Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 5**

**«Тестування REST Web-service»**

**з дисципліни «Управління якістю програмного забезпечення»**

**Лектор:**

асист. кафедри ПЗ

Ваврук І.Є.

**Виконав:**

студ. групи ПЗМ-11

Павленчик М. М.

**Прийняв:**

асист. кафедри ПЗ

Ваврук І.Є.

«\_\_\_» \_\_\_\_\_\_ 2019 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2019

**ТЕМА РОБОТИ**: Тестування REST Web-service.

**МЕТА РОБОТИ**: Освоїти основні принципи тестування REST Web-service.

### ЛАБОРАТОРНЕ ЗАВДАННЯ

1. Написати REST сервіс відповідно до варіанту
2. Написати REST-клієнт відповідно до варіанту.
3. Протестувати всі методи, використовуючи дата провайдер.
4. Здійснити логування усіх тестів в текстовий документ.
5. Сформувати звіт.

**Варіант 3 (Library)**

1. Створити веб-сервіс «Бібліотека», який емулює роботу автоматичної електронної бібліотеки з можливістю: 1) отримати список книжок, 2) отримати одну книжку, 3)повернути книжку, 4)Обміняти книжку, 5) отримати 5 книжок одного автора.

Книжка має назву, автора, жанр. Сервер має обробник помилок на випадок відсутності книжки/книжок (в залежності від випадку) з відповідними повідомленнями.

1. Створити клієнт веб-сервісу «Бібліотека». Створити тести для мінімум 10 кейсів з обробкою очікуваного результату. Всі операції мають логуватися в лог-file та результат тестування має бути виведений в репорт.

### РЕЗУЛЬТАТИ

**Код АРІ контроллеру:**

public class LibraryController : ApiController

{

public LibraryController(ILibraryService libraryService)

{

this.LibraryService = libraryService;

}

protected ILibraryService LibraryService { get; set; }

[HttpGet]

public IHttpActionResult GetAllBooks()

{

var booksList = LibraryService.GetAllBooks();

return Ok(booksList);

}

[HttpGet]

public IHttpActionResult GetBook(int bookId)

{

try

{

var book = LibraryService.GetBookById(bookId);

return Ok(book);

}

catch (Exception ex)

{

return BadRequest(ex.Message);

}

}

[HttpGet]

public IHttpActionResult GetBooksByAuthor(string author, int? count = null)

{

try

{

var booksList = LibraryService.GetBooksByAuthor(author, count);

return Ok(booksList);

}

catch (Exception ex)

{

return BadRequest(ex.Message);

}

}

[HttpPut]

public IHttpActionResult LoadNewBook([FromBody]Book newBook)

{

try

{

var newBookId = LibraryService.LoadNewBook(newBook);

return Ok(newBookId);

}

catch (Exception ex)

{

return BadRequest(ex.Message);

}

}

[HttpDelete]

public IHttpActionResult ReturnBook(int bookId)

{

try

{

LibraryService.ReturnBook(bookId);

return Ok("Book returned successfully!");

}

catch (Exception ex)

{

return BadRequest(ex.Message);

}

}

[HttpPost]

public IHttpActionResult ReplaceBooks([FromBody]ReplaceBooksModel model)

{

try

{

var newRentId = LibraryService.ReplaceBook(model);

return Ok(newRentId);

}

catch (Exception ex)

{

return BadRequest(ex.Message);

}

}

}

**Код клієнта:**

public class LibraryClient : RESTClient

{

private string baseUrl;

public LibraryClient()

{

baseUrl = ConfigurationManager.AppSettings["BaseUrl"];

}

public RESTResult<List<Book>, RequestError> GetAllBooks()

{

return Get<List<Book>, RequestError>(baseUrl);

}

public RESTResult<Book, RequestError> GetBook(int bookId)

{

return Get<Book, RequestError>($"{baseUrl}/GetBook?bookId={bookId}");

}

public RESTResult<List<Book>, RequestError> GetBooksByAuthor(string author, int? count)

{

return Get<List<Book>, RequestError>($"{baseUrl}?author={author}&count={count}");

}

public RESTResult<int?, RequestError> LoadNewBook(Book newBook)

{

return Put<int?, RequestError>($"{baseUrl}/LoadNewBook", newBook);

}

public RESTResult<string, RequestError> ReturnBook(int bookId)

{

return Delete<string, RequestError>($"{baseUrl}/ReturnBook?bookId={bookId}");

}

public RESTResult<int?, RequestError> ReplaceBooks(int oldBookId, int newBookId, int userId)

{

return Post<int?, RequestError>($"{baseUrl}/ReplaceBooks", new ReplaceBooksModel

{

OldBookId = oldBookId,

NewBookId = newBookId,

UserId = userId

});

}

}

**Код тестів:**

[TestFixture]

public class LibraryTest

{

private static readonly Logger Logger = LogManager.GetCurrentClassLogger();

private LibraryClient Client { get; set; }

public LibraryTest()

{

Client = new LibraryClient();

}

[TestCase]

public void GetAllBooks\_CorrectRequest\_LoadedListOfAllBooks()

{

//arrange

Logger.Info($"Loading all books.");

//act

var actualResult = Client.GetAllBooks();

var content = actualResult.model;

Logger.Info($"Result count: {content.Count}");

//assert

Assert.IsNotNull(content);

}

[TestCaseSource(typeof(LibraryDataProvider), "ExistingBooks")]

public void GetBook\_ExistingBookId\_LoadedSelectedBook(int bookId, string bookName)

{

//arrange

Logger.Info($"Loading bookId = {bookId}");

//act

var actualResult = Client.GetBook(bookId);

var content = actualResult.model;

Logger.Info($"Book name: {content.Name}");

//assert

Assert.AreEqual(bookName, content.Name);

}

[TestCaseSource(typeof(LibraryDataProvider), "NonexistentBookIds")]

public void GetBook\_NonexistentBookIds\_BadRequestResultReturned(int bookId)

{

//arrange

Logger.Info($"Loading bookId = {bookId}");

//act

var actualResult = Client.GetBook(bookId);

var badRequestResult = actualResult.error;

Logger.Info($"Request result: {badRequestResult.Message}");

//assert

Assert.AreEqual($"The book with ID = {bookId} has not been found.", badRequestResult.Message);

}

[TestCaseSource(typeof(LibraryDataProvider), "Authors")]

public void GetAllBooks\_CorrectRequest\_LoadedListOfAllBooks(string author, int? count)

{

//arrange

Logger.Info($"Loading {count?.ToString() ?? "all"} {author}'s books.");

//act

var actualResult = Client.GetBooksByAuthor(author, count);

var content = actualResult.model;

Logger.Info($"Result count: {content?.Count}");

//assert

Assert.IsTrue(content != null && (!count.HasValue || content.Count <= count));

}

[TestCaseSource(typeof(LibraryDataProvider), "NewBooks")]

public void LoadNewBook\_ValidBooks\_LoadedNewBookId(Book newBook)

{

//arrange

Logger.Info($"Loading book name = {newBook.Name}");

//act

var actualResult = Client.LoadNewBook(newBook);

var content = actualResult.model;

Logger.Info($"New book id: {content}");

//assert

Assert.IsNotNull(content);

}

[TestCaseSource(typeof(LibraryDataProvider), "BooksInRent")]

public void ReturnBook\_ValidBooks\_LoadedSuccessMessage(int bookId)

{

//arrange

Logger.Info($"Returning book with ID = {bookId}");

//act

var actualResult = Client.ReturnBook(bookId);

var content = actualResult.model;

Logger.Info($"Result: {content}");

//assert

Assert.AreEqual("Book returned successfully!", content);

}

[TestCaseSource(typeof(LibraryDataProvider), "NonexistentBookIds")]

public void ReturnBook\_NonexistentBookIds\_BadRequestResultReturned(int bookId)

{

//arrange

Logger.Info($"Returning book with ID = {bookId}");

//act

var actualResult = Client.ReturnBook(bookId);

var content = actualResult.error;

Logger.Info($"Result: {content.Message}");

//assert

Assert.AreEqual($"The book with ID = {bookId} has not been rent.", content.Message);

}

[TestCaseSource(typeof(LibraryDataProvider), "ReplaceBooks")]

public void ReplaceBooks\_ValidBooks\_LoadedNewRentId(int oldBookId, int newBookId, int userId)

{

//arrange

Logger.Info($"ReplaceBooks book with ID = {oldBookId}, to the book with ID = {newBookId} by {userId}.");

//act

var actualResult = Client.ReplaceBooks(oldBookId, newBookId, userId);

var content = actualResult.model;

Logger.Info($"New rent ID: {content}");

//assert

Assert.IsNotNull(content);

}

[TestCaseSource(typeof(LibraryDataProvider), "ReplaceBooksWithNonexistentUsers")]

public void ReplaceBooks\_InvalidUser\_BadRequestResultReturned(int oldBookId, int newBookId, int userId)

{

//arrange

Logger.Info($"ReplaceBooks book with ID = {oldBookId}, to the book with ID = {newBookId} by {userId}.");

//act

var actualResult = Client.ReplaceBooks(oldBookId, newBookId, userId);

var content = actualResult.error;

Logger.Info($"Result: {content.Message}");

//assert

Assert.AreEqual($"The user with ID = {userId} doesn't exist.", content.Message);

}

}

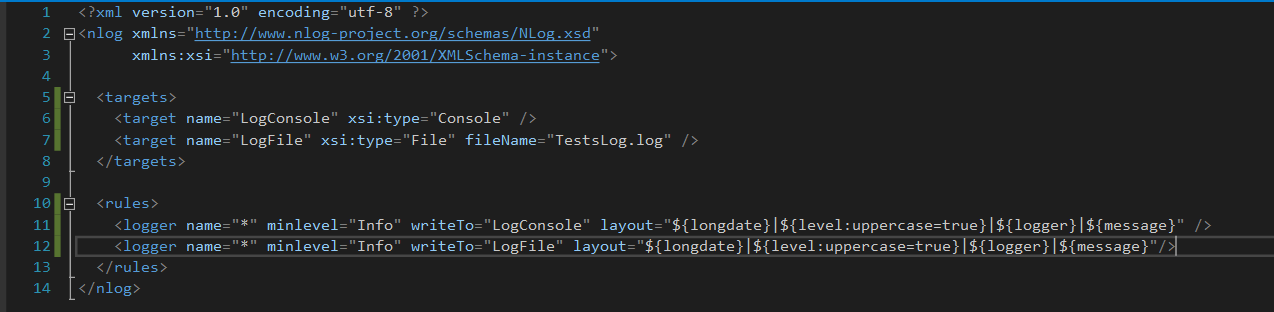


Рис. 1. Файл конфігурації для логування за допомогою бібліотеки NLog

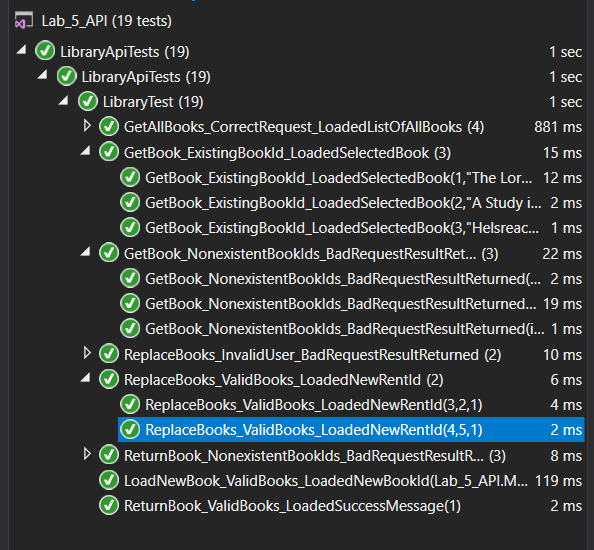


Рис. 2. Результати запуску тестів

### ВИСНОВКИ

На даній лабораторній роботі я освоїв основні принципи тестування REST Web-service.