



How-to: Web services

Document version: 1.01

This document describes, with a simple step by step example, how to use Nirva as a web service provider. The reader should have some understanding of web services and involved protocols (SOAP and WSDL).



Nirva can also be used as a web service client. This is generally done using the SYSTEM:XML:SEND command and the XSLT internal engine to prepare the SOAP messages to send and the received responses. This is not described in this document.

What is a web service

Web services represent a new architectural paradigm for applications. Web services implement capabilities that are available to other applications (or even other web services) via industry standard network, application interfaces and protocols. An application can use the capabilities of a web service by simply invoking it across a network without having to integrate it. As such, web services represent software building blocks that are URL addressable.

The capabilities provided by a web service can fall into a variety of categories, including:

- Functions, such as routine for calculating the integral square root of a number.
- Data, such as fetching the quantity of a particular widget a vendor has on hand.
- Business processes, such as accepting an order for a widget, shipping the desired quantity of widgets and sending an invoice.

Some of these capabilities are difficult or impractical to integrate within third party applications. When such capabilities are exposed as web services, they can be loosely coupled together, thereby achieving the benefits of integration without incurring the difficulties thereof.

Web services expose their capabilities to client applications, not their implementation. This allows web services to be implemented in any language and on any platform and still be compatible with all client applications.

Each building block (web service) is self-contained. It describes its own capabilities, publishes its own programming interface and implements its own functionality that is available as a hosted service. The business logic of the web service runs on a remote machine that is accessible by other applications through a network. The client application simply invokes the functionality of a web service by sending it messages, receives return messages from the web service and then uses the result within the application. Since there is no need to integrate the web service within the client application into a single monolithic block, development and testing times, maintenance costs, and overall errors are thereby reduced.

Practically, a web service is defined as set of operations, each of them having a well defined structured pair of input and output messages. The description of the web service operations and messages is done into an

XML data flow that respects the WSDL standard and is URL accessible. In this way, a web service client application knows what the web service operations are and how to invoke them. Some usual language tools allow automatically constructing web service client stubs from a particular favourite language (e.g. Java AXIS)

A client application invokes a web service operation by sending to it an XML message compliant with the SOAP standard. The web service executes the operation and returns the result message also SOAP formatted.

The protocol used for exchanging messages between a web service and its clients is usually HTTP.

NIRVA implementation of web services

NIRVA allows users creating web services in just few clicks by the way of the NIRVA configuration tool.

A NIRVA web service is defined at system level but is always executed by a NIRVA application. In this way, a single web service can be used by several NIRVA applications.

A permission associated to each operation of a web service allows NIRVA applications to control web service security access.

Any NIRVA command puts incoming data into an input container and delivers resulting data from an output container. A NIRVA web service operation is very similar to a single NIRVA command except that the input and output message structures are well defined.

A NIRVA web service message is a well defined NIRVA container having sub-containers and NIRVA objects.

A NIRVA web service operation is a NIRVA procedure (written in native, Perl, .Net or Java language) that takes data from the input container and delivers data to the output container.

A NIRVA web service itself is a collection of web service operations.

A NIRVA web service is accessible to external application like any other web service using HTTP, XML and SOAP standards but also from a NIRVA procedure or service by way of a dedicated NIRVA command. In this way, it is possible to integrate the web service business blocks into NIRVA applications or to create web services that call other web services.

Web service example

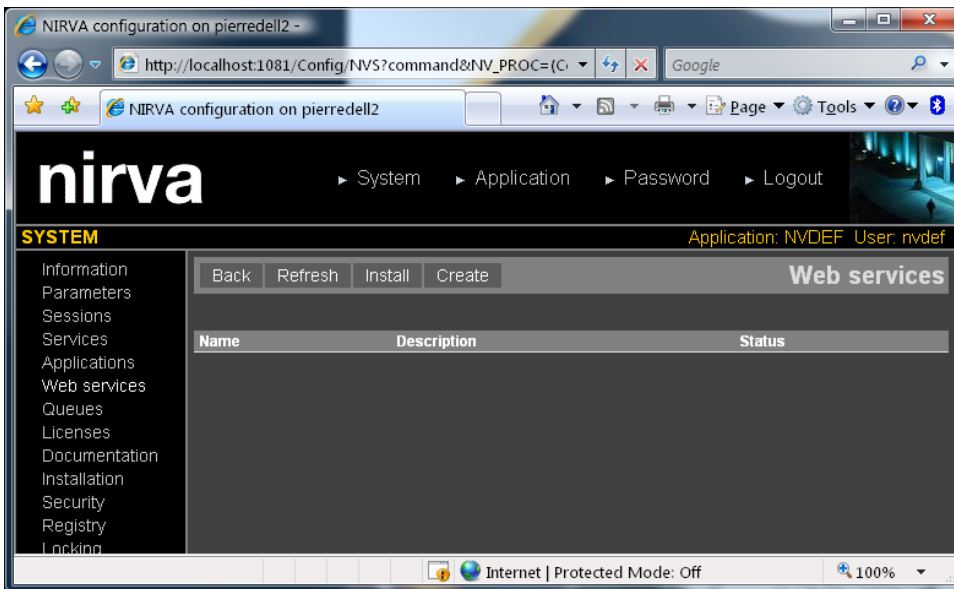
The following example creates a web service that takes in input the name of a person (first name and last name) and returns a welcome message.

Before trying this example, the NIRVA server must be running and the user must have enough rights to submit the necessary commands. The example runs on the NIRVA default application (NVDEF) and user (nvdef). Please consult the configuration chapter in order to give all the necessary permissions to the nvdef user in the NVDEF application. We can suggest to give the nvdef user all permissions.

The example shows nearly all steps in detail. In reality, these steps are carried out in a few seconds.

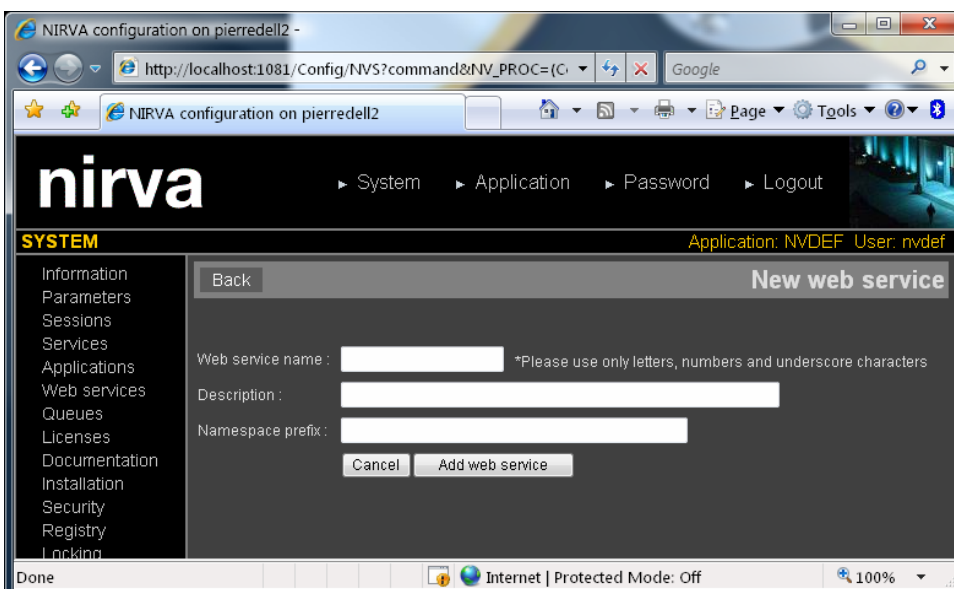
Creating the web service

For creating a web service, first run the configuration tool from your web browser (<http://127.0.0.1:1081/Config/login.htm> from the local machine) and use “nvadmin” user with default password “nirva” if you didn’t change it. Then go in the System/Web service menu. This should display the following screen:

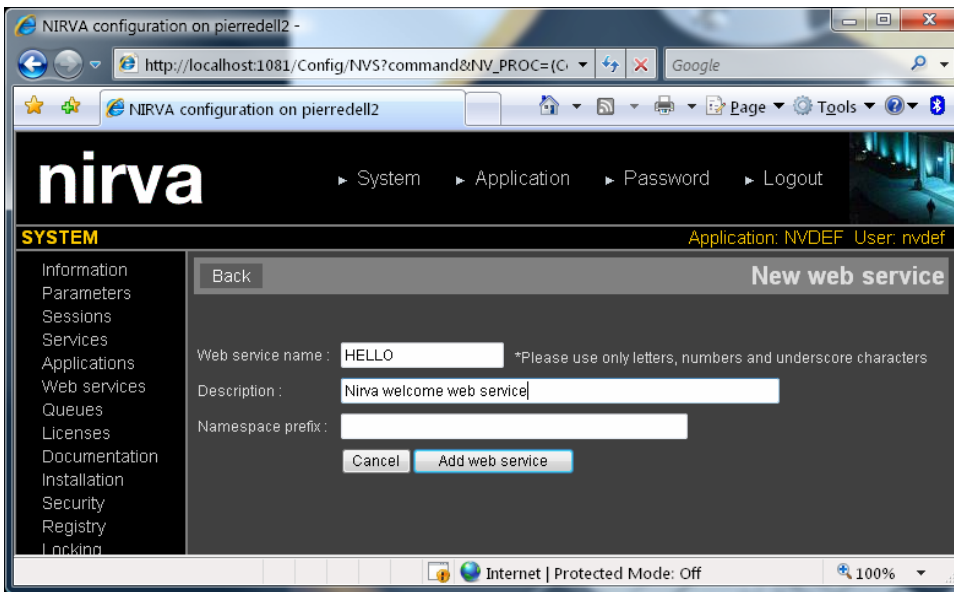


This is the list of available web services on your NIRVA server. The list should be empty if it is the first time you work with NIRVA web services.

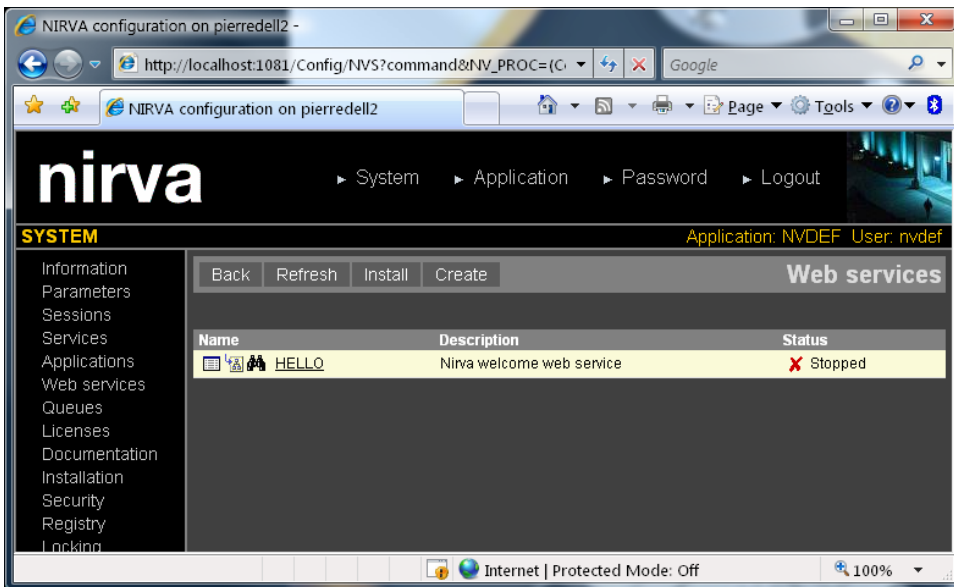
For creating a web service, press the “Create” button at the top of the screen:



In the Web service name, enter “HELLO” and in the description field, enter “Nirva welcome web service”: Leave the “Namespace prefix” field blank.



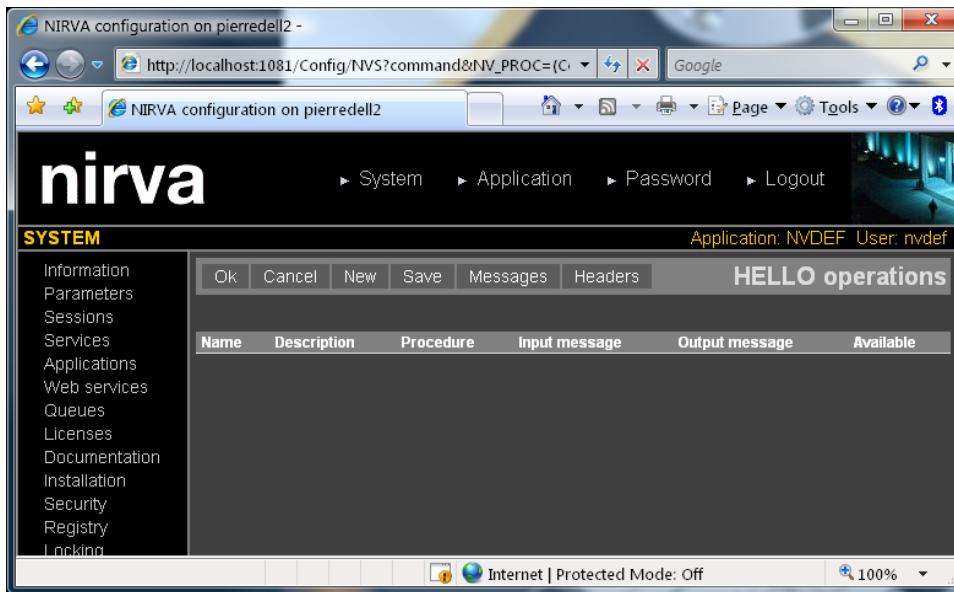
Press the “Add web service” button in order to create your “HELLO” web service. This returns to the web service list with the new HELLO web service listed:



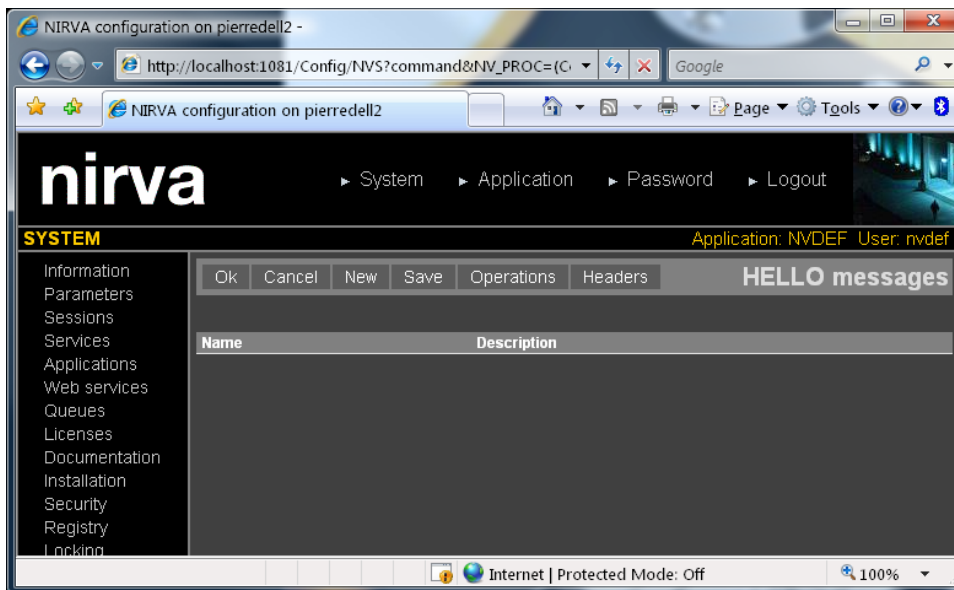
Editing the web service

After creating the web service we edit it in order to define its messages and operations. We will simply create an operation named “Welcome” accepting an input message named “You” that contains two string objects “firstname” and “lastname” and delivering an output message named “Message” containing a string object named “Welcome”.

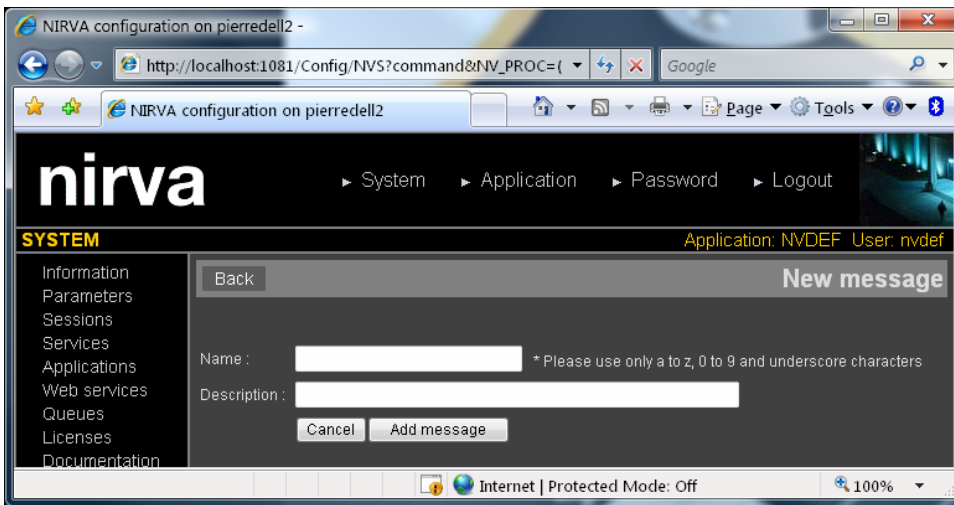
Enter the editing mode (the web service should be stopped for being able to enter editing mode). For that, click on the icon near the service name. This displays the following screen:



This is the list of operations. We'll create the "Welcome" operation later. Define the messages first by clicking on the "Messages" button:

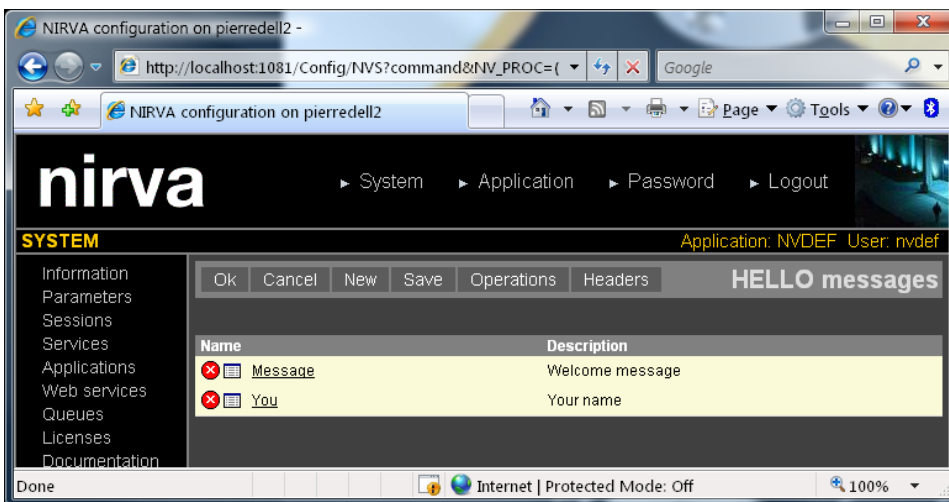


This is the list of defined messages for the HELLO web service. We now create a new message by pressing the "New" button:

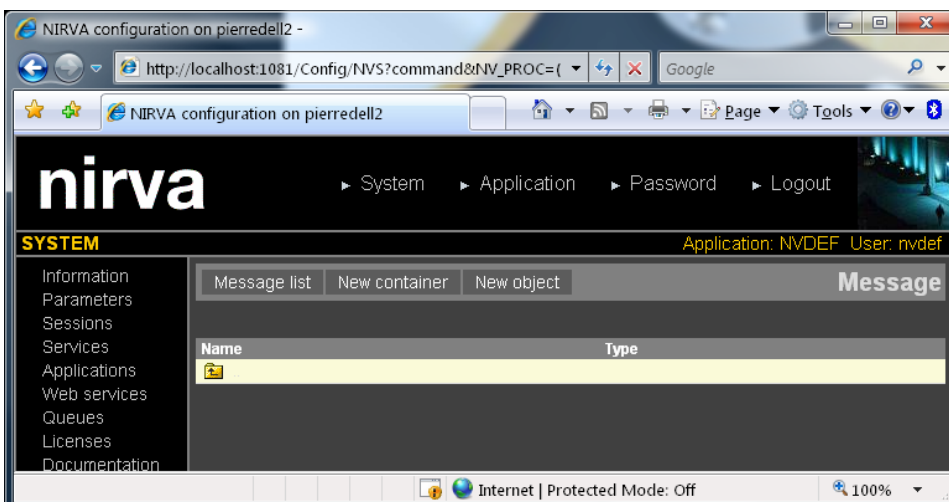


For the first message, enter “You” as the message name and “Your name” as description and press the “Add message” button. Repeat the operation for the second message with “Message” as name and “Welcome message” as description.

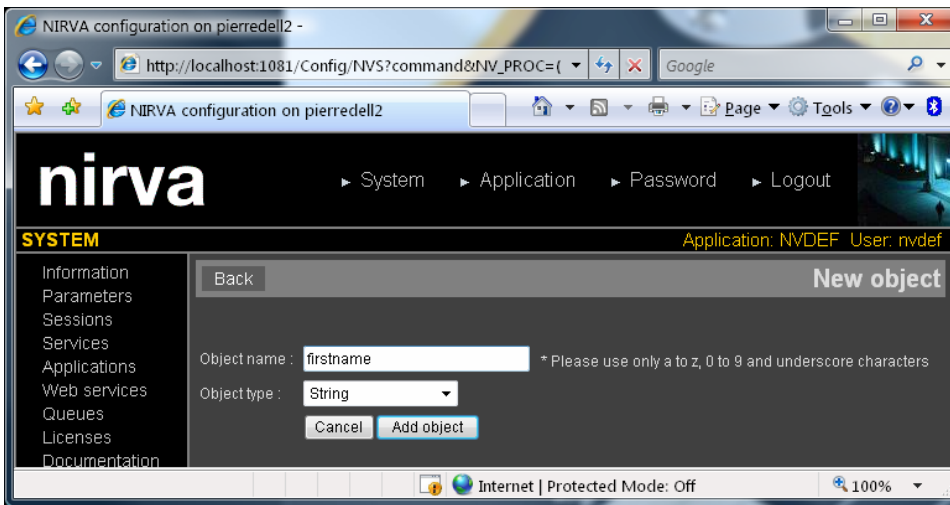
The message list should look like this:



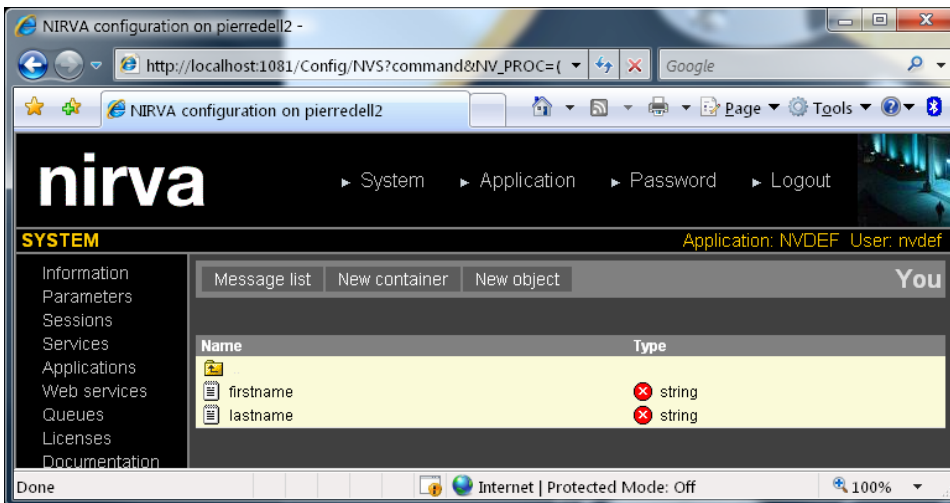
We have now the messages but no content. For editing message content: click on the message name. This enters into the structure of the message content. Let's do it first for the “You” message:



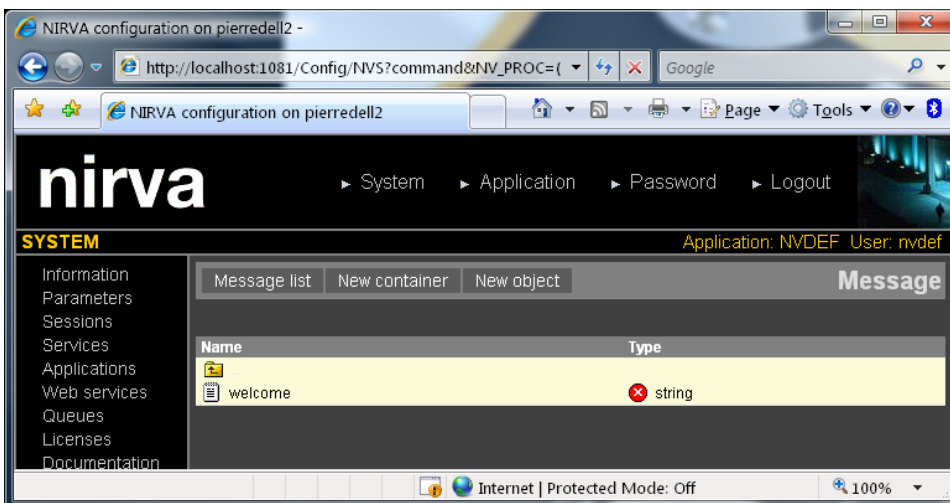
Into this message, create two string objects named respectively “firstname” and “lastname”. For that, press the “New object” button:



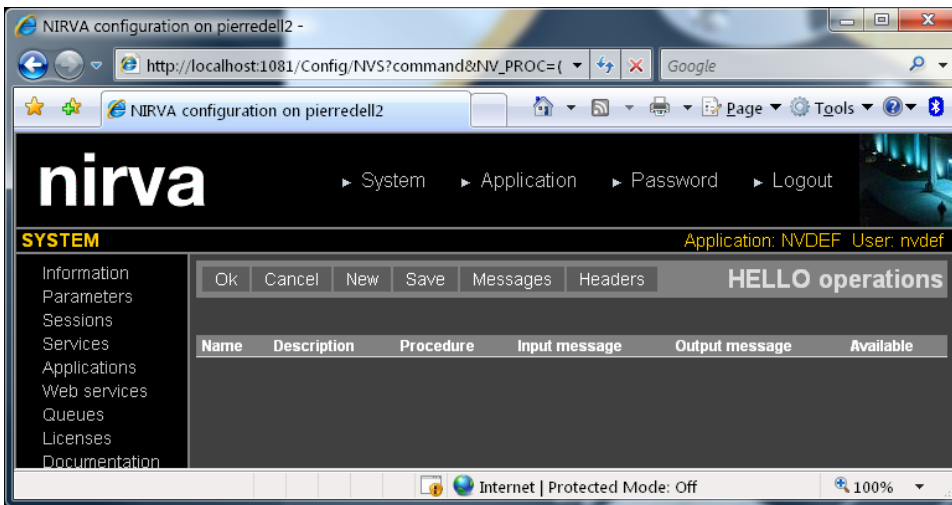
Then press the “Add object” button. Repeat the operation with the second object. The “You” message structure should look like this:



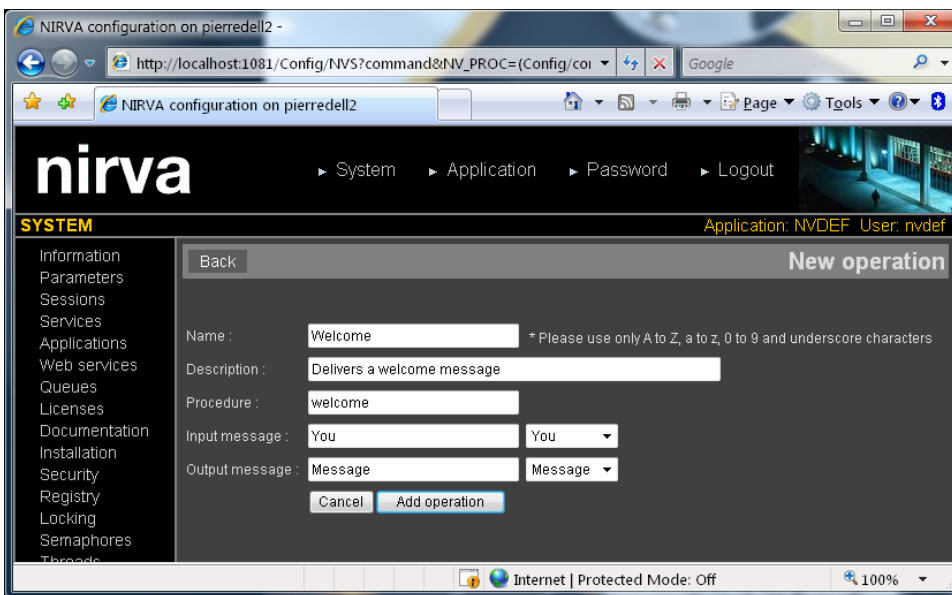
Do the same with the “Message” message by creating a string object named “welcome”:



Now we can come back to the message list by pressing the “Message list” button and switch to the operation list by clicking the “Operations” button:



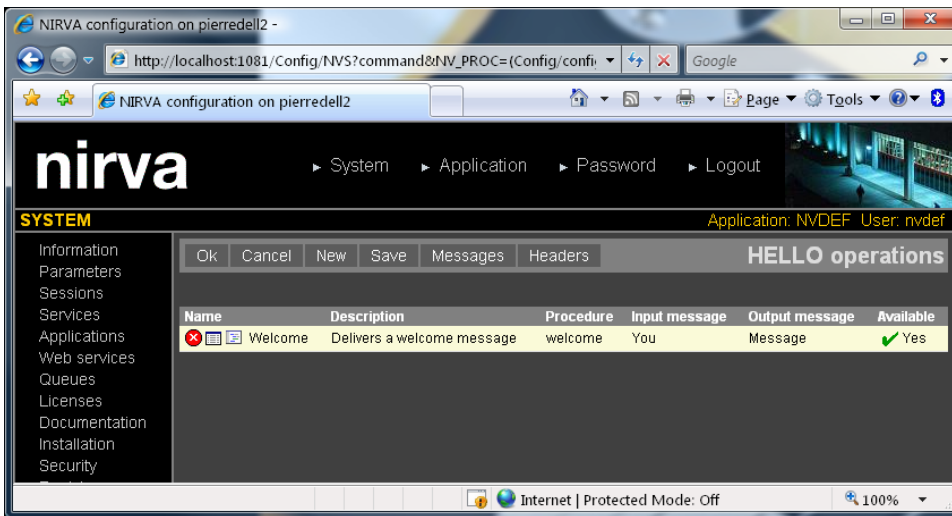
Create a new operation called “Welcome” by clicking on the “New” button:



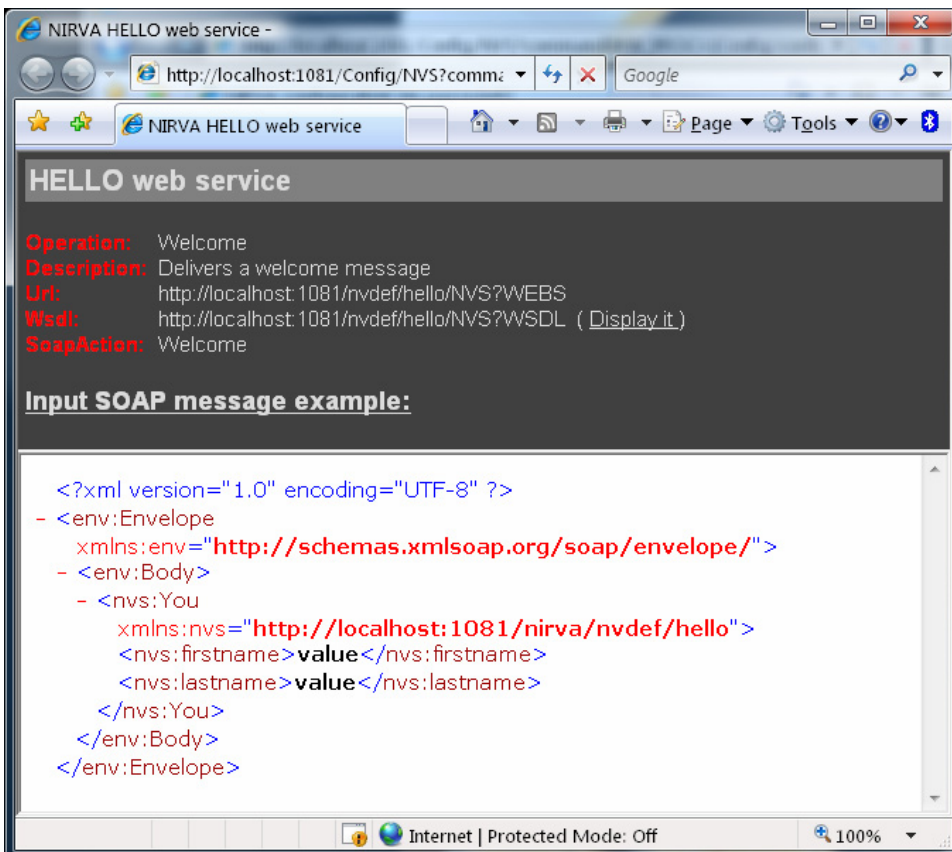
Enter the following information:

- Name is “Welcome”.
- Description is “Delivers a welcome message”.
- Procedure is “welcome”.
- Input message is “You” (chosen from the dropdown list).
- Output message is “Message” (chosen from the dropdown list).

And press the “Add operation” button. The operation list should then look like this:

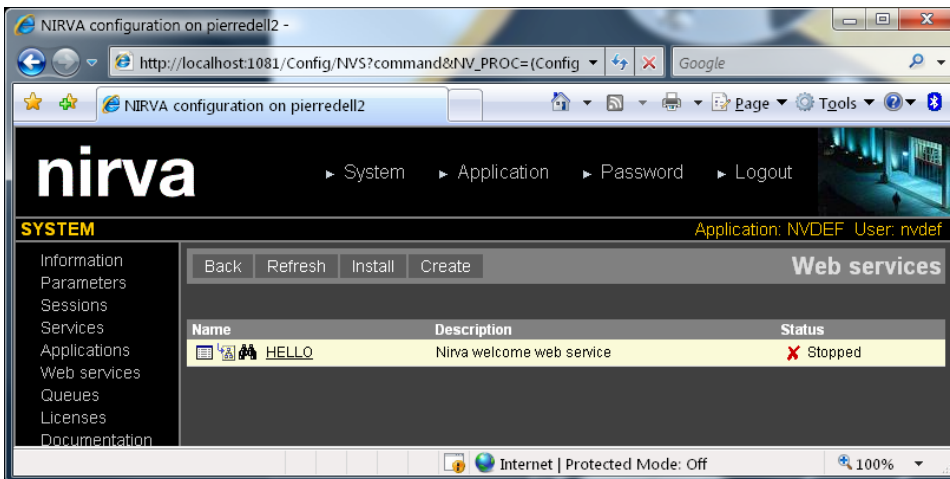



Our web service is now nearly ready. We can get some detailed information about the “Welcome” operation by clicking on the icon near the operation name:



The WSDL URL (in our example <http://localhost:1081/nvdef/hello/NVS?WSDL>) is available by clicking the “Display it” link from this screen. This URL will be useful for your web service client to automatically construct the messages or code for the web service (e.g. Java AXIS).

Go back to the web service list by clicking on the “Ok” button of the operation list. This also saves the HELLO web service. Nirva then displays the list of web services:



Note: the WSDL is also available from this screen clicking on the  icon near the web service name.


We have now finished with the web service definition itself but the web service doesn't do anything. We must create the procedure that processes input data and delivers output data. For that, we create a file named "welcome.nvp" (we have defined the procedure name as "welcome" in the web service operation) in the Nirva/Webservices/HELLO/Procs directory with the following content:

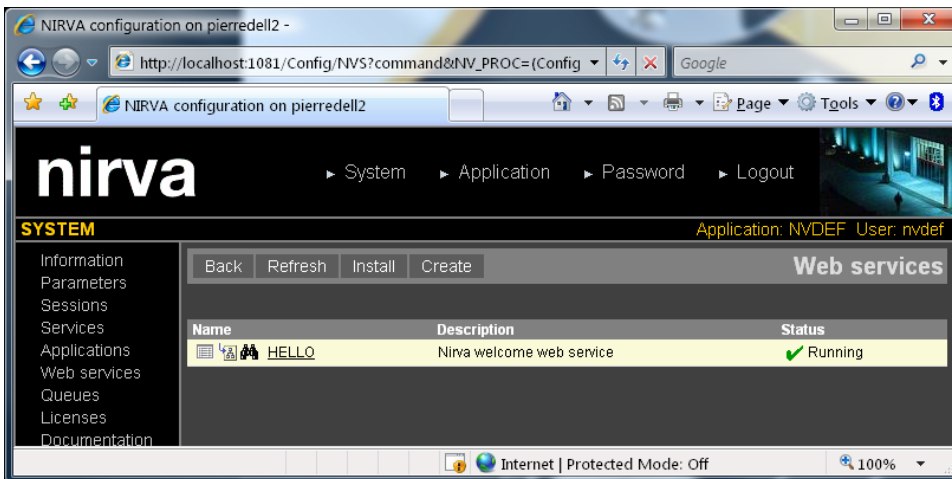
```
NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|firstname| NV_VAR=|FirstName|
NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|lastname| NV_VAR=|LastName|
NV_CMD=|OBJECT:CREATE| NAME=|welcome| TYPE=|STRING|
NV_CMD=|OBJECT:STRING_SET_VALUE| NAME=|welcome| VALUE=|Welcome to NIRVA web services, |
+|#FirstName| + | | +|#LastName|
```

Please respect the syntax and put one command on an entire line.

This simple procedure just constructs the welcome message from the given first and last names. In this example, we use the NIRVA native language, but the procedure can be also written directly in Perl, .Net or Java language.

Starting the web service

For starting, the web service, click on the  icon in the web service status column. The web service is now ready to run:



Running the web service

Testing the web service can be done with any dedicated web service tool on the market or the NIRVA `nvcc` tool. The first test shown is done with `nvcc`. The second tests is done with `soapUI`.

With `nvcc` tool

We must first create an input XML file named "welcomein.xml" with the following content:

```
<?xml version="1.0" encoding="UTF-8" ?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Body>
    <nvs:You xmlns:nvs="http://pierredell:1081/nirva/nvdef/hello">
      <nvs:firstname>John</nvs:firstname>
      <nvs:lastname>SMITH</nvs:lastname>
    </nvs:You>
  </env:Body>
</env:Envelope>
```

This is the input SOAP message for our hello web service. Please copy this file into the Nirva/Bin directory.

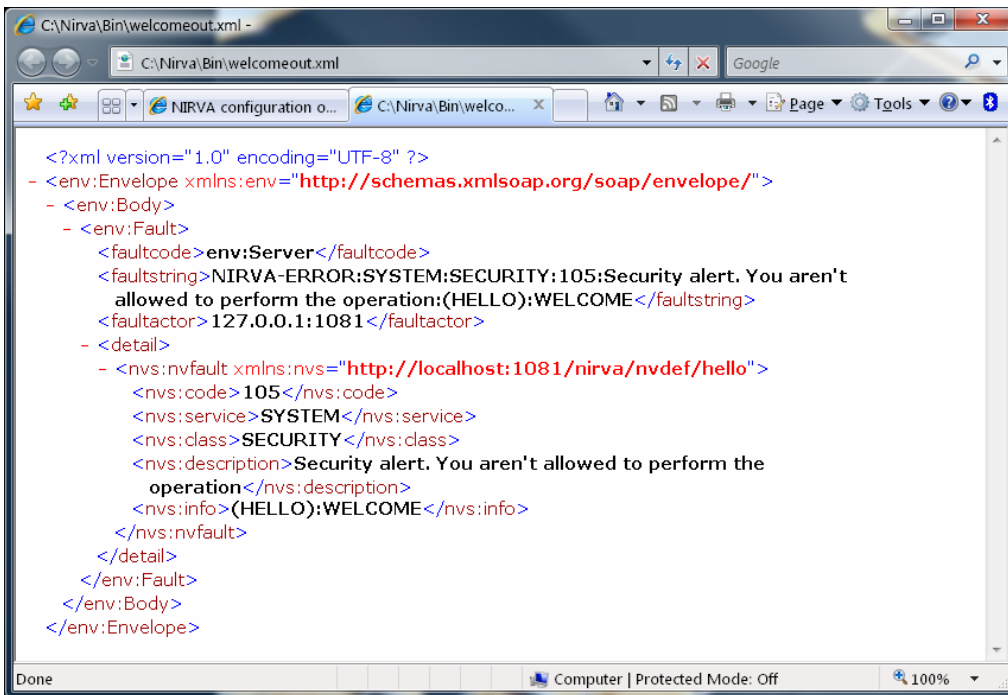
For testing the web service, open a console window, go into the Nirva/Bin directory and type the following command line:

```
nvcc -i testwebs.txt HELLO Welcome welcomein.xml welcomeout.xml
```

This instructs NIRVA to execute the Welcome operation of the HELLO web service with the input data of the "welcomein.xml".

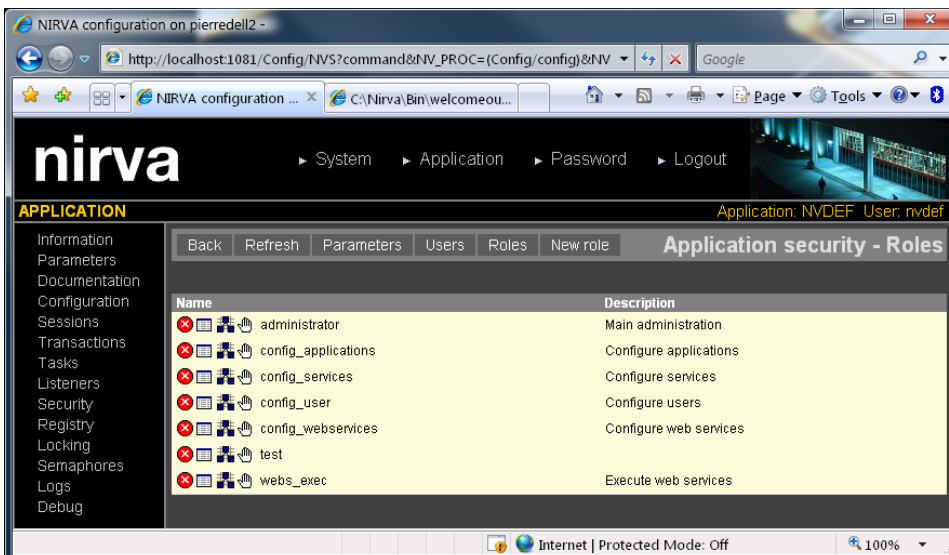
NIRVA delivers the output message into the welcomeout.xml file.

After running the command, the resulting welcomeout.xml file looks like this:

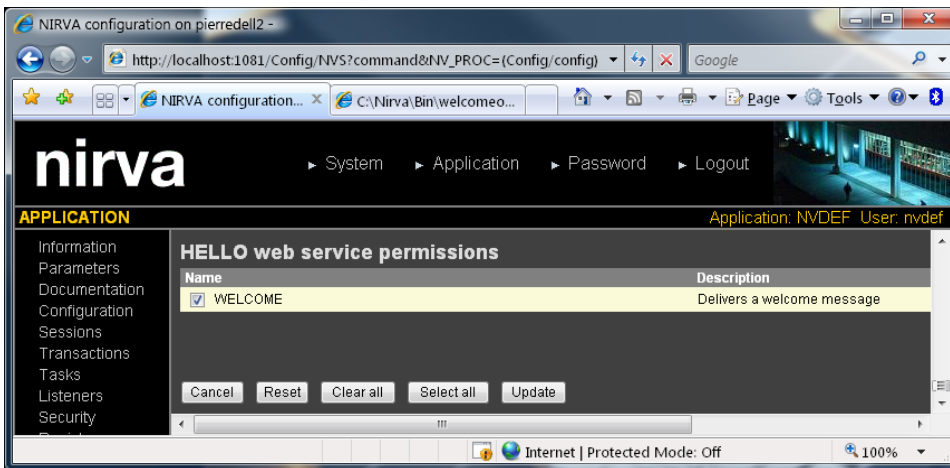


In fact, the returned message is a SOAP fault message because we forgot to allow the default user of the default application to use the web service.

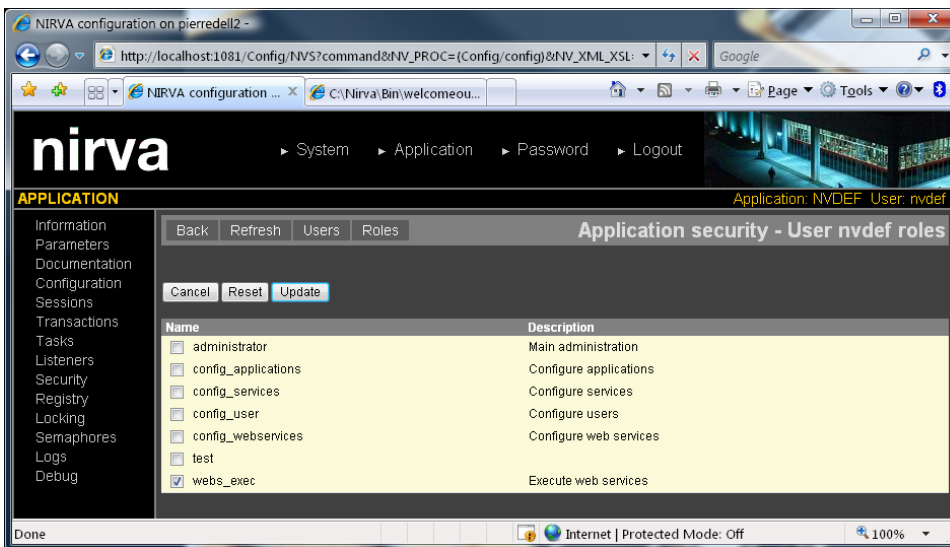
For that, we must create a new role to the default user and add the HELLO/Welcome web service permission to this role (we can also just add the permission to an existing role). Go into the Nirva configuration tool (<http://localhost:1081/Config/login.htm>) and login to the default application (do not enter anything in the Application field of the login screen). Then go into the application security and add a new role named "webs_exec":



In this role, display the list of permissions, check the WELCOME permission for the HELLO web service (this should be at the bottom of the permission screen) and press the update button:



Now give to "nvdef" user the webs_exec role and press the update button:

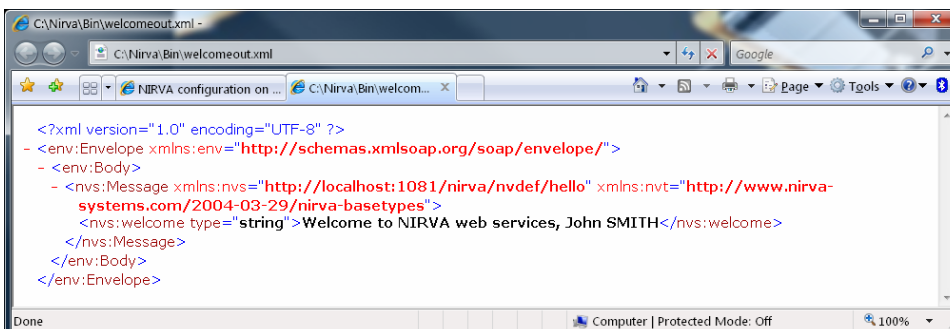


Nota: the screens may differ on your computer following the security rights you have already set.

Now, everything is OK and we can run again the web service with the command line:

```
nvcc -i testwebs.txt HELLO Welcome welcomein.xml welcomeout.xml
```

Now the welcomeout.xml file looks like this:

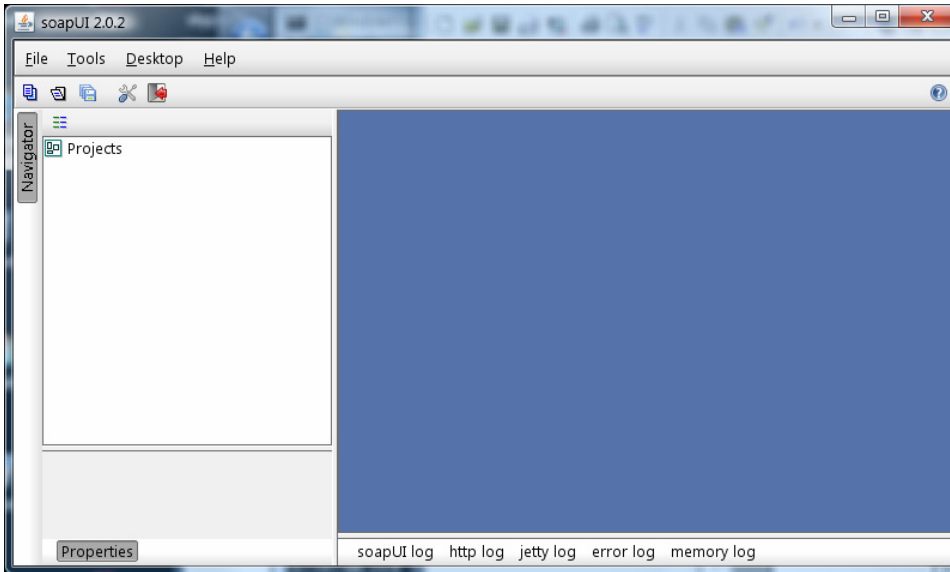


This is now the expected result of our simple HELLO web service.

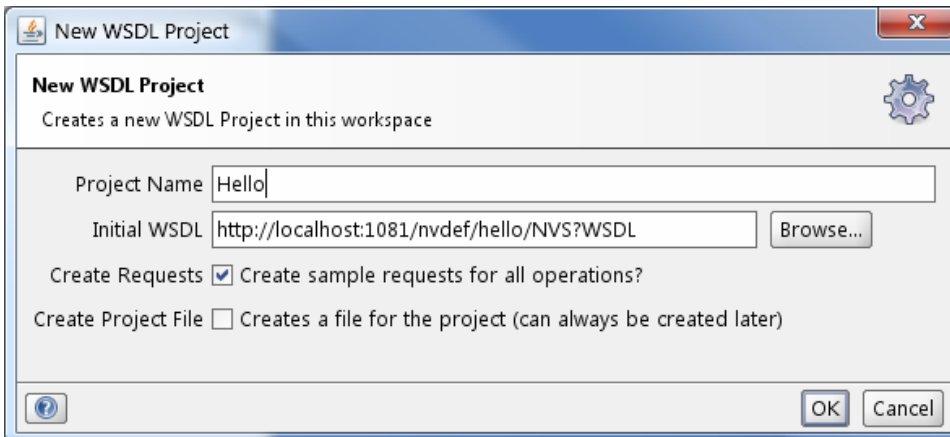
With SoapUI standard tool

The soapUI is a free tool for testing web services. It can be found on <http://www.soapui.org/>. We use version 2.0.2 in this documentation.

First start soapUI. This should display the following screen:

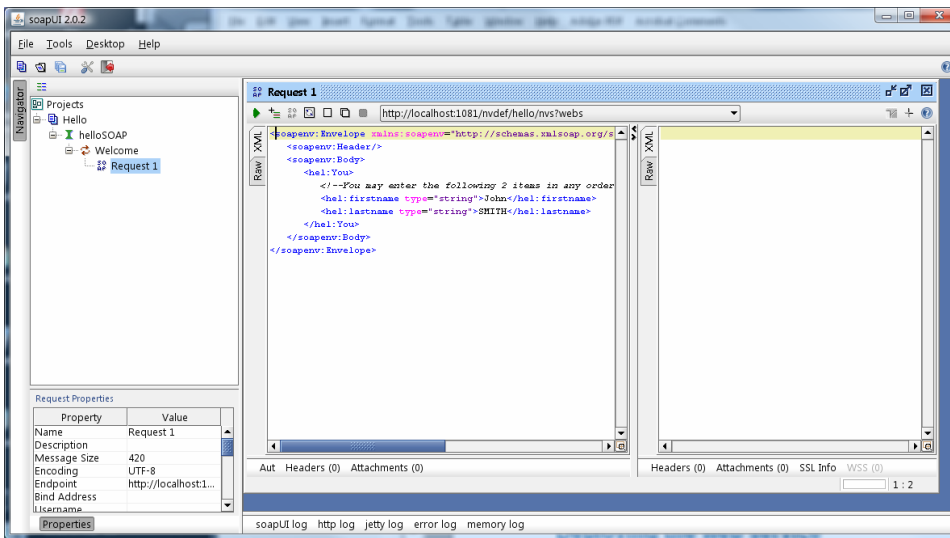


Right click on Projects and choose “New WSDL Project”:




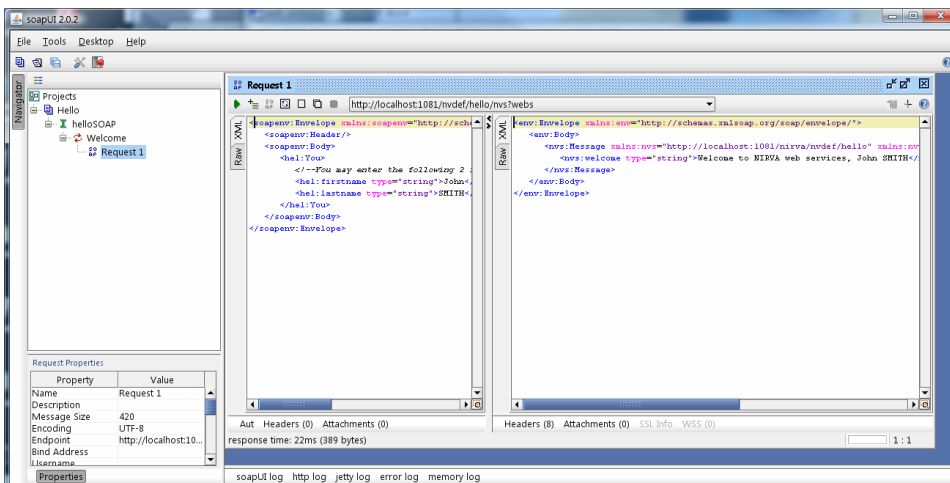
Name your project “Hello” and enter the URL of the web service as found in the Nirva configuration tool. If you run soapUI on the same computer than Nirva, this will be “<http://localhost:1081/nvdef/hello/NVS?WSDL>”, otherwise set the correct computer name and port in the URL.

Click “OK”. This creates your Hello project. Expand the Hello project and double click on “Request 1”. SoapUI then automatically generates your input soap message:



Change the message by adding your first and last names in the correct places.

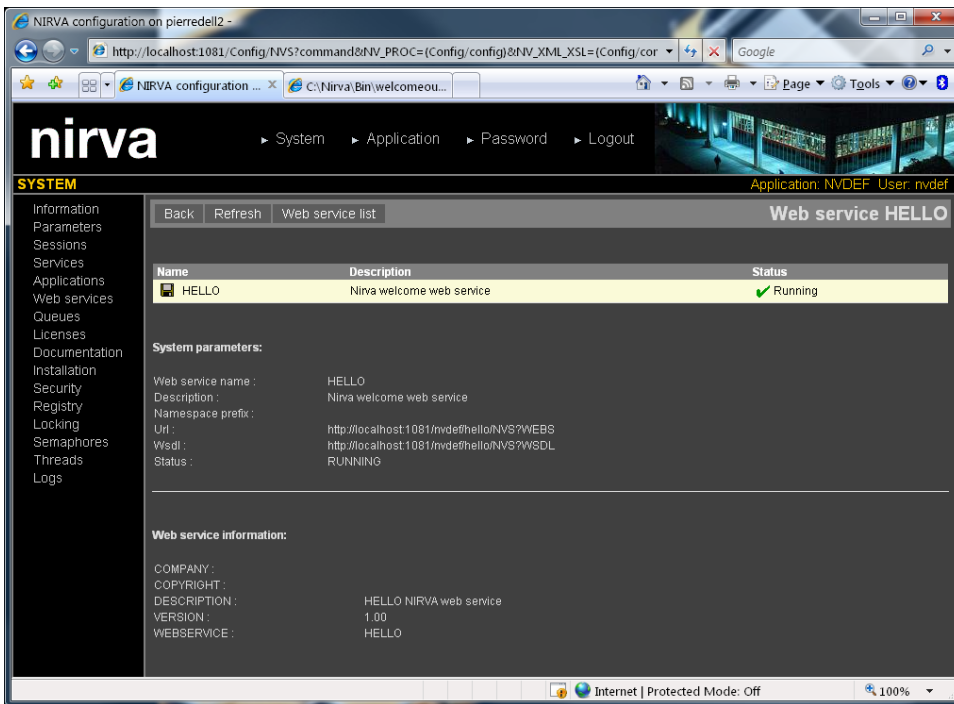
Then press the  button from the Request 1 window. This sends the input message to Nirva and returns back the result message:



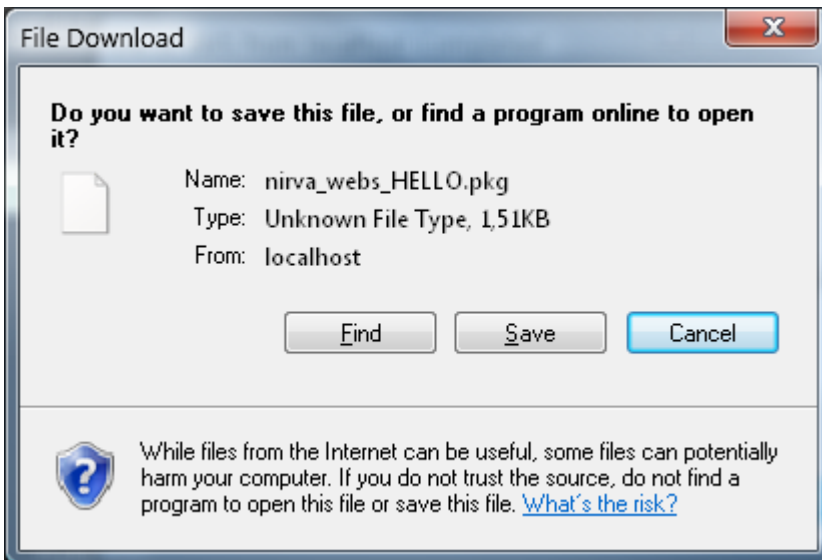
Deploying the web service

Deploying a web service means creating an installation package on one side and installing the application package on another side.

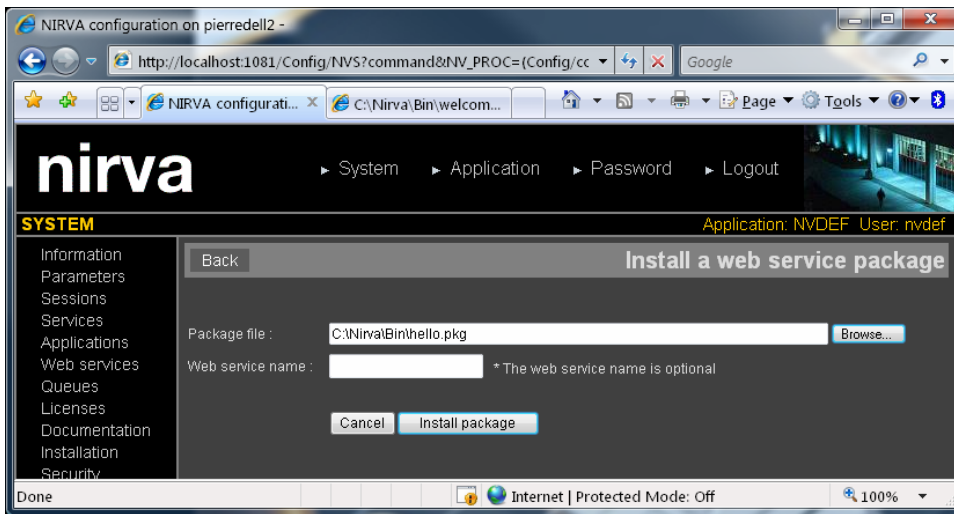
For creating the package file for the web service HELLO, go to the web service list in the configuration tool and click on the "HELLO" web service. This displays the following screen:



just click on the icon near the web service name. Then save your package file:



For installing this web service package on the target NIRVA, use the configuration tool of this target NIRVA, go to the System/Web services menu and press the install button. Then enters the path of your package file an press the “Install package” button:



The web service name is not mandatory since NIRVA is able to get it from the package file itself. Changing the web service name allows to create a copy of a web service.

After a confirmation message, the HELLO web service is installed on the target NIRVA.

Description file

The web service description file is a text file named “webservice.dsc”. The description file must reside in the webservice “Files” directory.

The description file is composed of well defined sections. A new section begins with a new line starting with the '[' character followed by the section name and terminating with the ']' character.

Each section contains a succession of lines with a meaning depending of the section itself.

The description file can include comments. A comment is a line starting with ';', '/' or '\’.

Any blank line is ignored.

Here are the description file available sections:

INFO General web service information.

SETTINGS General settings.

Here is an example of a description file for the service “HELLO”:

```
// hello.dsc : description file
// NIRVA web service
// This file should reside in the NIRVA/Webs/HELLO/Files directory

// This file contains the HELLO NIRVA web service description
// NIRVA tries to read it when loading the web service
// The hello.dsc file should be installed in the web service File director
// This file is not required but is very usefull for NIRVA configuration
```

```
// INFO section
// The INFO section gives some general web service information on the form
// infoname =info value
// Any new string can be added, removed or modified
[INFO]
WEBSERVICE = HELLO
VERSION = 1.00
DESCRIPTION = HELLO NIRVA web service
COMPANY =
COPYRIGHT =

[SETTINGS]
NSPREFIX =
```

INFO section

The INFO section gives some general web service information.

There is a single INFO section in the description file.

The section is composed of several entries of the form *infoname = infovalue*.

The WEBSERVICE entry is the name of the web service. It's used by Nirva as default web service name during installation.

SETTINGS section

The SETTINGS section gives some web service information.

There is a single SETTINGS section in the description file.

The section is composed of several entries of the form *paramname = paramvalue*.

The NSPREFIX section gives the default namespace prefix that will be used in the web service WSDL if not given, Nirva is using the web service host parameter defined at system level (see config/system parameters). This parameter is only used at installation time if the web service doesn't exist. After install the namespace prefix can be changed directly from the configuration tool.