Webbtjänster med API er

Mål med lektionen!

- En lite djupare inblick i RESTfulla tjänster

Vad lektionen omfattar

- RESTful Services

Överblick

- SOAP kan vara lite ”overkill” för vissa specifika web service scenarion.
- SOAP utan WS-* protokollen, Simple SOAP.
- XML meddelanden (POX)

REST

R.Fielding

Skrev en avhandling som beskrev en skalbar arkitekturför att bygga tjänster runt ett enhetligt interface.

Vilket han egentligen beskrev, var en arkitektonisk beskrivning för hur webben egentligen fungerar idag, och på grund av detta så är HTTP det protokoll som folk använder idag för att bygga RESTfulla implementationer.

REST är fundamentalt olika ifrån SOAP

SOAP definierar en transport-neutral modell, fokuserat på definiera customiserade tjänste-kontrakt med customiserade operationer och du kan anropa dessa operationer över en mängd olika transporter genom att använda olika sätt att koda för meddelanden.

REST å andra sidan definierar mer en transport specifik modell. REST binder dig inte till HTTP, men i realitet, HTTP är det enda protokollet som praktiskt används idag för att bygga RESTfulla arkitekturer. Så dessa två går hand i hand.

Förstå REST

- Definiera resurser med unikt id eller URI
- Enhetliga service kontrakt
- GET, PUT, POST, DELETE och HEAD
- HTML, XML, RSS eller JSON

Vad SOAP betonar

SOAP har betoningen på verb eller handlingar/händelser, medans **REST** har betoningen på adjektiv eller resurser.

När du definierar en SOAP tjänst så är ditt fokus på att definiera service kontraktet och uppsättningen med service operationer som vi skall exponera genom tjänsten.

REST vs SOAP

SOAP

```
getUser()
addUser()
removeUser()
...
getLocation()
addLocation()
...
```

REST

```
User { }
Location { }
```

Fördelar med HTTP verb

- ROA (Resurs-orienterad arkitektur) fokuserar på identifiera och namnge resurserna.
- Hur man navigerar mellan dessa resurser med länkar eller HREF’s.
- HTTP verb med fördelar.
- Ett utbrett samarbete mellan system/tjänster.
- Skalbarheten

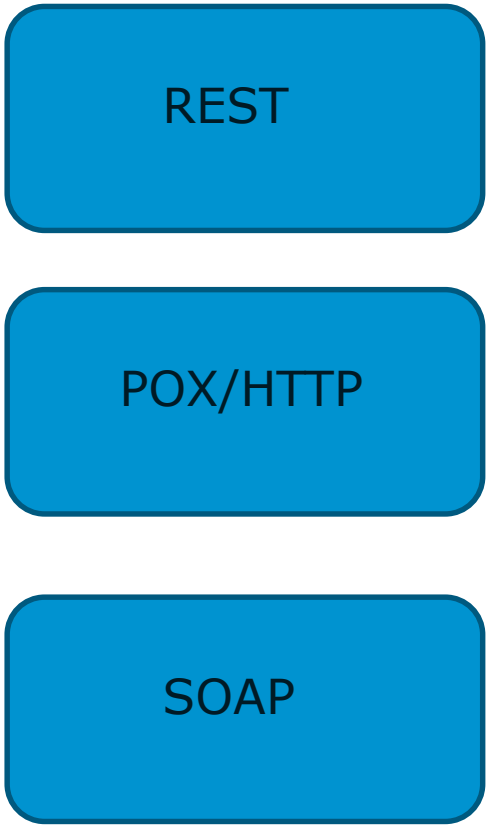
Vikten av GET metoden

- Ca 90% av dagens webb trafik använder GET.
- Idempotenta GET requests.
- POST PUT och DELETE, ger inte samma garantier som GET gör.
- REST & GET.
- SOAP anropen tunnlar genom ett POST request.

Vikten av webformaten

- Räckvidden på tjänsten begränsas av formatets stöd.
- XML vanligaste formatet.
  - RSS
  - Atom
- JSON

3 olika modeller och vad får vi ifrån dem?



Dessa modeller (SOAP)

- I grund och botten har varje arkitektur sina egna fördelar och det finns ingen tydlig vinnare för alla use-cases.
- SOAP och WS-* lösningar är för storföretag, med strikta krav.
- Men den kan vara ganska överdriven i vissa distribuerade scenarion.

Dessa modeller (REST)

- Skalbarhet, hög interoperabilitet.
- Verktygsstödet
- Ingen klar vinnare.

REST stöd

- WCF har ingen favorit.
- Många val.
- Hög flexibilitet.
- Behöver ingen ny modell

REST stöd (forts)

- Ny modell.
- Mappa HTTP anrop mot metoder.
- Ny bindning, nytt behavior.
- Tillämpar WebHttpBehavior för att räkna ut hur vi mappar mot metoder.

Konfigurera WCF för REST

- WebHttpBinding klassen producerar kanal stacken.
- Låter oss konfigurera en hel del.
- HTTP eller HTTPS.
- Kör över operation selection processen, och injicerar en anpassad operations selector.

Konfigurera WCF för REST

Hur man skriver koden för att koppla upp/koppla ihop dessa sakerna.

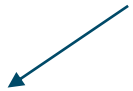
```
ServiceHost host = new ServiceHost(typeof(EvalService),
new Uri("http://localhost:8080/evals"));

host.AddServiceEndpoint(typeof(IEvals),
    new WebHttpBinding(), "");
host.Description.Endpoints[0].Behaviors.Add(
    new WebHttpBehavior());
host.Open();  // service is up and running

Console.ReadLine(); // hold process open
```

Lägga till WebBinding

Lägga till WebBehavior

WebServiceHost

Denna hosten lägger till
WebBinding & Behavior



```
WebServiceHost host = new WebServiceHost(
    typeof(EvalService),
    new Uri("http://localhost:8080/evals"));
host.Open();  // service is up and running
Console.ReadLine(); // hold process open
```

WebGetAttribute

- Nya attribut för att mappa HTTP request.
- WebGetAttribute, mappar HTTPGet mot en viss metod.
- UriTemplate.
- Vill vi använda XML eller JSON för meddelande?

WebGetAttribute

```
[ServiceContract]
public interface IEvalService
{
    [WebGet(UriTemplate="evals?name={name}&score={score}")]
    [OperationContract]
    List<Eval> GetEvals(string name, int score);
```

WebInvokeAttribute

- WebInvokeAttribute
- Optimerar GET separat.
- POST är default.
- Man antar att den sista parametern ska deserialiseras.

WebInvokeAttribute

```
[ServiceContract]
public interface IEvals
{
    [WebInvoke(UriTemplate ="/evals?name={name}",Method="PUT")]
    [OperationContract]
    void SubmitEval(string name, Eval eval /* body */);
```

UriTemplate

- UriTemplates mappar inkommande anrop till metoder.
- Kan använda wildcard i pathen (/*).
- Ny klass som heter System.UriTemplate.
- Matcha en URI:mot värdet i en UriTemplate.

```
/services/evals?name={name}&detailed={detailed}
```

WebOperationContext

- WebOperationContext klassen.
- HTTP request och response meddelandena. WebOperationContext.Current.
- Innehållslängd (ContentLength), Innehållstyp (ContentType).
- WCF försöker gömma detaljerna.

Syndication programming model

```
[ServiceKnownType(typeof(Atom10FeedFormatter))]
[ServiceKnownType(typeof(Rss20FeedFormatter))]
[ServiceContract]
public interface IEvalService{
    [WebGet(UriTemplate = "evalsfeed")]
    [OperationContract]
    SyndicationFeedFormatter GetEvalsFeed();
...
}
```

Syndication programming model

```
public class EvalService : IEvalService{
    public SyndicationFeedFormatter GetEvalsFeed(){

        List<Eval> evals = this.GetEvals();
        SyndicationFeed feed = CreateSyndicationFeed(evals);

        // figure out what format the client wants
        WebOperationContext ctx = WebOperationContext.Current;
        string format =
            ctx.IncomingRequest.UriTemplateMatch.QueryParameters["format"];

        // return the right type of formatted feed
        if (format != null && format.Equals("atom"))
           return new Atom10FeedFormatter(feed);
        else
           return new Rss20FeedFormatter(feed);
    }
```

Hypermedia?

- Olinjär information.
- Hypermedia är vad www är gjord av.
- Hypermedia går utöver hypertext.
- Ett ökande behov av olika komponenter inom en enda applikation

REST?

- Enhetligt gränssnitt
- Applikationstillståndets motor
- Definitionen av REST

REST? Hypermedia?

- Application state
- Hypermedia myntat av Ted Nelson
- Olika APIER har i regel problem med att förstå varandra.
- Ett gemensamt format för att skriva API:er?

Hypermedia & REST?

- Flickr API’et.
- Fixerat API, får ej ändras?
- Enkelt kunna gå mellan innehåll.
- Enhetligt gränssnitt.

RESTful övning ändring i ServiceLibrary

- Data kontraktet skall ha ett attribut inom parantes som är Namespace och tilldelas http://localhost:8080/evals.
- Lägg till en DataMember som är en public string Id.
- Lägg tillföljande OperationContracts (i Serviceklassen):
  - Eval GetEval(string Id)
  - List<Eval> GetAllEvals()
  - List<Eval> GetEvalsBySubmitter(string submitter)
  - void RemoveEval(string Id)
- Referera in System.ServiceModel.Web till projektet.

RESTful övning ändring i ServiceLibrary

- Annotera alla OperationContact med följande:
  - SubmitEval->[WebInvoke(Method="POST", UriTemplate="evals")]
  - GetEval(string id)->[WebGet(UriTemplate = "eval?Id={id}")]
  - GetAllEvals->[WebGet(UriTemplate = "evals")]
  - GetEvalsBySubmitter(string submitter)->
    [WebGet(UriTemplate = "evals?submitter={submitter}")]
  - RemoveEval(string id)->
    [WebInvoke(Method = "DELETE", UriTemplate = "eval?Id={id}")]

RESTful övning ändring i ServiceLibrary

- I serviceklassen så skall vi lägga till en variabel som heter evalCount som är 0.
- I metoden SubmitEval, innan vi lägger till Eval till listan så skall vi ta värdet ifrån ++evalCount och lägga till eval.Id.ToString().
- GetEval(string id) använd lambda uttryck för att hämta eval med id som matchar.
- GetAllEvals skall anropa GetEvalsBySubmitter (return satsen) med null som parameter.
- GetEvalsBySubmitter skall kontrollera om submitter är null eller tom sträng om så returnera listan evals, annars returnera alla evals som matchar submittern.
- RemoveEvals gör först en find som returnerar en eval som matchar id därefter gör vi remove på listan med den hittade eval som parameter.

RESTful övning ändring i ConsoleHost

I ConsoleHost så skall er app.config se ut enligt följande:

```
<system.serviceModel>
  <services>
    <service name="EvalServiceLibrary.EvalService">
      <host>
        <baseAddresses>
          <add baseAddress="http://localhost:8080/evalservice"/>
        </baseAddresses>
      </host>
    </service>
  </services>
</system.serviceModel>
```

RESTful övning ändring i ConsoleHost

- Referera in System.ServiceModel.Web till projektet.
- I program filen i ConsoleHost projektet skapa upp en instans av klassen WebServiceHost som tilldelas resultatet av new WebServiceHost(typ av (servicekontraktet));
- I try delen öppna hosten som vanligt och vänta på en ReadLine() innan vi stänger den.
- I catch delen så skall vi anropa abort på hosten.
- Referera in EvalServiceLibrary till projektet.

RESTful övning ändring i Client

- Referera in System.ServiceModel.Web till projektet.
- Referera in EvalServiceLibrary till projektet.
- app.config skall se ut enligt följande:

```
<startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.1" />
</startup>
<system.serviceModel>
    <bindings>
        <basicHttpBinding>
            <binding name="BasicHttpBinding_IEvalService" />
        </basicHttpBinding>
    </bindings>
    <client>
        <endpoint address="http://localhost:8080/evalservice" binding="basicHttpBinding"
            bindingConfiguration="BasicHttpBinding_IEvalService" contract="EvalServiceLibrary.IEvalService"
            name="BasicHttpBinding_IEvalService" />
    </client>
</system.serviceModel>
```

RESTful övning ändring i Client

- I programfilen skapa:
- WebChannelFactory<IEvalService> cf = new
  WebChannelFactory<IEvalService>(
  new Uri("http://localhost:8080/evalservice"));
- IEvalService client = cf.CreateChannel();
- Be användaren om ett kommando som vi sparar i en variabel
- Gör en loop som snurrar tills kommandot är lika med ”exit”.
- Gör en switch över kommandot och casen skall vara:
  - submit
  - get
  - list
  - remove
  - default

RESTful övning ändring i Client

- Submit skall skapa upp en eval och submitta denna.
- Get skall be om ett id som vi har som parameter i anropet till client.GetEval().
- List ber om submitterns namn som vi använder för att skicka in som parameter till GetEvalsBySubmitter() och sedan för varje eval i listan skriver vi ut värdena.
- Remove ber om ett id på en eval att ta bort som vi använder som parameter i anropet client.RemoveEval().
- Default skall bara tala om att kommandot inte stöds.
- Utanför loopen be om ett kommando.