If you enable your customers to register and become also users on your MVC site, the typical scenarios for registered customers are:

- Saving and pre-filling the customer's addresses when going through the checkout process to create an order.
- Saving and loading the customer's shopping cart in case of leaving some products in it.

This page describes the pre-filling of the customer's address when creating an order using the **Kentico.Ecommerce** integration package.

Accessing the current shopping cart

Storing and retrieving the customers' shopping carts works automatically in Kentico when using the **GetCurrentShoppingCart** method in the **ShoppingService** from the **Kentico.Ecommerce** integration package.

You can customize how Kentico retrieves the current shopping cart for cases such as:

- When the same shopping cart is used across all sites running under one Kentico instance
- When an older saved shopping cart overrides the current shopping cart
- What shopping cart information are removed when the shopping cart is retrieved from the database

See Retrieving the current shopping cart to learn more.

To use benefits of registered customers, you need to enable visitors of your MVC site to register. Learn more in Working with users on MVC sites.

To add the possibility to choose from all customer's saved addresses and to pre-fill them:

This process presumes you already have a working customer details step in your MVC checkout process as described in Creating the customer details step in MVC checkout processes.

1. Open your controller for the checkout process in your MVC project in Visual Studio.

   The following example uses the **ShoppingService**, **ICustomerAddressRepository** and **IShippingOptionRepository** classes initialized in the controller's constructor. See Using a shopping cart on MVC sites.

2. Modify your **DeliveryDetails** action method to contain loading of the customer.

```
        /// <summary>
        /// Displays the customer detail checkout process step with an address
selector for registered customers.
        /// </summary>
        public ActionResult DeliveryDetailsAddressSelector()
        {
            // Gets the current user's shopping cart
            ShoppingCart cart = shoppingService.GetCurrentShoppingCart();

            // If the shopping cart is empty, displays the shopping cart
            if (cart.IsEmpty)
            {
                return RedirectToAction("ShoppingCart");
            }

            // Gets all countries for the country selector
            SelectList countries = new SelectList(CountryInfoProvider.
GetCountries(), "CountryID", "CountryDisplayName");

            // Gets the current customer
            Customer customer = cart.Customer;

            // Gets all customer billing addresses for the address selector
            IEnumerable<CustomerAddress> customerAddresses = Enumerable.
Empty<CustomerAddress>();
            if (customer != null)
            {
                customerAddresses = addressRepository.GetByCustomerId(customer.
ID);
            }

            // Prepares address selector options
            SelectList addresses = new SelectList(customerAddresses, "ID",
"Name");

            // Gets all enabled shipping options for the shipping option selector
            SelectList shippingOptions = new SelectList(shippingOptionRepository.
GetAllEnabled(), "ShippingOptionID", "ShippingOptionDisplayName");

            // Loads the customer details
            DeliveryDetailsViewModel model = new DeliveryDetailsViewModel
            {
                Customer = new CustomerModel(cart.Customer),
                BillingAddress = new BillingAddressModel(cart.BillingAddress,
countries, addresses),
                ShippingOption = new ShippingOptionModel(cart.ShippingOption,
shippingOptions)
            };

            // Displays the customer details step
            return View(model);
        }
```

3. Add a method to the checkout controller. The method processes customer's address selection in the customer details step.

```
        /// <summary>
        /// Loads information of an address specified by its ID.
        /// </summary>
        /// <param name="addressID">ID of the address.</param>
        /// <returns>Serialized information of the loaded address.</returns>
        [HttpPost]
        public JsonResult CustomerAddress(int addressID)
        {
            // Gets the address with its ID
            CustomerAddress address = addressRepository.GetById(addressID);

            // Checks whether the address was retrieved
            if (address == null)
            {
                return null;
            }

            // Creates a response with all address information
            var responseModel = new
            {
                Line1 = address.Line1,
                Line2 = address.Line2,
                City = address.City,
                PostalCode = address.PostalCode,
                CountryID = address.CountryID,
                StateID = address.StateID,
                PersonalName = address.PersonalName
            };

            // Returns serialized information of the address
            return Json(responseModel);
        }
```

4. Modify your view of the customer details step to display the address selector. For example:

Adding the address selector

```
        <div class="js-address-selector-div" data-statelistaction='@Url.Action
("CustomerAddress", "Checkout")'>

            @Html.LabelFor(m => m.BillingAddress.AddressID)
            @Html.DropDownListFor(m => m.BillingAddress.AddressID, Model.
BillingAddress.Addresses, "(new)", new { @class = "js-address-selector" })

            <div class="message message-error">
                @Html.ValidationMessageFor(m => m.BillingAddress.AddressID)
            </div>
        </div>
```

Adding the JavaScript file

```
@Scripts.Render("~/Scripts/addressSelector.js")
```

5. Add a JavaScript file that loads the address information and pre-fills the information to the form.

```
»(function () {
    'use strict';

    $('.js-address-selector-div').change(function () {
        var $selectorDiv = $(this),
            $addressDiv = $selectorDiv.parent(),
            $selector = $selectorDiv.find('.js-address-selector'),
            url = $selectorDiv.data('statelistaction'),
            postData = {
                addressId: $selector.val()
            };

        if (!postData.addressId) {
            eraseFields($addressDiv);
            return;
        }

        $.post(url, postData, function (data) {
            fillFields($addressDiv, data);
        });
    });

    function fillFields($addressDiv, data) {
        fillBasicFields($addressDiv, data);
        fillCountryStateFields($addressDiv, data);
    }

    function fillBasicFields($addressDiv, data) {
        var basicFields = $addressDiv.data('fields'),
            addressType = $addressDiv.data('addresstype');

        $.each(basicFields, function (i, val) {
            var fieldId = '#' + addressType + '_' + val,
                fieldVal = data[val];

            $(fieldId).val(fieldVal);
        });
    }

    function fillCountryStateFields($addressDiv, data) {
        var $countryStateSelector = $addressDiv.find('.js-country-state-
selector'),
            countryField = $countryStateSelector.data('countryfield'),
            stateField = $countryStateSelector.data('statefield'),
            $countrySelector = $countryStateSelector.find('.js-country-selector');

        $countryStateSelector.data('stateselectedid', data[stateField]);
        $countrySelector.val(data[countryField]).change();
    }

    function eraseFields($addressDiv) {
        var data = {};
        fillFields($addressDiv, data);
    }
}());
```

When you now sign in on your website, you can select your address from the addresses that you have used in the past.