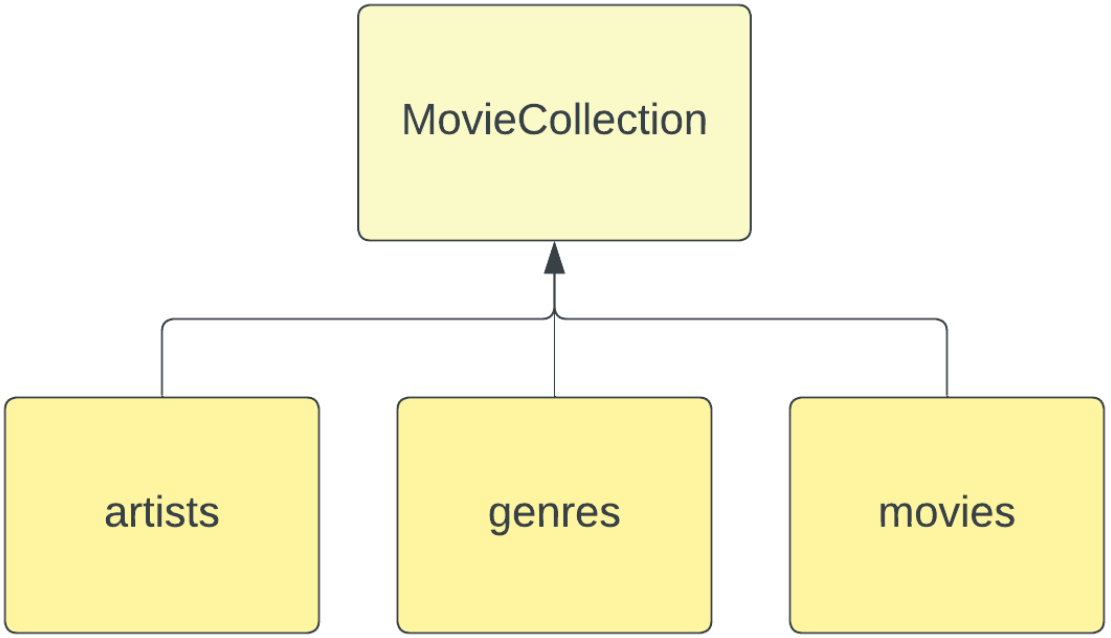
След като събрахме нужната информация относно филмите в ***XML*** документ, пристъпихме към направата на ***DTD*** файл, който да послужи за валидация на данните и структурата в документа. Тук отново дискутирахме и изпробвахме различните варианти, докато не стигнахме до подходящ краен резултат, който да можем да използваме пълноценно в разработката на проекта. Благодарение на възможностите на модерните среди за разработка, които използвахме, валидацията се случваше статично, в реално време, докато пишехме кода, позволявайки голяма гъвкавост и предотвратявайки множество евентуални грешки докато добавяхме и редактирахме данните в ***XML*** документа.

Последва създаването на ***XSL*** модул, чрез който трябваше да изпратим данните за интерпретация към ***PDF*** и да дефинираме стиловото оформление на каталога. Това е и най-времеемката част на нашия проект - откриването и внедряването на подходяща технология, която да позволява интерпретация до нужния формат. След щателно проучване открихме такава в лицето на Apache FOP - една open-source приставка, която използвахме както самостоятелно, така и вградена в средата за разработка (в случая на Oxygen XML Editor). ***Apache FOP*** е ***Java*** сървис, който притежава мощен ***XML*** парсър, като в контекста на нашия проект може с входни данни ***XML*** документа и прилежащия му ***XSL-FO*** файл да създаде междинен ***FO (Formatting Objects)*** файл, който от своя страна се интерпретира до завършен ***PDF*** файл.

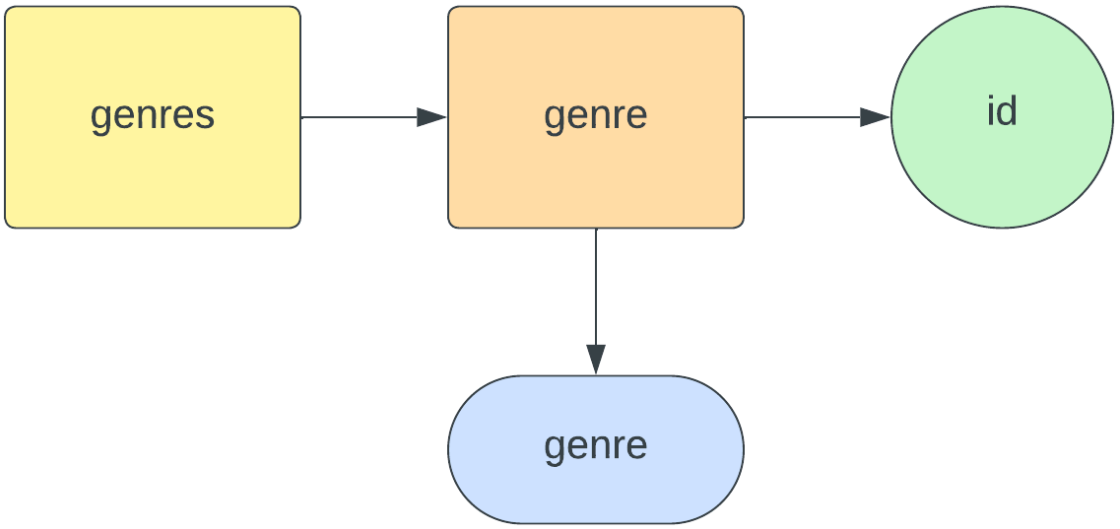
В процеса на разработка добавихме и графично съдържание, а именно - филмови постери, които подходящо представят всеки един филм. След като бяхме готови с проектната структура, започнахме да документираме самия проект.

2.2 Структура на съдържанието

Започваме с външната структура на XML документа. Той се състои от елемента ***MovieCollection***, чиито деца са съответно ***artists***, ***genres*** и ***movies***.

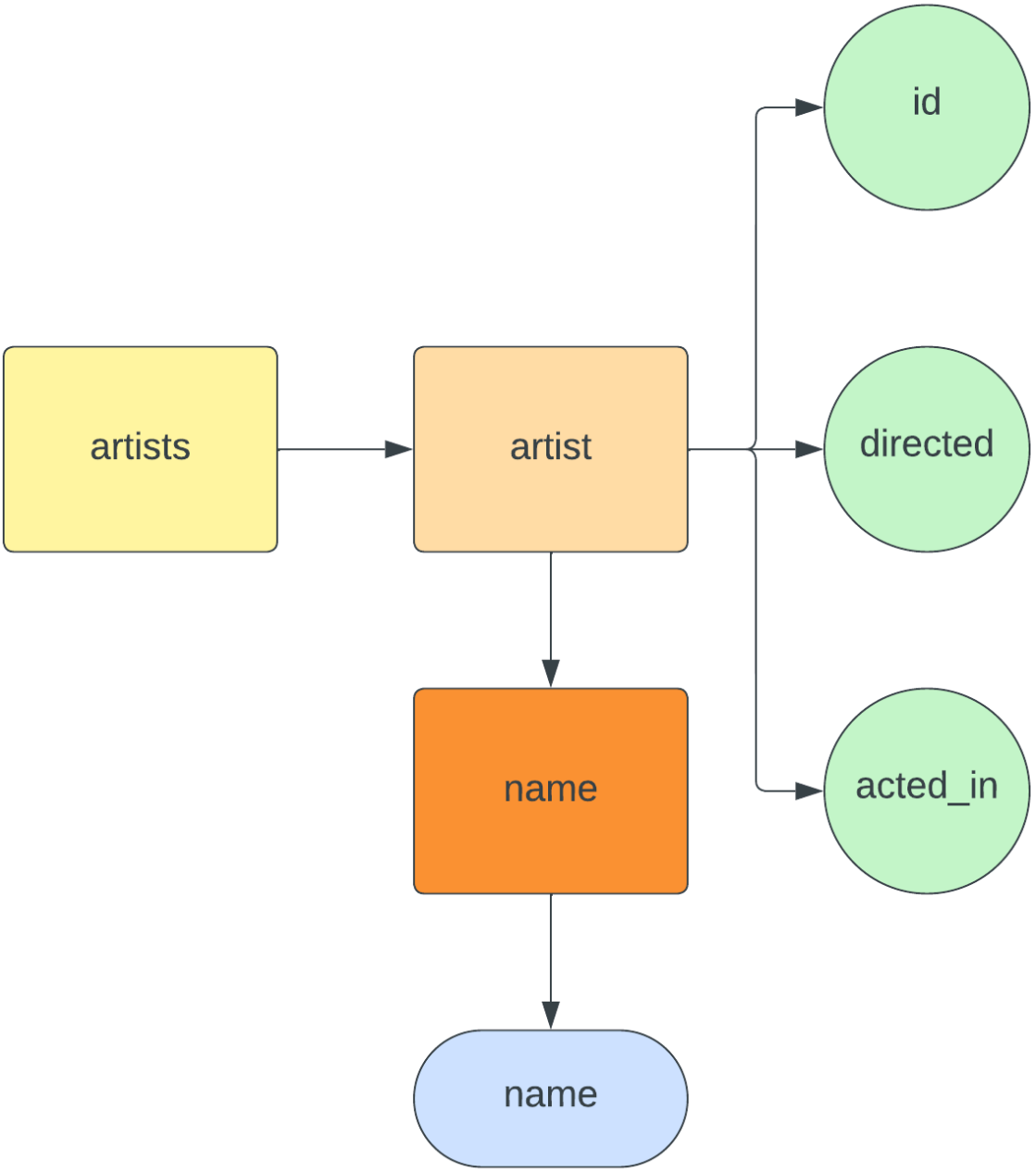


Елементът, който съдържа информацията за жанровете, е най-простият елемент на това ниво. Съдържа множество елементи ***genre***, всеки от които има атрибут ***id*** (идентификатор) и съдържа своето име.

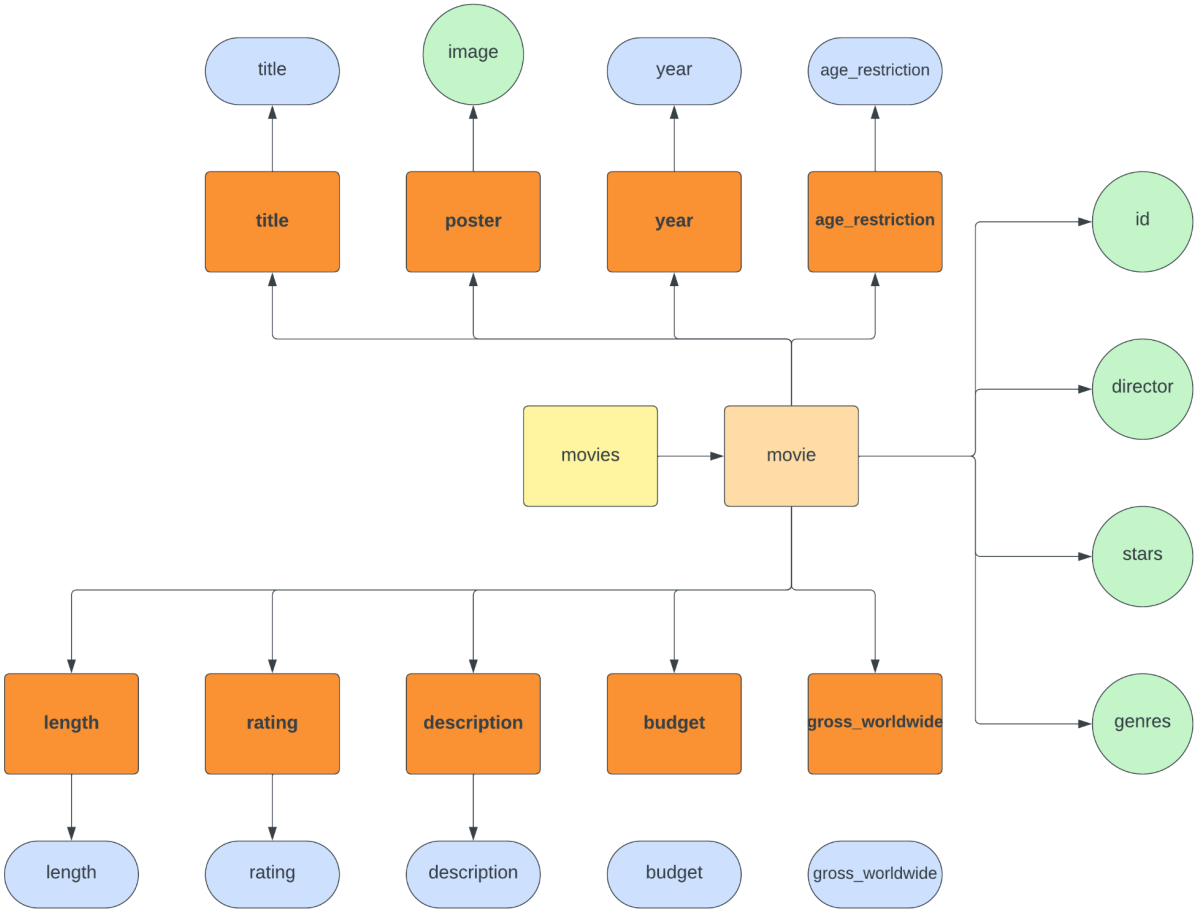


Елементът, който отговаря за данните на актьорите и режисьорите, носи генерализираното име ***artists***, защото актьорите и режисьорите в конкретни случаи

могат да изпълняват една и съща функция, например често режисьорите участват като актьори в собствените си филми. Нещо повече, данните, които се отнасят до актьорите и режисьорите, са еквивалентни. По този начин преизползваме конкретния елемент, вместо да създаваме два елемента с еднакво поведение, но различни по дефиниция. Всеки артист притежава атрибути ***id*** (идентификатор), ***directed*** (идентификатори на филмите, които режисира), ***acted_in*** (идентификатори на филмите, в които играе роля). Всеки артист има елемент ***name***, който съдържа името му.



Най-комплексният елемент в документа е именно ***movies***. Него ще опишем схематично.



- movies
  - movie
    - атрибут ***id*** (идентификатор)
    - атрибут ***director*** (идентификатор на режисьора на филма)
    - атрибут ***stars*** (идентификатори на актьорите в главните роли)
    - атрибут ***genres*** (идентификатори на жанровете, към които спада филмът)
    - елемент ***title*** (заглавие на филма)
      - съдържа текст със заглавието на филма
    - елемент ***poster*** (постер на филма)
      - атрибут ***image***, който реферира към относителния път на конкретното изображение
    - елемент ***year*** (годината, в която филмът започва да се излъчва в кината)
      - съдържа текст с годината

        - елемент ***age_restriction*** (възрастово ограничение на филма)
            - съдържа текст с възрастовото ограничение
        - елемент ***length*** (времетраене на филма)
            - съдържа текст с времетраене във формат ***Xh YYm***
        - елемент ***rating*** (средна потребителска оценка на филма в IMDb)
            - съдържа текст с оценка реално число между 1 и 10
        - елемент ***description*** (кратко описание на филма)
            - съдържа текст с кратко описание
        - елемент ***budget*** (бюджет на филма)
            - съдържа текст с бюджета на продукцията във формат ***$XXXXX***
        - елемент ***gross_worldwide*** (печалба в световен мащаб на филма)
            - съдържа текст с печалбата на продукцията във формат ***$XXXXX***

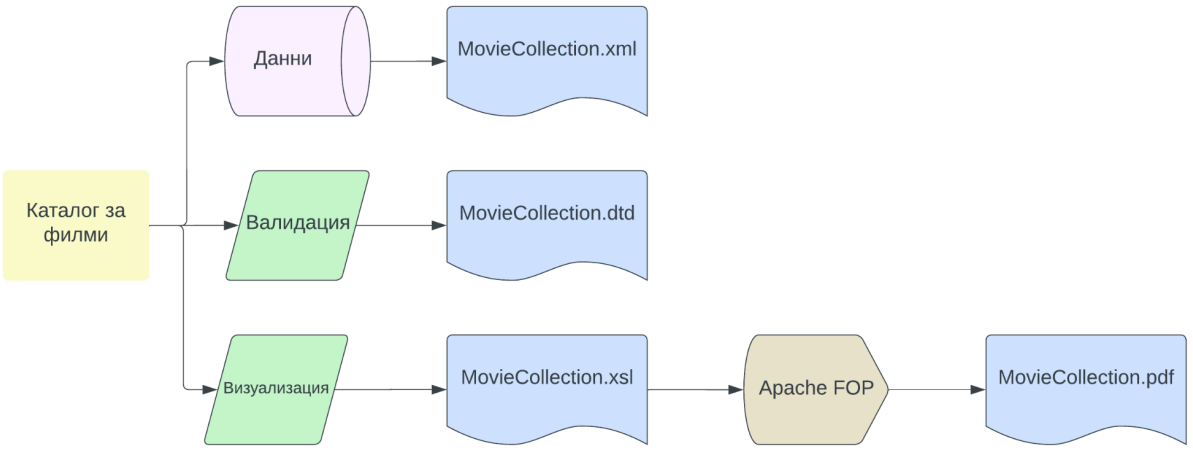
2.3 Тип и представяне на съдържанието

- 1× ***.xml*** файл (съдържа йерархична подредба на данните в проекта)
- 1× ***.dtd*** файл (служи за валидация)
- 1× ***.xsl*** (XSL-FO) файл (служи за прилагане на стилово оформление и подготвяне на данните за интерпретация в други файлови формати)
- 1× ***.pdf*** файл (служи за съхранение на крайния резултат от интерпретацията на данните
- 1× директория ***resources*** (съдържа всички външни графични ресурси с източник уикипедия)
    - 1× ***.png*** файл - ***Clapperboard.png - 122 KB*** (служи за графично оформление на титулната страница на PDF документа)
    - 10× ***.jpg*** файла (служат за представяне на атрибута ***image*** на елемента ***poster***, с други думи - представляват постери за всеки един от филмите).
        - ***A-Clockwork-Orange.jpg - 443 KB***
        - ***Blade-Runner.jpg - 2.75 MB***
        - ***Django Unchained.jpg - 131 KB***
        - ***Eternal-Sunshine.jpg - 3.35 MB***
        - ***Interstellar.jpg - 445 KB***
        - ***Lost-In-Translation.jpg - 1.24 MB***
        - ***Mr-Nobody.jpg - 967 KB***
        - ***Oldboy.jpg - 477 KB***
        - ***Pulp-Fiction.jpg - 142 KB***
        - ***The-Green-Mile.jpg - 67.3 KB***
- 1× директория documentation (съдържа необходимите за документацията ресурси)

- 1× ***.pdf*** файл и един ***.doc(x)*** файл (Прилежащият файл)
- 1x директория diagrams
  - 5× ***.png*** файла - Диаграми, описващи същността на проекта
    - ***Architecture.png - 81.6 KB***
    - ***Artists-Diagram.png - 45 KB***
    - ***Genres-Diagram.png - 22 KB***
    - ***Movies-Diagram.png - 143 KB***
    - ***Root-Structure-Diagram.png - 20 KB***

3 Дизайн

За да онагледим по-ефективно разпределението на работата между различните XML-базирани технологии в нашия проект, ще използваме една генерализирана диаграма, която цели да покаже по един лесен за възприемане начин архитектурата на каталога.



Данните представляват основата на проекта. Те включват цялата информация за филмите от каталога. В XML йерархията техният обобщен вариант се намира под тага ***movies***.

Всеки филм притежава собствен уникален атрибут ***id*** (идентификатор) и агрегация на отнасящата се за него информация. Добавили сме атрибути ***director, stars, genres***, които в DTD схемата декларирахме с тип ***IDREF(S)*** и реферират към уникалните идентификатори на други обекти в документа - съответно елементите в ***artists*** и ***genres.*** Целта на този тип реферираме е да се преизползват вече дефинирани XML структури и да бъде по-ефикасен и четим кодът. В случая на режисьора имаме релация **един филм → един режисьор**, ето защо използваме

единична референция към идентификатор. При другите два атрибута ***stars*** и ***genres*** наблюдаваме релация **един → много**, тъй като филмът може да има разнообразен актьорски състав и да се асоциира със съвкупност от жанрове.

```
<!ATTLIST movie
   id ID #REQUIRED
   director IDREF #REQUIRED
   stars IDREFS #REQUIRED
   genres IDREFS #IMPLIED>
```

Сега ще кажем няколко думи за елементите с метаинформацията за самия филм. Заглавието (елементът ***title***) е ***PCDATA***. Постерът (***poster***) е елемент, който има задължителен атрибут ***image*** - относителен път към локацията на конкретно изображение. Всеки един такъв път е дефиниран посредством ***ENTITY***. Чрез ***ENTITY*** декларацията можем да предефинираме даден обект така, че да реферираме по консистентен начин към него на множество места. За да добавим успешно референции към адресите на филмовите постери, използваме конструкцията ***NDATA***, следвана от типа на файловия формат на изображенията, а именно - ***JPEG (JPG)***. Посредством ключовата дума ***NOTATION*** внасяме нужната на XML процесора информация за типа на тези външни файлове и как да ги обработи.

```
<!NOTATION JPG SYSTEM "image/jpg">
<!ENTITY pulp_fiction SYSTEM "resources/Pulp-Fiction.jpg" NDATA JPG>
<!ENTITY blade_runner SYSTEM "resources/Blade-Runner.jpg" NDATA JPG>
<!ENTITY the_green_mile SYSTEM "resources/The-Green-Mile.jpg" NDATA JPG>
<!ENTITY mr_nobody SYSTEM "resources/Mr-Nobody.jpg" NDATA JPG>
<!ENTITY eternal_sunshine SYSTEM "resources/Eternal-Sunshine.jpg" NDATA JPG>
<!ENTITY oldboy SYSTEM "resources/Oldboy.jpg" NDATA JPG>
<!ENTITY a_clockwork_orange SYSTEM "resources/A-Clockwork-Orange.jpg" NDATA
JPG>
<!ENTITY lost_in_translation SYSTEM "resources/Lost-In-Translation.jpg"
NDATA JPG>
<!ENTITY interstellar SYSTEM "resources/Interstellar.jpg" NDATA JPG>
<!ENTITY django_unchained SYSTEM "resources/Django-Unchained.jpg" NDATA JPG>
```

Следват еднотипни представяния на ***PCDATA*** елементи, които в контекста на конструкцията са еквивалентни на елемента ***title***.

Общ вид на ***XML*** елемента ***movie:***

```
<movie id="M-DU" director="A-1"
   stars="A-37 A-38 A-39 A-4"
   genres="G-Drama G-Western">
   <title>Django Unchained</title>
   <poster image="django_unchained"/>
   <year>2012</year>
   <age_restriction>18</age_restriction>
   <length>2h 45m</length>
   <rating>8.5</rating>
   <description>
       With the help of a German bounty-hunter, a freed slave
       sets out to rescue his wife from a brutal plantation owner in
Mississippi.
   </description>
   <budget>$100,000,000</budget>
   <gross_worldwide>$426,074,373</gross_worldwide>
</movie>
```

Както вече показахме, използваме външен ***DTD*** файл за валидиране на цялата структура на ***XML*** документа. По този начин, ако решим например да добавим нов филм в каталога, ще трябва да се съобразим с всички правила, дефинирани от ***DTD***, да добавим всички задължителни елементи и те да се съобразят с декларирания формат. По този начин могат да се предотвратят всички структурни грешки на по-късен етап. ***XML*** парсърът извършва статична проверка на написания ***XML*** код и известява в реално време за съществуващи грешки.

Друг съществен детайл в разработката на проекта е рендърването към ***PDF***. В нашия проект това се осъществява посредством ***XSL***, а по-конкретно ***XSLT*** и ***XSL-FO***. Дефинирахме каталог, съставен от страници в стандартен размер ***A4***. Той се състои от една титулна страница и десет аналогични презентационни страници, представящи десетте филма от съдържанието на каталога. Всеки филм от каталога се състои от заглавие, година, постер и списък, изброяващ компонентите с метаинформацията му. Всеки компонент е предшестван от префиксна дефиниция, което способства за по-четимо съдържание. Например, имаме: ***Director: Stanley Kubrick***, вместо енигматичното ***Stanley Kubrick***, което не внася достатъчен контекст в информацията.

Една от най-порочните практики в писането на код е дублирането на фрагменти от код на различни места в рамките на един и същи проект. Това може да доведе до сериозни проблеми в по-късен етап и да понижи качеството на софтуерния продукт значително. Ето защо, за да преизползваме кода, който служи за стилизация и трансформация на всеки един от филмите в каталога, ние се позовахме на

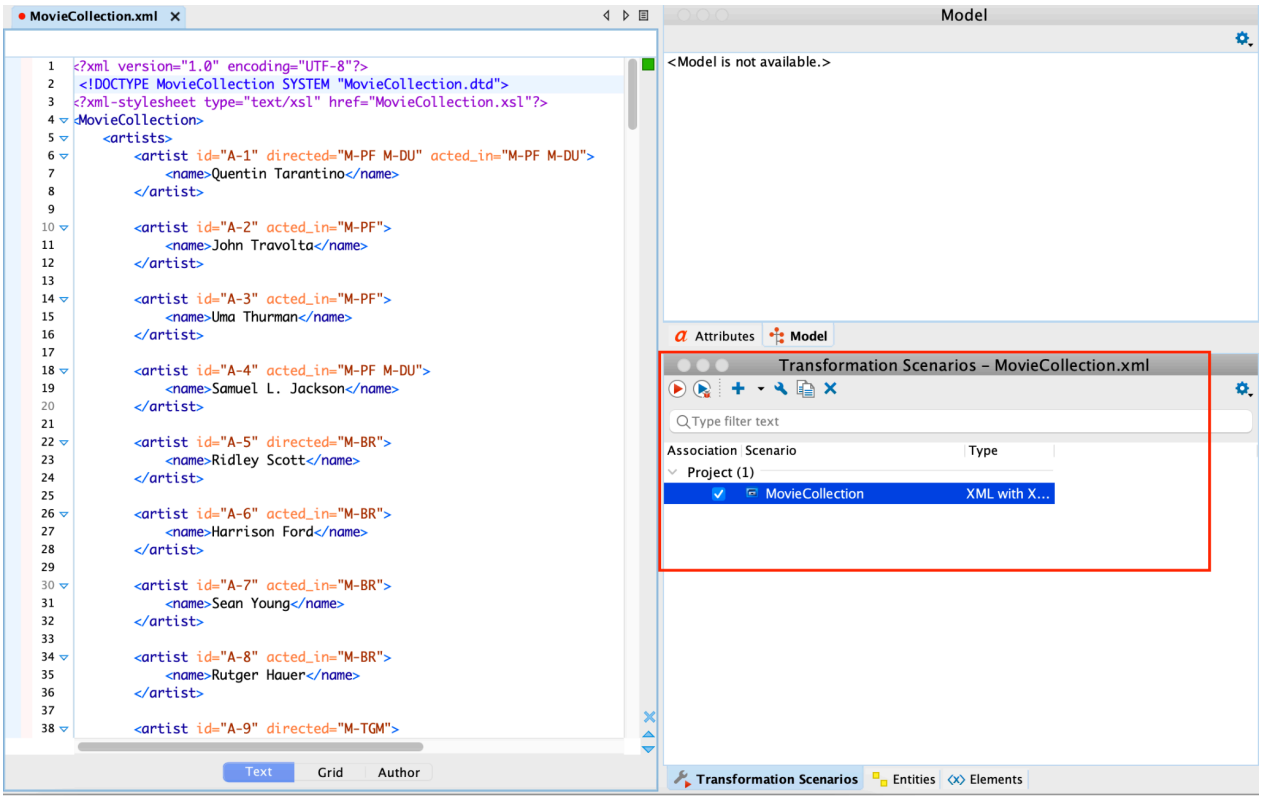
xsl:for-each и ***xsl:template***, благодарение на които декларациите за еднотипни компоненти могат да се оползотворят максимално след като сме ги дефинирали на едно единствено място.

Филмите са сортирани според заглавията, които носят, съответно подредбата в ***XML*** йерархията на елементите от тип movie не корелира с крайната подредба в ***PDF*** файла. Това се осъществява посредством ***xsl:sort*** в рамките на обхождането с ***xsl:for-each***.

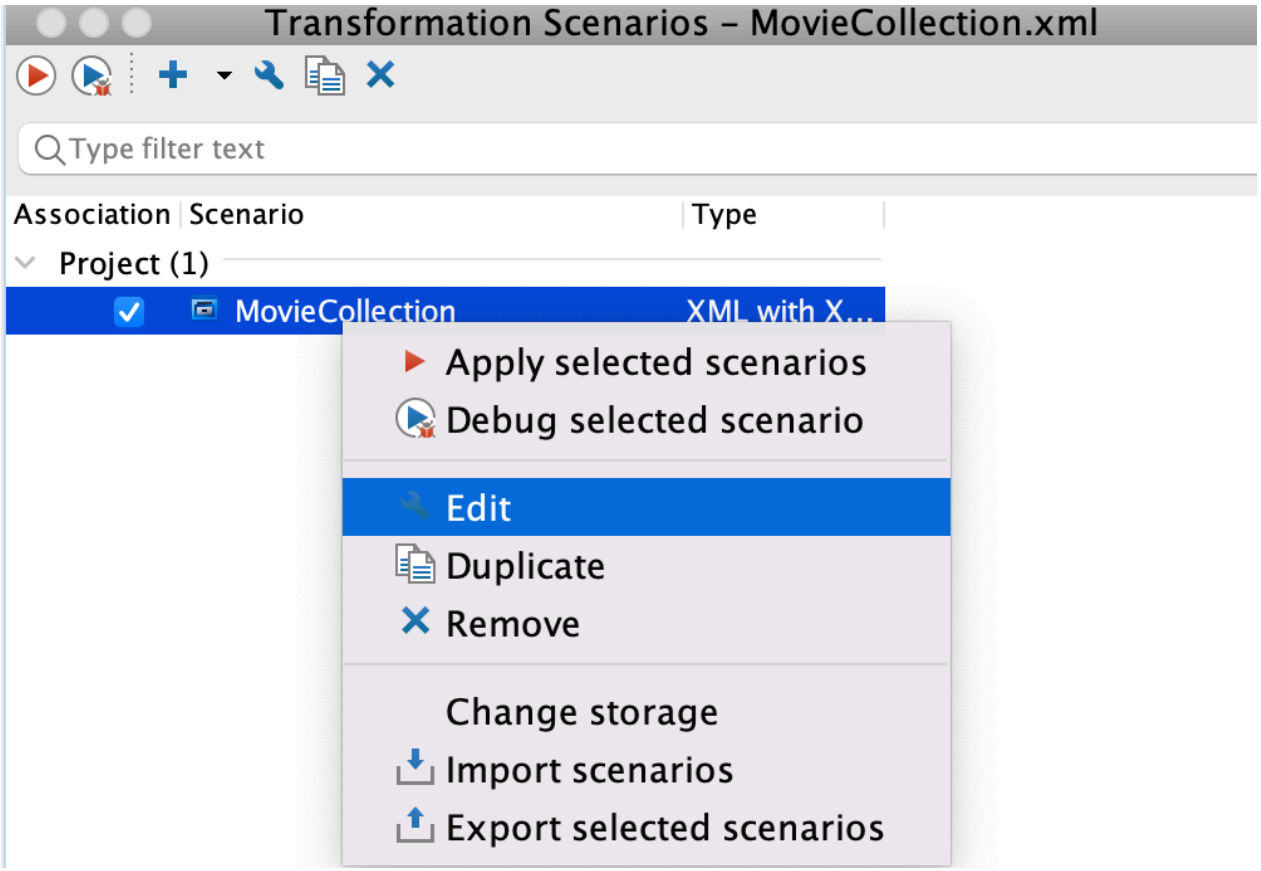
```
<xsl:for-each select="MovieCollection/movies/movie">
   <xsl:sort select="title"/>
      ... PAGE STYLING ...
</xsl:for-each>
```

За да опишем самия процес на трансформация към ***PDF***, нека за пример представим нагледно начина, по който това се прави чрез графичния интерфейс на средата за разработка ***Oxygen XML Editor***.

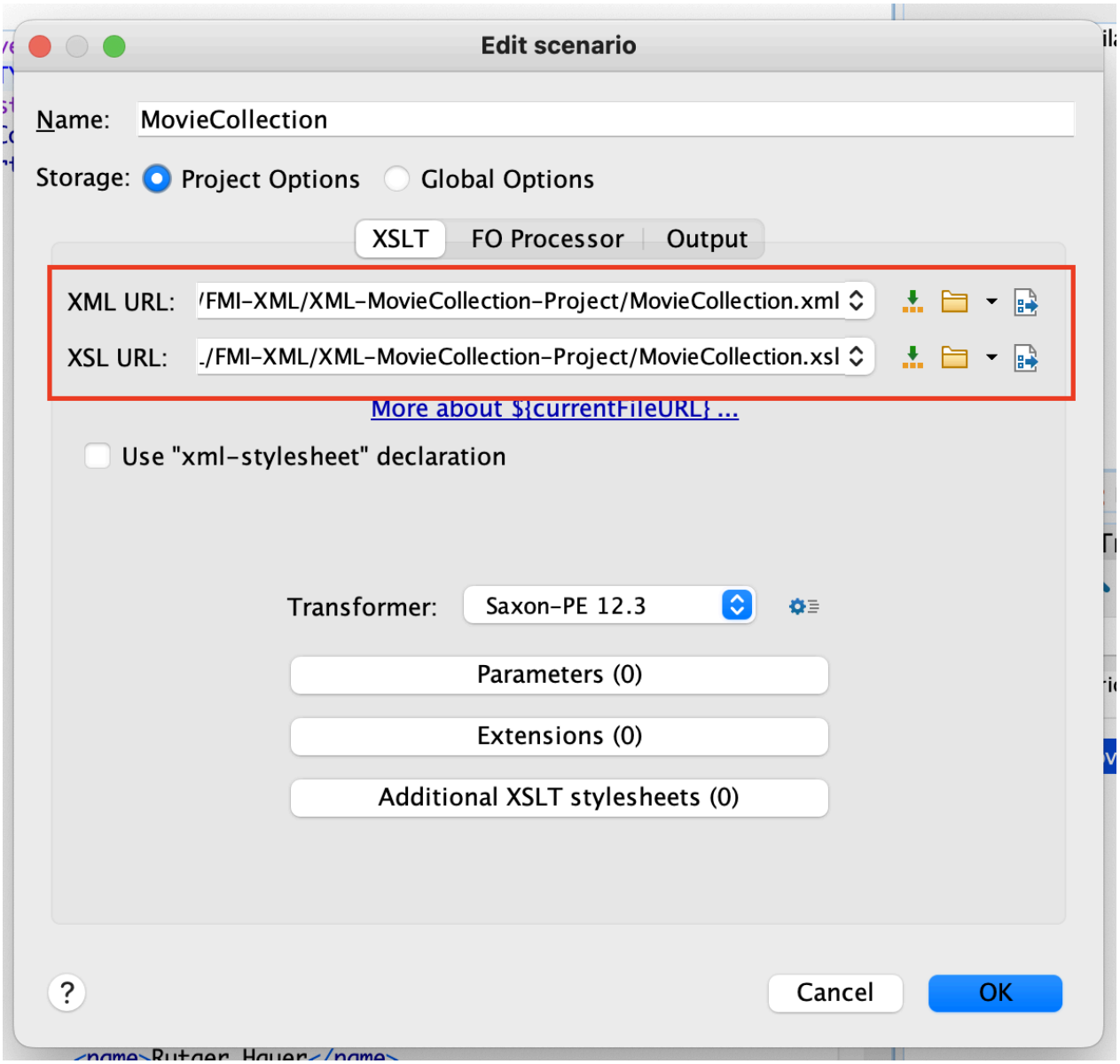
След като сме отворили проекта със съответните файлове, които са ни нужни, избираме ***Transformation Scenario*** менюто.



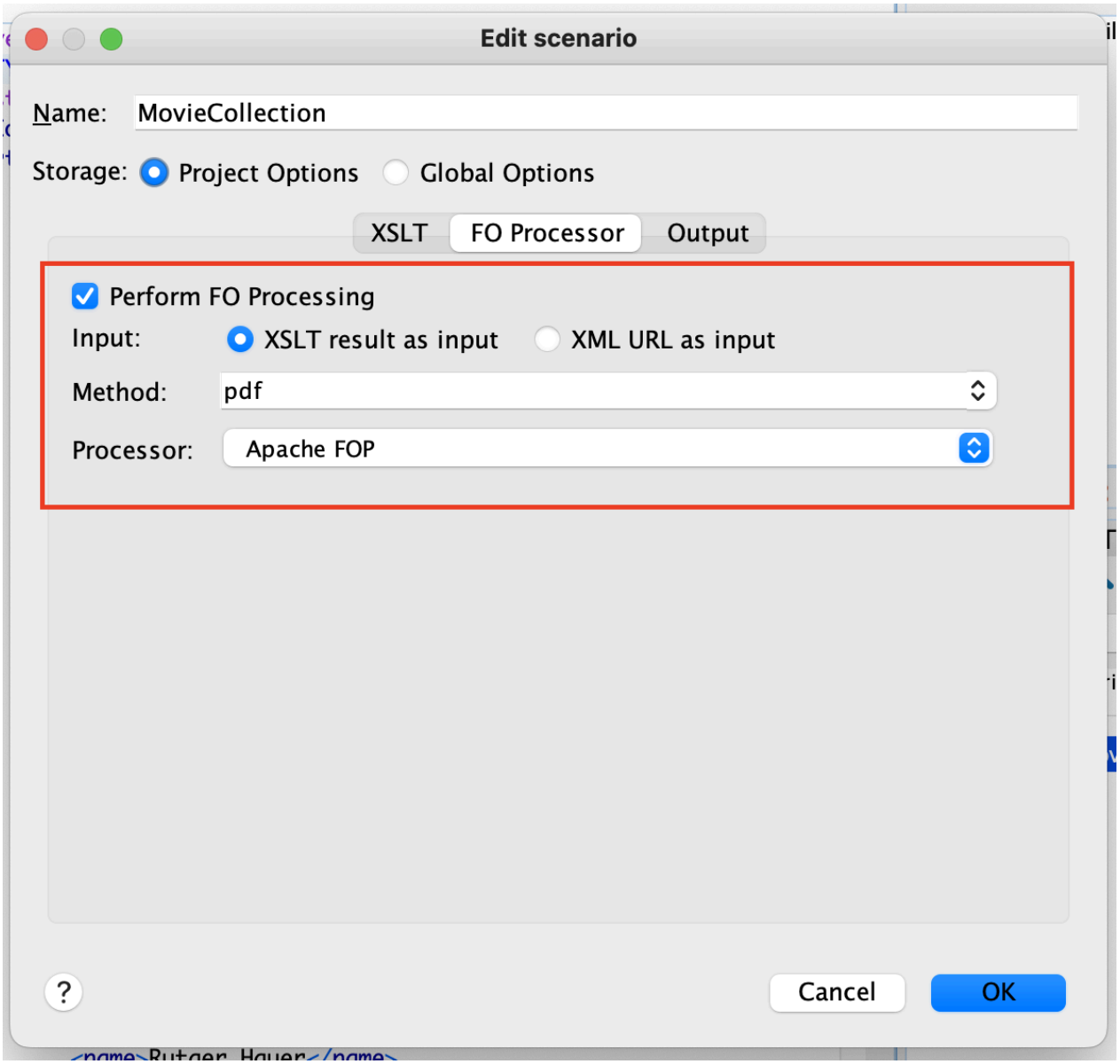
Можем да създадем нов сценарий за трансформиране, или да редактираме вече съществуващ такъв.



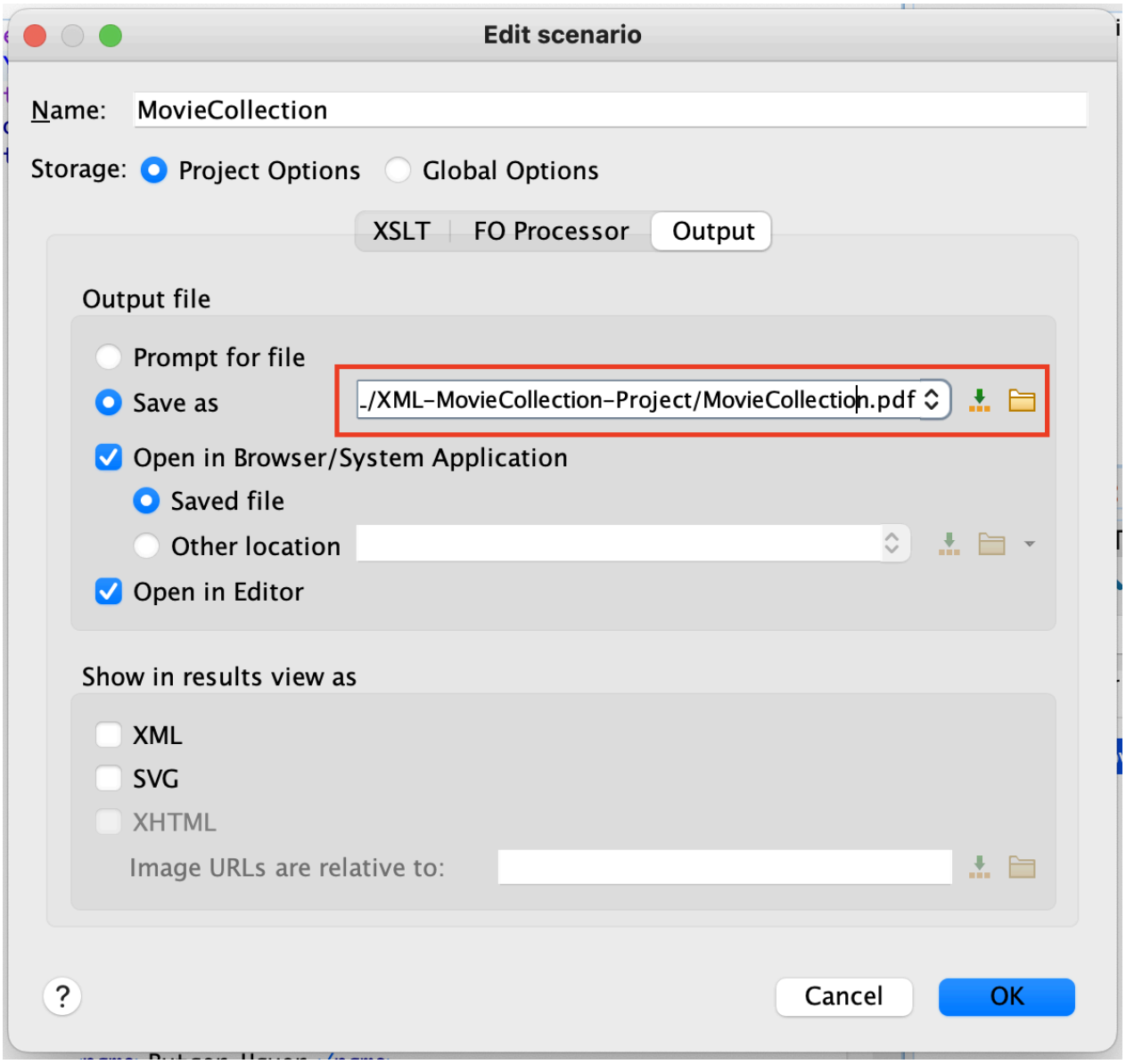
В ***XSLT*** менюто декларираме адресите на ***XML*** и ***XSL*** файловете.



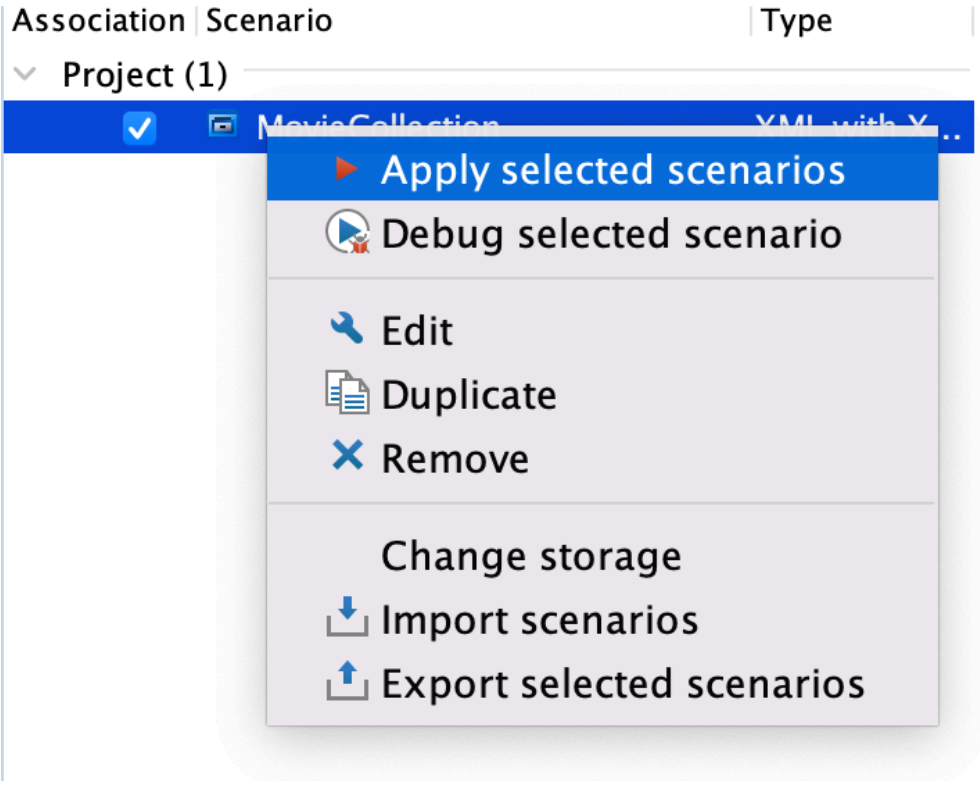
Във ***Formatting Objects Processor*** менюто избираме опцията за ***XSLT*** файл за входни данни, метод на рендърване - ***PDF***, и инструмент за конвертиране - ***Apache FOP***.



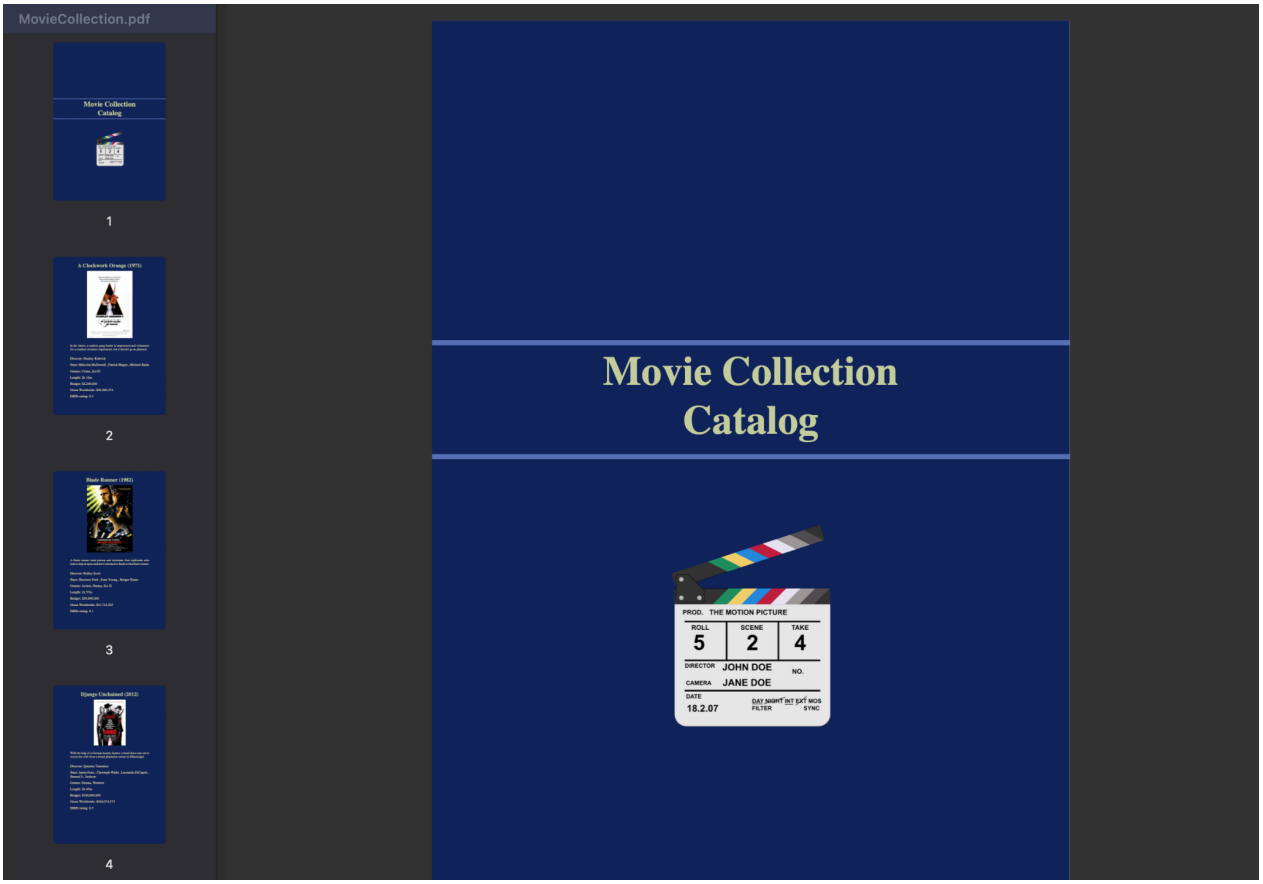
В менюто за изходния резултат избираме адреса и името на файла, в който искаме да запишем визуализирания каталог.



Вече можем да конвертираме.



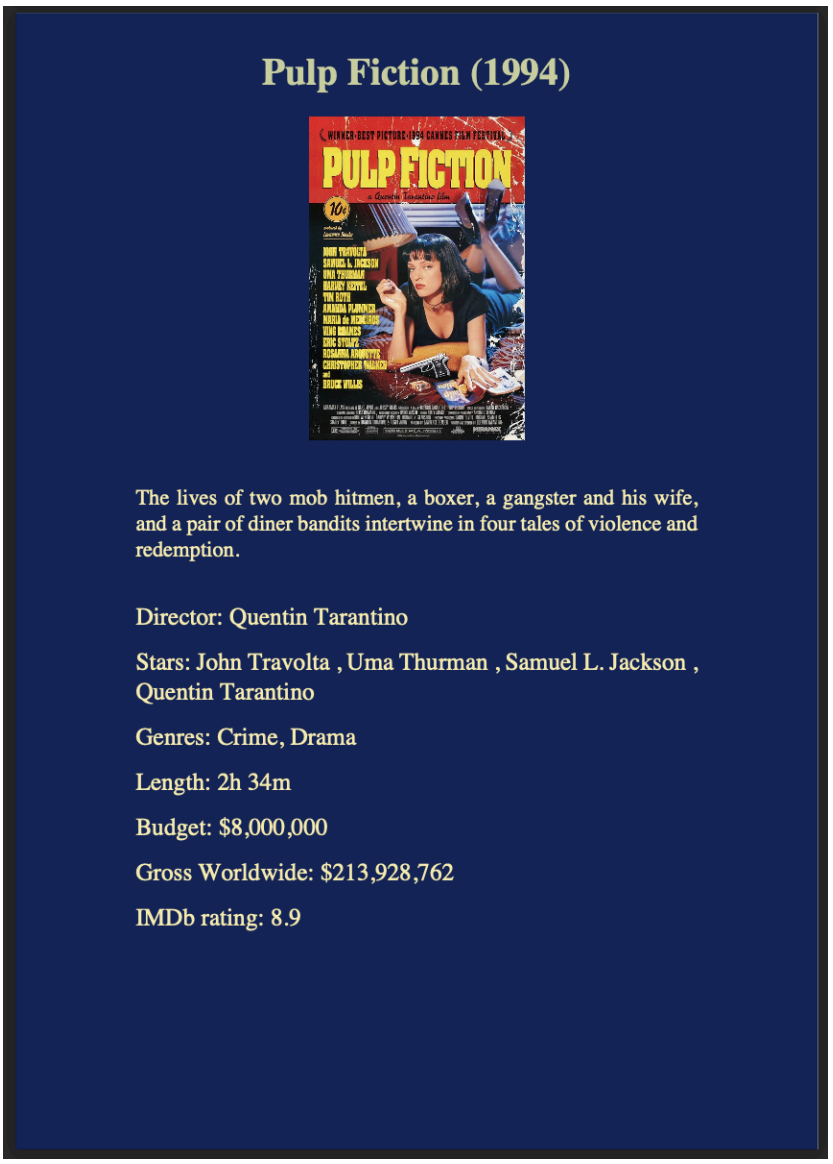
При завършено конвертиране можем вече да разгледаме готовия ***PDF*** документ:



4 Тестване

Пореди спецификите на заданието, нашият проект не се нуждае от конкретен тип тестване. Резултатът е видим и е *human-readable*, съответно всяка девиация в стандартния изглед на крайния резултат би била лесно откриваема. Погрижихме се да тестваме проекта на различни физически устройства, под различни операционни системи (***Windows OS и macOS***) и изходът бе еквивалентен. Използвахме наличните в средите за разработка инструменти за статична проверка на кода и форматиране, така, че да не допускаме грешки, и с помощта на ***DTD*** осъществихме валидация на каталога. По време на разработката тествахме резултата основно чрез оценка на PDF съдържанието - дали има разминавания в индентациите, шрифтовете, цветовете, какви са размерите на снимките и текста и т.н.

Пример - изглед на филм от каталога:



5 Заключение и възможно бъдещо развитие

Считаме, че в рамките на поставената задача успяхме да достигнем до успешен резултат. Разширихме своите практически умения в работата с ***XML*** технологии. Успяхме да направим завършен каталог, който можем да използваме за презентационни цели и извън контекста на учебния курс, поради актуалността на темата, което само по себе си е изключително позитивен ефект от работата по проекта. Технологиите като ***Apache FOP***, обаче са остарели към днешна дата и не разполагат с достатъчно добра документация и поддръжка в наши дни. Справянето с грешки при употреба на неактуални технологии понякога не е тривиално и може да доведе до проблеми. За бъдещо развитие на проекта е добра идея да се скалира неговият мащаб и да се преизползва идеята за визуализация в браузъри посредством ***HTML & CSS***.

6 Разпределение на работата

- Атанас Ников, ФN 3MI0600006
  - Проучване и тестване на инструменти за трансформация
  - Трансформация чрез XSLT и XSL-FO
  - Изготвяне на документация
- Светлозар Стефанов, ФN 4MI0600030
  - DTD валидация
  - Оптимизации на XSLT и XSL-FO
  - Проверка на документацията
- Съвместна работа
  - Подбор на филми и прилежащите им текстови данни и изображения
  - Създаване на XML структура
  - Дискусии и промени в плановете за разработка
  - Редакция на всеки един от файловете

7 Използвани литературни източници и Уеб сайтове

1. Записки и приложения към курса XML технологии за семантичен web
2. Конвертиране на файловия формат: Apache FOP
3. Входна информация: IMDb
4. Диаграми: Lucid
5. Контрол на версиите: Git & GitHub