Cel testu
Test `SignIn_Post_ValidUserName_SetsSessionAndRedirectsToIndex` sprawdza, czy:
1. **Poprawna nazwa użytkownika** (`UserName`) przesłana w modelu `SignInVm`:
   - Zostaje zapisana w sesji pod kluczem `"USER_KEY"`.
   - Jest poprawnie zakodowana w formacie `UTF-8`.
2. **Kontroler**:
   - Po zapisaniu nazwy użytkownika w sesji przekierowuje użytkownika na akcję `Index`.

Podział testu na części

1. Arrange (Przygotowanie środowiska testowego)

To etap konfiguracji testu. Tworzymy kontroler i symulujemy zachowanie jego zależności.

- **Mockowanie `HttpContext` i `ISession`:**
  - Sesja HTTP (`ISession`) jest mockowana, ponieważ nie można jej bezpośrednio użyć w testach jednostkowych.
  - Stworzono słownik (`sessionStorage`), który symuluje faktyczną pamięć sesji.
  - Metody `Set` i `TryGetValue` sesji są skonfigurowane, aby zapisywać i odczytywać dane w symulowanej pamięci (`sessionStorage`).

    ```csharp
    var sessionStorage = new Dictionary<string, byte[]>();
    mockSession.Setup(s => s.Set(It.IsAny<string>(), It.IsAny<byte[]>()))
        .Callback<string, byte[]>((key, value) => sessionStorage[key] = value);
    ```

- **Przygotowanie kontrolera:**
  - Mockowany `HttpContext` i sesja są przypisane do kontrolera za pomocą jego `ControllerContext`.

    ```csharp
    var controller = new HomeController(Mock.Of<ILogger<HomeController>>())
    {
        ControllerContext = new ControllerContext
        {
            HttpContext = mockHttpContext.Object
        }
    };
    ```

- **Przygotowanie danych wejściowych:**
  - Tworzymy model `SignInVm` z nazwą użytkownika `TestUser`, który symuluje dane przesłane w żądaniu `POST`.

    ```csharp
    var signInVm = new SignInVm { UserName = "TestUser" };
    ```

---

2. Act (Wykonanie testowanej akcji)

Na tym etapie wywołujemy metodę `SignIn` kontrolera, aby sprawdzić jej zachowanie.

- `controller.SignIn(signInVm)`:
  - Wywołuje metodę `SignIn` z poprawnym modelem.
  - Oczekujemy, że zapisze nazwę użytkownika w sesji i przekieruje na akcję `Index`.

---

3. Assert (Sprawdzanie wyników)

Na tym etapie sprawdzamy, czy metoda zachowuje się zgodnie z oczekiwaniami:

a) Sprawdzenie zapisania danych w sesji
- **Czy metoda `Set` została wywołana?**
  - Weryfikujemy, że `ISession.Set` została wywołana z kluczem `"USER_KEY"`.
    ```csharp
    mockSession.Verify(s => s.Set("USER_KEY", It.IsAny<byte[]>()), Times.Once);
    ```

- **Czy dane zostały poprawnie zapisane?**
  - Sprawdzamy, czy `sessionStorage` zawiera klucz `"USER_KEY"`, oraz czy przechowywana wartość to `"TestUser"`.
    ```csharp
    Assert.True(sessionStorage.ContainsKey("USER_KEY"));
    Assert.Equal("TestUser",
System.Text.Encoding.UTF8.GetString(sessionStorage["USER_KEY"]));
    ```

b) Sprawdzenie przekierowania
- **Czy nastąpiło przekierowanie?**
  - Sprawdzamy, czy metoda zwróciła wynik typu `RedirectToActionResult`.
  - Sprawdzamy, czy przekierowano na akcję `Index`.
    ```csharp
    var redirectResult = Assert.IsType<RedirectToActionResult>(result);
    Assert.Equal("Index", redirectResult.ActionName);
    ```

Jak działa sesja w tym teście?

1. Sesja jest symulowana przez `sessionStorage` (słownik `Dictionary<string, byte[]>`), ponieważ faktyczna sesja HTTP nie działa w testach jednostkowych.
2. `ISession.Set`:
   - Zapisuje klucz (`USER_KEY`) i wartość (`TestUser` zakodowany jako `byte[]`) do `sessionStorage`.
3. `ISession.TryGetValue`:
   - Odczytuje dane z `sessionStorage` na podstawie klucza.

Za co odpowiadają poszczególne elementy?

Element	**Rola**
Mock<ISession>	Symuluje sesję HTTP, umożliwia testowanie metod `Set` i `TryGetValue`.
sessionStorage	Symuluje rzeczywistą pamięć sesji, działa jako substytut przechowywania danych w testach.
SignInVm	Model wejściowy symulujący dane przesłane przez użytkownika w żądaniu `POST`.
HomeController	Testowany kontroler, jego metoda `SignIn` jest przedmiotem testu.
Mockowanie HttpContext	Umożliwia wstrzyknięcie mockowanej sesji (`ISession`) do kontrolera.
Assert.IsType<T>	Sprawdza, czy wynik jest oczekiwanego typu (`RedirectToActionResult`).
Assert.Equal / Assert.True	Weryfikuje, czy dane w sesji są zapisane poprawnie i czy wynik metody jest zgodny z oczekiwaniami.

Podsumowanie
Test sprawdza:
1. Czy nazwa użytkownika została poprawnie zapisana w sesji (`sessionStorage` symulującej `ISession`).
2. Czy użytkownik został poprawnie przekierowany na akcję `Index` po pomyślnym zalogowaniu.

Jest to kompletna weryfikacja kluczowego fragmentu logiki kontrolera, który obsługuje logowanie użytkownika. Dzięki mockowaniu sesji można dokładnie przetestować, czy zapis w sesji oraz przekierowanie działają poprawnie.