10 Ways to Build Web Services in .NET ServiceStack

Chad McCallum
@ChadEmm

pluralsight
hardcore developer training

Module Outline

- Introduction to ServiceStack
- Installing and configuring ServiceStack
- Creating our first Request and Service
- A Request with a route parameter
- A Request using the request body
- Reusing Request objects for multiple routes
- Getting data using the ServiceStack.Client library
- Sending data reusing a Request object
- Getting raw string data from the server
- Getting a raw HttpWebResponse from the server
- Review

ServiceStack

Thoughtfully architected, obscenely fast, thoroughly enjoyable web services for all

- **A fast, unified and integrated replacement for WCF, WebAPI, and MVC**
  - Supports REST, SOAP, and Message Queue services
  - Includes serializers for many formats
  - Provides a Code-First ORM
  - User-friendly HTML views of data
  - Includes packages for caching, authentication, clients, logging, dependency injection, profiling, and much more

ServiceStack

- **Very fast**
  - ORM benchmarks about 12x faster than EntityFramework
  - JSON serialization benchmarks about 6x faster than DataContractJsonSerializer
- **Revolves around the reuse of Plain Old CLR Objects (POCOs) in requests, responses, client-side code, ORM, and more**
- **Focuses on the Data Transfer Object/Message Pattern**
  - Request Object -> Processor -> Response Object

Review

- Introduction to ServiceStack
- Installing and configuring ServiceStack
- Creating our first Request and Service
- A Request with a route parameter
- A Request using the request body
- Reusing Request objects for multiple routes
- Getting data using the ServiceStack.Client library
- Sending data reusing a Request object
- Getting raw string data from the server
- Getting a raw HttpWebResponse from the server

ServiceStack – Requests & Responses

- **Request objects define parameters and routes for service**
  - Automatically deserialized from incoming data: route variables, query string, request body, etc.
  - Requests are also automatically mapped to generated routes, i.e.

  POST (format)/oneway/(request)
- **Response objects define output data, not type**
  - ServiceStack takes care of serialization in multiple formats, including HTML, JSON, XML, CSV, JSV, SOAP, and any others that are configured
  - Preferred response format can be defined by:
    - Query String (?format=json)
    - Extension (.json)
    - Request Header (Accept: application/json)

ServiceStack - Services

- **Services are responsible for processing requests and returning responses**
  - Get(RequestObject request)
  - Post(RequestObject request)
  - Put(RequestObject request)
  - Delete(RequestObject request)
  - Any(RequestObject request)

ServiceStack – Client Library

- **ServiceStack provides a separate client library that can reuse the Request and Response objects used on the server**
  - RequestObject : IReturn<Type> is used by the client library to determine the expected return type to deserialize to
  - client.Get(request) will automatically determine the route, serialize the request, and deserialize the response
- **ServiceClient also provides direct access to routes, raw data, response objects, byte arrays, and streams**
  - client.Get<string>("/route") provides the raw, serialized data from the server
  - client.Get<HttpWebResponse>("/route") provides the HttpWebResponse object from the operation
  - client.Get<byte[]>("/route") provides the byte array of data
  - client.Get<Stream>("/route") provides the output stream of the request