Working with HTTP Communication

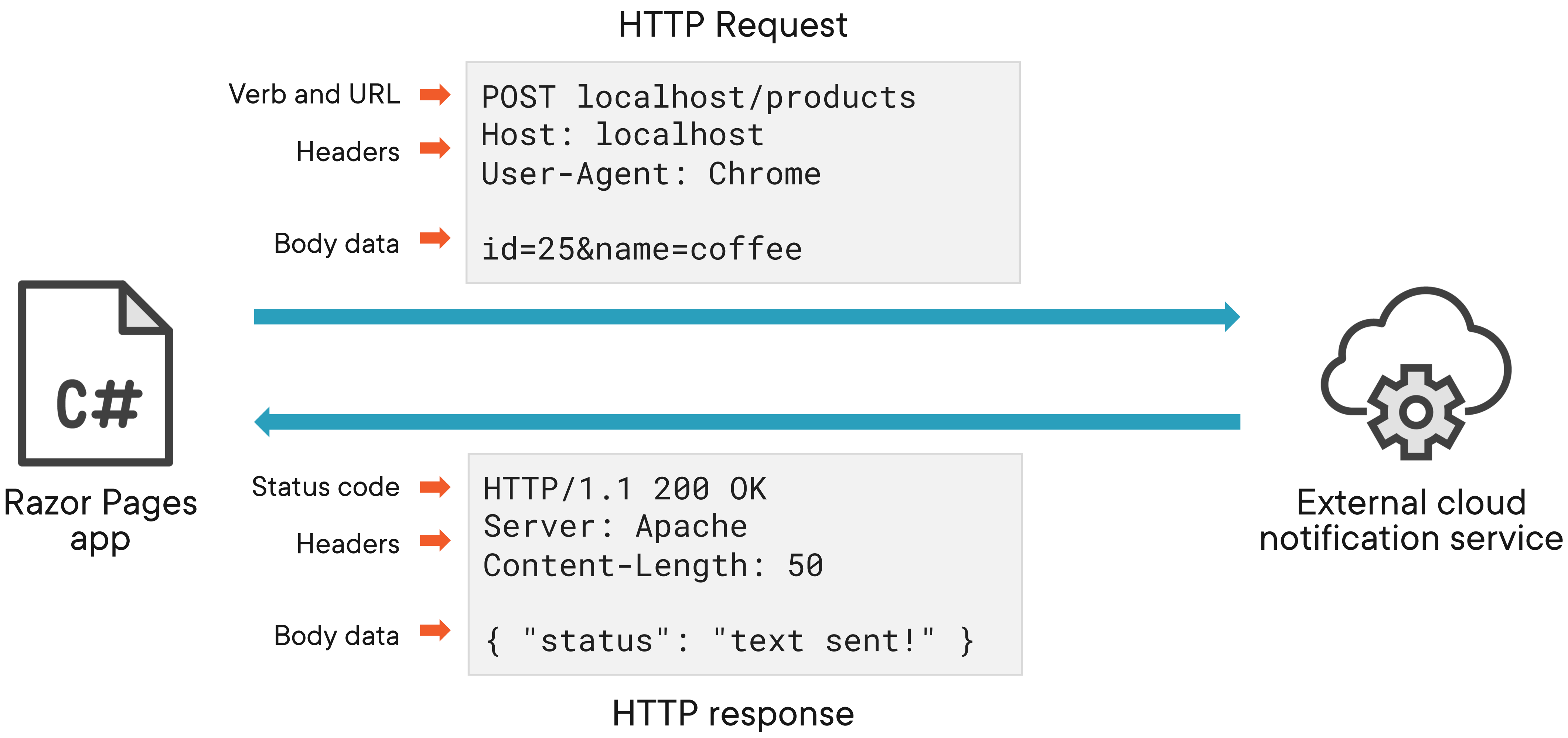
Alex Wolf

.NET Developer

www.thecodewolf.com

This module assumes basic working knowledge of HTTP requests.

HTTP Request and Response Basics



HTTP Request

Verb and URL

Headers

Body data

```
POST localhost/products
Host: localhost
User-Agent: Chrome

id=25&name=coffee
```

Razor Pages app

External cloud notification service

Status code

Headers

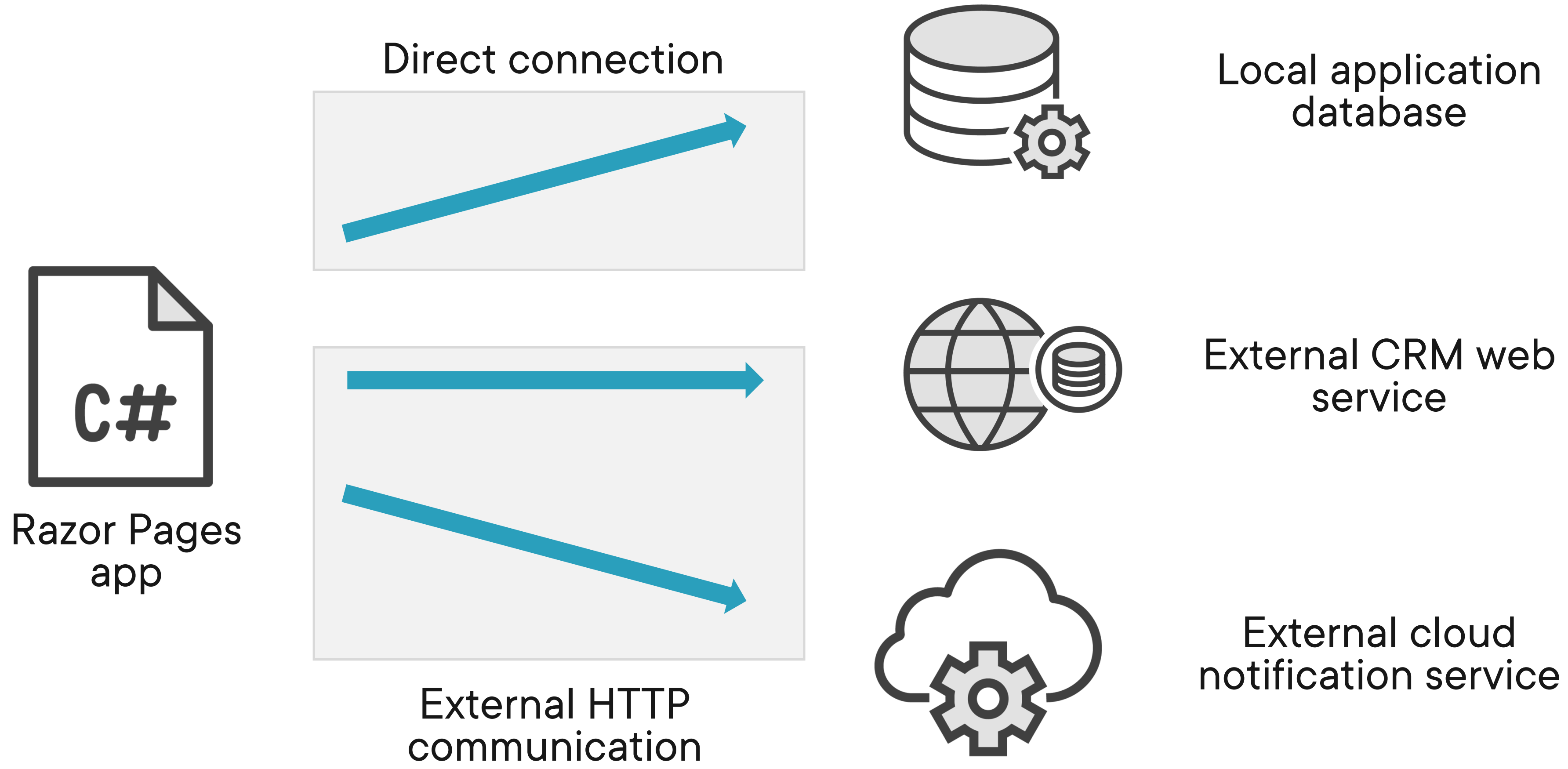
Body data

```
HTTP/1.1 200 OK
Server: Apache
Content-Length: 50

{ "status": "text sent!" }
```

HTTP response

Integrating with External Services



Direct connection
C#
Razor Pages app
External HTTP communication
Local application database
External CRM web service
External cloud notification service

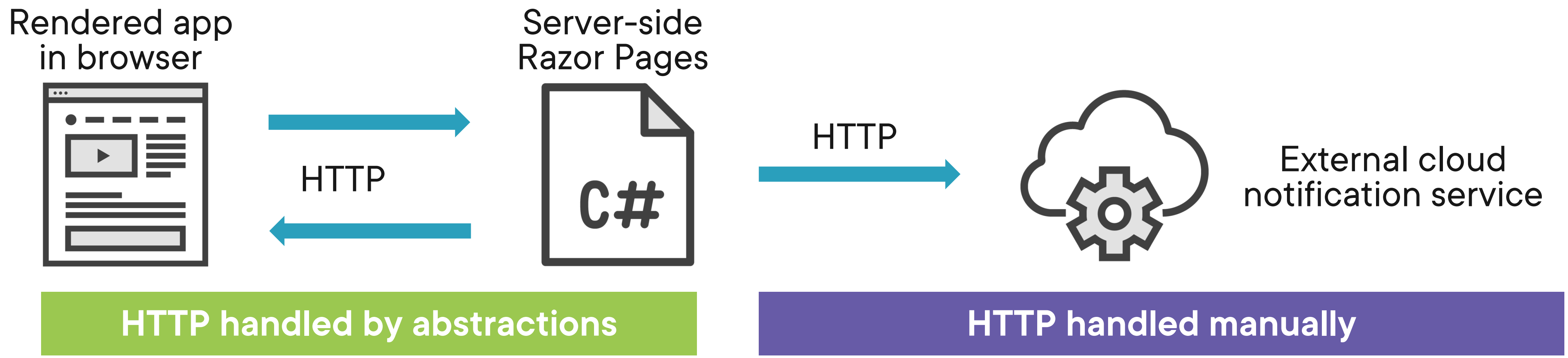
Web Service URLs

GET: https://wiredbraincoffeeadmin.azurewebsites.net/api/orderStats

GET: https://wiredbraincoffeeadmin.azurewebsites.net/api/tickets

POST: https://wiredbraincoffeeadmin.azurewebsites.net/api/ticket

Managing HTTP Communication



Rendered app in browser

HTTP

Server-side Razor Pages

HTTP

External cloud notification service

HTTP handled by abstractions

HTTP handled manually

HTTP Communication with Razor Pages

HttpClient

The main class provided by ASP.NET Core to manage HTTP requests and responses

A note about HttpClient.

HttpClientFactory

A class that manages HttpClient instances and addresses common challenges with them

```
// Registers HttpClientFactory and related services

builder.Services.AddHttpClient();
```

Registering the HttpClientFactory

The HttpClientFactory and related services are registered in program.cs using a helper method

This lets us inject the factory class into our pages and services to create HttpClient objects

```
private IHttpClientFactory factory;


public HelpPage(IHttpClientFactory factory)
{
   this.factory = factory;
}


public IActionResult OnGet()
{
   var client = factory.CreateClient();

   var response = client.GetAsync("{url}");
}
```

◀ Inject the registered HttpClientFactory

◀ Assign to field for use throughout class

◀ Create an instance of HttpClient

◀ Use the client to make a request to a given url

Demo

Creating a simple HTTP request

Understanding Typed Clients

HTTP Client Types

Basic clients

Named clients

✓ Typed clients

Generated clients

```
private HttpClient client;


public OrderService(HttpClient client)
{
   this.client = client;
}


public List<HelpTicket> GetAllOrders()
{
     // Method logic
}

public List<HelpTicket> GetOrderById()
{
     // Method logic
}

public List<HelpTicket> AddOrder()
{
     // Method logic
}
```

◀ Inject the HttpClient

◀ Assign to field for use throughout class

◀ Common operations over HTTP

```
// Register the typed client

builder.Services.AddHttpClient<ITicketService, TicketService(options =>
{
    options.BaseAddress = new Uri("https://wiredbraincoffeeadmin.azurewebsites.net")
});
```

Registering the Typed Client

Typed clients can be registered and configured in program.cs

The DI container and HttpClientFactory will manage injecting instances correctly

Demo

Exploring the need for a typed client

Demo

Creating a typed HTTP client

Demo

Retrieving data using the typed client

Demo

Sending data using the typed client

Summary

Razor Pages utilize the standard .NET Core HttpClient class to manage HTTP communication

The HttpClientFactory should be used to ensure client objects are created and disposed of properly

.NET Core provides different types of HTTP services to handle various scenarios, such as typed clients

Typed HTTP clients create strongly typed services that wrap interactions with an external service

Typed clients are registered and configured in Program.cs using a special helper method

HttpClient offers a variety of helper methods to streamline working with requests and JSON data