**Adatkezelés XML-ben**

**2. féléves beadandó feladat**

**JAXB programozás**

**Név:** Kiss Áron

**Neptun kód:** UHV61T

**Gyakorlatvezető:** Agárdi Anita

**Gyakorlat:** Szerda 10-12

**A feladat rövid leírása**

A feladat a korábban megírt XML sémát használó Java program írása, amely egy példa XML adatbázison hajt végre író, lekérdező, módosító, és törlő műveleteket.

A feladat megoldására JAXB programot írtam.

**Példa XML adatbázis**

A program a következő XML adatbázison hajtja végre a megfelelő műveleteket. Az adatbázis a korábban bemutatott XMLSchema-ra épül:

*<?***xml version="1.0" encoding="UTF-8" standalone="yes"***?>*<**database**>  
 <**users**>  
 <**user**>  
 <**name**>Toth Ferenc</**name**>  
 <**email**>tf@example.com</**email**>  
 <**mobileNumber**>06301234567</**mobileNumber**>  
 <**image src="https://example.com/img1.jpg"**/>  
 <**id**>ID-1</**id**>  
 <**username**>tf</**username**>  
 <**password**>  
 <**hash**>$2a$04$Pd3yH9dGEf6UTZli5s2M2O3.10rx0rJKDSSwuZjidZ6Tepch.agGm</**hash**>  
 </**password**>  
 </**user**>  
 <**user**>  
 <**name**>Kovacs Lajos</**name**>  
 <**email**>kl@example.com</**email**>  
 <**mobileNumber**>06307654321</**mobileNumber**>  
 <**image src="http://img.example.com/img.png"**/>  
 <**id**>ID-2</**id**>  
 <**username**>kovla</**username**>  
 <**password**>  
 <**hash**>$2a$04$7I0jg853YVVp8VDf1u6.fubfuNJ0D7bAmuHTAZddHEQyN1sS8cJi2</**hash**>  
 </**password**>  
 </**user**>  
 <**user**>  
 <**name**>Nagy Geza</**name**>  
 <**email**>nagy@example.com</**email**>  
 <**mobileNumber**>06201236547</**mobileNumber**>  
 <**image src="https://google.com/logo.png"**/>  
 <**id**>ID-3</**id**>  
 <**username**>nagyg</**username**>  
 <**password**>  
 <**hash**>$2a$04$qM38KQNWMym.FCY4k9zK4eS/RXUVCg1XVfPLvfDdW20U0CzMu5ZzW</**hash**>  
 </**password**>  
 </**user**>  
 </**users**>  
 <**librarians**>  
 <**librarian**>  
 <**employeeID**>ID-4</**employeeID**>  
 <**salary**>125000</**salary**>  
 <**userID**>ID-1</**userID**>  
 </**librarian**>  
 <**librarian**>  
 <**employeeID**>ID-5</**employeeID**>  
 <**salary**>250000</**salary**>  
 <**userID**>ID-2</**userID**>  
 </**librarian**>  
 </**librarians**>  
 <**readers**>  
 <**reader**>  
 <**id**>ID-6</**id**>  
 <**userID**>ID-1</**userID**>  
 </**reader**>  
 <**reader**>  
 <**id**>ID-7</**id**>  
 <**userID**>ID-3</**userID**>  
 </**reader**>  
 </**readers**>  
 <**books**>  
 <**book**>  
 <**ISBN**>ISBN-12345678</**ISBN**>  
 <**author**>Pasztor Imre</**author**>  
 <**title**>Analizis 8.</**title**>  
 <**publishYear**>2017</**publishYear**>  
 <**genre**>Scifi</**genre**>  
 </**book**>  
 <**book**>  
 <**ISBN**>ISBN-87654321</**ISBN**>  
 <**author**>Toth Palne</**author**>  
 <**title**>OS vizsga elsore</**title**>  
 <**publishYear**>2016</**publishYear**>  
 <**genre**>Fiction</**genre**>  
 </**book**>  
 </**books**>  
 <**bookInstances**>  
 <**bookInstance**>  
 <**inventoryNo**>IN-123456</**inventoryNo**>  
 <**bookISBN**>ISBN-12345678</**bookISBN**>  
 <**isLoaned**>true</**isLoaned**>  
 </**bookInstance**>  
 <**bookInstance**>  
 <**inventoryNo**>IN-123457</**inventoryNo**>  
 <**bookISBN**>ISBN-12345678</**bookISBN**>  
 <**isLoaned**>false</**isLoaned**>  
 </**bookInstance**>  
 <**bookInstance**>  
 <**inventoryNo**>IN-123458</**inventoryNo**>  
 <**bookISBN**>ISBN-12345678</**bookISBN**>  
 <**isLoaned**>false</**isLoaned**>  
 </**bookInstance**>  
 <**bookInstance**>  
 <**inventoryNo**>IN-1234569</**inventoryNo**>  
 <**bookISBN**>ISBN-87654321</**bookISBN**>  
 <**isLoaned**>true</**isLoaned**>  
 </**bookInstance**>  
 </**bookInstances**>  
 <**borrowings**>  
 <**borrowing**>  
 <**id**>ID-8</**id**>  
 <**creationDate**>2016-01-26</**creationDate**>  
 <**expirationDate**>2016-03-21</**expirationDate**>  
 <**status**>EXPIRED</**status**>  
 <**readerID**>ID-6</**readerID**>  
 <**bookInstanceInventoryNo**>IN-123456</**bookInstanceInventoryNo**>  
 </**borrowing**>  
 <**borrowing**>  
 <**id**>ID-9</**id**>  
 <**creationDate**>2017-01-23</**creationDate**>  
 <**expirationDate**>2017-01-25</**expirationDate**>  
 <**status**>RETURNED</**status**>  
 <**readerID**>ID-6</**readerID**>  
 <**bookInstanceInventoryNo**>IN-1234569</**bookInstanceInventoryNo**>  
 </**borrowing**>  
 <**borrowing**>  
 <**id**>ID-10</**id**>  
 <**creationDate**>2018-01-26</**creationDate**>  
 <**expirationDate**>2018-05-11</**expirationDate**>  
 <**status**>BORROWED</**status**>  
 <**readerID**>ID-7</**readerID**>  
 <**bookInstanceInventoryNo**>IN-1234569</**bookInstanceInventoryNo**>  
 </**borrowing**>  
 </**borrowings**>  
</**database**>

**JAXB program**

A program parancssori menürendszerrel működik, a következő műveletek végrehajtására képes:

* Új könyv hozzáadása
* Könyv lekérdezése ISBN alapján
* Könyvek lekérdezése az író neve alapján
* Könyv módosítása
* Könyv törlése

A program a sémából XJC-vel generált osztályok mellett 2 további osztályt tartalmaz:

* Runnable: A program belépési pontja, amely biztosítja a felhasználó számára a menürendszert, és továbbítja a kéréseit a Controller osztálynak.
* Controller: Ez az osztály kezeli az XML adatbázist, JAXB segítségével. Elérhető metódusai a következők:
  + **void** createBook (BookType bookToAdd)

Az átadott BookType objektumban szereplő adatokkal új könyvet hoz létre az adatbázisban.

* + BookType readBook (**long** isbn)

Visszaadja az adott ISBN-nel rendelkező könyvet az adatbázisból.

* + ArrayList<BookType> readBooksByAuthor (String authorToSearch)

Visszaadja az adott író könyveit az adatbázisból.

* + **void** updateBook (BookType bookToUpdate)

Módosítja a megfelelő könyvet az adatbázisban (ISBN alapján).

* + **void** deleteBook (String isbnToSearch)

Törli a megfelelő ISBN-nel rendelkező könyvet az adatbázisból.

Controller.java:

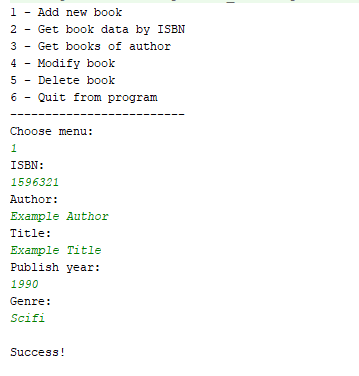
**package** hu.uni.miskolc.iit.uhv61t.XmlAssignment;  
  
**import** hu.uni.miskolc.iit.uhv61t.XmlAssignment.xjcModels.BookType;  
**import** hu.uni.miskolc.iit.uhv61t.XmlAssignment.xjcModels.Database;  
  
**import** javax.xml.bind.JAXBContext;  
**import** javax.xml.bind.JAXBException;  
**import** javax.xml.bind.Marshaller;  
**import** javax.xml.bind.Unmarshaller;  
**import** java.io.File;  
**import** java.util.ArrayList;  
**import** java.util.Collection;  
  
*/\*\*  
 \* This class does the data manipulation in XML database.  
 \*/***class** Controller {  
  
 */\*\*  
 \* Root element of the database XML.  
 \*/* **private** Database **database**;  
  
 Controller() **throws** JAXBException {  
 **this**.initializeDatabase();  
 }  
  
 */\*\*  
 \* Instantiates a new Database object, and fills it with the data from XML.  
 \** ***@throws*** *JAXBException  
 \*/* **private void** initializeDatabase () **throws** JAXBException {  
 File xml = **new** File(**"resources/database.xml"**);  
 JAXBContext jaxbContext = JAXBContext.*newInstance*(Database.**class**);  
 Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();  
  
 **this**.**database** = (Database) unmarshaller.unmarshal(xml);  
 }  
  
 */\*\*  
 \* Writes out the changes that are done in database field.  
 \** ***@throws*** *JAXBException  
 \*/* **private void** writeOutChanges () **throws** JAXBException {  
 File xml = **new** File(**"resources/database.xml"**);  
 JAXBContext jaxbContext = JAXBContext.*newInstance*(Database.**class**);  
 Marshaller marshaller = jaxbContext.createMarshaller();  
 marshaller.setProperty(Marshaller.***JAXB\_FORMATTED\_OUTPUT***, **true**); *//pretty print* marshaller.marshal(**this**.**database**, xml);  
 }  
  
 */\*\*  
 \* Creates a new book in database.  
 \** ***@param bookToAdd*** *The book to add to database.  
 \** ***@throws*** *JAXBException  
 \*/* **void** createBook (BookType bookToAdd) **throws** JAXBException {  
 **this**.**database**.getBooks().getBook().add(bookToAdd);  
 **this**.writeOutChanges();  
 }  
  
 */\*\*  
 \* Returns the book with the given ISBN.  
 \** ***@param isbn*** *ISBN to search.  
 \** ***@return*** *Books with the given ISBN.  
 \*/* BookType readBook (**long** isbn) {  
 Collection<BookType> books = **this**.**database**.getBooks().getBook();  
 String isbnToSearch = **"ISBN-"** + String.*valueOf*(isbn);  
  
 **for** (BookType book : books) {  
 **if** (book.getISBN().equals(isbnToSearch)) {  
 **return** book;  
 }  
 }  
  
 **return null**;  
 }  
  
 */\*\*  
 \* Returns the books of the given author.  
 \** ***@param authorToSearch*** *Author to search.  
 \** ***@return*** *The authors' books.  
 \*/* ArrayList<BookType> readBooksByAuthor (String authorToSearch) {  
 Collection<BookType> books = **this**.**database**.getBooks().getBook();  
 ArrayList<BookType> results = **new** ArrayList<>();  
  
 **for** (BookType book : books) {  
 **if** (book.getAuthor().equals(authorToSearch)) {  
 results.add(book);  
 }  
 }  
  
 **return** results;  
 }  
  
 */\*\*  
 \* Updates book based on its ISBN  
 \** ***@param bookToUpdate*** *The book with the appropriate ISBN and new information.  
 \** ***@throws*** *JAXBException  
 \*/* **void** updateBook (BookType bookToUpdate) **throws** JAXBException {  
 Collection<BookType> books = **this**.**database**.getBooks().getBook();  
 BookType found = **null**;  
  
 **for** (BookType book : books) {  
 **if** (book.getISBN().equals(bookToUpdate.getISBN())) {  
 found = book;  
 }  
 }  
  
 books.remove(found);  
 books.add(bookToUpdate);  
 **this**.writeOutChanges();  
 }  
  
 */\*\*  
 \* Deletes the book with the given ISBN.  
 \** ***@param isbnToSearch*** *The ISBN of book to delete.  
 \** ***@throws*** *JAXBException  
 \*/* **void** deleteBook (String isbnToSearch) **throws** JAXBException {  
 Collection<BookType> books = **this**.**database**.getBooks().getBook();  
  
 **for** (BookType book : books) {  
 **if** (book.getISBN().equals(isbnToSearch)) {  
 books.remove(book);  
 }  
 }  
  
 **this**.writeOutChanges();  
 }  
}

Runnable.java:

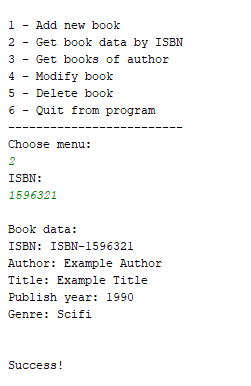
**package** hu.uni.miskolc.iit.uhv61t.XmlAssignment;  
  
**import** hu.uni.miskolc.iit.uhv61t.XmlAssignment.xjcModels.BookType;  
**import** hu.uni.miskolc.iit.uhv61t.XmlAssignment.xjcModels.GenreType;  
  
**import** javax.xml.bind.JAXBException;  
**import** java.util.ArrayList;  
**import** java.util.Scanner;  
  
*/\*\*  
 \* Main entry point of the program.  
 \*/***public class** Runnable {  
  
 */\*\*  
 \* The controller, that manipulate data in database.  
 \*/* **static** Controller *controller*;  
  
 **public static void** main(String[] args) **throws** JAXBException {  
 *controller* = **new** Controller();  
 **int** chosenMenu;  
  
 **do** {  
 *printMenu*();  
 chosenMenu = *readMenuChoice*();  
 **switch** (chosenMenu) {  
 **case** 1:  
 BookType bookToAdd = *getNewBook*();  
 *controller*.createBook(bookToAdd);  
 *printSuccess*();  
 **break**;  
 **case** 2:  
 BookType book = *getBookByISBN*();  
 *printOutBook*(book);  
 *printSuccess*();  
 **break**;  
 **case** 3:  
 ArrayList<BookType> books = *getBooksByAuthor*();  
  
 **if** (books.isEmpty()) {  
 System.***out***.println(**"Book not found by the given author"**);  
 **break**;  
 }  
  
 **for** (BookType bookOfAuthor : books) {  
 *printOutBook*(bookOfAuthor);  
 }  
  
 *printSuccess*();  
 **break**;  
 **case** 4:  
 BookType bookToModify = *getNewBook*();  
 *controller*.updateBook(bookToModify);  
 *printSuccess*();  
 **break**;  
 **case** 5:  
 BookType bookToDelete = *getBookByISBN*();  
 *controller*.deleteBook(bookToDelete.getISBN());  
 *printSuccess*();  
 **break**;  
 }  
 } **while** (chosenMenu != 6);  
  
 System.***out***.println();  
 System.***out***.println(**"Exit ..."**);  
 }  
  
 */\*\*  
 \* Prints the menu to stdout.  
 \*/* **private static void** printMenu () {  
 System.***out***.println(**"1 - Add new book"**);  
 System.***out***.println(**"2 - Get book data by ISBN"**);  
 System.***out***.println(**"3 - Get books of author"**);  
 System.***out***.println(**"4 - Modify book"**);  
 System.***out***.println(**"5 - Delete book"**);  
 System.***out***.println(**"6 - Quit from program"**);  
 System.***out***.println(**"-------------------------"**);  
 System.***out***.println(**"Choose menu: "**);  
 }  
  
 */\*\*  
 \* Reads the menu choice of the user from stdin.  
 \** ***@return*** *\*/* **private static int** readMenuChoice () {  
 Scanner scanner = **new** Scanner(System.***in***);  
 **return** Integer.*valueOf*(scanner.nextLine());  
 }  
  
 */\*\*  
 \* Prints success message to stdout.  
 \*/* **private static void** printSuccess () {  
 System.***out***.println();  
 System.***out***.println(**"Success!"**);  
 System.***out***.println();  
 }  
  
 */\*\*  
 \* Reads data of new book from stdin.  
 \** ***@return*** *The new book.  
 \*/* **private static** BookType getNewBook () {  
 Scanner scanner = **new** Scanner(System.***in***);  
  
 System.***out***.println(**"ISBN: "**);  
 **long** ISBN = Long.*valueOf*(scanner.nextLine());  
  
 System.***out***.println(**"Author: "**);  
 String author = String.*valueOf*(scanner.nextLine());  
  
 System.***out***.println(**"Title: "**);  
 String title = String.*valueOf*(scanner.nextLine());  
  
 System.***out***.println(**"Publish year: "**);  
 **int** year = Integer.*valueOf*(scanner.nextLine());  
  
 System.***out***.println(**"Genre: "**);  
 String genre = String.*valueOf*(scanner.nextLine());  
  
 BookType bookToAdd = **new** BookType();  
 bookToAdd.setISBN(**"ISBN-"** + String.*valueOf*(ISBN));  
 bookToAdd.setAuthor(author);  
 bookToAdd.setTitle(title);  
 bookToAdd.setPublishYear(year);  
 bookToAdd.setGenre(GenreType.*fromValue*(genre));  
  
 **return** bookToAdd;  
 }  
  
 */\*\*  
 \* Returns the book with the given ISBN.  
 \** ***@return*** *Book with the given ISBN.  
 \*/* **private static** BookType getBookByISBN () {  
 Scanner scanner = **new** Scanner(System.***in***);  
  
 System.***out***.println(**"ISBN: "**);  
 **long** isbn = Long.*valueOf*(scanner.nextLine());  
  
 **return** *controller*.readBook(isbn);  
 }  
  
 */\*\*  
 \* Prints out a book object to stdout.  
 \** ***@param book*** *Book to print out.  
 \*/* **private static void** printOutBook (BookType book) {  
 System.***out***.println();  
  
 **if** (book == **null**) {  
 System.***out***.println(**"Book not found with the given ISBN"**);  
 **return**;  
 }  
  
 System.***out***.println(**"Book data:"**);  
 System.***out***.println(**"ISBN: "** + book.getISBN());  
 System.***out***.println(**"Author: "** + book.getAuthor());  
 System.***out***.println(**"Title: "** + book.getTitle());  
 System.***out***.println(**"Publish year: "** + book.getPublishYear());  
 System.***out***.println(**"Genre: "** + book.getGenre().value());  
  
 System.***out***.println();  
 }  
  
 */\*\*  
 \* Returns an array list with the books of an author.  
 \** ***@return*** *Books of the author.  
 \*/* **private static** ArrayList<BookType> getBooksByAuthor () {  
 Scanner scanner = **new** Scanner(System.***in***);  
  
 System.***out***.println(**"Author: "**);  
 String author = scanner.nextLine();  
  
 **return** *controller*.readBooksByAuthor(author);  
 }  
}

A teljes program [elérhető a GitHub-on](https://github.com/aron123/XML-assignment) is.

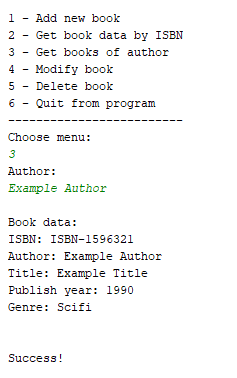
**Képernyőképek**



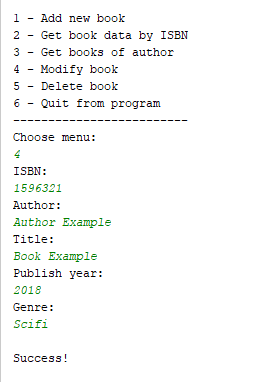
*Könyv hozzáadása*



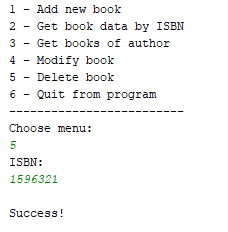
*Könyv keresése ISBN alapján*



*Író könyveinek keresése*



*Könyv adatainak módosítása*



*Könyv törlése*