The example on this page demonstrates how to create a custom shipping carrier provider. The sample carrier provider has the following features:

- Shipping costs are determined based on the country to which the order is shipped. The costs are defined directly in the code, without a user interface for setting the values.

  > If you want to have a user interface for configuring the shipping costs (or other options), you can create a configuration tab as described in Implementing custom shipping carrier providers or create a custom table and use the custom table API to load the shipping cost values from the given table. Do not forget to set caching (you can use the CacheHelper class).

- Shipping options that use the sample shipping carrier are available only for orders whose shipping address is in the USA, Canada or Mexico.

  > If your site's checkout process has the **Customer address** web part (where customer enter the shipping address) and the **Shipping option selection** web part (where customers select the desired shipping option) on the same page, you need to enable the **Propagate changes on postback** property for the **Customer address** web part.
  >
  > This ensures that when the customer changes the shipping country, the shipping options that the **Shipping option selection** web part offers are recalculated based on the selected shipping country.

- Does NOT use a packageable module for distribution to other instances. See Example - Creating a packageable module to learn how to create a module package containing custom functionality (such as the shipping carrier provider).

Start by preparing a separate project for custom classes in your Kentico solution:

1. Open your Kentico solution in Visual Studio.
2. Create a new *Class Library* project in the Kentico solution (or reuse an existing custom project).
3. Add references to the required Kentico libraries (DLLs) for the new project:
   a. Right-click the project and select **Add -> Reference**.
   b. Select the **Browse** tab of the **Reference manager** dialog, click **Browse** and navigate to the ***Lib*** folder of your Kentico web project.
   c. Add references to the following libraries (and any others that you may need in your custom code):
      - **CMS.Base.dll**
      - **CMS.Core.dll**
      - **CMS.DataEngine.dll**
      - **CMS.Ecommerce.dll**
      - **CMS.Globalization.dll**
      - **CMS.Helpers.dll**
4. Reference the custom project from the Kentico web project *(CMSApp* or *CMS)*.
5. Edit the custom project's **AssemblyInfo.cs** file (in the *Properties* folder).
6. Add the **AssemblyDiscoverable** assembly attribute:

```
using CMS;

[assembly:AssemblyDiscoverable]
```

> For custom shipping carrier provider classes, the **AssemblyDiscoverable** attribute is not required (the **Carrier provider** selector in the administration interface can find the class even without the attribute). However, we recommend adding the attribute to your custom Class Library project to ensure that other possible types of customizations work correctly if you (for example registration of custom modules, interface implementations or inherited providers).

Continue by implementing the shipping carrier provider:

1. Add a new class under the custom project, implementing the **ICarrierProvider** interface. For example, name the class *CountryCarrierProvider*.

```
using System;
using System.Collections.Generic;
using System.Linq;

using CMS.Ecommerce;
using CMS.Globalization;

/// <summary>
/// Sample carrier provider that sets shipping costs based on the delivery
country.
/// The carrier allows creation of shipping options that are available for
addresses in the USA, Canada and Mexico.
/// </summary>
public sealed class CountryCarrierProvider : ICarrierProvider
{
    private const string NAME = "Custom carrier provider";
    private List<KeyValuePair<string, string>> services;
    private Dictionary<string, decimal> countryCosts;

    /// <summary>
    /// Gets a dictionary of 3-letter country codes and shipping costs.
    /// The dictionary also defines all countries where delivery is available.
    /// </summary>
    private Dictionary<string, decimal> CountryCosts
    {
        get
        {
            return countryCosts ?? (countryCosts = new Dictionary<string, decimal>
                {
                    { "CAN", 12.99m },
                    { "MEX", 33.33m },
                    { "USA", 5.0m }
                });
        }
    }

    /// <summary>
    /// Gets the name of the carrier provider.
    /// </summary>
    public string CarrierProviderName
    {
        get
        {
            return NAME;
        }
    }

    /// <summary>
    /// Returns a list of service code names and display names.
    /// </summary>
    public List<KeyValuePair<string, string>> GetServices()
    {
        return services ?? (services = new List<KeyValuePair<string, string>>
        {
            new KeyValuePair<string, string>("CostByCountry", "Shipping cost to 
USA, CAN and MEX")
        });
    }

    /// <summary>
    /// Calculates the shipping cost of the given delivery.
    /// </summary>
    /// <param name="delivery">The shipped delivery.</param>
    /// <param name="currencyCode">The code of the currency in which the shipping 
price should be calculated (e.g. USD or EUR).</param>
    /// <returns>The price in the currency specified by currencyCode.</returns>
    public decimal GetPrice(Delivery delivery, string currencyCode)
    {
        if (delivery.DeliveryAddress != null)
        {
            // Gets the 3-letter country code of the delivery's target country
            var country = CountryInfoProvider.GetCountryInfo(delivery.
DeliveryAddress.AddressCountryID);
            string countryCode = country?.CountryThreeLetterCode;

            // Checks whether the shipping option is available for the specified 
country code
            if (CountryCosts.ContainsKey(countryCode))
            {
                // Returns the shipping price converted to the specified currency
                return CurrencyConverter.Convert(CountryCosts[countryCode], 
"USD", currencyCode, delivery.ShippingOption.ShippingOptionSiteID);
            }
        }

        return 0;
    }

    /// <summary>
    /// Checks whether the shipping option is available for the specified 
shipping address.
    /// The only supported shipping addresses are to USA, Canada or Mexico.
    /// </summary>
    public bool CanDeliver(Delivery delivery)
    {
        // Checks whether a shipping option and delivery address is assigned to 
the delivery
        // and evaluates whether the shipping option is available for the 
specified shipping country
        if ((delivery.ShippingOption == null) || (delivery.DeliveryAddress == 
null) || !CanDeliverToCountry(delivery.DeliveryAddress))
        {
            return false;
        }

        // Returns true if the carrier provider supports the shipping service
        return SupportsService(delivery.ShippingOption.
ShippingOptionCarrierServiceName);
    }

    /// <summary>
    /// Returns Guid.Empty. This carrier does not have a configuration interface.
    /// </summary>
    public Guid GetConfigurationUIElementGUID()
    {
        return Guid.Empty;
    }

    /// <summary>
    /// Returns Guid.Empty. This carrier does not have a carrier service 
configuration interface.
    /// </summary>
    public Guid GetServiceConfigurationUIElementGUID(string serviceName)
    {
        return Guid.Empty;
    }

    /// <summary>
    /// Returns true if the carrier provider can handle the specified shipping 
service.
    /// </summary>
    private bool SupportsService(string serviceName)
    {
        return GetServices().Any(item => item.Key == serviceName);
    }

    /// <summary>
    /// Checks whether the package can be delivered to the selected country.
    /// </summary>
    private bool CanDeliverToCountry(IAddress address)
    {
        // Gets the 3-letter country code from the delivery address
        var country = CountryInfoProvider.GetCountryInfo(address.
AddressCountryID);
        string countryCode = country?.CountryThreeLetterCode;

        // Returns true if the country code is found among the codes supported by 
the carrier provider
        return CountryCosts.ContainsKey(countryCode);
    }
}
```

2. Save all changes and **Build** the custom project.

Register the custom carrier in the Kentico administration interface:

1. Open the **Store configuration** application (or **Multistore configuration**).
2. Select the **Shipping -> Carriers** tab.
3. Click **New carrier**.
4. Type a **Name** for the carrier, and choose your custom assembly and *ICarrierProvider* class via the **Carrier provider** selector.
5. Click **Save**.

You can now create shipping options that use the custom carrier and its "Shipping cost to USA, CAN and MEX" service. If custome rs create an order with products that need shipping, and their delivery address is in the USA, Canada or Mexico, the given shipping options are available for selection and their cost is calculated by the custom shipping carrier provider.