

NServiceBus Hands-On Lab

**Lab 101: Hello NServiceBus World** (rev 1.3)

#### Greetings, NServiceBus user, and welcome to the first NServiceBus hands-on lab

We are glad that you have chosen to give NServiceBus a try, and hope that after running this hands-on lab you’ll see how simple it can be to build robust, scalable, and secure systems using NServiceBus.

Lab 1: Hello NServiceBus World will guide you in building a simple ordering system with NServiceBus. The ordering system includes three projects—Client, Server, and Messages—to hold the message contracts between the client and the server.

Terms of Use

© 2012, 2013 Particular Ltd. All rights reserved.

By using this Hands-on Lab, you agree to the following terms:

The technology/functionality described in this Hands-on Lab is provided by NServiceBus in a “sandbox” testing environment for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the Hands-on Lab to evaluate such technology features and functionality and provide feedback to NServiceBus. You may not use it for any other purpose. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this Hands-on Lab or any portion thereof.

COPYING OR REPRODUCTION OF THE HANDS-ON LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED.

THIS HANDS-ONLAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS HANDS-ON LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCITONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

**FEEDBACK**. If you give feedback about the technology features, functionality and/or concepts described in this Hands-on Lab to NServiceBus, you give to NServiceBus, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a NServiceBus software or service that includes the feedback. You will not give feedback that is subject to a license that requires NServiceBus to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

NServiceBus Ltd. HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE HANDS-ON LAB , INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. NServiceBus DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF THE VIRTUAL LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE VIRTUAL LAB FOR ANY PURPOSE.

Table of Contents

[Creating the Client project 4](#_Toc361844411)

[Creating the Messages project 5](#_Toc361844412)

[Creating the Server project 6](#_Toc361844413)

[Sending an order 7](#_Toc361844414)

[Running the solution 8](#_Toc361844415)

[Completed Solution 8](#_Toc361844416)

## Creating the Client project

Create a **Client** project, which you will use to send order requests to an NServiceBus endpoint:

* Open Visual Studio 2012 as administrator and open the OrderingSolution from C:\Hands on Labs\Exercises\Lab 101 – Hello World:   
  File 🡪 Open 🡪 Project/Solution…

Add references from the NServiceBus assemblies to the Client project. The quickest and easiest way is using NuGet:

* Open the NuGet Package Manager Console:  
  Tools 🡪 Library Package Manager 🡪 Package Manager Console  
  Type the following command at the Package Manager Console:

**Install-package NServiceBus.Host Client –DependencyVersion HighestMinor**

**NOTE**: Upon running the command, Visual Studio detects a change in the Client project file and asks you to reload the project. Click the Reload button.

The package installation process adds references to NServiceBus assemblies and creates a configuration file called EndpointConfig.cs in the Client project.

* Open the EndpointConfig.cs file that was just created for you and select persistence to InMemory:

Starting from NServiceBus version 5, it is essential to pick the Persistence of choice. The generated code is as below:

public class EndpointConfig : IConfigureThisEndpoint

{

public void Customize(BusConfiguration configuration)

{

// NServiceBus provides the following durable storage options

// To use RavenDB, install-package NServiceBus.RavenDB and then use configuration.UsePersistence<RavenDBPersistence>();

// To use SQLServer, install-package NServiceBus.NHibernate and then use configuration.UsePersistence<NHibernatePersistence>();

// If you don't need a durable storage you can also use, configuration.UsePersistence<InMemoryPersistence>();

// more details on persistence can be found here: http://docs.particular.net/nservicebus/persistence-in-nservicebus

//Also note that you can mix and match storages to fit you specific needs.

//http://docs.particular.net/nservicebus/persistence-order

configuration.UsePersistence<PLEASE\_SELECT\_ONE>();

}

}

Since we are using this for development/debugging, change the highlighted line to: **configuration.UsePersistence<InMemoryPersistence>();**

You will add more code to the **Client** project later on. For now, concentrate on the area that will handle your order requests.

## Creating the Messages project

The Messages project is the container of message definitions. This project will be shared between the client and server so both sides agree on the typed message descriptions:

Install the **NServiceBus** NuGet package for this new project:

* At the Package Manager Console, type

**Install-package NServiceBus Messages**

Add a command with a property to hold a product name:

Implement the PlaceOrder command in PlaceOrder.cs.

* Replace the content of PlaceOrder.cs with the following code:

namespace Messages

{

using NServiceBus;

public class PlaceOrder : ICommand

{

public string Product { get; set; }

}

}

## Creating the Server project

You are now ready to create the order processing server:

Install the **NServiceBus Host** NuGet package for this new project:

* At the Package Manager Console, type

**Install-package NServiceBus.Host Server –DependencyVersion HighestMinor**

**NOTE**: As before, upon running the command, Visual Studio detects a change in the Client project file and asks you to reload the project. Click the Reload button.

* Select persistence in the EndpointConfig.cs to use InMemoryPersistence.

Replace:

**configuration.UsePersistence<PLEASE\_SELECT\_ONE>();**

With:

**configuration.UsePersistence<InMemoryPersistence>();**

* For the server side to understand and interpret the message content, add a reference to the **Messages** project you created earlier:

Right click References in the Server Project 🡪 Add Reference 🡪 Messages

* Replace the content of PlaceOrderHandler.cs with the following code:

using System;

using NServiceBus;

using Messages;

namespace Server

{

public class PlaceOrderHandler : IHandleMessages<PlaceOrder>

{

public void Handle(PlaceOrder message)

{

Console.WriteLine("Order for Product: {0} placed.", message.Product);

}

}

}

## Sending an order

You are nearly done; you just have to add a Messages reference to the client side and add SendOrder logic:

* For the client side to understand and interpret the message content, return to the Client project and add a reference to the **Messages** project you created earlier:

Right click References at the Client Project 🡪 Add Reference 🡪 Messages

* Replace the content of SendOrder.cs with the following code:

using Messages;

using NServiceBus;

namespace Client

{

public class SendOrder : IWantToRunWhenBusStartsAndStops

{

public IBus Bus { get; set; }

public void Start()

{

Bus.Send("Server", new PlaceOrder() {Product = "New shoes"});

}

public void Stop()

{

}

}

}

## Running the solution

You’ve completed coding the example and now it’s time to run the solution. To see the complete system, run both the Client and the Server projects together:

* Start the Server (Right click **Server -> Debug -> Start new instance)**
* Start the Client (Right click **Client -> Debug -> Start new instance)**

In the Server window, you should now see: "Order for Product:New shoes placed".

**NOTE**: Warnings indicate that the NServiceBus framework could not find the queues on the running machine so it creates them on the fly. If no warnings appear, the queues already exist.

## Completed Solution

The complete solution for this exercise that uses NServiceBus version 4.x can be found under C:\Hands on Labs\Completed Solutions\Lab101-Hello World.

The complete solution using Version 5.x and this manual is here:

<https://github.com/Particular/HandsOnLabs/tree/master/Lab101>

Congratulations!

You've just built your first NServiceBus application.

Wasn't that easy?