Web Services

- A Web Service is ***like*** an isolated piece of code designed to carry out some specific task (like a function in programming languages).
- Any platform should be able to access Web Services irrespective of what platform they are running on.
- A programming language can connect to a Web Service irrespective of what it was written in.
- Web Services use XML for communication and invokation.

Web Service Protocols

- Web Services can make use the following internet standard protocols:
    - HTTP,
    - SMTP,
    - SOAP,
    - UDDI,
    - WSDL,
    - XML.

Scenario

1. An application calls a web service having a URL (an address).
2. The call is constructed as a SOAP message and transported over HTTP.
3. The service
    1. Receives the request,
    2. Decodes the SOAP message,
    3. Processes the request,
    4. Sends a response to the caller (as another SOAP message over HTTP).

B2B

- Web Services are especially useful in business-to-business transactions.
  - For example a bank exposing currency conversion services to an online retailer.
- However, such companies realise that since they developed using ‘closed’ languages and ‘close’ platforms they are limited in the services they offered.
- Web Services can be exposed by design.
- Custom integration design is
  - Expensive,
  - Time consuming,
  - Very sensitive to changes in logic.

Development Tools

- Web Services are gaining so much interest that whole development tools are now focusing on the concept and languages are being extended to make Web Service development easy.
- http://msdn.microsoft.com/webservices/
- http://java.sun.com/webservices/docs/1.0/tutorial/doc/IntroWS.html

Web Services Infrastructure

- Web Directories:
  - Central location to store information about published web services.
  - The directory to query might be a web service itself!
  - The Universal Description, Discovery, and Integration (UDDI) specification defines what has to be published.
  - E.g. uddi.microsoft.com
- Discovery:
  - The process used by the clients to ‘discover’ information about, and the location of web services.

Web Services Infrastructure

- Description:
  - Provides information about the web service (i.e. which operations are supported by the service).
  - Web services are described in an XML format Web Services Description Language (WSDL) document.
- Wire formats:
  - Open wire formats are the internet protocols that allow us to communicate with web services.
  - HTTP, SMTP and others can be used.

Discovering a Service

1. The client accesses a UDDI service to query it about a web service.
2. The UDDI service, returns the location (URL) to the discovery document of the web service.
3. The discovery document usually contains details such as the contact reference of the service publisher (amongst others).
4. The client uses the discovery document to locate the description document of the web service (the WSDL document).
5. Using the WSDL the client can now use the web service.

The Client Side

1. You create an object from the web service class (the class is really a proxy class).
2. The client calls the desired method (on the proxy object).
3. The call and all arguments are serialised into a request SOAP message.
4. The request SOAP message is sent over the network by the client side infrastructure.
5. The server receives the request SOAP message and deserialises it, extracting all info related to the call.

The Client Side

- The web service is instantiated on the server and is executed using the parameters of the request SOAP message.
- The return value and out parameters are serialised into a response SOAP message which is sent to the client over the network.
- The infrastructure on the client side, deserialises the SOAP message, extracts the return value and out parameters and sends them to the proxy object.
- The proxy object returns.
- ON THE CLIENT SIDE, THE WHOLE PROCESS APPEARS JUST LIKE ANY METHOD CALL. THE INFRASTRUCTURE HANDLES EVERYTHING!!!

Developing Web Services

- Modern tools and languages like Microsoft’s .NET and Sun’s Java come packaged with web service development functionality.
- We will be using .NET in our examples, but the same concepts should apply for Java and any other web service-aware tool or language.
- In .NET, web services are called **ASP.NET Web Services**.

ASP.NET

- Visual Studio comes with pre-built templates to develop ASP.NET Web Services.
- Once you name your project, Visual Studio will create the skeleton of your service(s).
- Files created:
  - AssemblyInfo: version number, product name, etc…
  - Web.config: the web service will live on IIS. This file contains configuration info such as the authentication modes, debug mode flags, etc…
  - Global.asax and Global.asax.cs: Manage application and session level events (e.g. Application_OnError).
  - MyService.asmx and MyService.asmx.cs: The ASMX file is a descriptor of the service, and the CS file is the c# code file.
  - WebService.vsdisco: the discovery file.

Web Methods

- To create a method to be exposed in a web service:

```
[webmethod]
public string SayHello()
{
  return “Hello World!”;
}
```

Consuming Web Services

1. Add a Web Reference to the web service – done by discovering the service.
2. Generate a proxy class (done automatically)
3. Instantiate an object from the proxy class.
4. Call the web method.

Advanced Properties

- Buffering Responses:
  - To minimise the number of connections mede between the server and the client, the server hosting the web service can buffer a batch of responsed and send them all at once.
  - To make a response ‘bufferable’:

```
[webmethod(BufferResponse=true)]
public string SayHello()
{
   return “Hello World!”;
}
```

Advanced Properties

- Caching Results:
  - The infrastructure can cache the responses for each method/parameter combination and reuse the response if the same method is called using the same parameter values.
  - Caching has a duration in seconds:

```
[webmethod(CacheDuration=60)]
public string SayHello()
{
  return “Hello World!”;
}
```

Advanced Properties

- Setting Service Descriptions that appear in the help page of the web service:

```
[webmethod(Description=“Says hello”)]
public string SayHello()
{
  return “Hello World!”;
}
```

Advanced Properties

- Can maintain session state:

```
[webmethod]
public string SayHello()
{
  string name;
  name = Session[“Name”];
  return “Hello “ + name + “!”;
}
```

Advanced Properties

- Asynchronous Methods:
  - Methods may take long to execute (due to the transport) and block interaction.
  - To solve the problem, a method can be called asynchronously.
  - The concept of a callback is used.