



The final step before payment is usually a preview of the future [order](#). In this step, visitors/[customers](#) can check content of their [shopping cart](#), billing/shipping [details they typed about themselves](#), and they can select their preferred [payment method](#). Finally, the page provides a button that finalizes the [checkout process](#) and creates an order. Also, the button can redirect to a payment gateway if the customer selects a suitable [payment method](#).

To create the checkout process steps, use the [Kentico.Ecommerce integration package](#).

 **Tip:** You can view a sample preview checkout process step on [the sample MVC Dancing Goat site](#).

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

 The following example uses the **ShoppingService** and **IPaymentMethodRepository** classes initialized in the controller's constructor. See [Using a shopping cart on MVC sites](#).

2. Create a model for payment methods. The model can use the **PaymentOptionInfo** class from the **CMS.Ecommerce** namespace.

```
public class PaymentMethodViewModel
{
    public int PaymentMethodID { get; set; }
    public SelectList PaymentMethods { get; set; }


    /// <summary>
    /// Creates a payment method model.
    /// </summary>
    /// <param name="paymentMethod">Selected payment method.</param>
    /// <param name="paymentMethods">List of all available payment methods.<
/param>
    public PaymentMethodViewModel(PaymentOptionInfo paymentMethod, SelectList
paymentMethods)
    {
        PaymentMethods = paymentMethods;

        if (paymentMethod != null)
        {
            PaymentMethodID = paymentMethod.PaymentOptionID;
        }
    }

    /// <summary>
    /// Creates an empty payment method model.
    /// </summary>
    public PaymentMethodViewModel()
    {
    }
}
```

 If you do not want to enable your customers to select their preferred payment method, you can skip this step.

3. Create a model for all previous steps – **PreviewAndPayViewModel**.

 This model uses the *DeliveryDetailsViewModel* model created during a process described in [Creating the customer details step in MVC checkout processes](#).



```
public class PreviewAndPayViewModel
{
    public DeliveryDetailsViewModel DeliveryDetails { get; set; }
    public ShoppingCart Cart { get; set; }
    public PaymentMethodViewModel PaymentMethod { get; set; }
}
```

4. Add methods that prepare the selection of payment methods.

```
/// <summary>
/// Decides whether the specified payment method is valid on the current
site.
/// </summary>
/// <param name="paymentMethodID">ID of the applied payment method.<
/param>
/// <returns>True if the payment method is valid.</returns>
private bool IsPaymentMethodValid(int paymentMethodID)
{
    // Gets the current user's shopping cart
    ShoppingCart cart = shoppingService.GetCurrentShoppingCart();

    // Gets a list of all applicable payment methods to the current
user's shopping cart
    List<PaymentOptionInfo> paymentMethods = GetApplicablePaymentMethods
(cart).ToList();

    // Returns whether an applicable payment method exists with the
entered payment method's ID
    return paymentMethods.Exists(p => p.PaymentOptionID ==
paymentMethodID);
}

/// <summary>
/// Gets all applicable payment methods on the current site.
/// </summary>
/// <param name="cart">Shopping cart of the site</param>
/// <returns>Collection of applicable payment methods</returns>
private IEnumerable<PaymentOptionInfo> GetApplicablePaymentMethods
(ShoppingCart cart)
{
    // Gets all enabled payment methods from Kentico
    IEnumerable<PaymentOptionInfo> enabledPaymentMethods =
paymentRepository.GetAll();

    // Returns all applicable payment methods
    return enabledPaymentMethods.Where(cart.IsPaymentMethodApplicable);
}
```

5. Add a method to the controller that creates the **PreviewAndPayViewModel**.



```
/// <summary>
/// Prepares a view model of the preview checkout process step including
the shopping cart,
/// the customer details, and the payment method.
/// </summary>
/// <returns>View model with information about the future order.</returns>
private PreviewAndPayViewModel PreparePreviewViewModel()
{
    // Gets the current user's shopping cart
    ShoppingCart cart = shoppingService.GetCurrentShoppingCart();

    // Prepares the customer details
    DeliveryDetailsViewModel deliveryDetailsModel = new
DeliveryDetailsViewModel
    {
        Customer = new CustomerModel(cart.Customer),
        BillingAddress = new BillingAddressModel(cart.BillingAddress,
null, null)
    };

    // Prepares the payment method
    PaymentMethodViewModel paymentViewModel = new PaymentMethodViewModel
    {
        PaymentMethods = new SelectList(GetApplicablePaymentMethods
(cart), "PaymentOptionID", "PaymentOptionDisplayName")
    };

    // Gets the selected payment method if any
    PaymentOptionInfo paymentMethod = cart.PaymentMethod;
    if (paymentMethod != null)
    {
        paymentViewModel.PaymentMethodID = paymentMethod.PaymentOptionID;
    }

    // Prepares a model from the preview step
    PreviewAndPayViewModel model = new PreviewAndPayViewModel
    {
        DeliveryDetails = deliveryDetailsModel,
        Cart = cart,
        PaymentMethod = paymentViewModel
    };

    return model;
}
```



If you provide a possibility to have a different shipping address, prepare also the shipping address for the *DeliveryDetailsViewModel*.

6. Add a method to the controller that displays the preview step.



```
/// <summary>
/// Display the preview checkout process step.
/// </summary>
public ActionResult PreviewAndPay()
{
    // Gets the current user's shopping cart
    ShoppingCart cart = shoppingService.GetCurrentShoppingCart();

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

    // Prepares a model from the preview step
    PreviewAndPayViewModel model = PreparePreviewViewModel();

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

7. Add a method to the controller that processes the order. If the processing is successful, the method displays a payment page.



```
/// <summary>
/// Validates that all information is correct to create an order, creates
an order,
/// and redirects the customer to payment.
/// </summary>
/// <param name="model">View model with information about the future
order.</param>
[HttpPost]
public ActionResult PreviewAndPay(PreviewAndPayViewModel model)
{
    // Gets the current user's shopping cart
    ShoppingCart cart = shoppingService.GetCurrentShoppingCart();

    // Validates the shopping cart
    ShoppingCartCheckResult checkResult = cart.ValidateContent();

    // Gets the selected payment method and assigns it to the shopping
cart
    cart.PaymentMethod = paymentRepository.GetById(model.PaymentMethod.
PaymentMethodID);

    // Evaluates the shopping cart
    cart.Evaluate();

    // If the validation was not successful, displays the preview step
again
    if (checkResult.CheckFailed || !IsPaymentMethodValid(model.
PaymentMethod.PaymentMethodID))
    {
        // Prepares a model from the preview step
        PreviewAndPayViewModel viewModel = PreparePreviewViewModel();

        // Displays the preview step again
        return View("PreviewAndPay", viewModel);
    }

    // Creates an order from the shopping cart
    Order order = shoppingService.CreateOrder(cart);

    // Deletes the shopping cart from the database
    shoppingService.DeleteShoppingCart(cart);

    // Redirects to the payment gateway
    return RedirectToAction("Index", "Payment", new { orderID = order.
OrderID });
}
```



To limit payment methods based on the selected shipping options, see [Making payment methods dependent on shipping options](#).



If you provide only [manual payment methods](#) (and therefore, you do not want to [connect any payment gateways](#)), you can redirect them directly to the thank-you page. For example:

```
return RedirectToAction("ThankYou", new { orderID = order.
OrderID });
```

8. Create a view for the preview step.

```
<div id="customerDetails">
  <h2>Customer Details</h2>
  <div>
    @Html.LabelFor(m => m.DeliveryDetails.Customer.FirstName)
    @Html.DisplayFor(m => m.DeliveryDetails.Customer.FirstName)
  </div>
  <div>
    @Html.LabelFor(m => m.DeliveryDetails.Customer.LastName)
    @Html.DisplayFor(m => m.DeliveryDetails.Customer.LastName)
  </div>
  <div>
    @Html.LabelFor(m => m.DeliveryDetails.Customer.Company)
    @Html.DisplayFor(m => m.DeliveryDetails.Customer.Company)
  </div>
</div>

<div id="cartContent">
  <ul>
    *@ Loops through all shopping cart items. *@
    @foreach (Kentico.Ecommerce.ShoppingCartItem cartItem in Model.Cart.Items)
    {
      *@ Displays the shopping cart items' properties. *@
      <li>
        @cartItem.Units&times; @cartItem.Name ... @currency.FormatPrice
        (cartItem.Subtotal)
      </li>
    }
  </ul>
</div>

<div id="shoppingCartTotals">
  <p>Total tax: @currency.FormatPrice(Model.Cart.TotalTax)</p>
  <p>Total shipping: @currency.FormatPrice(Model.Cart.Shipping)</p>
  <p>Total (incl. tax): @currency.FormatPrice(Model.Cart.GrandTotal)</p>
</div>

@using (Html.BeginForm("PreviewAndPay", "Checkout", FormMethod.Post))
{
  <div id="paymentMethod">
    @Html.LabelFor(m => m.PaymentMethod.PaymentMethodID)
    @Html.DropDownListFor(m => m.PaymentMethod.PaymentMethodID, Model.
    PaymentMethod.PaymentMethods)
  </div>

  <input type="submit" value="Create an order" />
}
```



When a visitor gets to the customer details step during their [checkout process](#), they review their information and the cart content and can continue with creating an order. You can add either a redirect to a payment gateway or a redirect to the thank-you page.

After creating an order, a notification email is sent based on the [Send order notification setting](#).