RESTclient for WCF

Enabling REST-Client Automatization (ERA for WCF)

**Documentation**

**(Version 1.0 wip)**

# Index

# Introduction and Features 3

# The REST Starter Kit Preview 2 4

# Install WCF REST Starter Kit Preview 2 7

# Create a new Singleton Service 8

# Create a new Collection Service 11

# How to use the RESTclient for WCFProgram 14

* Code 2 WADL 14
* WADL to Code 16

# The Web Application Description Language (WADL) 18

WADL Description (overview) and supported Features (of the RESTclient for WCF)

# Links 26

# Introduction and Features:

The goal of the REST Client for WCF is, to enable the possibility to create WADL from compiled Program Code or compile Program Code from a WADL, using the REST environment with Windows Communication Foundation (WCF) and .NET.

Features:

* Console Application based Program
* Generate a WADL from compiled Program Code or Code from a WADL
* Language independent approach by using CodeDOM

# The REST Starter Kit Preview 2

The REST Starter Kit contains .NET Frameworks, Templates and new Visual Studio Features which help users to access and create Windows Communication Foundation (WCF) services with REST in .Net 3.5 SP1.

The RSK are divided in three Parts:

* Server Features
* Client Features
* Templates

## Server features (overview):

The REST Starter Kit Server side features allow developing service-oriented applications by using the REST architecture.

### Help Page:

The Help Page is an automatic generated Help Page which is accessible by append */help* to your service.

### Representation Formats:

The Representation Formats supports the representation XML and JSON by specifying a HTTP request like GET with text/xml or application/json, you can choose the representation of the return value in JSON or XML.

### Security:

The REST Starter Kit contains several key capabilities to enable creation of secure services.

### Error Handling:

The Error Handler allows returning exceptions in a string or any serializable .Net type.

### Hosting Settings:

The Starter Kit is optimized to use basic functions without any configuration by user. The user can however chance most of the properties like maximum message size, XML reader quotas, throttling settings, transfer mode (buffered or streaming), principal permission mode, request interceptors and so forth by using the WebServiceHost2 class.

## Client features (overview):

The REST Starter Kit Client features support the user to access REST services by using HTTP requests.

### HTTP Client:

The HTTP Client is a class that allows accessing REST Services programmatically by sending and receiving HTTP response and requests. The HTTP Client can be used to access WCF REST services built using the starter kit, or any third-party REST service.

The HTTP Client contains following features:

#### Rich format support:

XML, JSON, serialization and DOM- based approaches are supported.

#### Extensibility:

The client contains two client implementations. The *PollingAgent* class issues GET requests against an URI and response with an event, when the resource of the URI chances. The *AtmoPubClient* Class is designed to consume *AtomPub* services easier.

#### Uniform error handling:

The Uniform error handling is an error handling model, which inspect the response status an throws exceptions for unexpected status code.

#### Typed header model:

A model, which allows the use of custom headers.

#### Async programming model:

The async model exposes the Begin/End and the event-based.

#### A pipeline/stage model:

Support HTTP/REST request /response outside the application logic

## Templates (overview):

The REST Starter Kit provides five templates to support developers to create RESTful Services using WCF.

### Atom Feed WCF Service:

The Atom Feed WCF Service template allows the user to create a service in a collection conform to the Atom Syndication Format.

### REST Singleton WCF Service:

With the REST Singleton WCF Service template the user can create a service which represents a single instance of recourse and can be implemented in XML or JSON.

### REST Collection WCF Service:

The REST Collection WCF Service template provides the user with a collection of resources exposed as XML or JSON.

### Atom Publishing Protocol WCF Service:

With the Atom Publishing Protocol WCF Service template, the user can create a service, that manages a collection of recourses using the Atom Publishing Protocol.

# Install WCF REST Starter Kit Preview 2

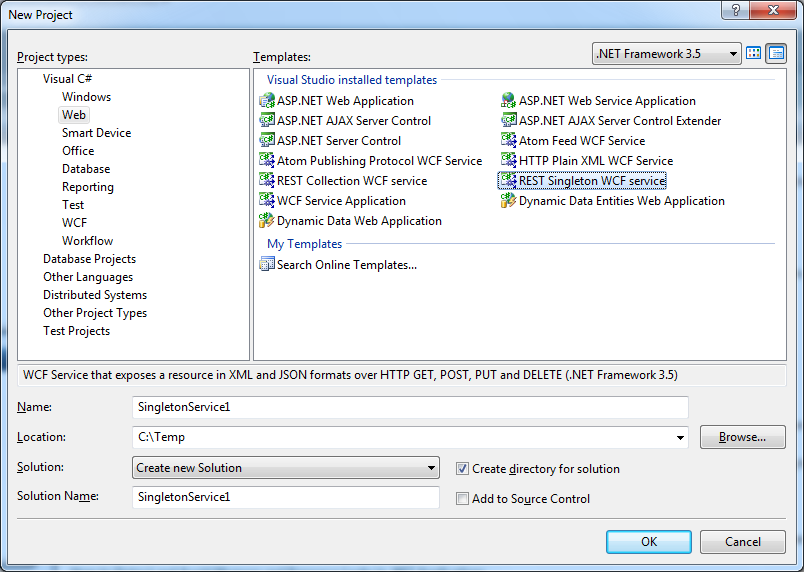
1. First you need to download the WCF REST Starter Kit Preview 2 from <http://aspnet.codeplex.com/Release/ProjectReleases.aspx?ReleaseId=24644#DownloadId=61777>
2. Installation of the WCF REST Starter Kit:
   1. To use the WCF REST Starter Kit Preview 2 you need the following System Requirements:
      1. Windows Server 2003, Windows Server 2008, Windows 7, Windows Vista, Windows XP
      2. Visual Studio 2008 SP1
      3. Important information for user of Visual Studio 2008 SP1 Express Editions: The Visual Studio 2008 SP1 Express Editions may not support some features.
   2. Run the *WCF Rest Starterkit Preview 2.msi* file, downloaded at step 1.
   3. Follow the installation Instructions :



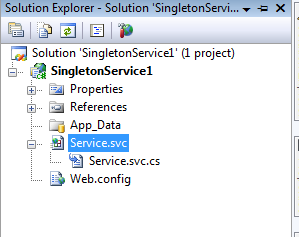
* 1. If you receive a "User Account Control" warning window, please select the option to allow the setup process to continue.

# Create a new Singleton Service:

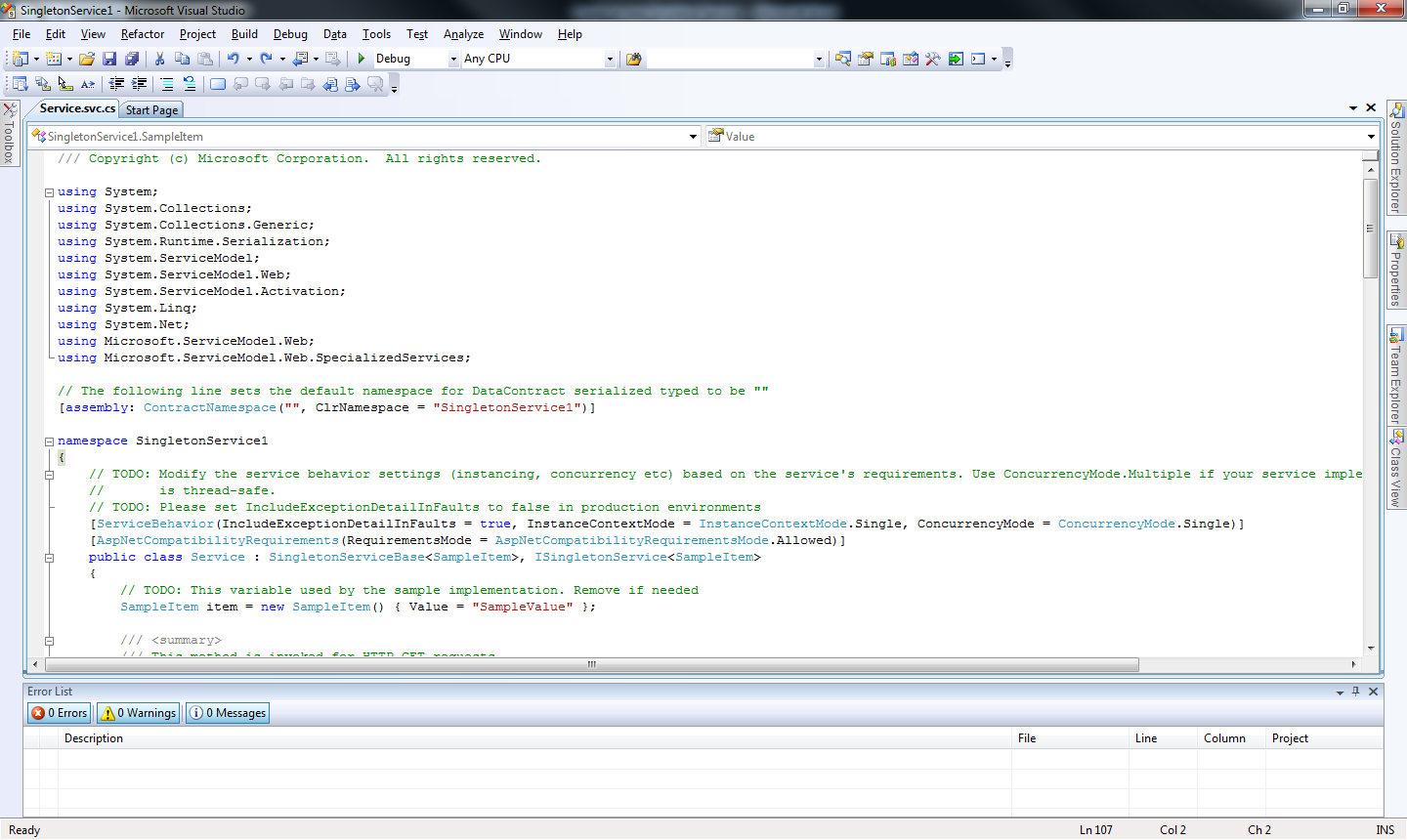
1. Run Visual Studio 2008:
   1. Navigate the Tab: ” File > New > Project”
   2. In” Project Types > Visual C# > Web” you will find the REST Singleton WCF Service



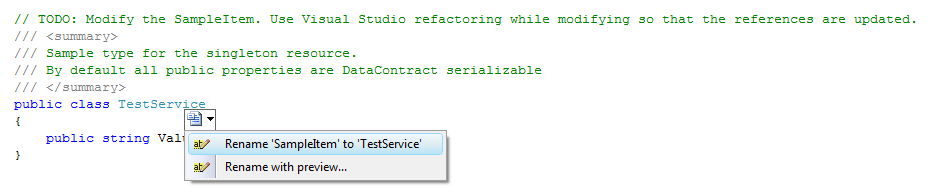
1. Name your Project and your Solution and choose your save Location
2. After clicking OK the Singleton Template will be generated

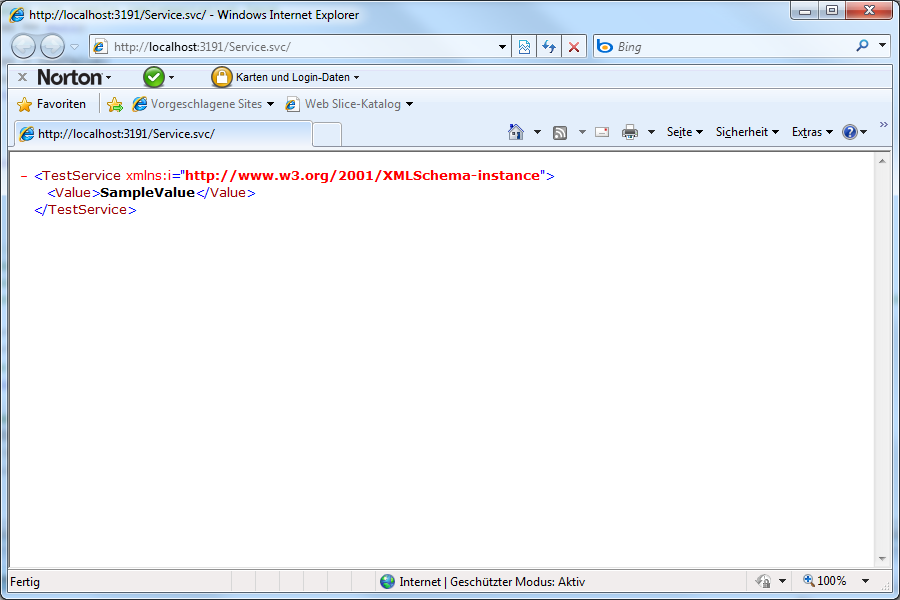
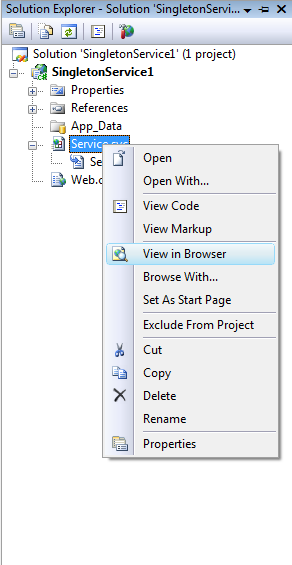


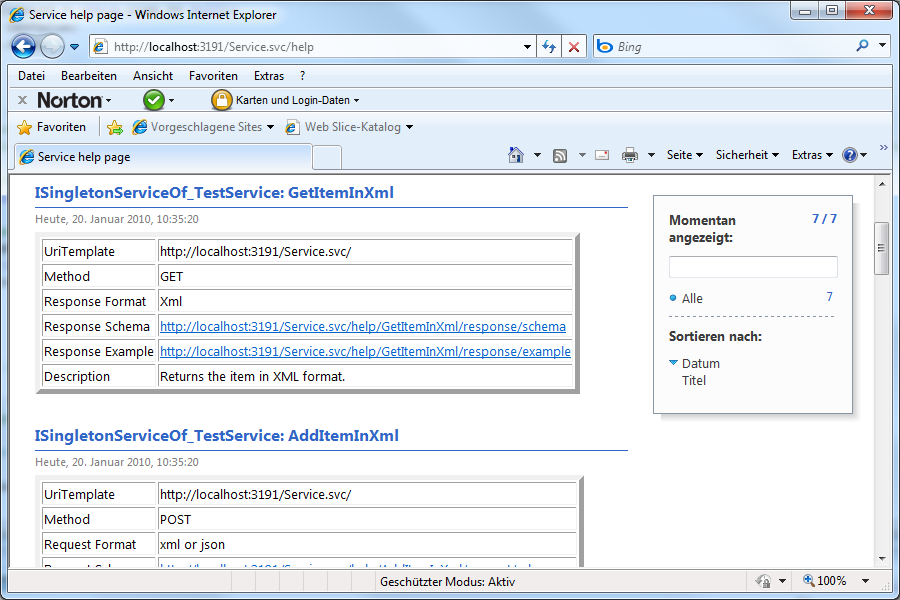
1. In the Solution Explorer the you will find the *Service.svc.cs* which contains the code for the service



1. The Template will provide you with the auto generated Infrastructure of the Singleton Service including the functions *OnGetItem, OnAdditem, OnDeleteitem* and *OnUpdateItem*.
2. At the End of the auto generated code you will find the *SampleItem* class which contains the content of the Singleton Service. Rename this class to a Classname of your choice and apply the renaming for the whole project (refactoring).



1. The Singleton Service is now created and can be viewed in a Browser.

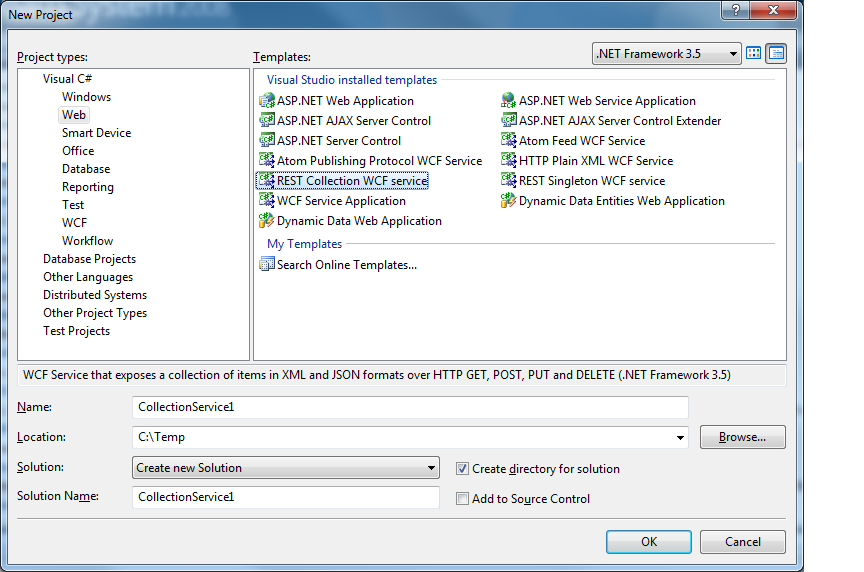
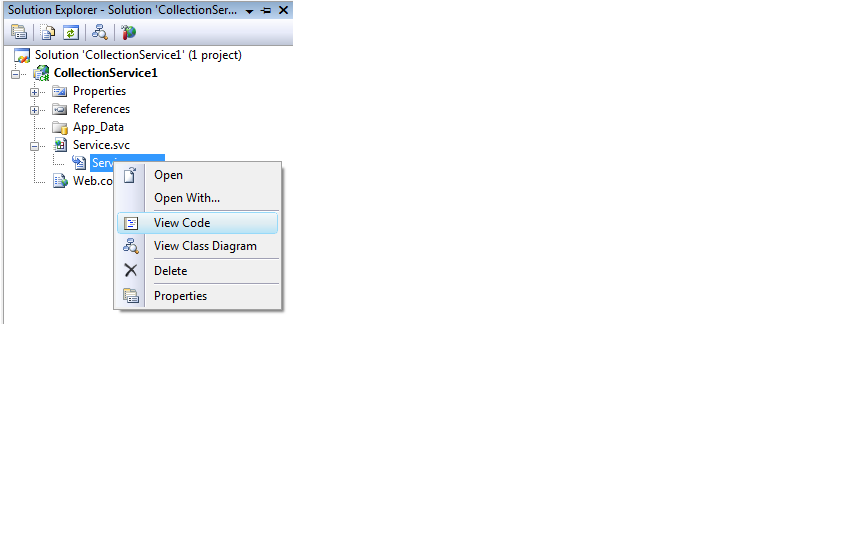
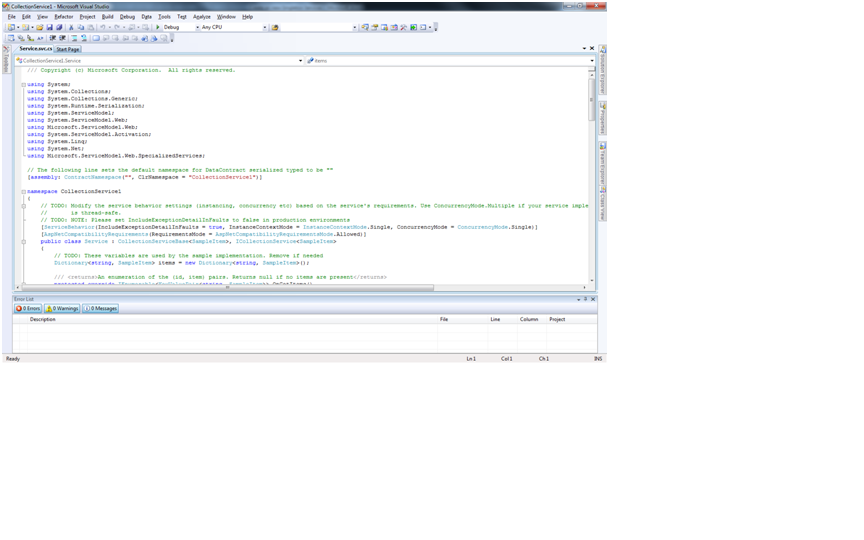


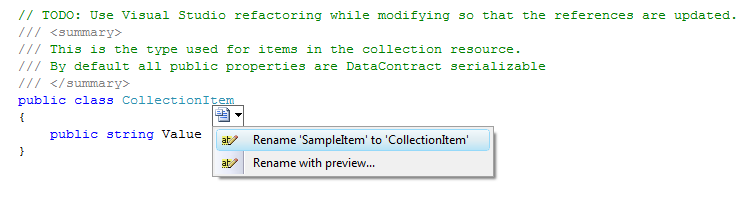
1. The Service template contains a Helppage, which contains Information’s

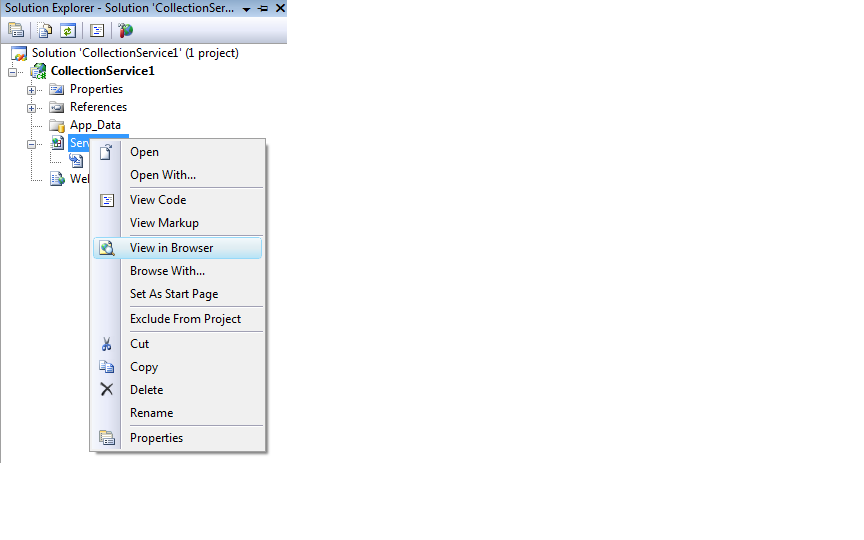
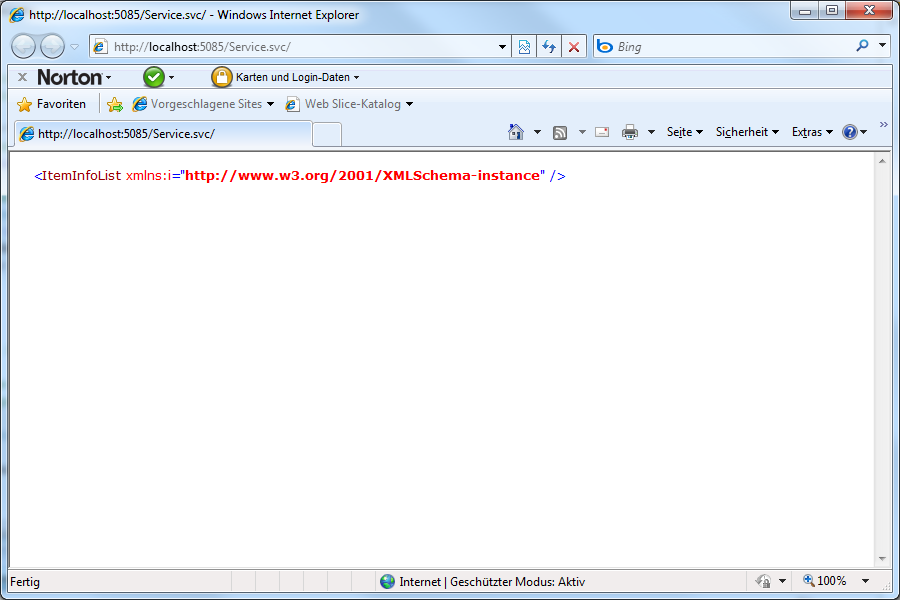
and examples about the xml schema and JSON for the different REST operations.

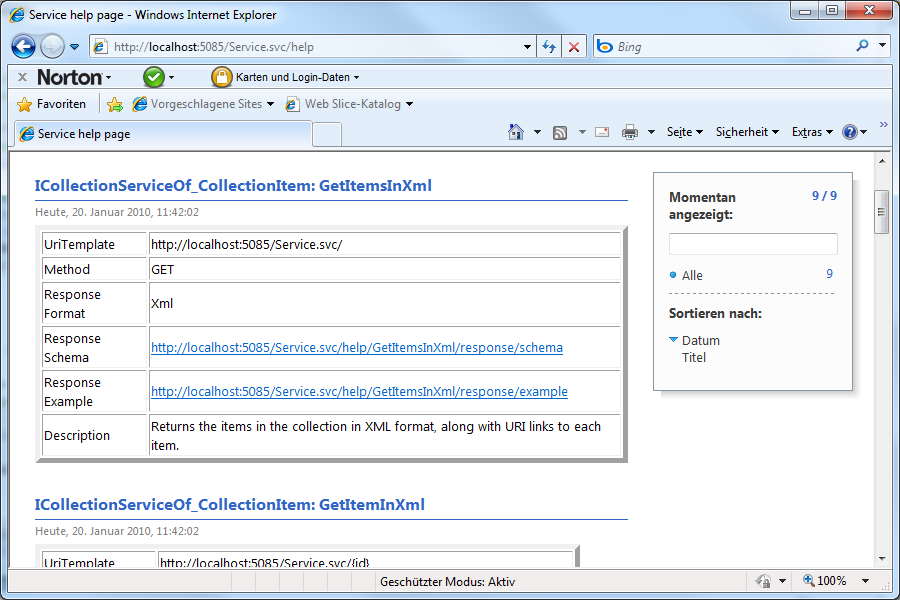
1. At <http://channel9.msdn.com/shows/Endpoint/endpointtv-Screencast-Building-resource-singleton-services-with-the-WCF-REST-Starter-Kit/> you will find a Screencast from Pluralsight, which show you how to create a Singleton Service with Visual Studio and the REST Starter Kit

# Create a new Collection Service:

1. Run Visual Studio 2008
   1. Navigate the Tab “File > New > Projects”
   2. In” Project Types > Visual C# > Web” you will find the REST Collection WCF Service
2. Name your Project and your Solution and choose a save Location
3. After clicking OK the Collection Template will be generated
4. In the Solution Explorer you will find the *Service.svc.cs* which contains the code of the Collection Service Template
5.  The Template will provide you with the auto generated Infrastructure of the Collection Service including the functions *OnGetItem, OnAdditem, OnDeleteitem* and *OnUpdateItem*.
6. At the End of the auto generated code you will find the *SampleItem* class which contains the content of the Collection Service. Rename this class to a Classname of your choice and apply the renaming for the whole project (refactoring).



1. The Collection Service is now created and can be viewed in a Browser



1. The Service template contains a Helppage, which contains Information’s and examples about the xml schema and JSON for the different REST operations.
2. At <http://channel9.msdn.com/shows/Endpoint/endpointtv-Screencast-Building-resource-collection-services-with-the-WCF-REST-Starter-Kit/> you will find a Screencast from Pluralsight, which show you how to create a Collection Service with Visual Studio and the REST Starter Kit

# How to use the RESTclient for WCFProgram:

## Step by Step instructions:

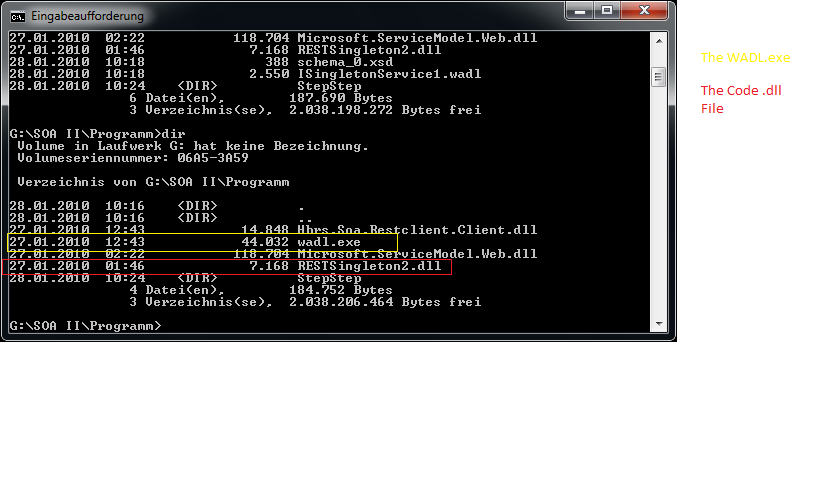
This Step by Step instruction shows the Codegeneration with a singleton Service. The generation with a collection Service is analog.

### Code 2 WADL:

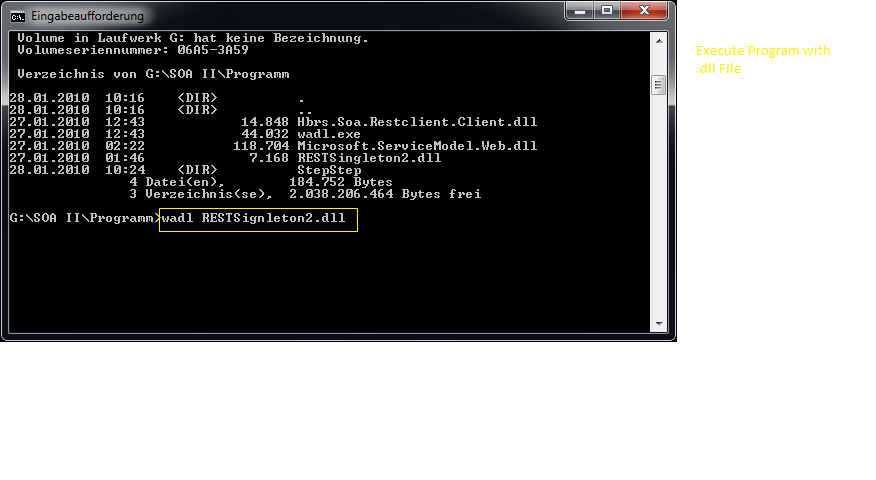
The RESTclient for WCF is a Console Application to generate the WADL files from an existing code file.

#### Step 1:

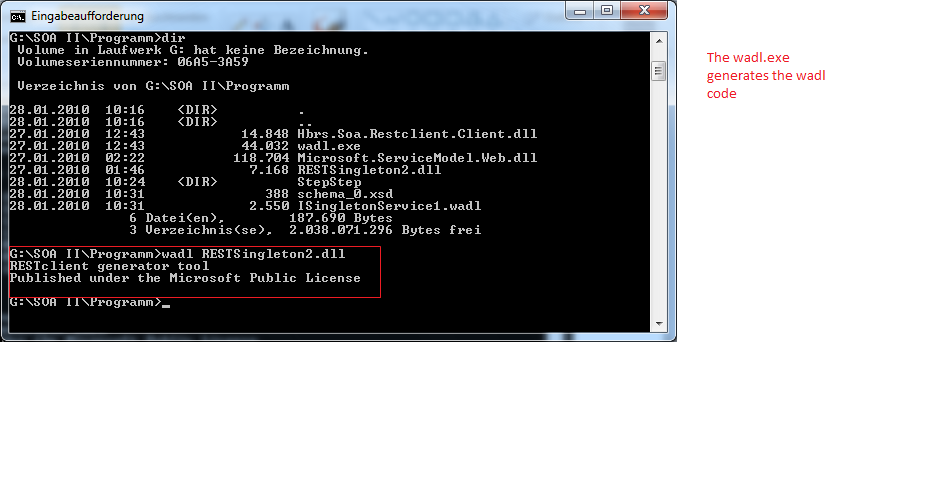
Open the Windows Console and browse to the location of the wadl.exe.

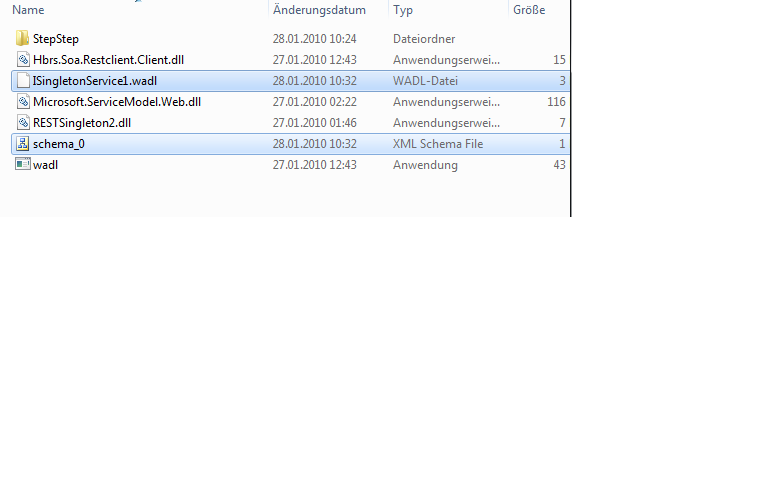


Step 2:

Invoke the wadl.exe with the .dll Codefile

#### Step 3:

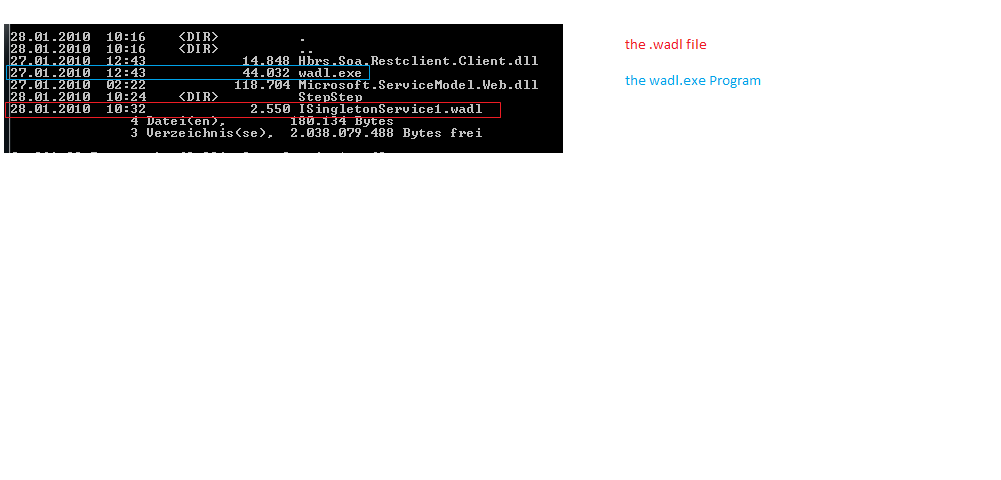
After pressing ENTER, the wadl.exe Program generates the WADL and a Schema File.

The two generated Files. The .wadl File and the XML schema file.

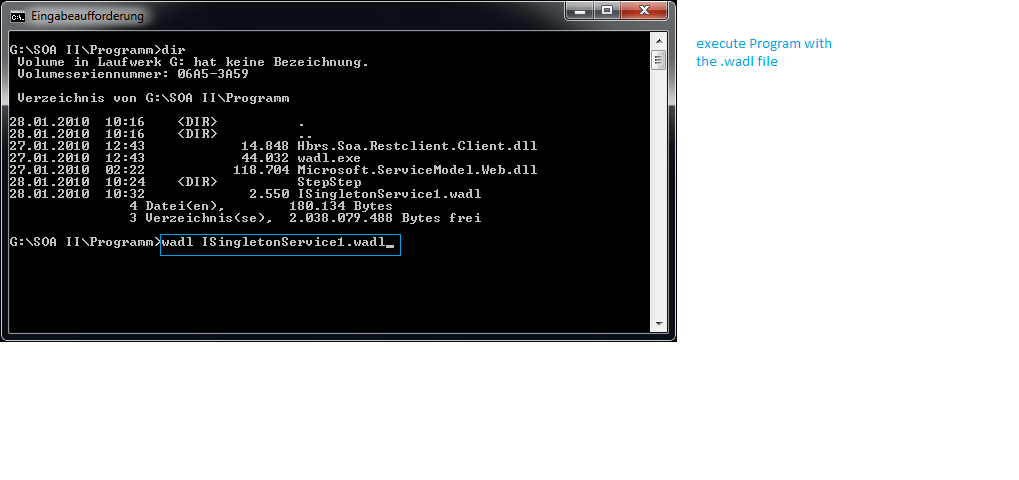
### WADL to Code:

The RESTclient for WCF is a Console Application to generate the Code files from an exisisting .wadl file.

#### Step 1:

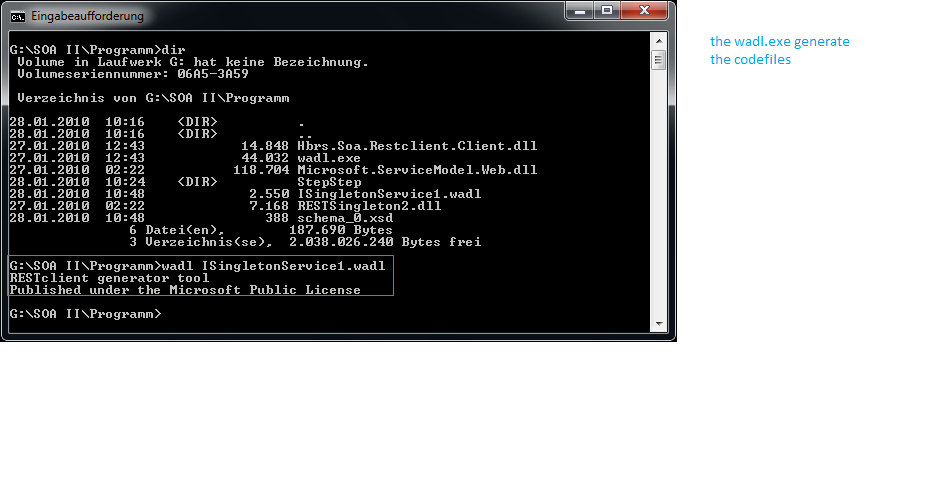
Open the Windows Console and browse to the location of the wadl.exe.

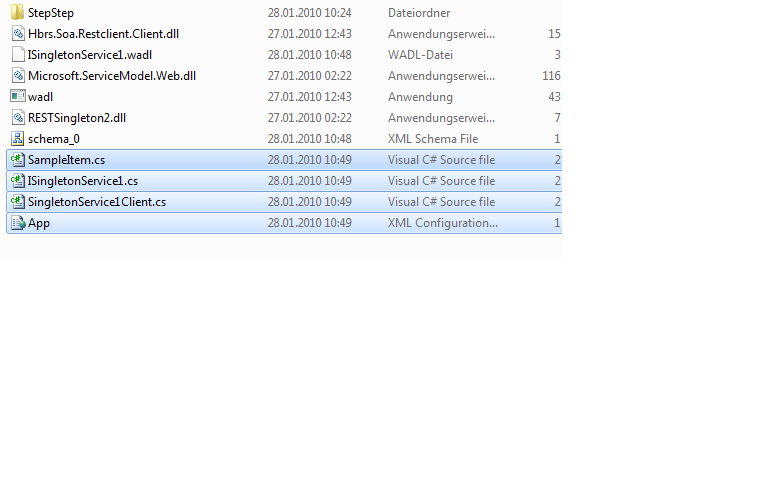
#### Step 2:

Invoke the wadl.exe with the .wadl Codefile

#### Step 3:

After pressing ENTER, the wadl.exe Program generates the Code files.



The Sampleitem.cs, ISingletonService1.cs, SingletonServicecClient.cs and App XML Configuration files are generated:

# The Web Application Description Language (WADL)

## Definition:

The WADL file Format provides machine – readable code in XML to describe HTTP-based Web applications usually using REST web services.

Like the WSDL for SOAP based Web Services, the WADL provides a model to describe Web Services. Prior to a WADL, this information had to be picked manually.

The WADL Standard was submitted by W3C Member Sun Microsystems, Inc. on 31 August 2009 and can be found at:

<http://www.w3.org/Submission/wadl/#x3-110002.5> .

## WADL Description (overview) and supported Features (of the RESTclient foe WCF)

### 1. Introduction

#### 1.1 Web Application:

In 1.1 a typical Web Application is defined as a HTTP-based application with following attributes:

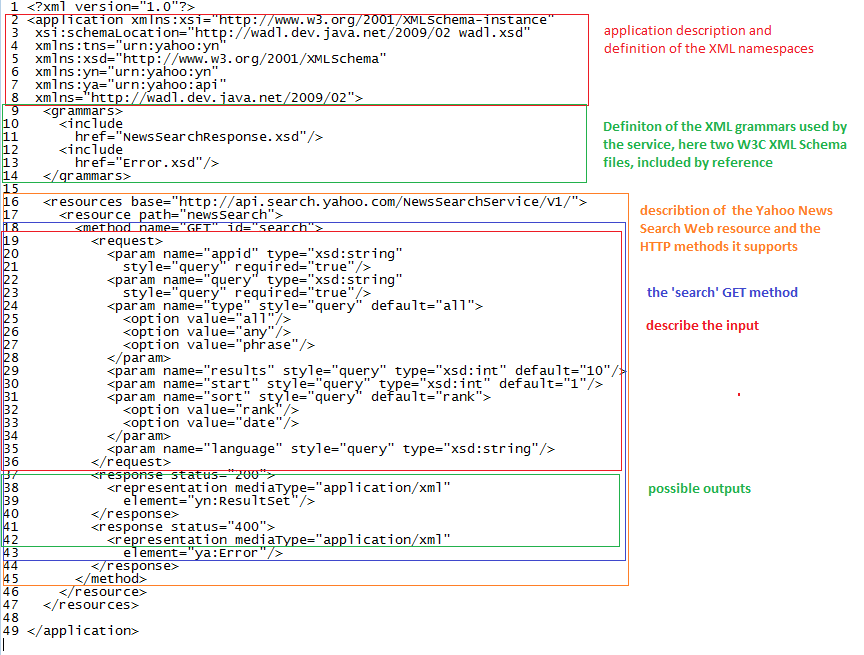
* based on existing Web architecture and infrastructure
* platform and programming language independent
* Promote re-use of the application beyond the browser
* Enable composition with other Web or desktop applications
* Require semantic clarity in content (representations) exchanged during their use

#### 1.2 Use Cases:

Use cases describe the three following machine friendly description formats:

* Application Modeling and Visualization:
  + Support for development of resource modeling tools for resource relationship and choreography analysis and manipulation.
* Code Generation:
  + Automated generation of stub and skeleton code and code for manipulation of resource representations.
* Configuration:
  + Configuration of client and server using a portable format.

#### 1.3 Example WADL Description:

1.3 shows an example WADL description of a YAHOO News Search application.

### 2. Description Components

#### 2.2 Application:

The application element forms the root of a WADL description and contains the following:

* Zero or more *doc* elements
* An optional *grammars* element
* Zero or more *resources* elements
* Zero or more of the following:
  + *resource\_type* elements
  + *method* element
  + *representation* elements
  + *param* elements

#### 2.3 Documentation:

A WADL-defined element may have one or more *doc* elements to document the element. The *doc* element has the followingattributes.

* xml:lang:
  + Defines the language for the title attribute value and the contents of the doc element. If an element contains more than one doc element then they MUST have distinct values for their *xml:lang* attribute.
* Title:
  + A short plain text description of the element being documented, the value SHOULD be suitable for use as a title for the contained documentation.

>> 2.3 are currently not supported or included into the project <<

#### 2.4 Grammars:

The *grammars* element is a container definition of the format of data exchanged during execution of the protocol described by the WADL document. There is no definition language mandated.

##### 2.4.1 Include:

The resources element acts as a container for the resources provided by the application. A resources element has a base attribute of type *xsd:anyURI* that provides the base URI for each child resource identifier. Descendent resource elements describe the resources provided by the application.

>> 1. At this time only *include* is implemented and supported in the project

2. At this time only *xsd* is accepted per include <<

#### 2.5 Resources:

The *resources* element is a container for the *resources* provided by the application. A *resources* element has a base attribute of type *xsd:anyURI* that provides the base URI for each child *resource* identifier.

#### 2.6 Resource:

2.6 describe the following attributes of a Resource:

* ID:
  + An optional attribute of type *xsd:ID* that identifies the resource element.
    - Current Restriction of the project: A resource ID is mandatory
* Path:
  + Path is an optional *xsd:string* attribute, which provides a relative URI template for the identifier of the resource
* Type:
  + Type is an optional *xsd:anyURI* attribute which value in the list is a cross reference that identifies a *resource\_type* element that defines a set of methods supported by the resource.
* Query type:
  + The query Type defines the media type for the query component of the resource URI

2.6 describe the following child elements for a resource:

* Zero or more *doc* elements
* Zero or more *param* elements
* Template:
  + Provides additional information about an embedded template parameter
* Matrix:
  + Specifies a Matrix URI Parameter
* Query:
  + Specifies a global URI query parameter for all child method elements of the resource.
* Header:
  + Specifies a global HTTP header for use in the request part of all child *method* elements of the resource

>> At this time the WADL in the Project only works with one Resource <<

#### 2.6.1 Generating Resource Identifiers:

The Resource Identifier set rules for the URI representation of a *resource* element.

>> Rules in 2.6.1 are not fully considered in the project at this time <<

#### 2.7 Resource Type:

The resource type provides a set of methods that define the behavior of a type of resource.

A resource type has the following attributes:

* ID:
  + A required attribute of type *xsd:ID* that identifies the *resource\_type* element.

A resource type has the following child elements:

* Zero or more *doc* elements
* Zero or more *param* elements
* Query:
  + Specifies a URI query parameter for all child method elements of the *resource* type
* Header:
  + Specifies a HTTP header for use in the request part of all child method elements of the *resource* type

>> Methods describe in 2.7 are not fully considered in the project at this time <<

#### 2.8 Methods:

A method element describes the input to and output from an HTTP protocol method that may be applied to a resource. A method element can either be a method definition or a reference to a method defined elsewhere.

##### 2.8.1 Method Reference:

A *method* reference element is a child of a *resource* element that has an *href* attribute whose type is *xsd:anyURI*. The value of the *href* attribute is a cross reference to a method definition element. Method reference elements MUST NOT have any other WADL-defined attributes or contain any WADL-defined child elements.

This form of method element may be used to reduce duplication when the same method applies to more than one resource.

##### 2.8.2 Method Definition:

A *method* definition element is a child of a resource or *application* element and has the following attributes:

* Name:
  + the HTTP method used
* ID:
  + An identifier for the method, required for globally defined methods

A method has the following child elements:

* DOC:
  + Zero or more doc elements
* Request:
  + Describes the input to the method as a collection of parameters and an optional resource representation
* Response:
  + Zero or more response elements that describe the possible outputs of the method

#### 2.9 Request:

A request element describes the input to be included when applying an HTTP method to a resource.

A request element may contain the following child elements:

* Zero or more *doc* element
* Zero or more *representation* elements
* Zero or more *param* elements
* Query:
  + Specifies a URI query parameter for all methods that apply to this resource
* Header:
  + Specifies a HTTP header for use in the request

#### 2.10 Response:

A *response* element describes the output that results from performing an HTTP method on a resource.

The response element has the following attributes:

* Status:
  + Optionally present on responses, provides a list of HTTP status codes associated with a particular response.

A response element may have the following child elements:

* Zero or more *doc* element
* Zero or more *representation* elements
* Zero or more *param* elements

#### 2.11 Representation:

A *representation* element describes a representation of a resource's state. A *representation* element can either be a representation definition or a reference to a representation defined elsewhere.

##### 2.11.1 Representation Reference:

A representation reference element can be a child of a request or response element

##### 2.11.2 Representation Definition:

A representation definition element can be a child of a request, response or application element

##### 2.12.3 Representation Parameters:

A child *param* element is used to parameterize its parent representation element. Representation parameters can have one of two different functions depending on the media type of the representation:

1. Define the content of the representation. For representation elements with a *mediaType* attribute whose value is either *'application/x-www-form-urlencoded'* or *'multipart/form-data'* the representation parameters define the content of the representation which is formatted according to the media type. The same may apply to other media types.

2. Provide a hint to processors about items of interest within a representation. For XML based representations, representation parameters can be used to identify items of interest with the XML. The path attribute of a representation parameter indicates the path to the value of the parameter within the representation. For XML-based representations this is an *XPath* expression.

>> The Representation of the Project supports at this time only XML <<

#### 2.12 Parameter:

A *param* element describes a parameterized component of its parent element. A *param* element can either be a parameter definition or a reference to a parameter defined elsewhere.

##### 2.12.1 Parameter Reference:

A *param* reference element is a *param* element that has an *href* attribute whose type is *xsd:anyURI*. The value of the *href* attribute is a cross reference to a *param* definition element. Param reference elements MUST NOT have any other WADL-defined attributes or contain any WADL-defined child elements.

##### 2.12.2 Parameter Definition:

A *param* definition element describes a parameterized component of its parent element and may be a child of a resource, application, request, response, or a representation element. A param definition element has zero or more doc child elements, zero or more option child elements, an optional link child element and has the following attributes:

* Id:
  + An optional identifier that may be used to refer to a parameter definition using a URI

reference.

* Name:
  + The name of the parameter as an *xsd:NMTOKEN*. Required.
* Style:
  + Indicates the parameter style, table 1 on page 25 lists the allowed values and shows the context(s) in which each value may be used.
* Type:
  + Optionally indicates the type of the parameter as an XML qualified name, defaults to *xsd:string*.
* Default:
  + Optionally provides a value that is considered identical to an unspecified parameter value.
* Path:
  + When the parent element is a representation element, this attribute optionally provides a path to the value of the parameter within the representation
* Required:
  + Optionally indicates whether the parameter is required to be present or not, defaults to false (parameter not required).
* Repeating:
  + Optionally indicates whether the parameter is single valued or may have multiple values, defaults to false (parameter is single valued)
* Fixed:
  + Optionally provides a fixed value for the parameter.

##### 2.12.3 Option:

An *option* element defines one of a set of possible values for the parameter represented by its parent *param* element.

##### 2.12.4 Link:

A link element is used to identify links to resources within representations.

# Links:

* Web Application Description Language W3C Member Submission 31 August 2009; Sun Microsystems, Inc; Santa Clara California 2009; <http://www.w3.org/Submission/wadl/#x3-100002.4.1>
* <http://channel9.msdn.com/shows/Endpoint/endpointtv-Screencast-Building-resource-singleton-services-with-the-WCF-REST-Starter-Kit/> Pluralsight Screencast How to Create a singleton Service
* <http://channel9.msdn.com/shows/Endpoint/endpointtv-Screencast-Building-resource-collection-services-with-the-WCF-REST-Starter-Kit/> Pluralsight Screencast How to Create a collection Service