Bài 8

MỘT SỐ VÍ DỤ SỬ DỤNG WINDOWS COMMUNICATION FOUNDATION

Mục lục

Các bài trước chúng ta đã biết các thành phần cấu thành một dịch vụ trên WCF cũng như cách xây dựng một dịch vụ WCF, đồng thời cũng đã được giới thiệu cách xây dựng chương trình ứng dụng để sử dụng các dịch vụ WCF. Trong bài cuối cùng này, chúng ta sẽ được làm quen với một số ứng dụng của dịch vụ WCF theo một số cách khác nhau:

- Tạo ứng dụng trên Web sử dụng công nghệ AJAX làm việc với dịch vụ WCF
- Tạo dịch vụ WCF làm việc với REST

1 Tạo dịch vụ WCF hỗ trợ làm việc với AJAX

Để phục vụ cho mục đích giới thiệu việc hỗ trợ của WCF trong làm việc với REST hay JSON, đầu tiên chúng ta tạo ra một dịch vụ WCF có hỗ trợ tốt khi client là trình duyệt với AJAX. Mục tiêu ứng dụng của chúng ta vẫn là ứng dụng quản lý nhân viên như trong các ví dụ ở các bài trước. Tóm tắt lại, ứng dụng quản lý nhân viên đơn giản gồm có các mục sau:

- Thêm mới một nhân viên
- Sửa thông tin một nhân viên
- Xoá một nhân viên
- Lấy thông tin chi tiết của một nhân viên
- Lấy danh sách các nhân viên
- Tìm kiếm nhân viên

1.1 Tạo ứng dụng web để quản lý nhân viên

1. Mở Visual Studio, chọn File->New Web Site, sau cho chọn ASP.NET Web Site
2. Đặt tên Web site là HRManagement

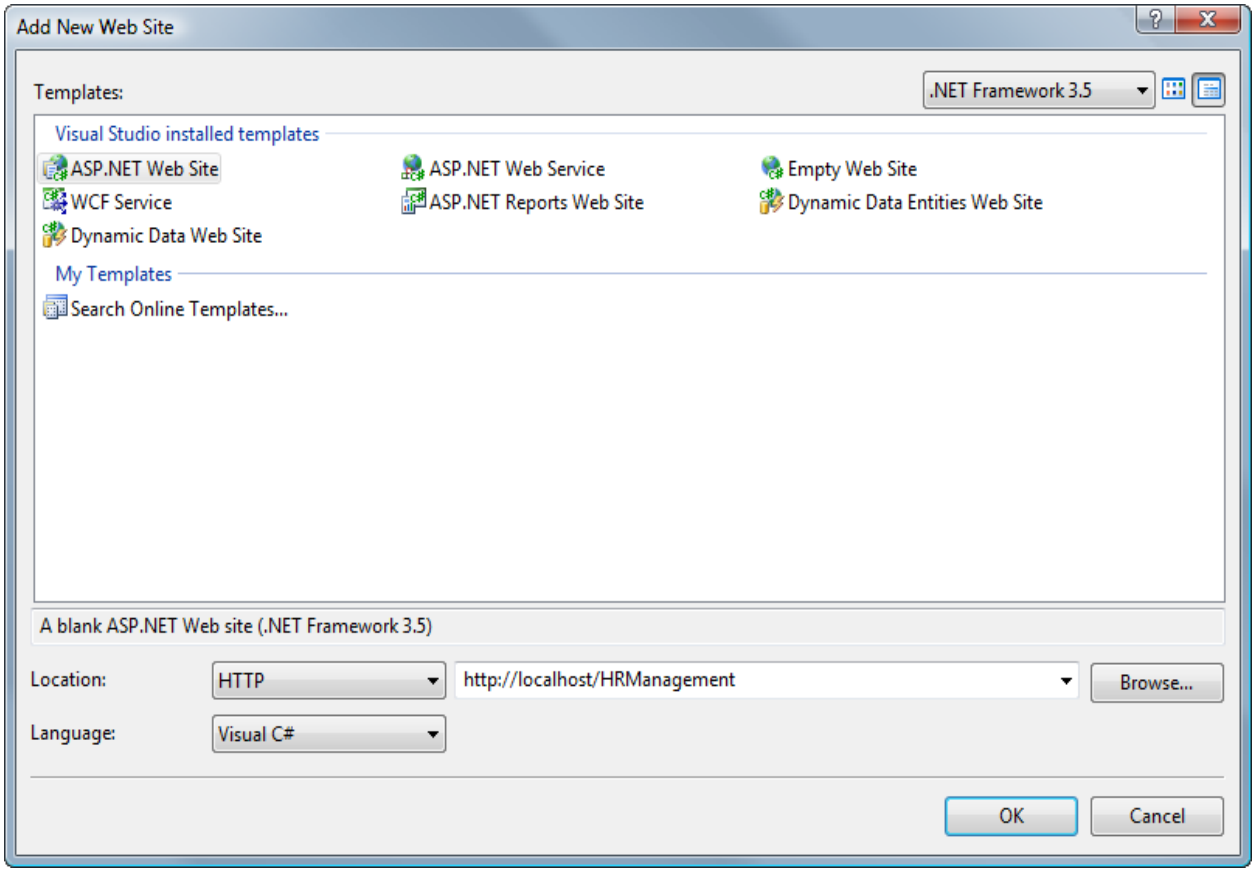

Figure 1 Tạo một web site mới

Do phần client cho web site tương đối dài, nên các bạn xem trong phần source code ví dụ để tiếp tục làm việc cho tiện. Ở đây, phần client sẽ có 3 trang.

- **Trang chủ**: hiển thị toàn bộ nhân viên và các thông tin chi tiết của nhân viên đó
- **Trang soạn thảo**: Sửa thông tin của nhân viên
- **Trang thêm mới**: Thêm một nhân viên

Cả 3 trang này đều sử dụng một trang master gọi là Application.master. Khi mở solution kèm theo bạn sẽ thấy web site project như sau:

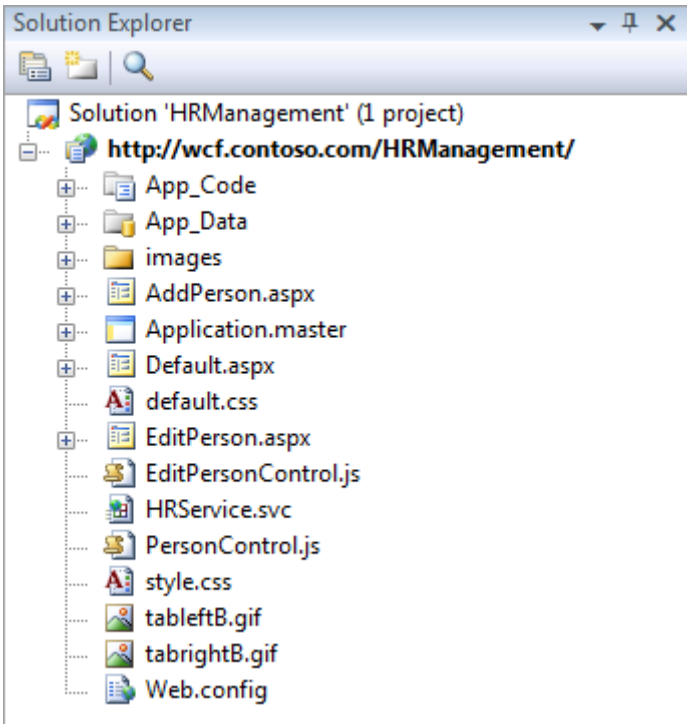

Figure 2 Web site project

1.2 Tạo dịch vụ quản lý nhân viên

Để ứng dụng web của bạn làm việc với các dịch vụ WCF, bạn cần thực hiện các bước sau:

1. Click chuột phải vào HRManagement project, chọn Add New Item
2. Chọn Thêm AJAX-enabled WCF Service
3. Đặt tên cho dịch vụ của bạn là HRService.svc

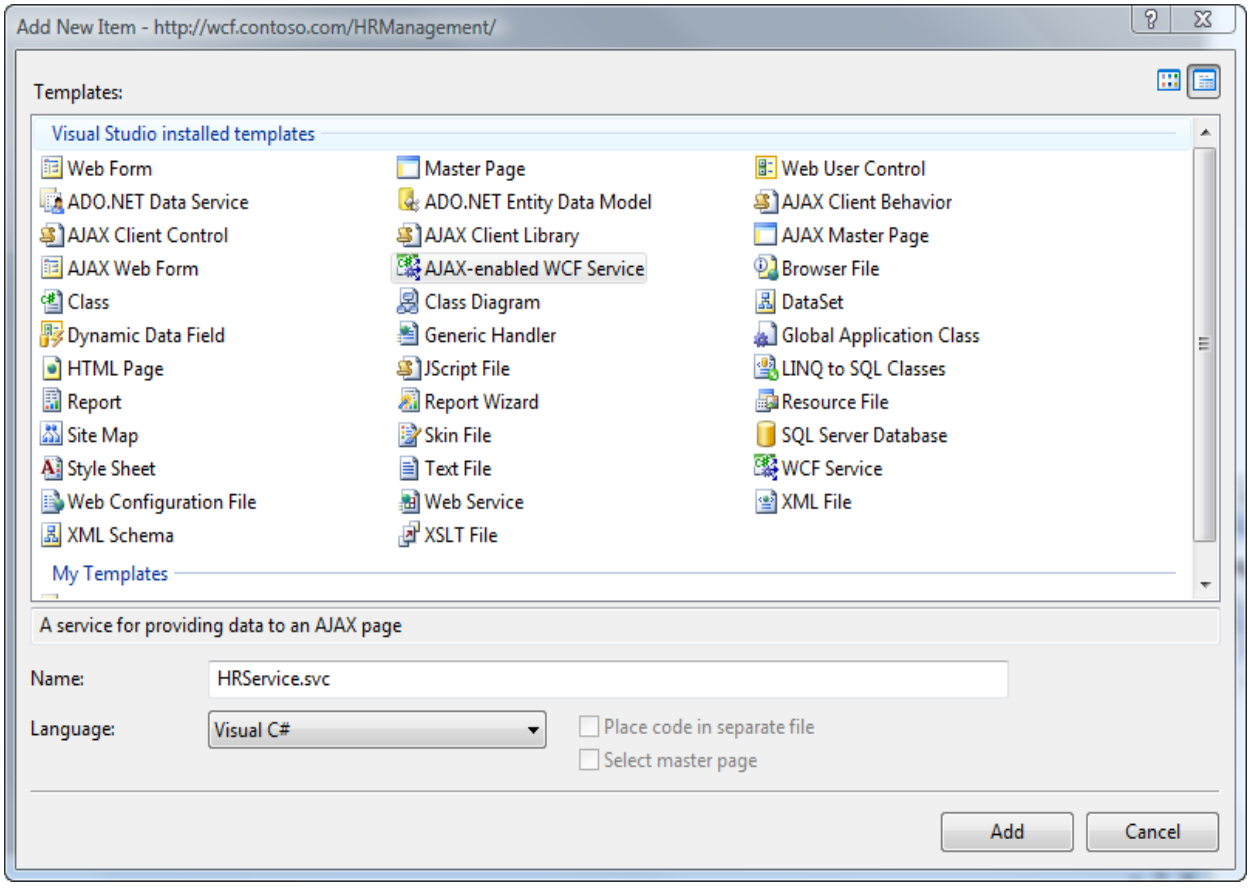

Figure 3 Thêm mới dịch vụ WCF

4. Thêm đoạn mã sau đây để cài đặt dịch vụ

```
        using System.Collections.Generic;
        using System.Linq;
        using System.Runtime.Serialization;
        using System.ServiceModel;
        using System.ServiceModel.Activation;

        [DataContract]
        public class PersonData
        {
            [DataMember]
            public int PersonId;
            [DataMember]
            public string FirstName;
            [DataMember]
            public string LastName;
            [DataMember]
            public string EmailAddress;
            [DataMember]
            public string Department;
        }

        [ServiceContract(Namespace = "urn:hr")]
        [AspNetCompatibilityRequirements(RequirementsMode                    =
AspNetCompatibilityRequirementsMode.Allowed)]
        public class HRService
        {
            [OperationContract]
            public IList<PersonData> GetAll()
```

```
        {
            using (var personCtx = new DataClassesDataContext())
            {
                // Set up the query
                var persons = from p in personCtx.Persons
                              orderby p.FirstName
                              select new PersonData
                              {
                                      PersonId = p.PersonId,
                                      FirstName = p.FirstName,
                                      LastName = p.LastName,
                                      EmailAddress = p.EmailAddress,
                                      Department = p.Department
                              };

                return persons.ToList();
            }
        }

        [OperationContract]
        public PersonData GetPerson(int personId)
        {
            using (var personCtx = new DataClassesDataContext())
            {
                // Set up the query
                var person = (from p in personCtx.Persons
                              where p.PersonId == personId
                              select
                                      new PersonData
                                      {
                                              PersonId = p.PersonId,
                                              FirstName = p.FirstName,
                                              LastName = p.LastName,
                                              EmailAddress                =
p.EmailAddress,
                                              Department = p.Department
                                      }).Single();

                return person;
            }
        }

        [OperationContract]
        public void UpdatePerson(int personId, PersonData newData)
        {
            using (var personCtx = new DataClassesDataContext())
            {
                // Set up the query
                var  person  =  personCtx.Persons.Single(p  =>  p.PersonId  ==
personId);
                if (person != null)
                {
                    person.FirstName = newData.FirstName;
                    person.LastName = newData.LastName;
                    person.EmailAddress = newData.EmailAddress;
                    person.Department = newData.Department;
                    personCtx.SubmitChanges();
                }
            }
```

```
        }

        [OperationContract]
        public void AddPerson(PersonData p)
        {
            using (var personCtx = new DataClassesDataContext())
            {
                var person = new Person
                {
                    PersonId = p.PersonId,
                    FirstName = p.FirstName,
                    LastName = p.LastName,
                    EmailAddress = p.EmailAddress,
                    Department = p.Department
                };

                personCtx.Persons.InsertOnSubmit(person);
                personCtx.SubmitChanges();
            }
        }

        [OperationContract]
        public void DeletePerson(int personId)
        {
            using (var personCtx = new DataClassesDataContext())
            {
                var  person  =  personCtx.Persons.Single(p  =>  p.PersonId  ==
personId);
                if (person != null)
                {
                    personCtx.Persons.DeleteOnSubmit(person);
                    personCtx.SubmitChanges();
                }
            }
        }
    }
```

5. Trong đoạn mã trên có 2 điểm cần chú ý:

   a. Giống như các dịch vụ khác, khi định nghĩa kiểu dữ liệu của chúng ta, chúng ta cần phải thêm vào data contract (trong trường hợp này là lớp `PersonData` được thêm thuộc tính `DataContract`)

   b. Khi làm việc với AJAX, ta nên đăng ký namespace của dịch vụ, sử dụng tham số `Namespace` trong thuộc tính `ServiceContract`. Trong ví dụ này là `[ServiceContract(Namespace = "urn:hr")]`

1.3 Thiết lập cấu hình để hỗ trợ AJAX

Trong thực tế, khi bạn tạo ra một project ASP.NET Web Site và thêm vào dịch vụ WCF như trên, Visual Studio sẽ tự động thêm vào các cấu hình cần thiết, bạn sẽ không cần làm gì mà đã có thể sử dụng dịch vụ. Để chắc chắn, hay kiểm tra tệp tin cấu hình xem có giống như sau không:

```
    <system.serviceModel>
        <behaviors>
          <endpointBehaviors>
```

```
                <behavior name="HRServiceAspNetAjaxBehavior">
                    <enableWebScript/>
                </behavior>
            </endpointBehaviors>
        </behaviors>
        <serviceHostingEnvironment aspNetCompatibilityEnabled="true"/>
        <services>
            <service name="HRService">
                <endpoint   address=""
                     behaviorConfiguration="HRServiceAspNetAjaxBehavior"
                     binding="webHttpBinding" contract="HRService"/>
            </service>
        </services>
    </system.serviceModel>
```

1.4 Sử dụng các phương thức của dịch vụ WCF bằng AJAX

Để sử dụng các phương thức của dịch vụ WCF, đầu tiên bạn cần thêm vào một `ScriptManager` trong trang aspx của bạn, sau đó thêm vào `Services` và trỏ tới dịch vụ bạn muốn sử dụng, ví dụ trong trang default.aspx, bạn thêm vào ngay sau thẻ `form` như sau:

```
    <asp:ScriptManager ID="ScriptManager1" runat="server">
        <Services>
            <asp:ServiceReference Path="~/HRService.svc" />
        </Services>
    </asp:ScriptManager>
```

Sau khi thêm vào tham chiếu dịch vụ WCF, bạn hoàn toàn có thể gọi các phương thức của dịch vụ như sau:

```
function GetData() {
    var hrs = new hr.HRService();
    hrs.GetAll(PersonsReturnedEventHandler, null, null);
}
function PersonsReturnedEventHandler(value) {
    alert("Tong cong co " + value.length, " nhan vien");
}
```

Như vậy cách sử dụng dịch vụ WCF là:

- Tạo một instace (thể hiện) của dịch vụ: `var hrs = new hr.HRService();`
- Gọi hàm của dịch vụ đó: `hrs.GetAll(...)`

Như vậy chúng ta đã có một ứng dụng web tương đối hoàn chỉnh để quản lý nhân viên.

Trang chủ mặc định:

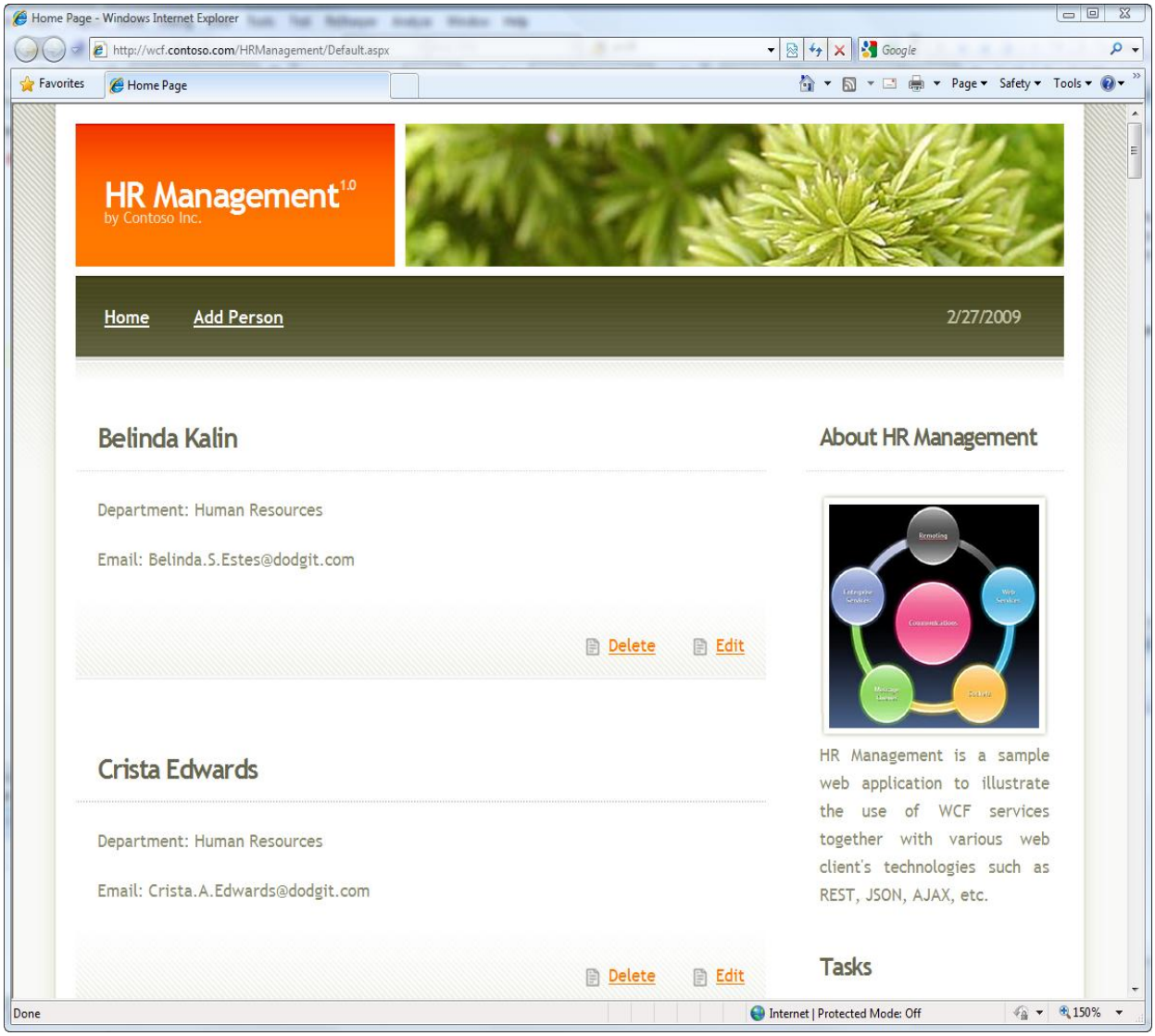

Figure 4 Trang chủ

Xoá nhân viên

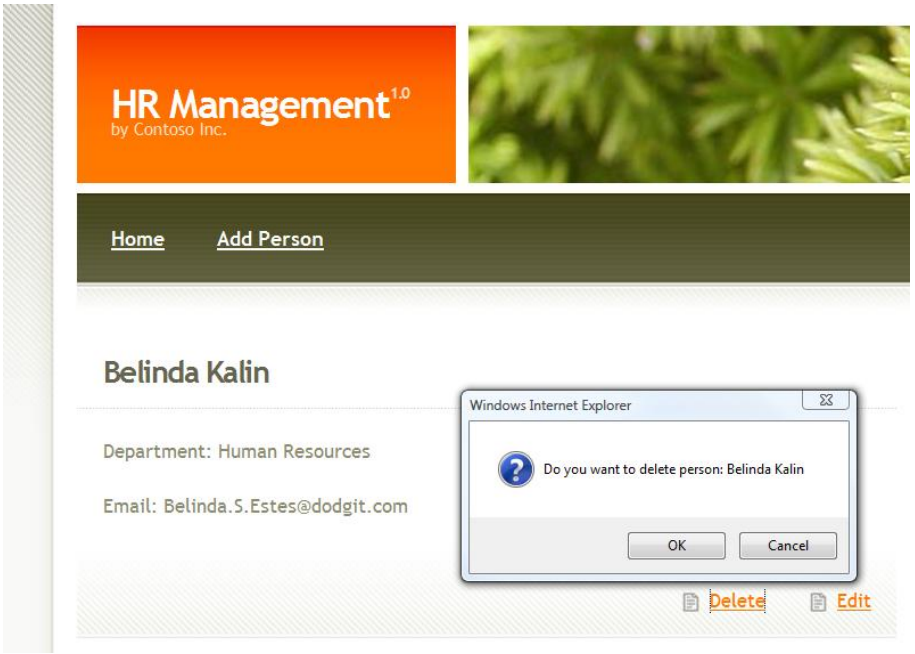

Figure 5 Xoá nhân viên

Sửa thông tin

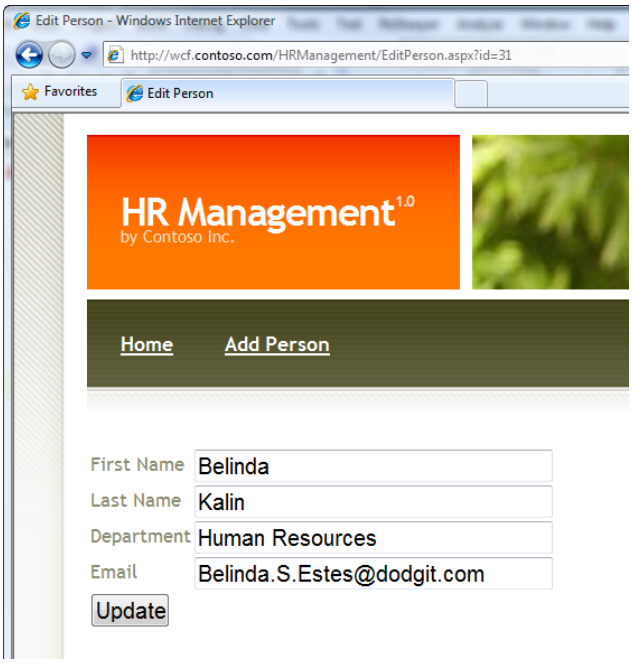

Figure 6 Sửa thông tin về một nhân viên

2 Tạo dịch vụ WCF làm việc với REST

Dựa trên cơ sở project chúng ta đã xây dựng ở trên. Thêm vào một dịch vụ WCF như sau:

1. Click chuột phải vào project HRManagement, chọn Add New Item, sau đó chọn thêm vào WCF Service
2. Đặt tên cho service là RESTHRService.svc

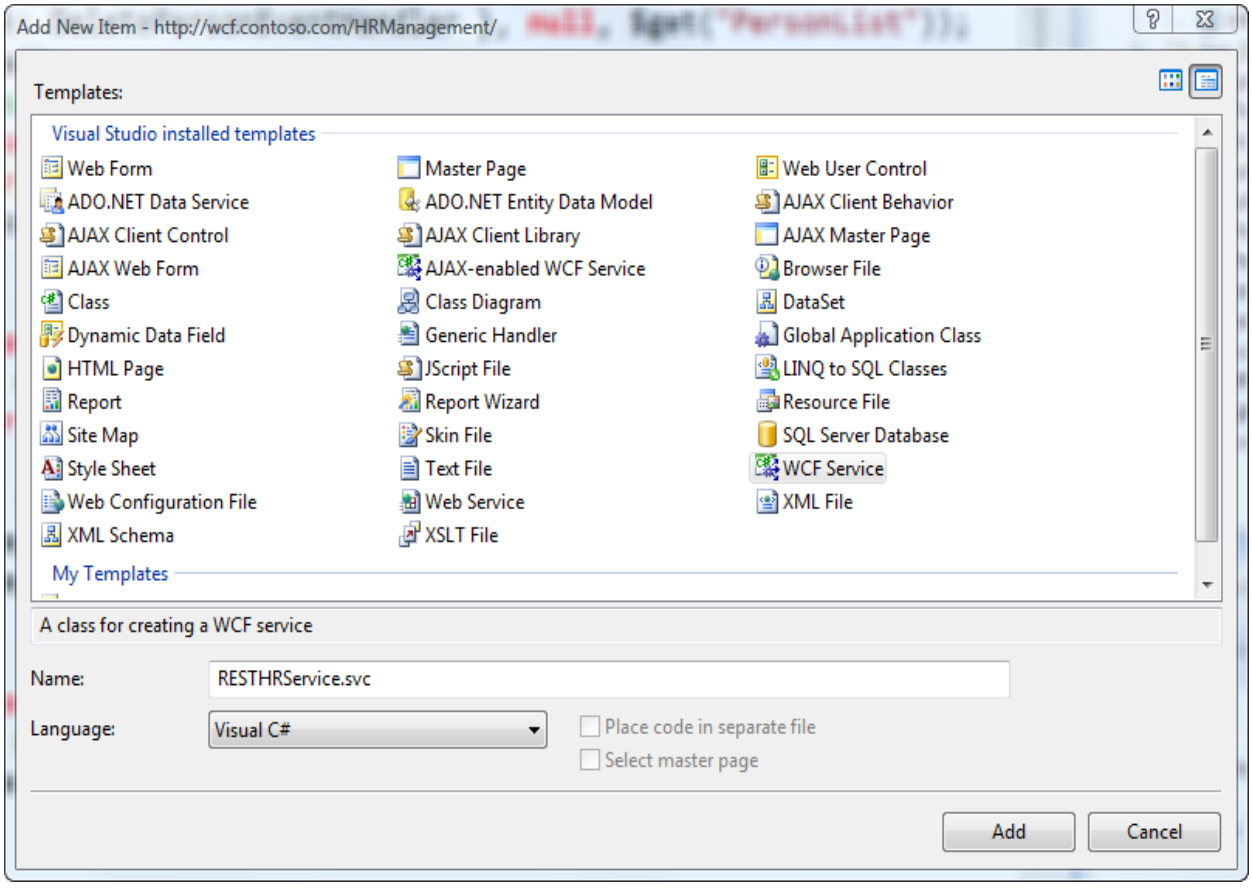

Figure 7 Thêm dịch vụ REST WCF

3. Sửa tập tin cấu hình như sau:

```
<system.serviceModel>
  <behaviors>
    <endpointBehaviors>
      <behavior name="AJAXFriendly">
        <enableWebScript />
      </behavior>
      <behavior name="RESTFriendly">
        <webHttp />
      </behavior>
    </endpointBehaviors>
  </behaviors>
  <serviceHostingEnvironment aspNetCompatibilityEnabled="true"/>
  <services>
    <service name="HRService">
      <endpoint   address=""
                  behaviorConfiguration="AJAXFriendly"
                  binding="webHttpBinding"
                  contract="HRService" />
    </service>
    <service name="RESTHRService">
      <endpoint   address=""
                  behaviorConfiguration="RESTFriendly"
                  binding="webHttpBinding"
                  contract="IRESTHRService" />
    </service>
  </services>
</system.serviceModel>
```

4. Chúng ta vẫn sử dụng DataContract là `PersonData`. Thêm vào khai báo dịch vụ `IRESTHRService` như sau:

```
[ServiceContract]
public interface IRESTHRService
{
    [OperationContract]
    IList<PersonData> GetAll();

    [OperationContract]
    PersonData GetPerson(int personId);

    [OperationContract]
    void UpdatePerson(int personId, PersonData newData);

    [OperationContract]
    void AddPerson(PersonData p);

    [OperationContract]
    void DeletePerson(int personId);
}
```

2.1 Xây dựng URI Template cho việc lấy dữ liệu (HTTP GET)

Để thực hiện dịch vụ thông qua REST chúng ta cần đưa vào các verb tương ứng trong REST như HTTP GET, POST, PUT, DELETE. Với verb HTTP GET chúng ta có thể đưa vào bằng cách thêm thuộc tính WebGet cho các hàm trong dịch vụ, ví dụ như:

```
[OperationContract]
[WebGet]
PersonData GetPerson(int personId);
```

Giá trị trả về chúng ta có thể quy định là dạng XML hoặc cũng có thể là JSON. Ở đây ta quy định giá trị trả về là dưới dạng XML như sau:

```
[OperationContract]
[WebGet(ResponseFormat = WebMessageFormat.Xml)]
PersonData GetPerson(int personId);
```

Ngoài ra khi trả về giá trị, dịch vụ của chúng ta cũng cần phải báo cho client biết là request có thành công hay không. Việc trả về trạng thái lỗi được thực hiện thông qua `WebOperationContext.Current`. Như vậy, phần cài đặt cho hàm GetPerson sẽ như sau:

```
public PersonData GetPerson(int personId)
{
    var ctx = WebOperationContext.Current;
    try
    {
        using (var personCtx = new DataClassesDataContext())
        {
            // Set up the query
            var person = personCtx.Persons.SingleOrDefault(p => p.PersonId ==
personId);

            if (person == null)
            {
                ctx.OutgoingResponse.SetStatusAsNotFound();
                return null;
```

```
                }
                var personData = new PersonData
                                        {
                                            PersonId = person.PersonId,
                                            FirstName = person.FirstName,
                                            LastName = person.LastName,
                                            EmailAddress = person.EmailAddress,
                                            Department = person.Department
                                        };

                ctx.OutgoingResponse.StatusCode = System.Net.HttpStatusCode.OK;
                return personData;
            }
        }
        catch
        {
            ctx.OutgoingResponse.StatusCode =
System.Net.HttpStatusCode.BadRequest;
            return null;
        }
    }
```

Một trong những đặc điểm cơ bản của REST là chúng ta chỉ làm việc với các URI. Do đó để dịch vụ của chúng ta theo đúng kiểu của REST, ta cần thêm tham số UriTemplate cho thuộc tính WebGet như sau:

```
    [OperationContract]
    [WebGet(UriTemplate = "person/{personId}", ResponseFormat =
WebMessageFormat.Xml)]
    PersonData GetPerson(int personId);
```

Khi đó giả sử dịch vụ của chúng ta ở tại địa chỉ sau:

`http://wcf.contoso.com/HRManagement/RESTHRService.svc`

Hàm GetPerson sẽ được kích hoạt với personId=31 tại địa chỉ sau:

`http://wcf.contoso.com/HRManagement/RESTHRService.svc/person/31`

Mở trình duyệt Internet Explorer vào địa chỉ trên ta có thể nhận được kết quả như sau:

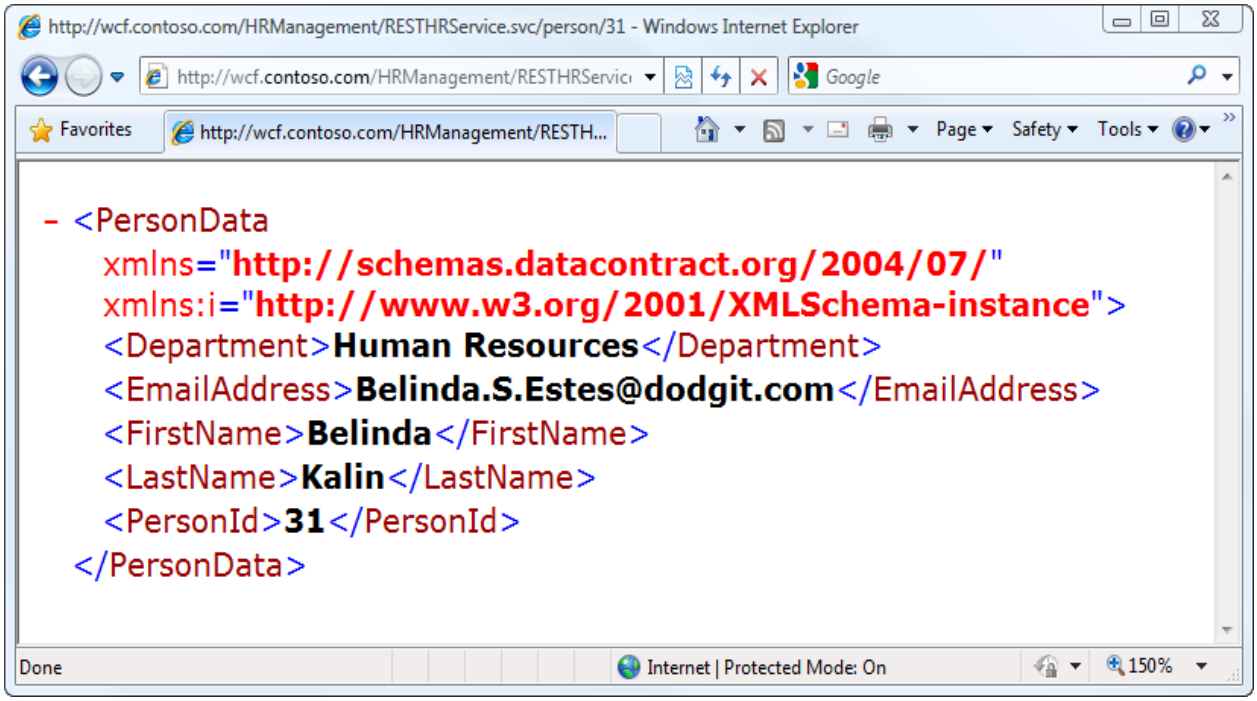

Figure 8 Kết quả lấy thông tin một nhân viên

Trong trường hợp bạn sử dụng định dạng trả về là Json như sau:

```
        [OperationContract]
        [WebGet(UriTemplate = "person/{personId}", ResponseFormat =
WebMessageFormat.Json)]
        PersonData GetPerson(int personId);
```

Khi đó truy xuất tới địa chỉ

`http://wcf.contoso.com/HRManagement/RESTHRService.svc/person/31`

sẽ cho ta kết quả như sau:

```
        {"Department":"Human
Resources","EmailAddress":"Belinda.S.Estes@dodgit.com","FirstName":"Belinda",
"LastName":"Kalin","PersonId":31}
```

2.2 Xây dựng URI Template cho việc cập nhật dữ liệu (HTTP PUT)

Việc cập nhật dữ liệu bao gồm sửa một bản ghi (thông tin nhân viên) sẵn có hoặc thêm mới một nhân viên. Việc này được thực hiện dựa vào verb HTTP PUT trong REST. Để mô tả một phương thức trong dịch vụ WCF sẽ được kích hoạt bằng verb PUT, ta sử dụng thuộc tính mô tả là `WebInvoke`, với tham số `Method` là `PUT`. Để chỉ ra ta sẽ cập nhật thông tin vào nhân viên nào, ta cũng sử dụng URI Template như phần trên. Như vậy, hàm cập nhật (hoặc thêm mới) nhân viên sẽ được khai báo như sau:

```
            [OperationContract]
            [WebInvoke(Method = "PUT",
                    UriTemplate = "person/{personId}",
                    RequestFormat = WebMessageFormat.Json)]
            void UpdatePerson(string personId, PersonData personData);
```

Mã nguồn cài đặt chi tiết cho hàm này như sau:

```
        public void UpdatePerson(string personId, PersonData personData)
```

```
        {
            WebOperationContext ctx = WebOperationContext.Current;
            System.Net.HttpStatusCode status = System.Net.HttpStatusCode.OK;

            try
            {
                using (var dataContext = new DataClassesDataContext())
                {
                    Person person = dataContext.Persons.SingleOrDefault(
                        prod => prod.PersonId == Convert.ToInt32(personId));

                    if (person == null)
                    {
                        person = new Person();
                        dataContext.Persons.InsertOnSubmit(person);
                        status = System.Net.HttpStatusCode.Created;
                    }

                    person.FirstName = personData.FirstName;
                    person.LastName = personData.LastName;
                    person.Department = personData.Department;
                    person.EmailAddress = personData.EmailAddress;

                    dataContext.SubmitChanges();
                    ctx.OutgoingResponse.StatusCode = status;
                    return;
                }
            }
            catch
            {
                ctx.OutgoingResponse.StatusCode =
System.Net.HttpStatusCode.BadRequest;
                return;
            }
        }
```

Trên trang web ta gọi hàm dịch vụ WCF như sau:

```
        function updatePersonEventHandler(sender, args) {
          var url = "../RESTHRService.svc/person/" + args.PersonId;
          var proxy = new Sys.Net.WebServiceProxy();
          proxy.restInvoke(url,
                 "PUT",
                 args,
                 "GetData",
                 updatePersonSucceeded,
                 null,
                 null);
        }
```

2.3 Xây dựng URI Template để xoá một nhân viên (HTTP DELETE)

Để ra lệnh xoá một resource (trong trường hợp này là nhân viên) bằng REST, chúng ta sử dụng verb HTTP `DELETE`. Tương tự như phần cập nhật, chúng ta khai báo phương thức với thuộc tính mô tả là `WebInvoke` với `Method` là `DELETE`:

```
            [OperationContract]
            [WebInvoke(Method = "DELETE", UriTemplate = "person/{personId}")]
```

```
        void DeletePerson(string personId);
```

Và mã nguồn cài đặt cho việc xoá nhân viên:

```
    public void DeletePerson(string personId)
    {
        WebOperationContext ctx = WebOperationContext.Current;

        try
        {
            using (var dataContext = new DataClassesDataContext())
            {
                Person person = dataContext.Persons.SingleOrDefault(
                 p => p.PersonId == Convert.ToInt32(personId));
                if (person == null)
                {
                    ctx.OutgoingResponse.StatusCode =
System.Net.HttpStatusCode.NotFound;
                    return;
                }

                dataContext.Persons.DeleteOnSubmit(person);
                dataContext.SubmitChanges();

                ctx.OutgoingResponse.StatusCode = System.Net.HttpStatusCode.OK;
                return;
            }
        }
        catch
        {
            ctx.OutgoingResponse.StatusCode =
System.Net.HttpStatusCode.BadRequest;
            return;
        }
    }
```

Phía client gọi hàm DeletePerson như sau:

```
    function DeletePersonEventHandler(sender, args) {
       if (confirm("Do you want to delete person: " + args.FirstName + " " +
args.LastName)) {
            var url = "../RESTHRService.svc/person/" + args.PersonId;
            var proxy = new Sys.Net.WebServiceProxy();
            proxy.restInvoke(url,
              "DELETE",
              null,
              "DeletePersonEventHandler", OnDeleted, null, null);
       }
    }
```

3 Tài liệu tham khảo

1. REST in WCF Blog Series Index (URL: http://blogs.msdn.com/bags/archive/2008/08/05/rest-in-wcf-blog-series-index.aspx)
2. Using WebHttpBinding & JSON Support in WCF (URL: http://spellcoder.com/blogs/bashmohandes/archive/2008/01/05/9423.aspx)
3. Consuming JSON data in .NET with WCF (URL: http://www.codeproject.com/KB/WCF/consuming-json-wcf.aspx)