**Chapter 6: The Threading Patterns**

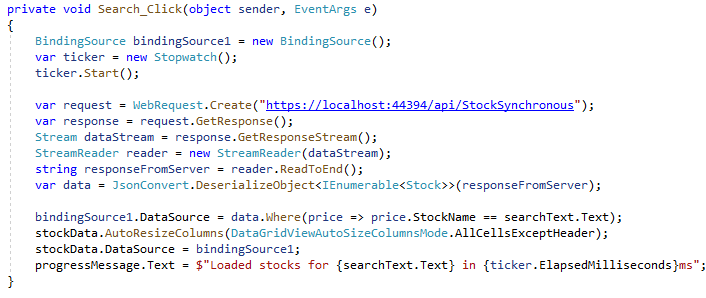
**Task-based Asynchronous Pattern (TAP)**

**Overview**

Task based Asynchronous Pattern (TAP) is recommended pattern to implement async programming in .NET. Task objects are one of the central components of TAP. This pattern is based on System.Threading.Tasks namespace using Task, Task<T> constructs. In this pattern we create a single method that represent beginning and ending of asynchronous operation.

**Implementing pattern**

To implement this pattern we will start with prefixing method with async keyword and add await keyword to the method that can be performed asynchronously typically a method retrieving data from database, reading file from disk or an API call. This is illustrated in below example



private async void Search\_Click(object sender, EventArgs e)

{

BindingSource bindingSource1 = new BindingSource();

var ticker = new Stopwatch();

ticker.Start();

var request = WebRequest.Create("https://localhost:44394/api/StockSynchronous");

var response = request.GetResponse();

Stream dataStream = response.GetResponseStream();

StreamReader reader = new StreamReader(dataStream);

string responseFromServer = reader.ReadToEnd();

var data = JsonConvert.DeserializeObject<IEnumerable<Stock>>(responseFromServer);

bindingSource1.DataSource = data.Where(price => price.StockName == searchText.Text);

stockData.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCellsExceptHeader);

stockData.DataSource = bindingSource1;

progressMessage.Text = $"Loaded stocks for {searchText.Text} in {ticker.ElapsedMilliseconds}ms";

}

**Figure 6.1 – A button click event on a win form loading data from API synchronously**



private async void Search\_Click(object sender, EventArgs e)

{

BindingSource bindingSource1 = new BindingSource();

var ticker = new Stopwatch();

ticker.Start();

#region Async Calls

using (HttpClient client = new HttpClient())

{

var response = await client.GetAsync($"https://localhost:44394/api/StockS");

var content = await response.Content.ReadAsStringAsync();

var data = JsonConvert.DeserializeObject<IEnumerable<Stock>>(content);

bindingSource1.DataSource = data.Where(price => price.StockName == searchText.Text);

}

stockData.DataSource = bindingSource1;

#endregion

progressMessage.Text = $"Loaded stocks for {searchText.Text} in {ticker.ElapsedMilliseconds}ms";

}

**Figure 6.2 – A button click event on a win form loading data from API asynchronously**

**CPU Bound vs I/O Bound**