# WCF “Make a Cat” Service Application & Client "Tester" Windows Form Application

## About

This project presents a simple, but fun “Make a Cat” Service Application. The service provides a factory creational design pattern to create and return a single Cat object (complete with a cute photo), or group of five cats in a CatList object (with multiple cats). It is hosted using IIS Express to quickly demo and test the service with a client.

A client “tester” windows form application tests the service and provides output to the user in a simple GUI.

## Architecture

The demo project consists of these component topics:

* Make A Cat WCF Service Application
  + Cat (Data Model Contract)
  + IMakeACatService (Interface for Service)
  + MakeACatService (Code that Implements the Service Interface)
* Client “Tester to Service” Windows Form Application
  + Connected Service “Proxy Reference” MakeACatServiceRef
  + Main Form GUI User Interface
  + Form Code – Processes GUI User Interface

## Make A Cat WCF Service Application

A WCF Service Application project was added to my Visual Studio solution. The code is available on GitHub here [].

### Cat (Data Model Contract)

The *Cat* class is a DataContract with five DataMember representing the name, breed, gender, age, and photo (bitmap) of the individual cat implemented as automatic properties. The Cat class also implements two constructors for the server-side factory creation pattern. The parameterized constructor is used to create a cat based on the client-side user input requested name, breed, gender, and age. A generic photo of that cat breed is set on the newly created cat object. Photos of the cat breeds are stored in the App\_Data.photos project directory on the server and named to match the Breed enum CatBreed. The Cat DataContract also specifies two enumerations: CatBreed (five breeds) and GenderType (two genders). Lastly, the CatList class is defined to specify multiple entities that can be returned and inherits from List<Cat>. The code is available on GitHub here [].

[insert code here]

### IMakeACatService (Interface for Service)

The ServiceContract for the Make A Cat Service allows for two possible options: make single cat entity or make a list of five cat entities (CatList). The first OperationContract will return a single Cat object, representing a factory created Cat object with the name, breed, age, and gender that was requested by the client.

The second OperationContract will return a CatList (List<Cat>) with five factory created Cat objects. The second interface returns five cats with random names selected from each of the five breed categories. This interface and implementation could be further expanded to allow the client to specify the number of random Cats to create, but this was a short demo project to be done quickly in a day. The code is available on GitHub here [].

[insert code here]

### MakeACatService (Code that Implements the Service Interface)

The service implementation code is a factory that accepts parameters and returns a built Cat object to the client. If the client requested multiple cats, it randomizes (name and age only) five Cat objects to create and return to the client. The multiple Cat factory implementation loads a list of possible cat names from a file in the project into a list using reflection. The static class LoadNames() takes care of loading the cat names from the file in the project directory on the server and putting them into memory. The Random number generator is also static on the class level to retain the same seeding that implements Random generator correctly. The code is available on GitHub here [].

[insert code here]

## Client “Tester to Service” Windows Form Application

The client “tester to service” is a simple windows form application project in the same solution that connects to the “Make A Cat Service” by use of a proxy generated by SVCUTIL. The client program will use this proxy to test each OperationContract or method available in the ServiceContract and return the results to the user on the form window.

The user enters in cat details in the form window, then selects “Make A Cat” button. The form will validate the user input and then pass the data to the service to receive a newly created Cat object. If the user clicks the “Make 5 Cats!” button, the form does not need to validate or send any parameters to build the factory list of Cat objects, it just receives the Cat objects.

After the service builds and sends either single entity or five Cat objects back to the client, the windows form GUI will update and add the names of the Cat objects to the list. The user can then select on the individual Cat name and it will populate the details in the entry boxes above and change the photo to the correct breed of cat. Note: The Photo is a bitmap that was stored on the server and transmitted in the DataContract for the Cat Photo property.

Connected Service “Proxy Reference” MakeACatServiceRef

I used the simple “Add Service Reference” wizard to create a Service Reference to an existing service in my Visual Studio solution. After selecting the “Discover” button on the wizard, it was able to first load, instance, and then recognize the Make A Cat service and build the service reference using the exposed meta-exchange data “WSDL”. See the demo section for more information on how to use SVCUTIL without the wizard to generate client “tester” proxies to an existing service. The auto-generated code is available on GitHub here [].

Main Program

The main program in the client “tester” windows application first connects to the Make A Cat Service using a client proxy with classes built from the “Add Service Reference” wizard. Based on this proxy, it then can call upon and retrieves either a single Cat or an enumerable list of Cats (five). The Program handles the user interface events, button clicks, validation, service calls, etc. The code is available on GitHub here [].

[insert code here]

## Demo

## Code

The entire project code repository is available on GitHub here.