UNIVERSITY of STIRLING

www.cs.stir.ac.uk

Web Services

Introduction

Some Logistics

- **Dr. Mario Kolberg**, 4B123, mko@cs.stir.ac.uk
- **Lectures and labs:**
  - Labs and lectures alternate weeks. (details follow …)

Logistics cont.

- **Tutorials**
  - No formal tutorials
  - Use the labs to ask questions
- **Assignments & Examination**
  - Lab Checkpoints (10%)
  - Assignment: (40%)
  - Examination: (50%)
- **On line @ CANVAS**

Resources for private study

- Internet: lectures & lab sheets will be available on webpage
- Books
  - Java Web Services, Martin Kalin, O'Reilly (recommended)
  - Restful Java with JAX-RS 2.0, Bill Burke, O'Reilly (recommended)
  - Building RESTful Web Services with Java EE 8, Mario-Leander Reimer, Packt Publishing (background)
- Emails are welcome
  - mko@cs.stir.ac.uk

Origins

- Distributed computing for
  - Linking networked computers and applications
  - Sharing computation
  - Sharing data
- Previous approaches
  - Java RMI (Remote Method Invocation)
  - OMG CORBA (Common Object Request Broker Architecture)
  - Microsoft DCOM (Distributed Component Object Model)
  - ODP (Open Distributed Model)
- Web was rapidly adopted as a means to share information, initially through static web pages, later dynamic and interactive web pages
- Web focussed on accessing information
- Web services focus on B2B communication

UNIVERSITY of STIRLING

Web Services Slide 5

5

Characteristics

- Jeff Bezos (CEO Amazon), Tech. Review 01/2005
  - "Web 1.0 was making the Internet for people; Web 2.0 is making the Internet better for computers"
- Gartner Research,
  - "Web services are loosely coupled software components delivered over Internet standard technologies"

- emphasise communication among applications rather than users
- follow open standards that are widely supported by industry
- architecture is loosely coupled, so web services can be designed in isolation
- can interwork even if they were not explicitly designed to do so
- supported by three classes of system: service consumers (clients), service providers (servers), and service brokers (registries)

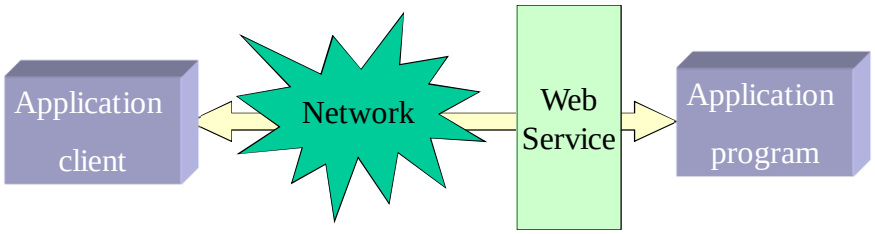
UNIVERSITY of STIRLING

Slide 6

6

What is a Web Service?

- A web service is a network accessible interface to application programs, built using standard Internet technologies.
- Clients of web services do NOT need to know how it is implemented.
- A Web Service is a URL-addressable software resource that performs functions (or a function).



UNIVERSITY of STIRLING

Slide 7

7

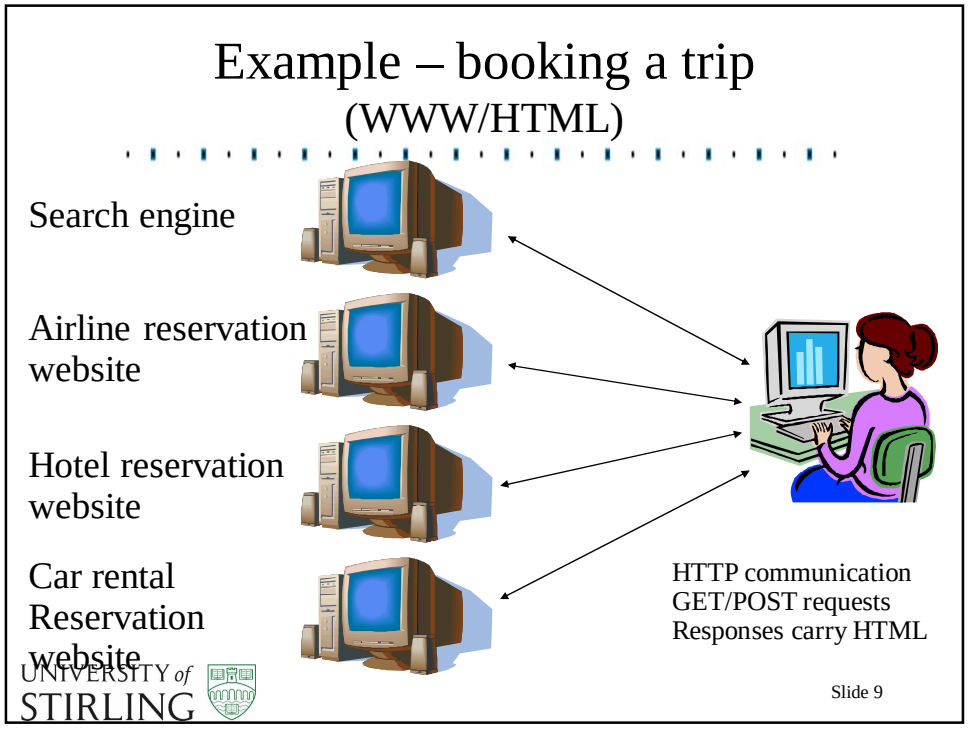
What is a Web Service?

- "Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. … Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service." *IBM web service tutorial*

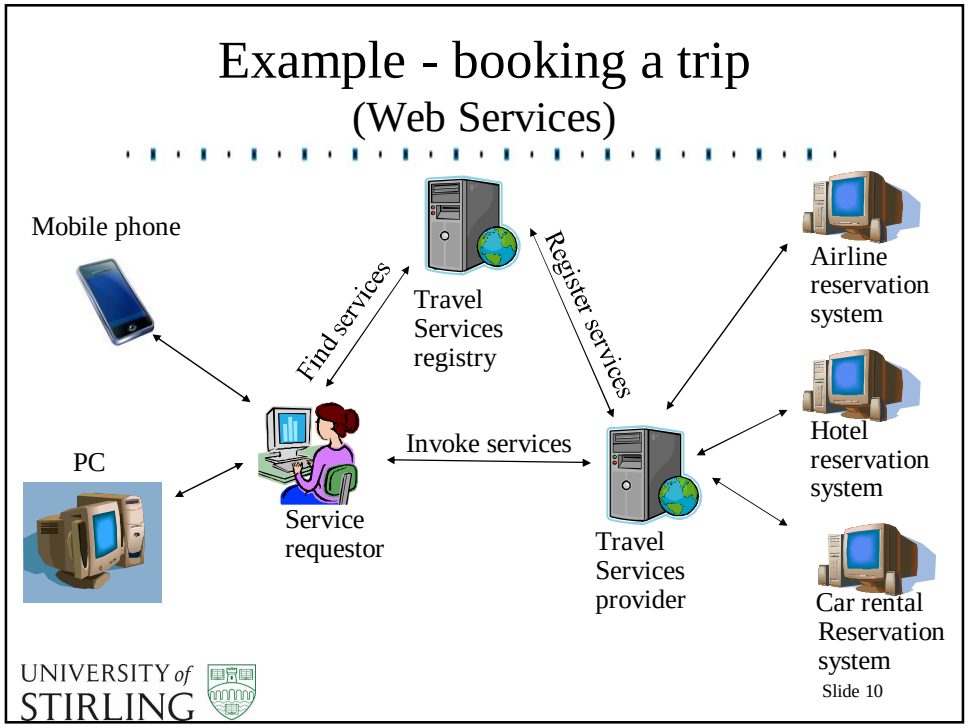
UNIVERSITY of STIRLING

Slide 8

8

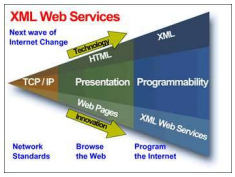

Example – booking a trip
(WWW/HTML)
Search engine
Airline reservation website
Hotel reservation website
Car rental Reservation website
HTTP communication
GET/POST requests
Responses carry HTML
UNIVERSITY of STIRLING
Slide 9

9


Example - booking a trip
(Web Services)
Mobile phone
PC
Find services
Travel Services registry
Register services
Invoke services
Service requestor
Travel Services provider
Airline reservation system
Hotel reservation system
Car rental Reservation system
UNIVERSITY of STIRLING
Slide 10

10

Evolution



Technology	TCP/IP	HTML	XML
Purpose	Connectivity	Presentation	Programmability
Applications	E-Mail, FTP…	Web Pages	Web Services
Outcome	Create the Web	Browse the Web	Program the Web

UNIVERSITY of STIRLING

Slide 11

11

Usage

- Web services are used by business for managing partnerships
  - supply chains (e.g. a brake manufacturer automatically maintains stock levels of parts for a car manufacturer)
  - outsourcing (e.g. an electronics manufacturer has its web pages managed by an IT company)
  - contracting (e.g. an Internet shop goes to tender for management of online purchases)
  - combined services (e.g. a travel agent uses the services of airlines, hotel chains and car rental companies to offer a complete travel booking service)
- Web services support virtual organisations across the boundaries of conventional organisations

UNIVERSITY of STIRLING

Slide 12

12

Web Services Standards

- standards for web services are defined by a number of organisations such as:
  - IETF (Internet Engineering Task Force, www.ietf.org)
  - OASIS (Organization for The Advancement of Structured Information Standards, www.oasis-open.org)
  - W3C (World Wide Web Consortium, www.w3.org)
- web services share certain communication mechanisms with conventional use of the Web:
  - HTTP (HyperText Transfer Protocol, IETF RFC 2616) to support message exchange between web services
  - TCP (Transmission Control Protocol, IETF RFC 793) for reliable transfer of data across a collection of subnetworks (e.g. the Internet)
  - IP (Internet Protocol, IETF RFC 791) for routing across a collection of subnetworks

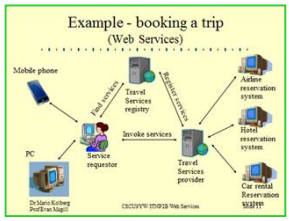
UNIVERSITY of STIRLING

Slide 13

13

Web Services Standards

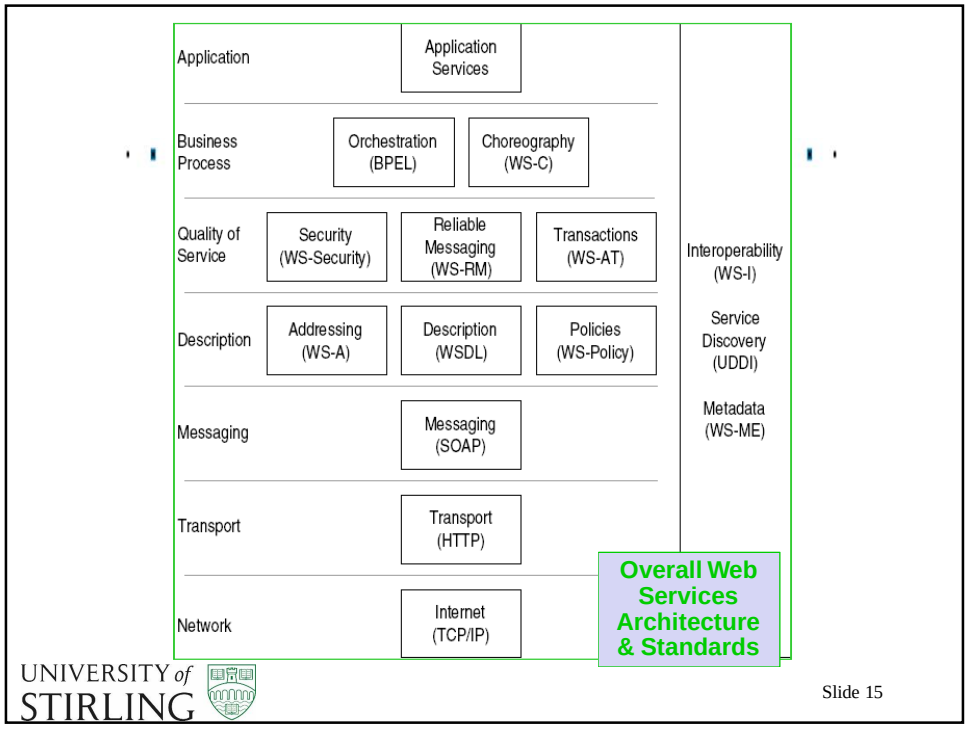
- many web service aspects have been (or are being) standardised in areas such as:
  - service description
  - service discovery
  - service addressing
  - security and authentication
  - reliable messaging and transaction
  - service orchestration and choreography
  - service policies



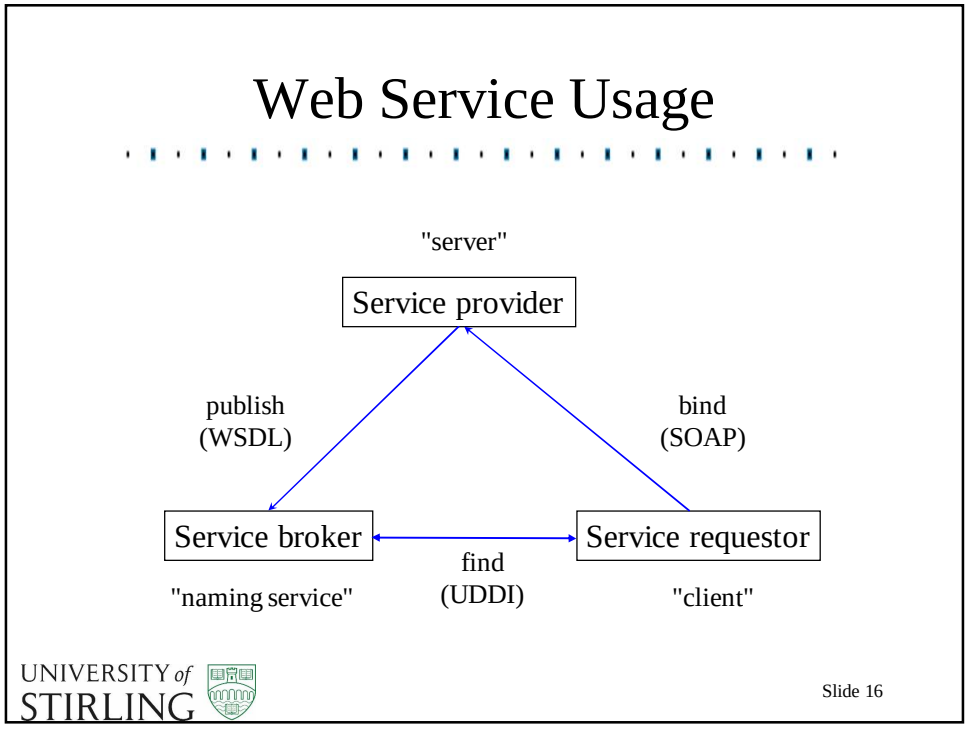
UNIVERSITY of STIRLING

Slide 14

14

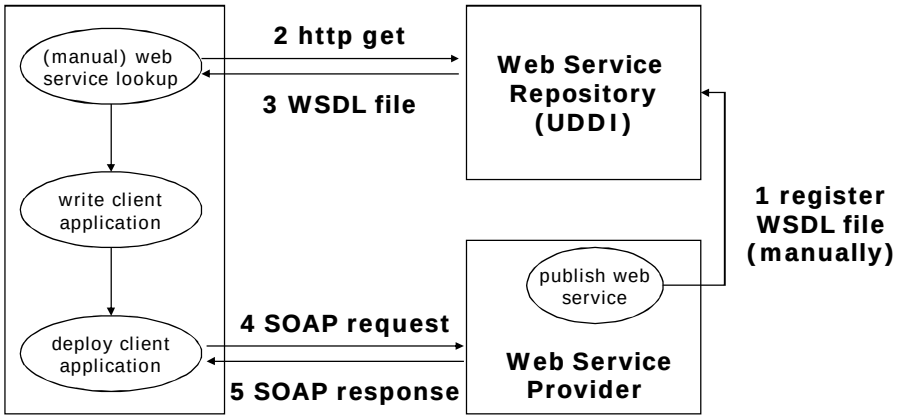

Application
Application Services
Business Process
Orchestration (BPEL)
Choreography (WS-C)
Quality of Service
Security (WS-Security)
Reliable Messaging (WS-RM)
Transactions (WS-AT)
Description
Addressing (WS-A)
Description (WSDL)
Policies (WS-Policy)
Messaging
Messaging (SOAP)
Transport
Transport (HTTP)
Network
Internet (TCP/IP)
Interoperability (WS-I)
Service Discovery (UDDI)
Metadata (WS-ME)
Overall Web Services Architecture & Standards
UNIVERSITY of STIRLING
Slide 15

15


Web Service Usage
"server"
Service provider
publish (WSDL)
bind (SOAP)
Service broker
Service requestor
find (UDDI)
"naming service"
"client"
UNIVERSITY of STIRLING
Slide 16

16

Web Services Usage Example



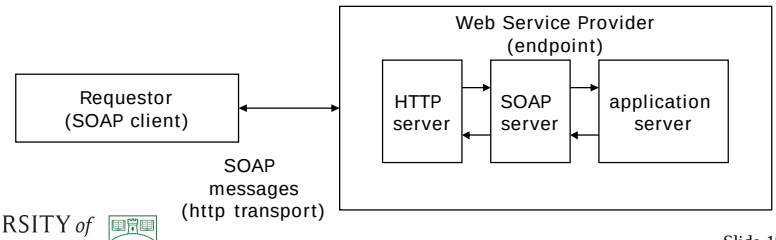
UNIVERSITY of STIRLING

Slide 17

17

Web Services Implementation

- Drastically simplified!
- Application Server (web service-enabled)
  - provides implementation of services and exposes it through WSDL/SOAP
  - implementation in Java, as EJB, as .NET (C#) etc.
- SOAP server, implements the SOAP protocol
- HTTP server, standard Web server
- SOAP client, implements the SOAP protocol on the client site



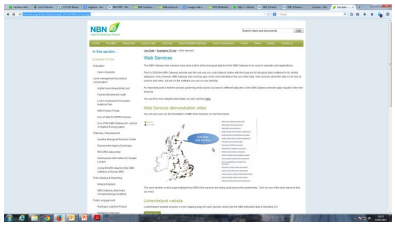
UNIVERSITY of STIRLING

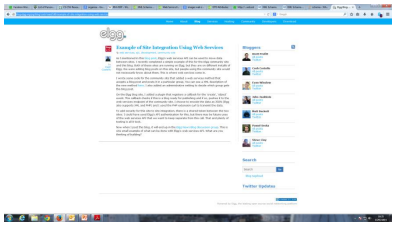
Slide 18

18

Examples

- http://aws.amazon.com/
  - Exposes world's largest product database through Web Services
  - Idea: let others figure out how to sell products for us
  - Associates program enables Web sites to link to Amazon.com and earn referral fees
- Some other interesting examples:
  - http://www.nbn.org.uk/Use-Data/Examples-Of-Use/Web-services.aspx
  - http://blog.elgg.org/pg/blog/cash/read/149/example-of-site-integration-using-web-services





UNIVERSITY of STIRLING

Slide 19

19

SOA/SOC

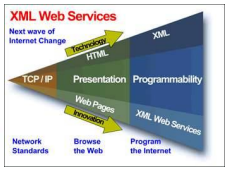
- web services support what is generically called SOA (Service Oriented Architecture) in support of SOC (Service Oriented Computing)
- SOC views a distributed system from the perspective of the services it offers and how these relate to each other
- previous work on distributed computing was object oriented, but this describes the (close) coupling among distributed objects
- instead, web services maintain a loose coupling – only the services offered by a distributed application are exposed:
  - legacy applications can easily be given a web service wrapping
  - the internal design of a web service can readily be changed
  - new services can be created by combining existing services

UNIVERSITY of STIRLING

Slide 20

20

Background: XML



- Meta-language
- XML uses **markup** to describe data.
  - So it is used to develop **our own** markup languages.
- Text files
- XML and HTML are for different purposes.
  - HTML is concerned with display (e.g. <body>, <p>, <ul>)
  - XML is concerned with data representation
- The markup facilities in HTML are **predefined**
- In XML **we define our own!**

UNIVERSITY of STIRLING

Slide 21

21

Background: XML cont.

- Some XML:

```
<pets>
  <animal>
    <type>Dog</type>
    <name>Jasper</name>
    <disposition>Enthusiastic</disposition>
  </animal>
  <animal>
    <type>Cat</type>
    <name>Barney</name>
    <disposition>Cynical</disposition>
  </animal>
</pets>
```

UNIVERSITY of STIRLING

Slide 22

22

Background XML cont.

- XML *attributes* further define *elements*:
  - `<price currency="GBP">1.15</price>`
  - `<price currency="USD">1.75</price>`
- #Required vs #Implied
- Well formed *vs* valid XML
- Two ways to specify the structure of XML documents:
  - Document Type Definitions (DTDs)
  - XML Schemas
- We need to specify
  - what elements (tags) will be used
  - how the various elements may be nested
  - what attributes they may contain
  - what types of data an element can contain

UNIVERSITY of STIRLING

Slide 23

23

DTD vs Schemas

- DTDs are rather limited in how they describe the content of data (e.g. it cannot be stated that a particular element contains integers or strings)
- **XSD** (**XML Schema Definition**, W3C version 1.0) is now the preferred way to define applications of XML, giving:
  - the structural elements in the data, defining the types of data they contain
  - definitions of data types, including sophisticated constraints on their contents
  - the relationships among the structural elements
  - the attributes of elements
- **XSI** (**XML Schema Instance**, W3C version 1.0) defines XML documents as instances of their schemas

UNIVERSITY of STIRLING

Slide 24

24

Namespaces

- a namespace is essentially just a unique string, though namespaces typically take the form of a URI (Uniform Resource Indicator)
- a namespace URI is typically a URL (Uniform Resource Locator) for where a schema is defined (e.g. www.cs.stir.ac.uk/schemas/mustard.xsd)
- a namespace URI may simply be a URN (Uniform Resource Name) that gives a (relatively) unique identifier (e.g. urn:MustardDefinition)
- since a namespace URI may be lengthy, it is commonly referred to by a short prefix – a string that is unique only within a document (e.g. mstd)

UNIVERSITY of STIRLING

Slide 25

25

Defining a Namespace

- namespace prefixes are defined and used as follows (xmlns means XML NameSpace):
  - xmlns:mstd="www.cs.stir.ac.uk/schemas/mustard.xsd"
  - ...
  - mstd:sequence
- mstd:sequence is an example of using a namespace prefix
- a document may declare a default namespace (xmlns on its own) for elements and attributes that are used without an explicit namespace prefix
- a document may also declare a target namespace (targetNamespace) that applies to all elements and attributes that it defines
- it is often convenient to have a prefix corresponding to the target namespace; this is typically, but need not be, named tns
- the namespace prefix for XML Schema Definition is usually xsd (though sometimes xs), while that for XML Schema Instance is usually xsi

UNIVERSITY of STIRLING

Slide 26

26

XML Schema

- With a schema, instead of having a definition in a file such as **note.dtd**, it is held in a file **note.xsd**
- When the XML file is specified by an XSD document held in **note.xsd**, the attributes within the **note** element in the XML file are:

```
<note xmlns="http://www.w3schools.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3schools.com note.xsd">
```

- The notation is therefore in XML.
- We have also defined namespaces.
- The default namespace in this example is: **http://www.w3schools.com**

UNIVERSITY of STIRLING

Slide 27

27

XML Schema

- Let us now look at the structure of the XSD document.
- The <schema> element is the root element of every XML Schema and it normally has attributes.
- The following specifies that elements and data types used in the schema (schema, element, complexType, sequence, string, boolean, etc.) come from the namespace: **http://www.w3.org/2001/XMLSchema**

and that the elements and data types from that namespace should be prefixed with xs:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.w3schools.com"
   xmlns="http://www.w3schools.com"
   elementFormDefault="qualified">
```

elements used by a XML document which are declared in this schema must be namespace **qualified**

UNIVERSITY of STIRLING

Slide 28

28

An example …

The root element in our XML is `note`. Suppose that it is composed of a sequence of four elements `to`, `from`, `heading` and `body`, it is an example of a ***complex element*** and its definition has the structure:

```
<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      ...
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- The term `sequence` indicates that the inner elements must appear in the specified order.

UNIVERSITY of STIRLING

Slide 29

29

XML Schema Datatypes

- XML Schema have a lot of built-in data types. Examples are:
  - **`xs:string`**
  - **`xs:decimal`**
  - **`xs:integer`**
  - **`xs:boolean`**
  - **`xs:date`**
  - **`xs:time`**
- This gives us a lot more control than we had with DTDs to specify what can go into our XML document. If the XML document contains a value of the wrong type then it will not validate.

UNIVERSITY of STIRLING

Slide 30

30

Example datatypes

- Suppose that we had the following simple elements in XML:

```
<lastname>Smith</lastname>
<age>36</age>
<dateborn>1968-03-27</dateborn>
```

- Note that date is given as YYYY-MM-DD
- The corresponding simple element definitions in XSD are:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

UNIVERSITY of STIRLING

Slide 31

31

Example cont.

In our **note** example, the inner elements are built-in simple elements. Their definition has the structure:

```
<xs:element name="aname" type="atype"/>
```

In fact, they are all strings. Hence the full definition of **note** is:

```
<xs:element name="note">
 <xs:complexType>
   <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
   </xs:sequence>
 </xs:complexType>
</xs:element>
```

UNIVERSITY of STIRLING

Slide 32

32

Defining new types

- So, here we have defined a new element called **`note`** and described its structure.
- It has an *anonymous* type.

- An alternative would be to define a new **`complexType`** (e.g. **`NoteType`**) and then define the **`note`** element as:

```
<xs:element name= "note" type= "NoteType"/>
```

This approach is much better if we are going to have several elements with the same structure (type).

UNIVERSITY of STIRLING

Slide 33

33

A new type

- We now define **`NoteType`** as:

```
<xs:complexType name="NoteType">
   <xs:sequence>
     <xs:element name="to" type="xs:string"/>
     <xs:element name="from" type="xs:string"/>
     <xs:element name="heading" type="xs:string"/>
     <xs:element name="body" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
```

UNIVERSITY of STIRLING

Slide 34

34

Another new type

- We can also define a new **`simpleType`**.
- We start from an existing **`simpleType`** (the base type) and impose a ***restriction*** by means of a ***facet***.
- Example facets are:
  - `maxInclusive` and `maxExclusive`
  - `minInclusive` and `minExclusive`
  - `pattern`
  - `enumeration`

UNIVERSITY of STIRLING

Slide 35

35

Restrictions on types

- Suppose that we wanted to restrict the range of allowable values in our `age` element.
- Instead of defining it as:

```
<xs:element name="age" type="xs:integer"/>
```

- We can define it as:

```
<xs:element name="age">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="13"/>
    <xs:maxInclusive value="19"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

```
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<xs:element name="initials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z][A-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- Can also define default or fixed values for a type:

```
<xs:element name="color" type="xs:string" default="red"/>
<xs:element name="color" type="xs:string" fixed="red"/>
```

UNIVERSITY of STIRLING

Slide 36

36

XML Schema summary

- schema data types use the following:
  - predefined types like **boolean**, **date**, **double**, **float**, **int**, **integer** (arbitrary length), **nonNegativeInteger**, **string**, **time**
  - **element** for a field in a type
  - **complexType** for structured types

```
<complexType name="fieldCount">                          <!-- start 'fieldCount' type -->
    <sequence>                                           <!-- start fields -->
    <element name="field"   type="xsd:string"/>          <!– 'field' field -->
    <element name="count" type="xsd:nonNegativeInteger"/> <!– 'count' field -->
    </sequence>                                          <!-- end fields -->
</complexType>                                           <!-- end 'fieldCount' type -->

<complexType name="analysis">                            <!-- start 'analysis' type -->
    <sequence>                                           <!-- start fields -->
    <element name="fieldCount" type="defs:fieldCount"
    minOccurs="0" maxOccurs="unbounded"/>                <!– 'fieldCount' array -->
    </sequence>                                          <!-- end fields -->
</complexType>                                           <!-- end 'analysis' type -->
```

UNIVERSITY of STIRLING

Slide 37

37