**СОФИЙСКИ УНИВЕРСИТЕТ  
„СВ. КЛИМЕНТ ОХРИДСКИ“**

****

**ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА**

**ПРОЕКТ**

**„КОМПЮТЪРЕН МАГАЗИН“**

Изготвен от: Кристиан Йорданов Васиков

Фак.№ **0MI0700194**

Специалност „Информационни системи“, **курс 3; група 2**

София, 2025г.

Съдържание:

[I. Версия “XML схема” 3](#_Toc187600789)

[Описание на предметната област 3](#_Toc187600790)

[Описание на схемата 3](#_Toc187600791)

[Примерно съдържание 5](#_Toc187600792)

[Реализация на схемата 7](#_Toc187600793)

[II. Версия „описание на DOM съответстващ на схемата“ 8](#_Toc187600794)

[III. Версия „набор от XSLT трансформации“ 10](#_Toc187600795)

[Трансформиране на XML документи в друг XML код 14](#_Toc187600796)

[Трансформиране на XML документи в чист текст 16](#_Toc187600797)

[IV. Заключения 18](#_Toc187600798)

# **Версия “XML схема”**

## **Описание на предметната област**

Проектът "Компютърен магазин" е разработен с цел структуриране и управление на данните за компютърна техника и аксесоари в магазин.

За валидиране на XML документите е избрана XML Schema (XSD) вместо DTD, поради няколко ключови предимства: по-богати възможности за дефиниране на типове данни, по-добра поддръжка на пространства от имена (namespaces), и по-стриктен контрол върху структурата на данните. XML Schema позволява създаването на по-сложни и прецизни модели на данните, което е особено важно при работа с продуктов каталог на компютърен магазин.

## **Описание на схемата**

XML схемата на проекта дефинира йерархична структура, която включва следните основни елементи:

* Основен елемент `computerStore`, който съдържа информация за целия магазин;
* За всеки продукт се съхраняват атрибути като: уникален идентификатор (ID), марка, модел, цена, наличност и технически характеристики;
* Дефинирани са сложни типове (complex types) на местата, на които е нужно по-детайлна информация.

1. Основен елемент: computerStore

* Описание: Коренов (root) елемент, който обхваща цялата информация за компютърния магазин, включително детайли за магазина, категории на частите и продуктите.
* Структура:
  + storeInfo (StoreInfoType): Съдържа основната информация за магазина.
  + partCategories (CategoriesType): Групира категориите на частите, продавани в магазина.
  + Products (ProductsType, неограничено): Изброява всички продукти, налични в магазина.

1. Сложен тип: StoreInfoType

* Описание: Описва информацията, свързана с магазина.
* Елементи:
  + storeName (string): Името на магазина.
  + location (LocationType): Детайли за местоположението на магазина.
  + contactNumber (string): Телефонен номер за контакт с магазина.

1. Сложен тип: LocationType

* Описание: Съдържа града и адреса на магазина.
* Елементи:
  + city (string): Градът, в който се намира магазинът.
  + address (string): Конкретният адрес на магазина.

1. Сложен тип: CategoriesType

* Описание: Съдържа всички категории на части, с които магазинът работи.
* Елементи:
  + category (CategoryType, неограничено): Всяка категория на части, налична в магазина.

1. Сложен тип: CategoryType

* Описание: Представя една категория на части.
* Елементи:
  + name (string): Името на категорията.
  + subcategories (SubcategoriesType, опционално): Подкатегории в рамките на категорията.

1. Сложен тип: SubcategoriesType

* Описание: Групира подкатегориите под една категория.
* Елементи:
  + subcategory (SubcategoryType, неограничено): Всяка подкатегория на части.

1. Сложен тип: SubcategoryType

* Описание: Подкатегория в рамките на основна категория.
* Елементи:
  + name (string): Името на подкатегорията.

1. Сложен тип: ProductsType

* Описание: Дефинира структурата за изброяване на продукти.
* Елементи:
  + product (ProductType, неограничено): Всеки продукт, изброен в магазина.

1. Сложен тип: ProductType

* Описание: Представя един продукт в магазина.
* Структура:
  + productName (string): Името на продукта.
  + category (string): Категорията, към която принадлежи продуктът.
  + price (float): Цената на продукта.
  + description (string): Описание на продукта.
  + specifications (SpecificationsType): Спецификации на продукта.
  + атрибути:
    - productID (ID, задължителен): Уникален идентификатор за продукта.

1. Сложен тип: SpecificationsType

* Описание: Съдържа спецификации на продукт.
* Елементи:
  + specification (SpecificationType, неограничено): Всяка спецификация на продукта.

1. Сложен тип: SpecificationType

* Описание: Единичен запис за спецификация.
* Елементи:
  + key (string): Името на спецификацията.
  + value (string): Стойността на спецификацията.

## **Примерно съдържание**

XSD схемата:

*<?*xml version="1.0" encoding="UTF-8"*?>  
<*xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"*>* <!-- Главен елемент за магазина -->  
 *<*xs:element name="computerStore"*>  
 <*xs:complexType*>  
 <*xs:sequence*>  
 <*xs:element name="storeInfo" type="StoreInfoType"*/>  
 <*xs:element name="partCategories" type="CategoriesType"*/>  
 <*xs:element name="products" type="ProductsType" maxOccurs="unbounded"*/>  
 </*xs:sequence*>  
 </*xs:complexType*>  
 </*xs:element*>* <!-- Информация за магазина -->  
 *<*xs:complexType name="StoreInfoType"*>  
 <*xs:sequence*>  
 <*xs:element name="storeName" type="xs:string"*/>  
 <*xs:element name="location" type="LocationType"*/>  
 <*xs:element name="contactNumber" type="xs:string"*/>  
 </*xs:sequence*>  
 </*xs:complexType*>* <!-- Адрес на магазина -->  
 *<*xs:complexType name="LocationType"*>  
 <*xs:sequence*>  
 <*xs:element name="city" type="xs:string"*/>  
 <*xs:element name="address" type="xs:string"*/>  
 </*xs:sequence*>  
 </*xs:complexType*>* <!-- Категории части -->  
 *<*xs:complexType name="CategoriesType"*>  
 <*xs:sequence*>  
 <*xs:element name="category" type="CategoryType" maxOccurs="unbounded"*/>  
 </*xs:sequence*>  
 </*xs:complexType*>* <!-- Отделна Категория и подкатегория, които в случая са различните марки части-->  
 *<*xs:complexType name="CategoryType"*>  
 <*xs:sequence*>  
 <*xs:element name="name" type="xs:string"*/>  
 <*xs:element name="subcategories" type="SubcategoriesType" minOccurs="0"*/>  
 </*xs:sequence*>  
 </*xs:complexType*>* <!-- Подкатегориите -->  
 *<*xs:complexType name="SubcategoriesType"*>  
 <*xs:sequence*>  
 <*xs:element name="subcategory" type="SubcategoryType" maxOccurs="unbounded"*/>  
 </*xs:sequence*>  
 </*xs:complexType*>* <!-- Подкатегория, която в случая е марката на определената част -->  
 *<*xs:complexType name="SubcategoryType"*>  
 <*xs:sequence*>  
 <*xs:element name="name" type="xs:string"*/>  
 </*xs:sequence*>  
 </*xs:complexType*>* <!-- Продукти -->  
 *<*xs:complexType name="ProductsType"*>  
 <*xs:sequence*>  
 <*xs:element name="product" type="ProductType" maxOccurs="unbounded"*/>  
 </*xs:sequence*>  
 </*xs:complexType*>* <!-- Продукт -->  
 *<*xs:complexType name="ProductType"*>  
 <*xs:sequence*>  
 <*xs:element name="productName" type="xs:string"*/>  
 <*xs:element name="category" type="xs:string"*/>  
 <*xs:element name="price" type="xs:float"*/>  
 <*xs:element name="description" type="xs:string"*/>  
 <*xs:element name="specifications" type="SpecificationsType"*/>  
 </*xs:sequence*>  
 <*xs:attribute name="productID" type="xs:ID" use="required"*/>  
 </*xs:complexType*>* <!-- Спецификации на продукт -->  
 *<*xs:complexType name="SpecificationsType"*>  
 <*xs:sequence*>  
 <*xs:element name="specification" type="SpecificationType" maxOccurs="unbounded"*/>  
 </*xs:sequence*>  
 </*xs:complexType*>* <!-- Единична спецификация -->  
 *<*xs:complexType name="SpecificationType"*>  
 <*xs:sequence*>  
 <*xs:element name="key" type="xs:string"*/>  
 <*xs:element name="value" type="xs:string"*/>  
 </*xs:sequence*>  
 </*xs:complexType*>  
  
</*xs:schema*>*

## **Реализация на схемата**

*<?*xml version="1.0" encoding="UTF-8"*?>  
<*computerStore  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:noNamespaceSchemaLocation="../XMLSchema/schema.xsd"  
*>  
 <*storeInfo*>  
 <*storeName*>*ТехноЛюбимец*</*storeName*>  
 <*location*>  
 <*city*>*София*</*city*>  
 <*address*>*ул. Цариградско шосе 120*</*address*>  
 </*location*>  
 <*contactNumber*>*+359 88 888 8888*</*contactNumber*>  
 </*storeInfo*>  
 <*partCategories*>  
 <*category*>  
 <*name*>*Процесори*</*name*>  
 <*subcategories*>  
 <*subcategory*>  
 <*name*>*Intel*</*name*>  
 </*subcategory*>  
 <*subcategory*>  
 <*name*>*AMD*</*name*>  
 </*subcategory*>  
 </*subcategories*>  
 </*category*>  
 <*category*>  
 <*name*>*Видеокарти*</*name*>  
 <*subcategories*>  
 <*subcategory*>  
 <*name*>*NVIDIA*</*name*>  
 </*subcategory*>  
 <*subcategory*>  
 <*name*>*AMD*</*name*>  
 </*subcategory*>  
 </*subcategories*>  
 </*category*>  
 </*partCategories*>  
 <*products*>  
 <*product productID="P001"*>  
 <*productName*>*Intel Core i9-9900K*</*productName*>  
 <*category*>*Процесори*</*category*>  
 <*price*>*799.99*</*price*>  
 <*description*>*Високопроизводителен процесор от 9-то поколение*</*description*>  
 <*specifications*>  
 <*specification*>  
 <*key*>*Ядра*</*key*>  
 <*value*>*8*</*value*>  
 </*specification*>  
 <*specification*>  
 <*key*>*Нишки*</*key*>  
 <*value*>*16*</*value*>  
 </*specification*>  
 </*specifications*>  
 </*product*>  
 <*product productID="P002"*>  
 <*productName*>*NVIDIA GeForce RTX 3080*</*productName*>  
 <*category*>*Видеокарти*</*category*>  
 <*price*>*1499.99*</*price*>  
 <*description*>*Графична карта за гейминг и професионална употреба*</*description*>  
 <*specifications*>  
 <*specification*>  
 <*key*>*Памет*</*key*>  
 <*value*>*10GB GDDR6X*</*value*>  
 </*specification*>  
 <*specification*>  
 <*key*>*Технология на изработка*</*key*>  
 <*value*>*8nm*</*value*>  
 </*specification*>  
 </*specifications*>  
 </*product*>  
 </*products*>  
</*computerStore*>*

# **Версия „описание на DOM съответстващ на схемата“**

В тази версия на проекта се демонстрира създаването и манипулирането на DOM (Document Object Model) дърво, което съответства на XML схемата от версия 1.

DOM представлява програмен интерфейс за работа с XML документи, който позволява динамично създаване, модифициране и валидиране на XML структури. За целта ще бъдат разработени следните компоненти:

* Създаване на DOM дърво, което отразява структурата на компютърния магазин, включващо всички основни елементи като продукти, категории и техните характеристики
* Имплементиране на методи за добавяне на нови продукти към съществуващата структура
* Демонстрация на валидация на DOM дървото спрямо дефинираната XML Schema

Този подход позволява динамична работа с данните и демонстрира практическото приложение на DOM интерфейса в контекста на системата за компютърен магазин.

let doc = *document*.implementation.createDocument*(*null, "computerStore", null*)*;  
  
let root = doc.documentElement;  
let storeInfo = doc.createElement*(*"storeInfo"*)*;  
  
let storeName = doc.createElement*(*"storeName"*)*;  
storeName.textContent = "Tech Store";  
  
let location = doc.createElement*(*"location"*)*;  
  
let city = doc.createElement*(*"city"*)*;  
city.textContent = "София";  
  
let address = doc.createElement*(*"address"*)*;  
address.textContent = "ул. Иван Вазов 23";  
  
location.appendChild*(*city*)*;  
location.appendChild*(*address*)*;  
  
let contactNumber = doc.createElement*(*"contactNumber"*)*;  
contactNumber.textContent = "+359 2 123 456";  
  
storeInfo.appendChild*(*storeName*)*;  
storeInfo.appendChild*(*location*)*;  
storeInfo.appendChild*(*contactNumber*)*;  
  
root.appendChild*(*storeInfo*)*;

В тази версия на проекта демонстрираме създаването и работата с DOM структура, която съответства на XML схемата от версия 1. Реализацията включва следните компоненти:

#### Екземпляри на документа

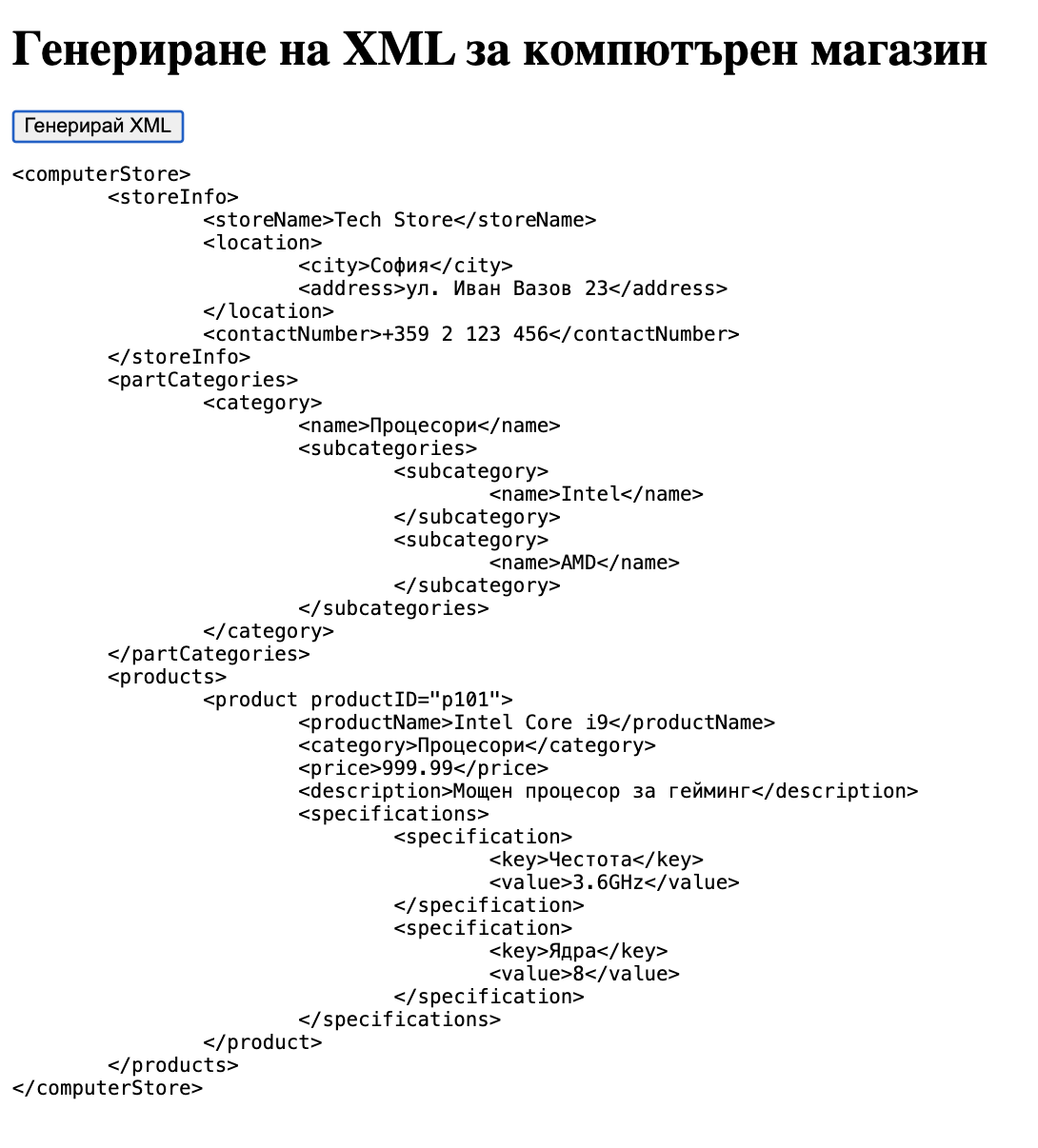
Създаването на DOM документа започва с използването на функцията `createDocument()`. Процесът включва:

* Създаване на основния (root) елемент на документа;
* Дефиниране на структурата чрез последователно добавяне на елементи:
* Създаване на комплексния тип "storeInfo" чрез `createElement()`;
* Добавяне на подчинени елементи към "storeInfo":
  + Информация за магазина
  + Данни за продуктите
  + Контактна информация
* Задаване на конкретни стойности на елементите;
* Прикрепване на създадената структура към root елемента чрез `appendChild()`.

#### Конструиране и валидиране чрез DOM

Процесът на конструиране и валидиране включва:

* Създаване на йерархична структура от възли;
* Добавяне на атрибути към елементите където е необходимо;
* Визуализация на резултата в браузър за проверка на коректността.



# **Версия „набор от XSLT трансформации“**

В тази версия на проекта се демонстрират възможностите за трансформация на XML документи чрез XSLT (eXtensible Stylesheet Language Transformations). Фокусът е върху два основни типа трансформации, които са реализирани чрез различни стилови таблици:

#### 1. Трансформации от XML към XML

Разработени са две различни XSLT таблици за трансформиране на XML документите в друг XML формат:

**Първа трансформация** (базирана на ExampleOne):

* Преструктуриране на данните за продуктите в по-опростен XML формат;
* Извличане на специфични характеристики за всеки продукт;

**Втора трансформация**:

Създаване на обобщена статистика за продуктите;

* Трансформиране на техническите спецификации в стандартизиран формат.

#### 2. Трансформации от XML към текст

Създадени са две отделни XSLT таблици за конвертиране на XML документите в чист текст:

**Първа текстова трансформация:**

* Генериране на каталог с продукти в четим формат
* Форматиране на цени и технически характеристики

**Втора текстова трансформация:**

* Информация за магазина
* Форматиране на продуктова информация за печат/каталог.

За демонстрация на трансформациите са използвани примерни документи от папка "DOMExamples", като особено внимание е отделено на "ExampleOne" и "ExampleFive.

Всяка трансформация е придружена от подробна документация и примери за входните и изходните данни, което позволява лесно разбиране и възпроизвеждане на процеса.

**Пример 1 (ExampleOne):**

*<*computerStore  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:noNamespaceSchemaLocation="../XMLSchema/schema.xsd"  
*>  
 <*storeInfo*>  
 <*storeName*>*PC Components*</*storeName*>  
 <*location*>  
 <*city*>*Русе*</*city*>  
 <*address*>*бул. Свобода 9*</*address*>  
 </*location*>  
 <*contactNumber*>*+359 82 876 543*</*contactNumber*>  
 </*storeInfo*>  
 <*partCategories*>  
 <*category*>  
 <*name*>*Захранване*</*name*>  
 <*subcategories*>  
 <*subcategory*>  
 <*name*>*Modular*</*name*>  
 </*subcategory*>  
 <*subcategory*>  
 <*name*>*Semi-Modular*</*name*>  
 </*subcategory*>  
 <*subcategory*>  
 <*name*>*Non-Modular*</*name*>  
 </*subcategory*>  
 </*subcategories*>  
 </*category*>  
 </*partCategories*>  
 <*products*>  
 <*product productID="p1"*>  
 <*productName*>*Cooler Master MWE 650W*</*productName*>  
 <*category*>*Захранване*</*category*>  
 <*price*>*75.00*</*price*>  
 <*description*>*Мощно и надеждно захранване за геймърски системи*</*description*>  
 <*specifications*>  
 <*specification*>  
 <*key*>*Мощност*</*key*>  
 <*value*>*650W*</*value*>  
 </*specification*>  
 <*specification*>  
 <*key*>*Тип*</*key*>  
 <*value*>*Modular*</*value*>  
 </*specification*>  
 <*specification*>  
 <*key*>*Ефективност*</*key*>  
 <*value*>*80+ Bronze*</*value*>  
 </*specification*>  
 </*specifications*>  
 </*product*>  
 </*products*>  
</*computerStore*>*

**Пример 2 (ExampleFive):**

*<?*xml-stylesheet type = html/xml" src = "../XSLT/ExampleOneToHTML.xslt"*?>  
<*computerStore  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:noNamespaceSchemaLocation="../XMLSchema/schema.xsd"  
*>  
 <*storeInfo*>  
 <*storeName*>*Tech Store*</*storeName*>  
 <*location*>  
 <*city*>*София*</*city*>  
 <*address*>*ул. Иван Вазов 23*</*address*>  
 </*location*>  
 <*contactNumber*>*+359 2 123 456*</*contactNumber*>  
 </*storeInfo*>  
 <*partCategories*>  
 <*category*>  
 <*name*>*Процесори*</*name*>  
 <*subcategories*>  
 <*subcategory*>  
 <*name*>*Intel*</*name*>  
 </*subcategory*>  
 <*subcategory*>  
 <*name*>*AMD*</*name*>  
 </*subcategory*>  
 </*subcategories*>  
 </*category*>  
 </*partCategories*>  
 <*products*>  
 <*product productID="p1"*>  
 <*productName*>*Intel Core i9*</*productName*>  
 <*category*>*Процесори*</*category*>  
 <*price*>*999.99*</*price*>  
 <*description*>*Мощен процесор за гейминг*</*description*>  
 <*specifications*>  
 <*specification*>  
 <*key*>*Честота*</*key*>  
 <*value*>*3.6GHz*</*value*>  
 </*specification*>  
 <*specification*>  
 <*key*>*Ядра*</*key*>  
 <*value*>*8*</*value*>  
 </*specification*>  
 </*specifications*>  
 </*product*>  
 <*product productID="p2"*>  
 <*productName*>*Intel Core i7*</*productName*>  
 <*category*>*Процесори*</*category*>  
 <*price*>*899.99*</*price*>  
 <*description*>*Средно мощен процесор за гейминг*</*description*>  
 <*specifications*>  
 <*specification*>  
 <*key*>*Честота*</*key*>  
 <*value*>*3.1GHz*</*value*>  
 </*specification*>  
 <*specification*>  
 <*key*>*Ядра*</*key*>  
 <*value*>*6*</*value*>  
 </*specification*>  
 </*specifications*>  
 </*product*>  
 <*product productID="p3"*>  
 <*productName*>*Intel Core i5*</*productName*>  
 <*category*>*Процесори*</*category*>  
 <*price*>*599.99*</*price*>  
 <*description*>* Процесор за гейминг*</*description*>  
 <*specifications*>  
 <*specification*>  
 <*key*>*Честота*</*key*>  
 <*value*>*2.6GHz*</*value*>  
 </*specification*>  
 <*specification*>  
 <*key*>*Ядра*</*key*>  
 <*value*>*4*</*value*>  
 </*specification*>  
 </*specifications*>  
 </*product*>  
 </*products*>  
</*computerStore*>*

## **Трансформиране на XML документи в друг XML код**

**Пример 1: Трансформация на продуктова информация**

За първата XSLT трансформация използваме файла от Пример 1, където демонстрираме преобразуване на оригиналната XML структура в нов, опростен XML формат. Трансформацията извлича следните ключови данни за всеки продукт в компютърния магазин:

* Наименование на продукта
* Детайлно описание
* Продажна цена

Резултатният XML документ представя информацията в по-компактен и фокусиран вид, което прави данните по-лесни за обработка и визуализация.

Структурата на изходния XML документ е организирана йерархично, като всеки продукт е представен като отделен елемент със съответните под-елементи за име, описание и цена. Това позволява лесна навигация и извличане на конкретна информация за всеки артикул в магазина.

* 1. **Трансформация:**

*<?*xml version="1.0" encoding="UTF-8"*?>  
<*xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"*>  
 <*xsl:output method="xml" encoding="UTF-8"*/>  
 <*xsl:template match="/"*>  
 <*products*>  
 <*xsl:for-each select="computerStore/products/product"*>  
 <*product*>  
 <*name*><*xsl:value-of select="productName"*/></*name*>  
 <*price*><*xsl:value-of select="price"*/></*price*>  
 <*description*><*xsl:value-of select="description"*/></*description*>  
 </*product*>  
 </*xsl:for-each*>  
 </*products*>  
 </*xsl:template*>  
</*xsl:stylesheet*>*

* 1. **Резултат след трансформацията:**

*<*products*>  
 <*product*>  
 <*name*>*Cooler Master MWE 650W*</*name*>  
 <*price*>*75.00*</*price*>  
 <*description*>*Мощно и надеждно захранване за геймърски системи*</*description*>  
 </*product*>  
</*products*>*

Изпълнението на XSLT трансформацията върху входния XML документ (ExampleOne) генерира XML изход, съдържащ данни за един продукт. Това е очакван резултат, тъй като в примерния XML документ е дефиниран само един продукт в структурата на магазина.

**Пример 2: Разглеждаме вторият пример и прилагаме трансформацията върху него. Трансформацията ни връща първо информацията за конкретния магазин и след това ни връща всички продукти в магазина както и техните имена, описания и цени.**

1. **Трансформация:**

*<?*xml version="1.0" encoding="UTF-8"*?>  
<*xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"*>  
 <*xsl:output method="xml" encoding="UTF-8"*/>  
 <*xsl:template match="/"*>  
 <*storeInfo*>  
 <*storeName*><*xsl:value-of select="computerStore/storeInfo/storeName"*/></*storeName*>  
 <*location*>  
 <*city*><*xsl:value-of select="computerStore/storeInfo/location/city"*/></*city*>  
 <*address*><*xsl:value-of select="computerStore/storeInfo/location/address"*/></*address*>  
 </*location*>  
 <*contactNumber*><*xsl:value-of select="computerStore/storeInfo/contactNumber"*/></*contactNumber*>  
 </*storeInfo*>  
 <*products*>  
 <*xsl:for-each select="computerStore/products/product"*>  
 <*product*>  
 <*name*><*xsl:value-of select="productName"*/></*name*>  
 <*price*><*xsl:value-of select="price"*/></*price*>  
 <*description*><*xsl:value-of select="description"*/></*description*>  
 </*product*>  
 </*xsl:for-each*>  
 </*products*>  
 </*xsl:template*>  
</*xsl:stylesheet*>*

1. **Резултат след трансформацията:**

*<*storeInfo*>  
 <*storeName*>*Tech Store*</*storeName*>  
 <*location*>  
 <*city*>*София*</*city*>  
 <*address*>*ул. Иван Вазов 23*</*address*>  
 </*location*>  
 <*contactNumber*>*+359 2 123 456*</*contactNumber*>  
</*storeInfo*>*<products*>  
<*product*>  
 <*name*>*Intel Core i9*</*name*>  
 <*price*>*999.99*</*price*>  
 <*description*>*Мощен процесор за гейминг*</*description*>  
</*product*>  
<*product*>  
 <*name*>*Intel Core i7*</*name*>  
 <*price*>*899.99*</*price*>  
 <*description*>*Средно мощен процесор за гейминг*</*description*>  
</*product*>  
<*product*>  
 <*name*>*Intel Core i5*</*name*>  
 <*price*>*599.99*</*price*>  
 <*description*>* Процесор за гейминг*</*description*>  
</*product*>  
</*products*>*

Изпълнената XSLT трансформация успешно постига поставените цели, като генерира коректно форматиран XML документ, съдържащ:

* Структурирана информация за компютърния магазин
* Данни за наличните продукти, включващи изискваните характеристики

Изходният XML документ съответства на предварително дефинираните изисквания за структура и съдържание, като запазва релевантната информация от входния документ в трансформиран вид. Това потвърждава правилното функциониране на създадената XSLT трансформация.

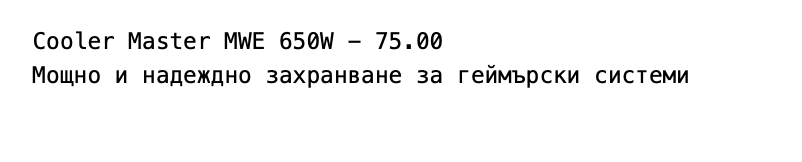
## **Трансформиране на XML документи в чист текст**

**Пример 1: Разглеждаме първият пример и прилагаме трансформацията върху него. Трансформацията трябва да ни върне всички продукти в магазина и тяхното описание.**

Трансформация:

*<?*xml version="1.0" encoding="UTF-8"*?>  
<*xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"*>  
 <*xsl:output method="text" encoding="UTF-8"*/>  
 <*xsl:template match="/"*>  
 <*xsl:for-each select="computerStore/products/product"*>  
 <*xsl:value-of select="productName"*/>* - *<*xsl:value-of select="price"*/>  
 <*xsl:text*>*&#10;*</*xsl:text*>  
 <*xsl:value-of select="description"*/>  
 <*xsl:text*>*&#10;*</*xsl:text*>  
 </*xsl:for-each*>  
 </*xsl:template*>  
</*xsl:stylesheet*>*

Резултат след трансформацията:



Резултатът от XSLT трансформацията демонстрира точно съответствие с дефинираните изисквания в кода:

1. Коректно извличане на данни:

* Трансформацията успешно обработва единствения продукт, наличен в ExampleOne;
* Структурата на изходния XML отразява точно логиката, заложена във "for-each" цикъла.

1. Форматиране на изхода:

* Подредбата на елементите следва последователността, дефинирана в XSLT шаблона;
* Начинът на представяне на информацията съответства на зададените правила за трансформация.

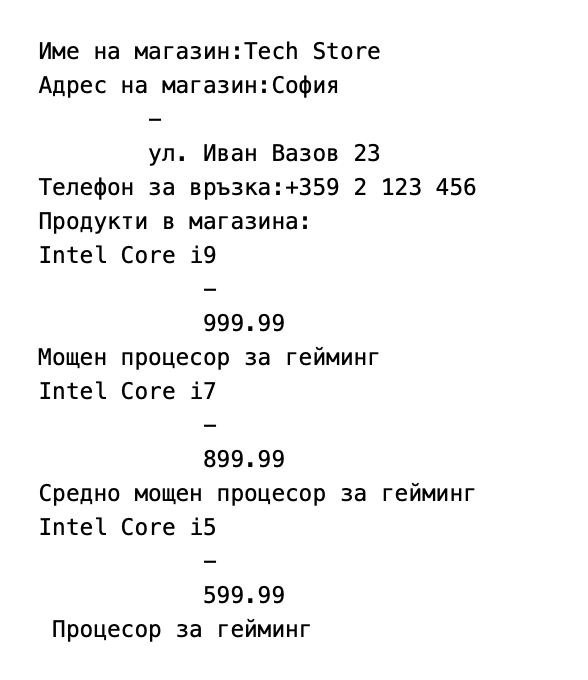
Това потвърждава, че имплементираната XSLT трансформация работи коректно и изпълнява точно заложената логика за обработка на данните.

**Пример 2:** **Ще разгледаме втория пример и ще приложим трансформацията върху него. Трансформацията трябва да върне информацията за магазин и всички продукти в него.**

Трансформация:

*<?*xml version="1.0" encoding="UTF-8"*?>  
<*xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"*>  
 <*xsl:output method="text" encoding="UTF-8"*/>  
 <*xsl:template match="/"*>  
 <*xsl:text*>*Име на магазин:*</*xsl:text*>  
 <*xsl:value-of select="computerStore/storeInfo/storeName"*/>  
 <*xsl:text*>*&#10;*</*xsl:text*>  
 <*xsl:text*>*Адрес на магазин:*</*xsl:text*>  
 <*xsl:value-of select="computerStore/storeInfo/location/city"*/>* -  
 *<*xsl:value-of select="computerStore/storeInfo/location/address"*/>  
 <*xsl:text*>*&#10;*</*xsl:text*>  
 <*xsl:text*>*Телефон за връзка:*</*xsl:text*>  
 <*xsl:value-of select="computerStore/storeInfo/contactNumber"*/>  
 <*xsl:text*>*&#10;*</*xsl:text*>  
 <*xsl:text*>*Продукти в магазина:*</*xsl:text*>  
 <*xsl:text*>*&#10;*</*xsl:text*>  
 <*xsl:for-each select="computerStore/products/product"*>  
 <*xsl:value-of select="productName"*/>* -  
 *<*xsl:value-of select="price"*/>  
 <*xsl:text*>*&#10;*</*xsl:text*>  
 <*xsl:value-of select="description"*/>  
 <*xsl:text*>*&#10;*</*xsl:text*>  
 </*xsl:for-each*>  
 </*xsl:template*>  
</*xsl:stylesheet*>*

Резултат след трансформацията:



Изпълнението на XSLT трансформацията генерира резултат, който напълно отговаря на зададените изисквания за извеждане на данните:

1. Пълнота на данните:

* Всички специфицирани елементи са успешно извлечени;
* Информацията е представена в изисквания формат.

1. Точност на трансформацията:

* Елементите са коректно трансформирани според дефинираните правила;
* Изходният формат съответства на очакваната структура.

Това потвърждава успешното изпълнение на трансформацията и точното съответствие с изискванията за представяне на данните.

# **Заключения**

В рамките на този курсов проект по XML успешно демонстрирах практическите умения и знания, придобити през обучението. Проектът беше реализиран в три ключови етапа, всеки от които допринесе за цялостното разбиране и практическо приложение на различни аспекти на XML технологиите:

* Разработка на XML Schema:
* Дефиниране на основната структура за данните
* Създаване на типове и релации между елементите
* Внедряване на базови валидационни правила
* Имплементация на DOM:
* Работа с DOM
* Управление на XML документи
* Валидация на данните
* XSLT трансформации:
* Създаване на базови трансформации
* Преобразуване на XML формати
* Генериране на структурирани изходи

**Възможности за усъвършенстване**

Проектът предоставя солидна основа за бъдещо развитие в следните направления:

* Разширяване на XML схемата:
  + - Добавяне на атрибути за:
      * Статус на наличност
      * Рейтинг и отзиви
    - Въвеждане на допълнителни елементи за:
      * Доставчици
      * Складови локации
      * История на продажбите
* Разработка на нови XSLT трансформации:
  + - Създаване на трансформации за:
      * Генериране на продуктови каталози
      * Ценови листи по категории
      * Справки за наличности

Тези подобрения ще обогатят функционалността на системата и ще разширят възможностите за обработка и анализ, както и представяне на данните за компютърния магазин, включително обогатена визуализация.