https://www.halvorsen.blog

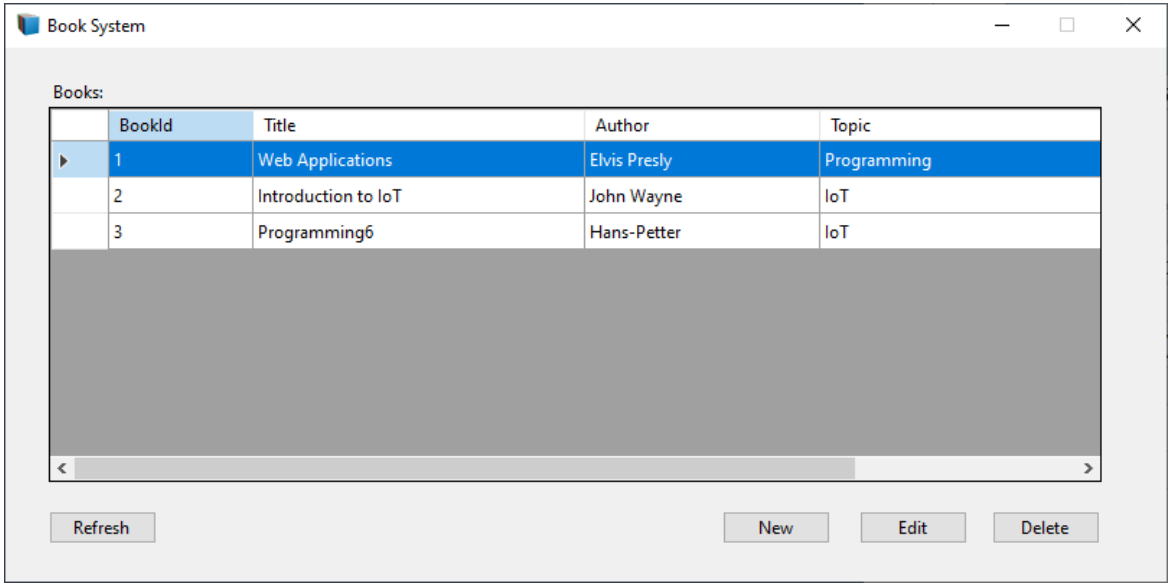
Consuming PHP REST API in WinForms App

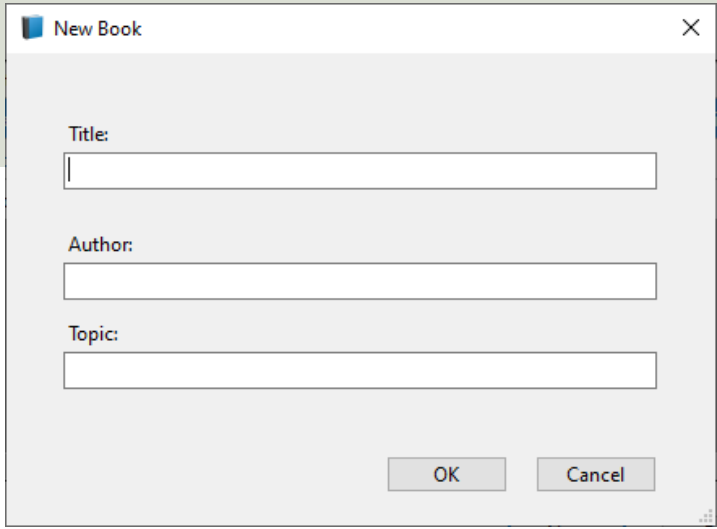
Hans-Petter Halvorsen

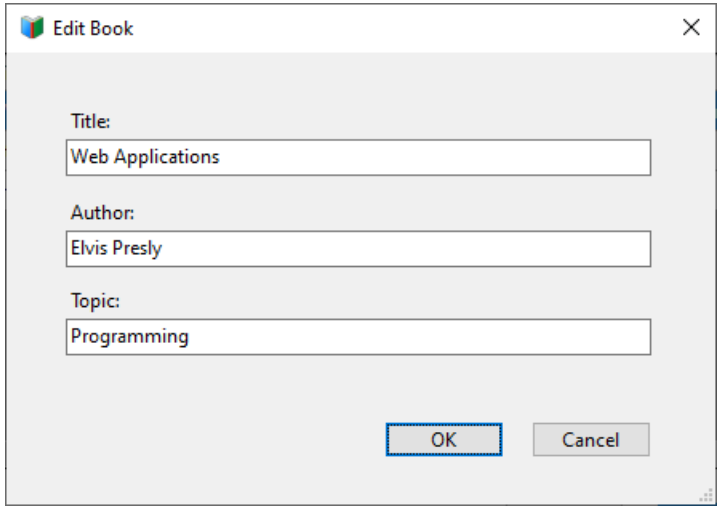
Contents

- The background is that in a previous Tutorial a simple Windows Forms Desktop Application was made that was directly communicating with a SQL Database.
- 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?
- In a previous Tutorial we made a REST API using PHP.
  - The API has CRUD functionality.
  - CRUD means Create, Read, Update and Delete data in the Database.
  - The REST API implements the GET, POST, PUT and DELETE methods in HTTP to handle the CRUD operations.
- In this Tutorial we will use, or consume, that REST API in a Windows Forms Desktop Application.

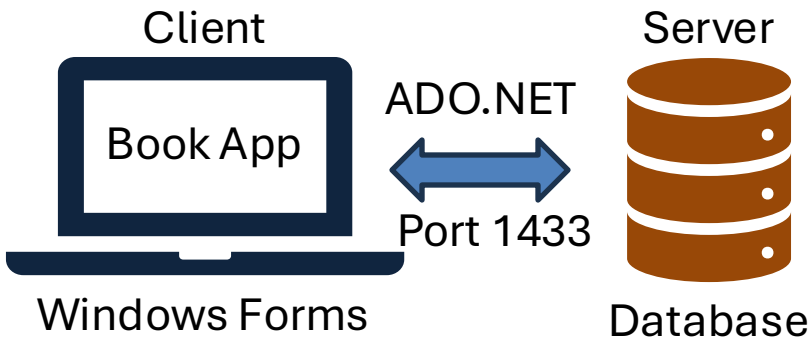
Book App

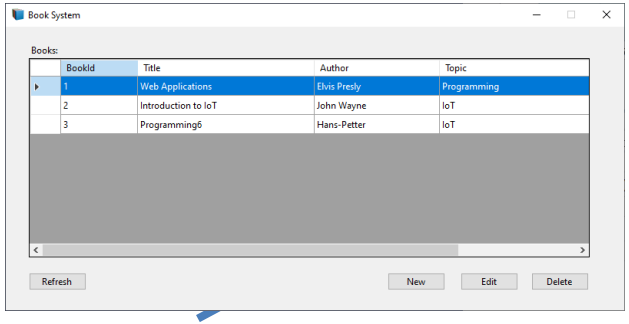

Book System
Books:
BookId
Title
Author
Topic
1
Web Applications
Elvis Presly
Programming
2
Introduction to IoT
John Wayne
IoT
3
Programming6
Hans-Petter
IoT
Refresh
New
Edit
Delete


New Book
Title:
Author:
Topic:
OK
Cancel

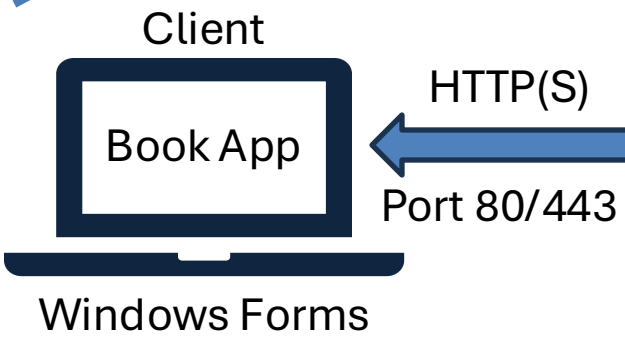

Edit Book
Title:
Web Applications
Author:
Elvis Presly
Topic:
Programming
OK
Cancel

Old Solution vs New Solution





The GUI will remain the same



Previous Tutorials

- Windows Forms CRUD App:
  <Link to YouTube>
- Simple PHP REST API:
  <Link to YouTube>

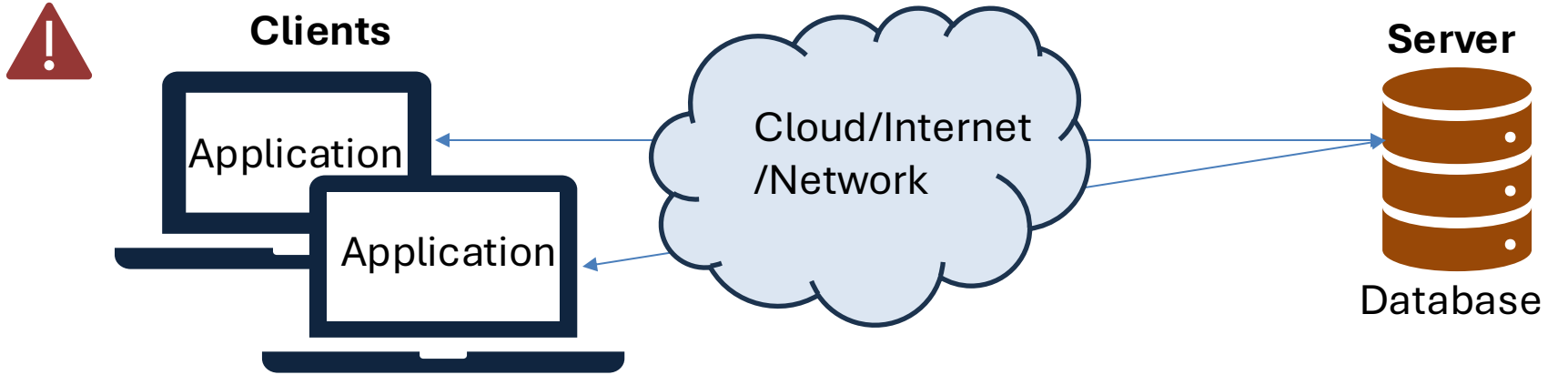
Introduction

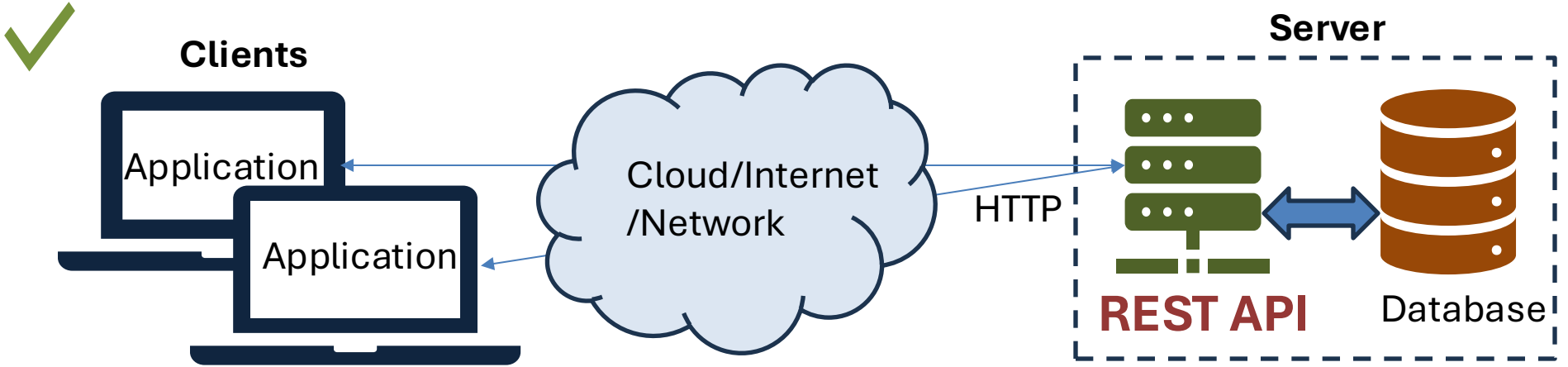
Hans-Petter Halvorsen

Table of Contents

Why use REST 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

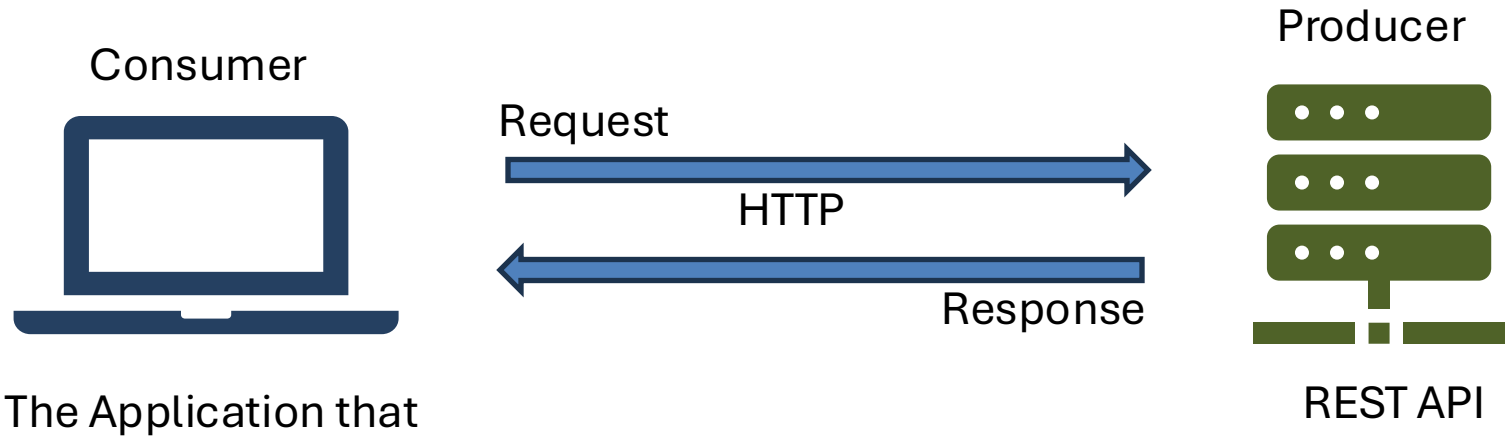
- Make HTTP requests with the HttpClient class: https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/http/httpclient

REST API

Hans-Petter Halvorsen

Table of Contents

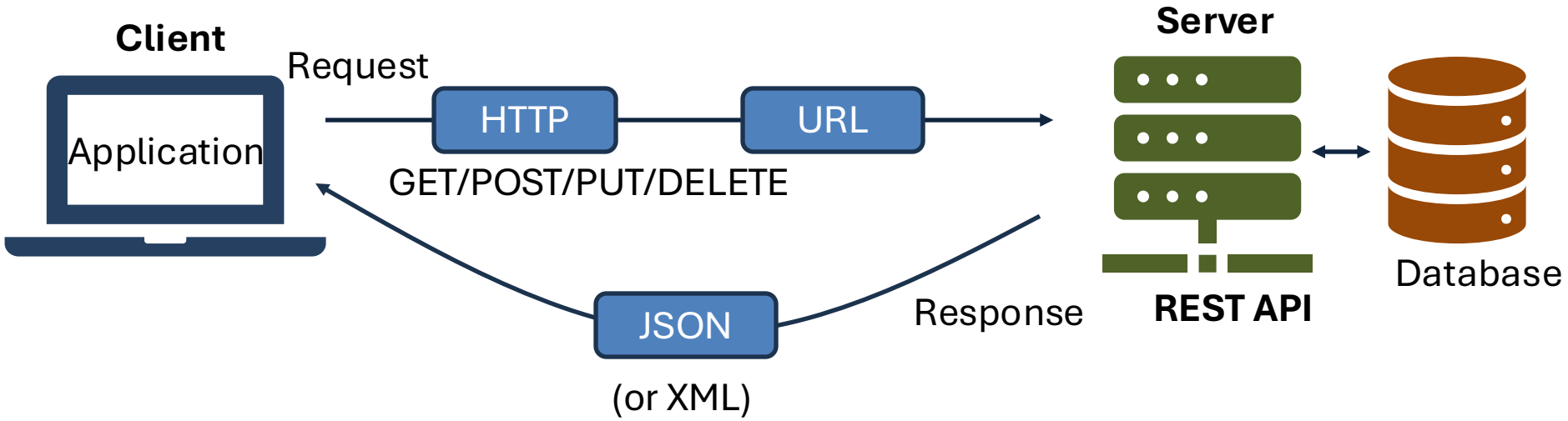
REST API


Consumer
Request
HTTP
Response
Producer
The Application that uses the REST API
REST API

REST API and HTTP

- REST APIs are based on/using the HTTP protocol.
- HTTP consists of different methods:
  - **GET** – This method is used to retrieve information from the server.
  - **POST** – This is used to send data to the server. Typically used to store data from a web page (an HTTML Form) to ,e.g., a database.
  - **PUT** – This is used to update information on the server.
  - **DELETE** – This is used to delete information on the server.
- You usually refer to these four methods as **CRUD** operations because they allow you to Create (POST), Read (GET), Update (PUT), and Delete (DELETE) resources, such as information in a database.

REST API


Client
Application
Request
HTTP
URL
GET/POST/PUT/DELETE
Server
REST API
Database
JSON
(or XML)
Response

API Summary

- Basically, Web APIs, REST APIs or HTTP APIs are basically the same.
- It is just different names for the same.
- They all communicate via Internet and use HTTP as communication protocol.
- And they use JSON (or sometimes XML) as Data Format.

REST API Example

Hans-Petter Halvorsen

Table of Contents

Database

We start by creating a simple Database Table, e.g.:

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

API Code

```php
<?php
require_once 'config.php’;

// Set the content type to JSON
header('Content-Type: application/json');
// Handle HTTP methods
$method = $_SERVER['REQUEST_METHOD'];

switch ($method) {
   case 'GET':
      ...
   break;

   case 'POST':
      ...
   break;

   case 'PUT':
      ...
   break;

   case 'DELETE':
      ...
   break;

   default:
      http_response_code(405);
      echo json_encode(['error' => 'Method not allowed']);
   break;
}
?>
```

API GET Code

```
case 'GET':
$json = file_get_contents('php://input');
$data = json_decode($json,true);

$id = $_GET['id'];
if ($id > 0)
{
   $stmt = $pdo->prepare('SELECT * FROM BOOK WHERE BookId=?');
   $stmt->execute([$id]);
}
else
{
   $stmt = $pdo->query('SELECT * FROM BOOK');
}
$result = $stmt->fetchAll(PDO::FETCH_ASSOC);

echo json_encode($result);
break;
```

Get All Books or get a specific Book specified by its BookId

API POST Code

Create New Book

```
case 'POST':
$json = file_get_contents('php://input');
$data = json_decode($json,true);
$title = $data['title'];
$author = $data['author'];
$topic = $data['topic'];

$stmt = $pdo->prepare('INSERT INTO BOOK (Title, Author, Topic)
      VALUES (?, ?, ?)');
$stmt->execute([$title, $author, $topic]);

echo json_encode(['message' => 'New Book added successfully']);
break;
```

API PUT Code

```
case 'PUT':
$json = file_get_contents('php://input');
$data = json_decode($json,true);
$id = $data['id'];
$title = $data['title'];
$author = $data['author'];
$topic = $data['topic'];

$stmt = $pdo->prepare('UPDATE BOOK SET Title=?, Author=?, Topic=?
      WHERE BookId=?');
$stmt->execute([$title, $author, $topic, $id]);

echo json_encode(['message' => 'Book updated successfully']);
break
```

Update a specific Book specified by its BookId

API DELETE Code

```
case 'DELETE':
$json = file_get_contents('php://input');
$data = json_decode($json,true);
$id = $data['id’];
if ($id == "")
   $id = $_GET['id'];

$stmt = $pdo->prepare('DELETE FROM BOOK WHERE BookId=?');
$stmt->execute([$id]);

echo json_encode(['message' =>'Book deleted successfully']);
break
```

Delete a specific Book specified by its BookId

Windows Form App Example

Hans-Petter Halvorsen

Table of Contents

Windows Forms App

- The Windows Forms App will be updated
- The GUI will remain the same
- MainForm.cs, NewBookForm.cs and EditBookForm.cs will remain the same
- Only the Book Class (Book.cs) will be updated
  - We will replace the ADO.NET code with new Http API code

Windows Form App Example

- We need to be able to communicate with the PHP REST API hosted on the server.
- We use the built-in **HttpClient** Class in C#.

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

string requestUrl = "..";

HttpResponseMessage response = await client.GetAsync(requestUrl);
string contentJson = await response.Content.ReadAsStringAsync();
```

HttpClient Class in C#

Here you see a basic example using the HttpClient Class in C#:

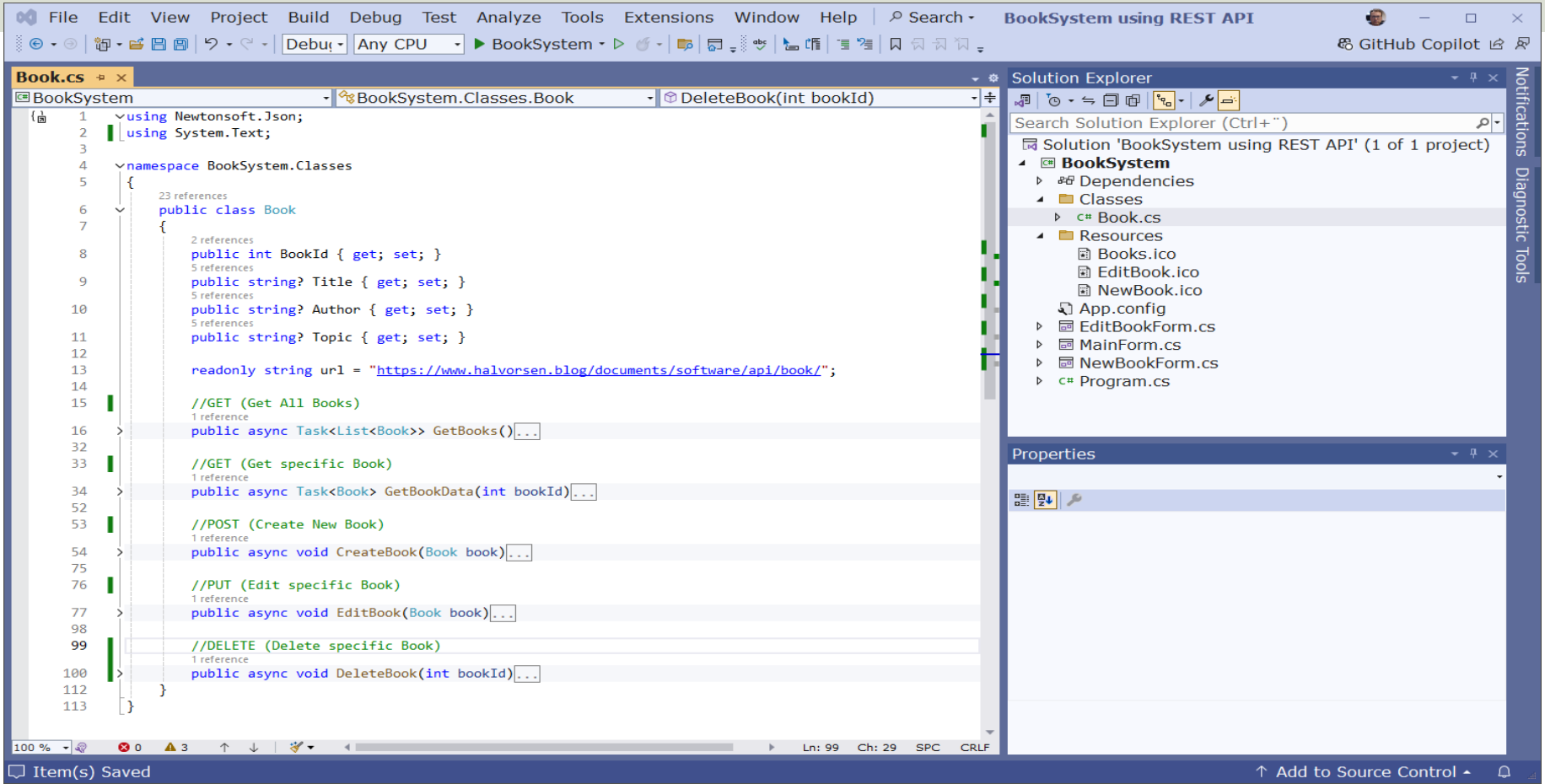
```
HttpClient client = new HttpClient();
client.BaseAddress = new Uri(url);

string requestUrl = “..";

HttpResponseMessage response = await client.GetAsync(requestUrl);
string contentJson = await response.Content.ReadAsStringAsync();
```

You can use the methods GetAsync(..), PostAsync(..), PutAsync(..) and DeleteAsync(..) for the HTTP methods GET, POST, PUT and DELETE

Visual Studio



GET

This method is used to retrieve information from the server/database

C# GET Code (All Books)

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

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

    string requestUrl = "";

    HttpResponseMessage response = await client.GetAsync(requestUrl);
    string contentJson = await response.Content.ReadAsStringAsync();

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

    return bookList;
}
```

C# GET Code (Specific Book)

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

    string requestUrl = "?id=" + bookId;

    HttpResponseMessage response = await client.GetAsync(requestUrl);
    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;
}
```

POST

This method is used to send data to the server/database

Hans-Petter Halvorsen

C# POST Code (New Book)

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

    string requestUrl = "";

    using StringContent contentJson = new(
    System.Text.Json.JsonSerializer.Serialize(new
    {
        title = book.Title,
        author = book.Author,
        topic = book.Topic
    }),
    Encoding.UTF8,
    "application/json");

    HttpResponseMessage response = await client.PostAsync(requestUrl, contentJson);
    string result = await response.Content.ReadAsStringAsync();
}
```

PUT

This method is used to update information on the server/database

Hans-Petter Halvorsen

C# PUT Code (Update Book)

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

    string requestUrl = "index.php";

    using StringContent contentJson = new(
    System.Text.Json.JsonSerializer.Serialize(new
    {
        id = book.BookId,
        title = book.Title,
        author = book.Author,
        topic = book.Topic
    }),
    Encoding.UTF8,
    "application/json");

    HttpResponseMessage response = await client.PutAsync(requestUrl, contentJson);
    string result = await response.Content.ReadAsStringAsync();
}
```

DELETE

This method is used to delete information on the server/database

C# DELETE Code

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

    string requestUrl = "?id=" + bookId;

    HttpResponseMessage response = await client.DeleteAsync(requestUrl);
    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 REST API
- In this Tutorial we updated the WinForm App so it used the Web API instead of direct Daatabse Communication using ADO.NET
- The code is very basic and don’t follow best practice, can be better structured, include error handling, authentication, 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

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog