

<https://www.halvorsen.blog>

# Consuming PHP Web API in WinForms App

Hans-Petter Halvorsen



# Contents

- We have in a previous Tutorial made a simple **CRUD Web API** using **PHP** and **MySQL**.
  - CRUD means Create, Read, Update and Delete data in the Database.
- In a previous Tutorial I have also made a Windows Forms CRUD Application that has direct access to a SQL Database and the Windows Forms App communicate directly with the database using the ADO.NET package in Visual Studio/C#.
- In real-life scenarios you normally don't have direct access to the database due to security issues.
- Also to be able to get direct access to the database you need to specify access to your IP address in the server firewall settings.
  - That may be OK for 1 or 2 computers, but what if hundreds or thousands of computer need access?
- So, the focus in this Tutorial is to make an updated version of the previous Windows Forms CRUD Application that has direct access to a SQL Database where we use the **CRUD PHP Web API** instead.
  - The CRUD PHP Web API relays entirely on HTTP so no IP access, etc. need to be set up.

<https://www.halvorsen.blog>

# Introduction



[Table of Contents](#)

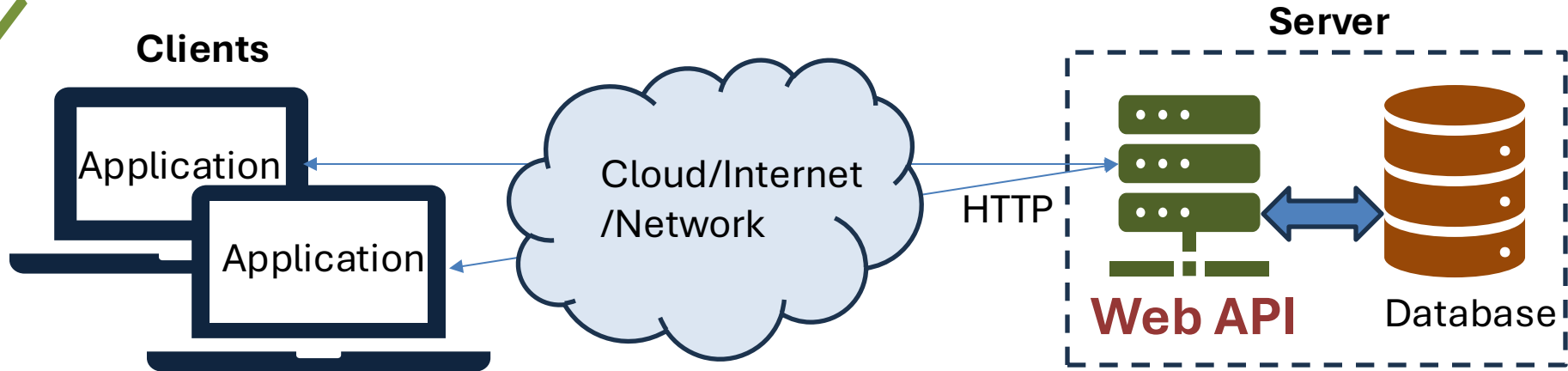
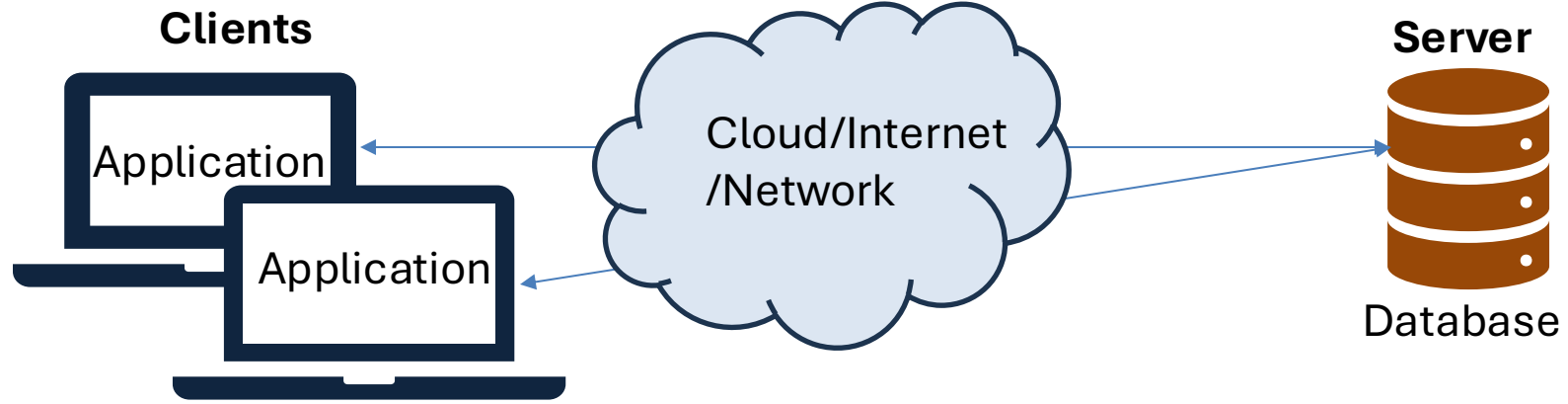
Hans-Petter Halvorsen

# Web API

- We can create/use APIs for internal use inside an Application or between 2 or more Applications.
- Basically, an API can be just a Class with Methods that you use several places inside an Application or that you share between multiple Applications.
- A set of Stored Procedures in a Database can also be an API.
- When the Application that consume/use the API is on a local PC and the API itself is located on a Server, we can talk about so-called “Web APIs”.
- Such Web APIs also very often perform CRUD operations against a Database located on the Web.
- Normally it is not allowed to connect directly to a Database located in the Cloud from a local computer unless you configure and give access to the IP addresses for those clients.

# Web API

Normally it is not allowed to connect directly to a Database located in the Cloud from a local computer unless you configure and give access to the IP addresses for those clients.



# References

- Make HTTP requests with the HttpClient class:

<https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/http/httpclient>

<https://www.halvorsen.blog>

# Backend/Server-side



Hans-Petter Halvorsen

[Table of Contents](#)

# Database

A simple Database with the following Table is used:

```
CREATE TABLE BOOK
(
  BookId int PRIMARY KEY AUTO_INCREMENT,
  Title varchar(100) NOT NULL,
  Author varchar(100) NOT NULL,
  Topic varchar(100) NOT NULL
);
```



# CRUD PHP Web API

In a previous Tutorial we have made a simple Web API in PHP that has CRUD functionality and communicated with a MySQL Database.

The following methods was created in the Web API:

- **GetBooks.php**
- **GetBookById.php**
- **InsertBook.php**
- **UpdateBook.php**
- **DeleteBook.php**

<https://www.halvorsen.blog>

# Windows Forms App

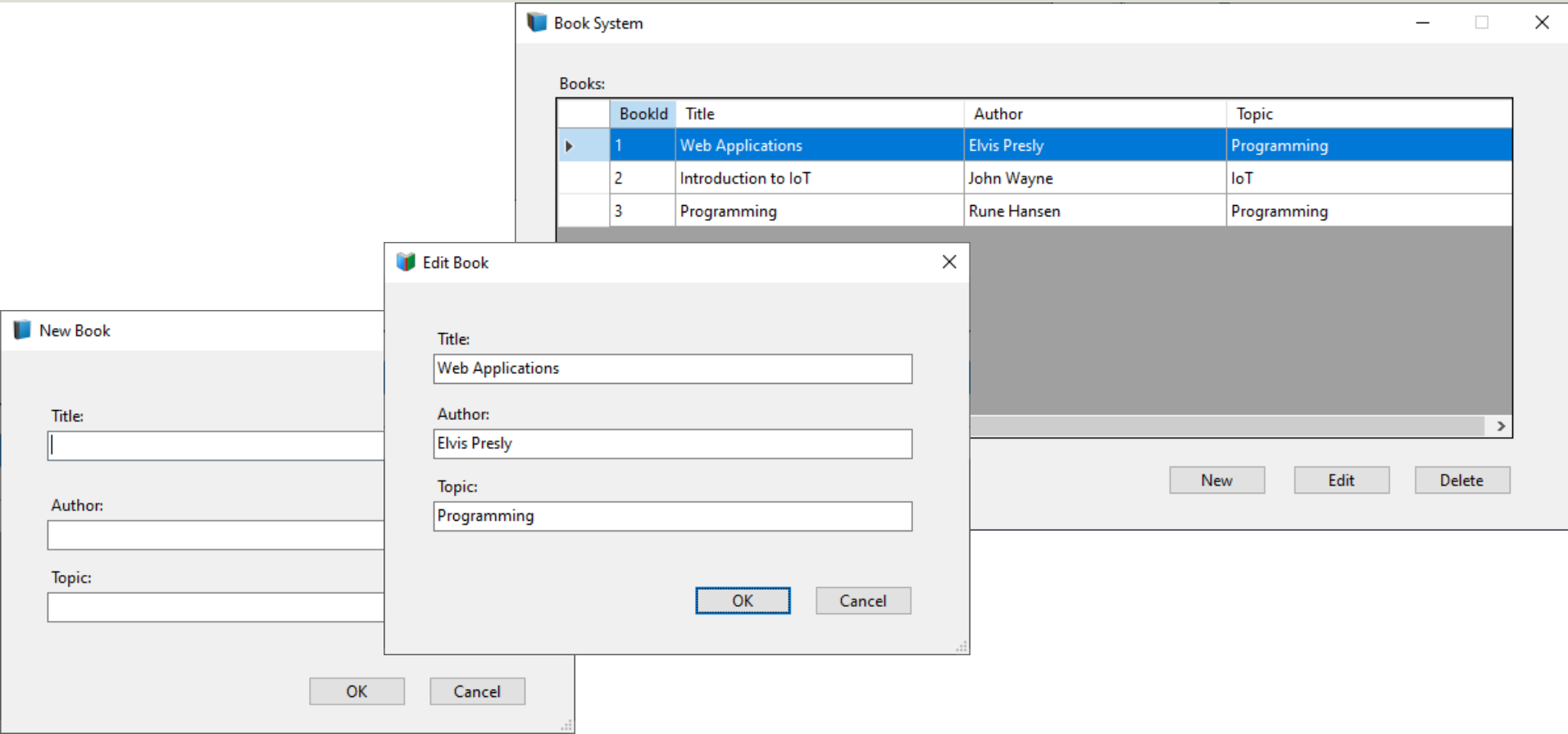
with direct Database Communication



Hans-Petter Halvorsen

[Table of Contents](#)

# WinForm App



# WinForm App Visual Studio

The screenshot displays the Visual Studio IDE with a WinForm application named "BookSystem" in design mode. The main window shows a form with a "Books:" label, a large empty rectangular area, and three buttons labeled "New", "Edit", and "Delete".

**Toolbox:** Lists various controls under "Common Controls" and "All Windows Forms".

**Solution Explorer:** Shows the project structure for "BookSystem", including "Classes", "Resources", and "Program.cs".

**Properties:** Shows the properties for the selected "MainForm" control, such as "Padding", "Size", and "Text".

**Error List:** Shows "0 Errors", "0 Warnings", and "0 Messages".

**Menu Bar:** Includes File, Edit, View, Project, Build, Debug, Format, Test, Analyze, Tools, Extensions, Window, Help, and Search.

**Toolbar:** Contains icons for various development actions like Run, Stop, and Undo.

**Status Bar:** Shows "Ready" and "Add to Source Control".

# Book Class deals with Database

The screenshot displays the Visual Studio IDE with the following components:

- Menu Bar:** File, Edit, View, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, Search, BookSystem.
- Toolbox:** Search Toolbox, General. A message states: "There are no usable controls in this group. Drag an item onto this text to add it to the toolbox."
- Code Editor:** Shows the `Book.cs` file in the `BookSystem.Classes` namespace. The code includes references to `System.Configuration`, `System.Data`, and `Microsoft.Data.SqlClient`. It defines a `Book` class with properties `BookId`, `Title`, `Author`, and `Topic`. The `GetBooks()` method uses `SqlConnection`, `SqlCommand`, and `SqlDataReader` to query a database table named `BOOK`.
- Solution Explorer:** Shows the project structure for `BookSystem`, including `Dependencies`, `Classes`, `Resources`, and `Program.cs`.
- Properties Window:** Empty.
- Status Bar:** Shows "100%", "No issues found", and "Ln: 18 Ch: 1 SPC CRLF".

```
1 using System.Configuration;
2 using System.Data;
3 using Microsoft.Data.SqlClient;
4
5 namespace BookSystem.Classes
6 {
7     22 references
8     public class Book
9     {
10         4 references
11         public int BookId { get; set; }
12         7 references
13         public string? Title { get; set; }
14         7 references
15         public string? Author { get; set; }
16         7 references
17         public string? Topic { get; set; }
18
19         readonly string connectionString = ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
20
21         1 reference
22         public List<Book> GetBooks()
23         {
24             List<Book> bookList = new List<Book>();
25
26             SqlConnection con = new SqlConnection(connectionString);
27
28             string selectSQL = "select BookId, Title, Author, Topic from BOOK";
29
30             con.Open();
31
32             SqlCommand cmd = new SqlCommand(selectSQL, con);
33
34             SqlDataReader dr = cmd.ExecuteReader();
35
36             if (dr != null)
37             {
38                 while (dr.Read())
39                 {
40                     Book book = new Book();
41
42                     book.BookId = Convert.ToInt32(dr["BookId"]);
```

# Book Class

```
Book.cs
BookSystem
BookSystem.Classes.Book
Topic

1 using System.Configuration;
2 using System.Data;
3 using Microsoft.Data.SqlClient;
4
5 namespace BookSystem.Classes
6 {
7     22 references
8     public class Book
9     {
10         4 references
11         public int BookId { get; set; }
12         7 references
13         public string? Title { get; set; }
14         7 references
15         public string? Author { get; set; }
16         7 references
17         public string? Topic { get; set; }
18
19         readonly string connectionString = ConfigurationManager.ConnectionStrings["ConnectionString"].ToString();
20
21         1 reference
22         public List<Book> GetBooks()...
23
24         1 reference
25         public Book GetBookData(int bookId)...
26
27         1 reference
28         public void CreateBook(Book book)...
29
30         1 reference
31         public void EditBook(Book book)...
32
33         1 reference
34         public void DeleteBook(int bookId)...
35
36     }
37 }
```

## Methods:

- GetBooks
- GetBookData
- CreateBook
- EditBook
- DeleteBook

<https://www.halvorsen.blog>

# Windows Forms App

using simple PHP Web API



Hans-Petter Halvorsen

[Table of Contents](#)

# Updated WinForms App

- Now we will update the WinForms App using the PHP Web API instead.
- We have already tested the API in the Web Browser and in Python.
- Now we need to implement API communication in Visual Studio/C#.
- We use the built-in **HttpClient** Class in C#.
- The only thing we need to do in our WinForm App is to update the Book Class that's communicates with the API instead of direct Database access.



# HttpClient

```
HttpClient client = new HttpClient();

string urlApi = "https://server/api/book/1.0/";
client.BaseAddress = new Uri(urlApi);

string apiMethod = "GetBooks.php";
HttpResponseMessage response = await client.GetAsync(apiMethod);

string contentJson = await response.Content.ReadAsStringAsync();
```

```
[ { "BookId": "1", "Title": "Arduino", "Author": "Hans-Petter", "Topic":
"Programming" }, { "BookId": "2", "Title": "Raspberry Pi", "Author": "John Wayne",
"Topic": "IoT" }, { "BookId": "22", "Title": "Arduino", "Author": "Hans-Petter",
"Topic": "Microcontrollers" }, { "BookId": "24", "Title": "SQL", "Author": "Hans-
Petter", "Topic": "Programming" } ]
```

# Visual Studio

The screenshot displays the Visual Studio IDE with the following components:

- Menu Bar:** File, Edit, View, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, Search.
- Toolbar:** Includes icons for file operations, debugging, and search.
- Toolbox:** Located on the left, showing a message: "There are no usable controls in this group. Drag an item onto this text to add it to the toolbox."
- Code Editor:** Displays the content of `Book.cs`. The code includes:

```
1 using Newtonsoft.Json;
2 using System.Text.Json;
3
4 namespace BookSystem.Classes
5 {
6     public class Book
7     {
8         public int BookId { get; set; }
9         public string? Title { get; set; }
10        public string? Author { get; set; }
11        public string? Topic { get; set; }
12
13        readonly string url = "https://1.../api/book/1.0/";
14        readonly string key = "...";
15
16        public async Task<List<Book>> GetBooks()
17        {
18            List<Book> bookList = new List<Book>();
19
20            HttpClient client = new HttpClient();
21            client.BaseAddress = new Uri(url);
22
23            string method = "GetBooks";
24            string query = method + ".php" + "?key=" + key;
25            HttpResponseMessage response = await client.GetAsync(query);
26
27            string contentJson = await response.Content.ReadAsStringAsync();
28
29            bookList = (List<Book>)JsonConvert.DeserializeObject<IEnumerable<Book>>(contentJson);
30
31            return bookList;
32        }
33
34        public async Task<Book> GetBookData(int bookId)
35        {
36            HttpClient client = new HttpClient();
37            client.BaseAddress = new Uri(url);
38
39            string method = "GetBookById";
40            string query = method + ".php" + "?key=" + key + "&id=" + bookId;
41            HttpResponseMessage response = await client.GetAsync(query);
42
43            string contentJson = await response.Content.ReadAsStringAsync();
```
- Solution Explorer:** Shows the project structure for "BookSystem using Web API", including folders like Dependencies, Classes, Resources, and files like App.config, EditBookForm.cs, MainForm.cs, NewBookForm.cs, and Program.cs.
- Properties Window:** Located at the bottom right, currently empty.
- Status Bar:** Shows "Ln: 14 Ch: 43 SPC CRLF".

# Book Class

```
Book.cs
BookSystem
BookSystem.Classes.Book
url

6 public class Book
7 {
8     2 references
    public int BookId { get; set; }
9     5 references
    public string? Title { get; set; }
10    5 references
    public string? Author { get; set; }
11    5 references
    public string? Topic { get; set; }
12
13    readonly string url = "https://www.googleapis.com/books/v1/api/book/1.0/";
14    readonly string key = "XXXXXXXXXXXX";
15
16    1 reference
    public async Task<List<Book>> GetBooks()...
33
17
18    1 reference
34    public async Task<Book> GetBookData(int bookId)...
52
19
20    1 reference
53    public async void CreateBook(Book book)...
65
21
22    1 reference
66    public async void EditBook(Book book)...
78
23
24    1 reference
79    public async void DeleteBook(int bookId)...
91
92 }
93 }
```

## Methods:

- GetBooks
- GetBookData
- CreateBook
- EditBook
- DeleteBook

# GetBooks Method

```
public async Task<List<Book>> GetBooks ()
{
    List<Book> bookList = new List<Book>();

    HttpClient client = new HttpClient();
    client.BaseAddress = new Uri(url);

    string method = "GetBooks";
    string query = method + ".php" + "?key=" + key;
    HttpResponseMessage response = await client.GetAsync(query);

    string contentJson = await response.Content.ReadAsStringAsync();

    bookList =
(List<Book>) JsonConvert.DeserializeObject<IEnumerable<Book>>(contentJson);

    return bookList;
}
```

# GetBookData Method

```
public async Task<Book> GetBookData(int bookId)
{
    HttpClient client = new HttpClient();
    client.BaseAddress = new Uri(url);

    string method = "GetBookById";
    string query = method + ".php" + "?key=" + key + "&id=" + bookId;
    HttpResponseMessage response = await client.GetAsync(query);

    string contentJson = await response.Content.ReadAsStringAsync();
    contentJson = contentJson.Replace("[", "");
    contentJson = contentJson.Replace("]", "");

    Book? book = new Book();
    book = Newtonsoft.Json.JsonConvert.DeserializeObject<Book>(contentJson);

    return book;
}
```

# CreateBook Method

```
public async void CreateBook(Book book)
{
    HttpClient client = new HttpClient();
    client.BaseAddress = new Uri(url);

    string method = "InsertBook";
    string query = method + ".php" + "?key=" + key + "&title=" +
        book.Title + "&author=" + book.Author + "&topic=" +
        book.Topic;

    HttpResponseMessage response = await client.GetAsync(query);

    string result = await response.Content.ReadAsStringAsync();
}
```

# EditBook Method

```
public async void EditBook(Book book)
{
    HttpClient client = new HttpClient();
    client.BaseAddress = new Uri(url);

    string method = "UpdateBook";
    string query = method + ".php" + "?key=" + key + "&id=" +
        book.BookId + "&title=" + book.Title + "&author=" +
        book.Author + "&topic=" + book.Topic;

    HttpResponseMessage response = await client.GetAsync(query);

    string result = await response.Content.ReadAsStringAsync();
}
```

# DeleteBook Method

```
public async void DeleteBook(int bookId)
{
    HttpClient client = new HttpClient();
    client.BaseAddress = new Uri(url);

    string method = "DeleteBook";
    string query = method + ".php" + "?key=" + key + "&id=" +
        bookId;

    HttpResponseMessage response = await client.GetAsync(query);

    string result = await response.Content.ReadAsStringAsync();
}
```



# Summary

- In previous Tutorials we have made
  - A basic **CRUD WinForm Desktop App** that communicates directly with a SQL Database
    - This is “bad practice” and very often not allowed
  - So, In another Tutorial we made a simple **PHP CRUD Web API**
- In this Tutorial we updated the WinForm App using the Web API instead
- The code is very basic and don't follow best practice, can be better structured, include error handling, etc.
- The code is made simple to illustrate the basic principles using Web APIs

# Hans-Petter Halvorsen

University of South-Eastern Norway

[www.usn.no](http://www.usn.no)

E-mail: [hans.p.halvorsen@usn.no](mailto:hans.p.halvorsen@usn.no)

Web: <https://www.halvorsen.blog>

