

# Computer Science E-259

XML with Java

## Lecture 10: XML Schema, Continued

11 April 2007

David J. Malan

`malan@post.harvard.edu`

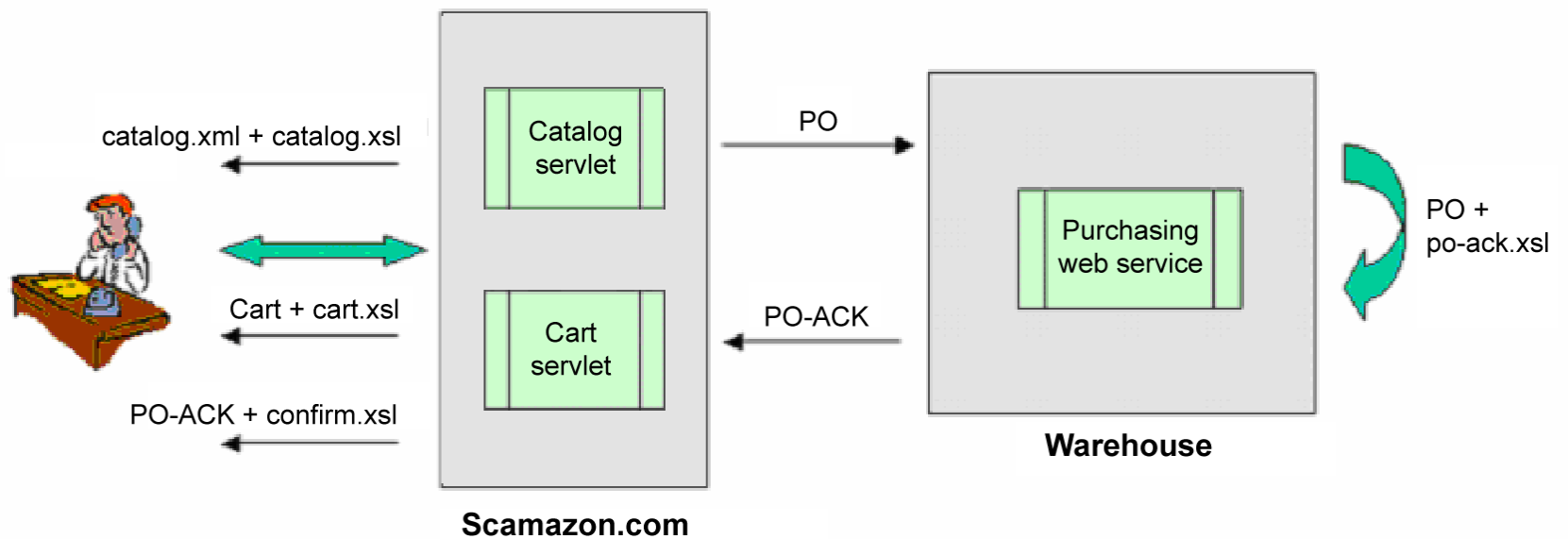
# Last Time

## XML Schema (Second Edition)

- XML Schema (Second Edition)
- Project 4

# Last Time

**scamazon.com**



# Last Time

## XML Schema (Second Edition)

- Declarations v. Definitions
- Global v. Local Components
- Element and Attribute Declarations
- Simple v. Complex Types
- Named v. Anonymous Types
- Type Definition Hierarchy
- Simple Types
- Complex Types
- Namespaces
  - Multiple
  - Default
  - Target
- Relating Instances to Schemas

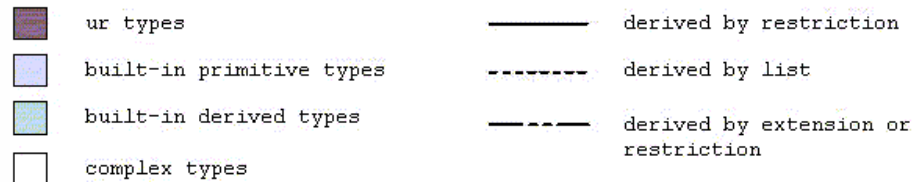
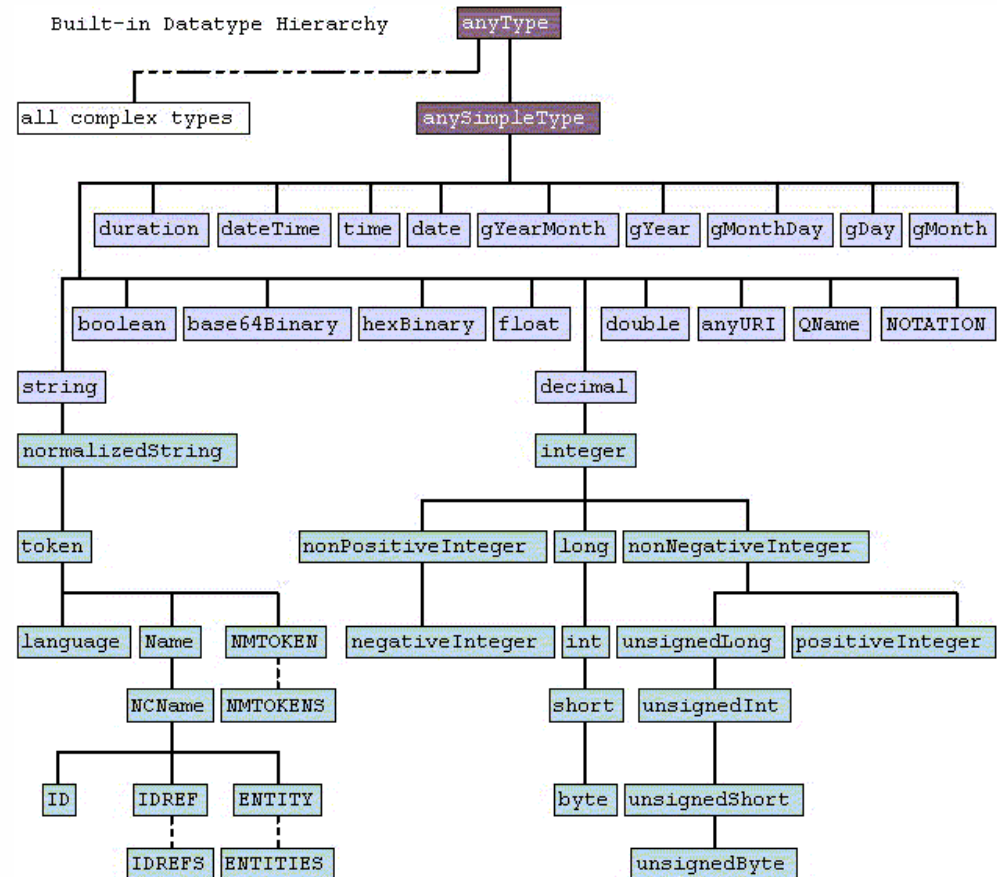
# Computer Science E-259

## This Time

- XML Schema 1.0, Continued

# Datatypes

## Built-In



# Structures

## Elements

```
<element
  abstract = boolean : false
  block = (#all | List of (extension | restriction | substitution))
  default = string
  final = (#all | List of (extension | restriction))
  fixed = string
  form = (qualified | unqualified)
  id = ID
  maxOccurs = (nonNegativeInteger | unbounded) : 1
  minOccurs = nonNegativeInteger : 1
  name = NCName
  nillable = boolean : false
  ref = QName
  substitutionGroup = QName
  type = QName
  {any attributes with non-schema namespace . . .}>
  Content: (annotation?, ((simpleType | complexType)?, (unique | key |
    keyref)*))
</element>
```

# Structures

## Attributes

```
<attribute
  default = string
  fixed = string
  form = (qualified | unqualified)
  id = ID
  name = NCName
  ref = QName
  type = QName
  use = (optional | prohibited | required) : optional
  {any attributes with non-schema namespace . . .}>
  Content: (annotation?, (simpleType?))
</attribute>
```



# Simple Types

## Elements of Unrestricted, Simple Types

- Canonical Declaration  
`<xsd:element name="..." type="..." />`
- Examples of Instances  
`<name>John Harvard</name>`  
`<year>1636</year>`
- Examples of Declarations  
`<xsd:element name="lastname" type="xsd:string" />`  
`<xsd:element name="age" type="xsd:integer" />`

# Simple Types

## Attributes of Unrestricted, Simple Types

- Canonical Declaration  
`<xsd:attribute name="..." type="..." />`
- Example of an Instance  
`<student gender="male">John Harvard</student>`
- Example of a Declaration  
`<xsd:attribute name="gender" type="xsd:string" />`

# Simple Types

## Attributes with Default or Fixed Values

- In the absence of a value, a default can be assigned  
`<xsd:attribute name="country" type="xsd:string" default="US"/>`
- A value, if present and declared fixed, must appear as declared;  
in the absence of a value, the declared will be assigned  
`<xsd:attribute name="country" type="xsd:string" fixed="US"/>`

# Simple Types

## Optional and Required Attributes

- By default, attributes are optional  
`<xsd:attribute name="country" type="xsd:string" use="optional"/>`
- If declared required, attribute must be present  
`<xsd:attribute name="country" type="xsd:string" use="required"/>`

# Simple Types

## Restrictions through Facets

Facet	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) are handled

# Simple Types

## Restricting by Value

```
<xsd:element name="year">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="2007"/>
      <xsd:maxInclusive value="2010"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

```
<xsd:element name="year">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minExclusive value="2006"/>
      <xsd:maxExclusive value="2011"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

# Simple Types

## Restricting by Value

```
<xsd:element name="major">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="English"/>
      <xsd:enumeration value="Math"/>
      <xsd:enumeration value="Physics"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

# Simple Types

## Restricting by Value

```
<xsd:element name="major" type="majors"/>
```

```
<xsd:simpleType name="majors">  
  <xsd:restriction base="xsd:string">  
    <xsd:enumeration value="English"/>  
    <xsd:enumeration value="Math"/>  
    <xsd:enumeration value="Physics"/>  
  </xsd:restriction>  
</xsd:simpleType>
```



# Simple Types

## Restricting by Pattern

```
<xsd:element name="choice">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[abcd]"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

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

# Simple Types

## Restricting by Pattern

```
<xsd:element name="gender">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="male|female"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="password">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[a-zA-Z0-9]{8}"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

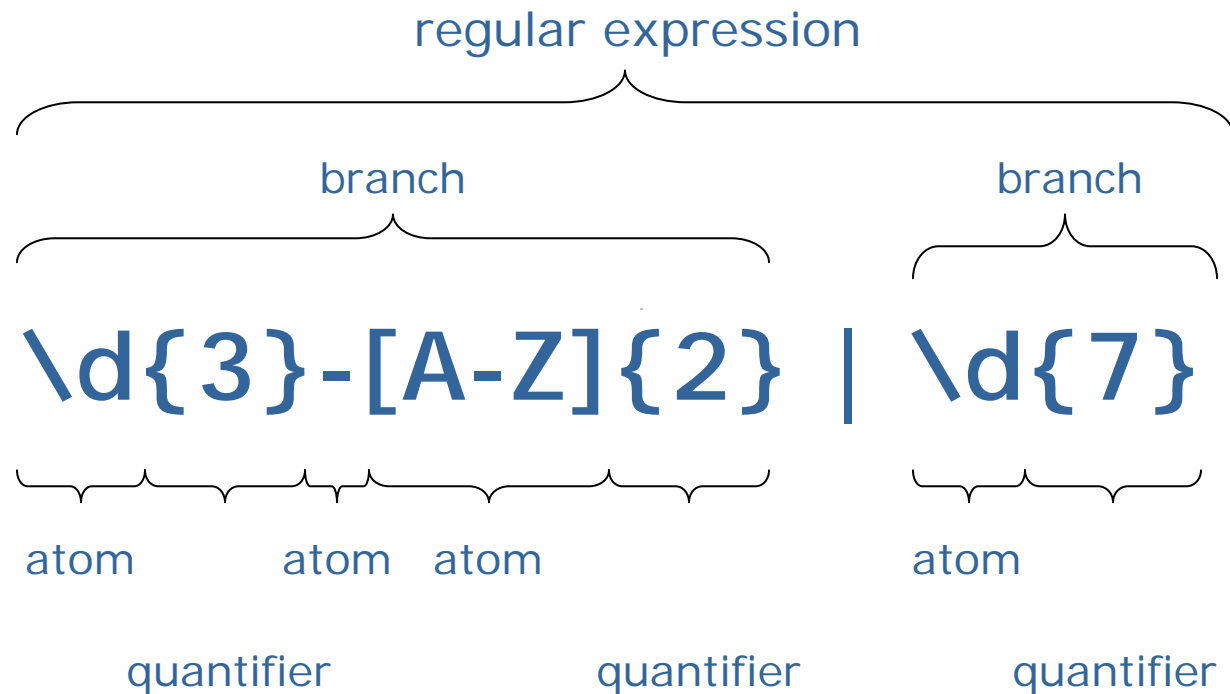
# Simple Types

## Restricting by Pattern

```
<xsd:simpleType name="ProdNumType">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="\d{3}-[A-Z]{2} | \d{7}"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

# Simple Types

## Regular Expressions



# Simple Types

## Regular Expressions

- An atom describes one or more character through
  - a single normal character (*e.g.*, **a** or **c**)
  - a parenthesized regular expression (*e.g.*, (**a|c**))
  - an escape, such as
    - \n** for newline, **\?** for **?**, **.** for any character but **\n** and **\r**
    - \d** for any digit, **\D** for any character but a digit
    - \s** for any whitespace character
  - a character class expression
    - [abc]** matches any of a list of characters
    - [0-9]** or **[a-z]** matched any character from a range
- A quantifier indicates how many times an atom should repeat (*e.g.*, **?**, **\***, **+**, **{n}**, **{n,}**, **{n,m}**)

# Simple Types

## Restricting Whitespace

```
<xsd:element name="name">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:whiteSpace value="preserve"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

```
<xsd:element name="name">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:whiteSpace value="replace"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

```
<xsd:element name="name">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:whiteSpace value="collapse"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

# Simple Types

## Restricting by Length

```
<xsd:element name="password">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

```
<xsd:element name="password">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength value="5"/>
      <xsd:maxLength value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

# Simple Types

## Lists

```
<grades>90 85 77 100 99 45</grades>
```

```
<xsd:element name="grades">  
  <xsd:simpleType>  
    <xsd:list itemType="xsd:nonNegativeInteger"/>  
  </xsd:simpleType>  
</xsd:element>
```



# Simple Types

## Unions

```
<xsd:element name="jeans_size">
  <xsd:simpleType>
    <xsd:union memberTypes="sizebyno sizebystring"/>
  </xsd:simpleType>
</xsd:element>
```

```
<xsd:simpleType name="sizebyno">
  <xsd:restriction base="xsd:positiveInteger">
    <xsd:maxInclusive="42"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="sizebystring">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="small"/>
    <xsd:enumeration value="medium"/>
    <xsd:enumeration value="large"/>
  </xsd:restriction>
</xsd:simpleType>
```

# Complex Types

## Content Types for Elements

- Simple (*i.e.*, children include text only)  
`<name>Jerry Seinfeld</name>`
- Element-Only (*i.e.*, children include elements only)  
`<name><first>Jerry</first><last>Seinfeld</last></name>`
- Mixed (*i.e.*, children contain text and/or elements)  
`<p><name>Jerry Seinfeld</name> is a <title>comedian</title>.</p>`
- Empty (*i.e.*, no children)  
`<comedian name="Jerry Seinfeld"/>`

# Complex Types

## Simple Content

```
<xsd:element name="...">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:restriction base="...">
        ....
      </xsd:restriction>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="...">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="...">
        ....
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

# Complex Types

## Element-Only Content

```
<student>  
  <name>John Harvard</name>  
  <year>1636</year>  
</student>
```

```
<xsd:element name="student">  
  <xsd:complexType>  
    <xsd:sequence>  
      <xsd:element name="name" type="xsd:string"/>  
      <xsd:element name="year" type="xsd:integer"/>  
    </xsd:sequence>  
  </xsd:complexType>  
</xsd:element>
```

# Complex Types

## Element-Only Content

```
<name>
  <first>John</first>
  <last>Harvard</last>
</name>

<xsd:element name="name">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="first" type="xsd:string"/>
      <xsd:element name="last" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

# Complex Types

## Mixed Content

```
<letter>
Dear Mr.<name>John Smith</name>.
Your order <orderid>1032</orderid>
will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

```
<xsd:element name="letter">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="orderid" type="xsd:positiveInteger"/>
      <xsd:element name="shipdate" type="xsd:date"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

# Complex Types

## Empty Content

```
<foo bar="baz" />
```

```
<xsd:element name="foo">  
  <xsd:complexType>  
    <xsd:attribute name="bar" type="xsd:string"/>  
  </xsd:complexType>  
</xsd:element>
```

# Complex Types

## Model Groups

- A **sequence** group of element declarations is used to indicate the order in which the elements should appear
- A **choice** group of element declarations is used to indicate that only one of the elements should appear
- An **all** group is used to indicate that all elements should appear, in any order, but no more than once each



# Complex Types

## The sequence Model

```
<xsd:element name="name">
  <xsd:complexType>
    <xsd:sequence minOccurs="0">
      <xsd:element name="first" type="xsd:string"/>
      <xsd:element name="last" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

# Complex Types

## The sequence Model

```
<xsd:element name="name">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="first" type="xsd:string"/>
      <xsd:element name="last" type="xsd:string"/>
      <xsd:element name="nick" type="xsd:string"
        maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

# Complex Types

## The choice Model

```
<xsd:complexType name="ProductType">  
  <xsd:choice minOccurs="0" maxOccurs="3">  
    <xsd:element name="size" type="SizeType"/>  
    <xsd:element name="color" type="ColorType"/>  
  </xsd:choice>  
  <xsd:attribute name="effDate" type="xsd:date"/>  
</xsd:complexType>
```

# Complex Types

## Nesting Models!

```
<xsd:complexType name="ProductType">
  <xsd:sequence>
    <xsd:element name="number" type="xsd:integer"/>
    <xsd:choice minOccurs="0" maxOccurs="3">
      <xsd:element name="size" type="SizeType"/>
      <xsd:element name="color" type="ColorType"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="effDate" type="xsd:date"/>
</xsd:complexType>
```

# Complex Types

## The all Model

```
<xsd:element name="name">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="first" type="xsd:string"/>
      <xsd:element name="last" type="xsd:string"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

# Complex Types

## Defining Model Groups

```
<xsd:group name="persongroup">
  <xsd:sequence>
    <xsd:element name="firstname" type="xsd:string"/>
    <xsd:element name="lastname" type="xsd:string"/>
    <xsd:element name="birthday" type="xsd:date"/>
  </xsd:sequence>
</xsd:group>

<xsd:element name="person" type="personinfo"/>
<xsd:complexType name="personinfo">
  <xsd:sequence>
    <xsd:group ref="persongroup"/>
    <xsd:element name="country" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

# Complex Types

## Defining Attribute Groups

```
<xsd:attributeGroup name="personattrgroup">
  <xsd:attribute name="firstname" type="xsd:string"/>
  <xsd:attribute name="lastname" type="xsd:string"/>
  <xsd:attribute name="birthday" type="xsd:date"/>
</xsd:attributeGroup>

<xsd:element name="person">
  <xsd:complexType>
    <xsd:attributeGroup ref="personattrgroup"/>
  </xsd:complexType>
</xsd:element>
```

# Complex Types

## Extending Simple Types

```
<xsd:simpleType name="size">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="small" />
    <xsd:enumeration value="medium" />
    <xsd:enumeration value="large" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="jeans">
  <xsd:simpleContent>
    <xsd:extension base="size">
      <xsd:attribute name="sex">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="male" />
            <xsd:enumeration value="female" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```



# Complex Types

## Extending Complex Types

```
<xsd:complexType name="ProductType">
  <xsd:sequence>
    <xsd:element name="number" type="ProdNumType"/>
    <xsd:element name="name" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="ShirtType">
  <xsd:complexContent>
    <xsd:extension base="ProductType">
      <xsd:choice maxOccurs="unbounded">
        <xsd:element name="size" type="SizeType"/>
        <xsd:element name="color" type="ColorType"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

# Complex Types

## Allowing for Any Elements

```
<xsd:element name="name">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="first" type="xsd:string"/>
      <xsd:element name="last" type="xsd:string"/>
      <xsd:any minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

# Complex Types

## Allowing for Any Attributes

```
<xsd:element name="name">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="first" type="xsd:string"/>
      <xsd:element name="last" type="xsd:string"/>
    </xsd:sequence>
    <xsd:anyAttribute/>
  </xsd:complexType>
</xsd:element>
```

# Complex Types

## Allowing for Substitutes

```
<customer>  
  <name>John Smith</name>  
</customer>
```

```
<cliente>  
  <nome>Giovanni Smith</nome>  
</cliente>
```

```
<xsd:element name="name" type="xsd:string"/>  
<xsd:element name="nome" substitutionGroup="name"/>  
<xsd:complexType name="custinfo">  
  <xsd:sequence>  
    <xsd:element ref="name"/>  
  </xsd:sequence>  
</xsd:complexType>  
<xsd:element name="customer" type="custinfo"/>  
<xsd:element name="cliente" substitutionGroup="customer"/>
```

# XML Schema (Second Edition)

## Summary

Element	Explanation
<a href="#"><u>all</u></a>	Specifies that the child elements can appear in any order. Each child element can occur 0 or 1 time
<a href="#"><u>annotation</u></a>	Specifies the top-level element for schema comments
<a href="#"><u>any</u></a>	Enables the author to extend the XML document with elements not specified by the schema
<a href="#"><u>anyAttribute</u></a>	Enables the author to extend the XML document with attributes not specified by the schema
<a href="#"><u>appInfo</u></a>	Specifies information to be used by the application (must go inside annotation)
<a href="#"><u>attribute</u></a>	Defines an attribute
<a href="#"><u>attributeGroup</u></a>	Defines an attribute group to be used in complex type definitions
<a href="#"><u>choice</u></a>	Allows only one of the elements contained in the <choice> declaration to be present within the containing element
<a href="#"><u>complexContent</u></a>	Defines extensions or restrictions on a complex type that contains mixed content or elements only
<a href="#"><u>complexType</u></a>	Defines a complex type element
<a href="#"><u>documentation</u></a>	Defines text comments in a schema (must go inside annotation)
<a href="#"><u>element</u></a>	Defines an element
<a href="#"><u>extension</u></a>	Extends an existing simpleType or complexType element
<a href="#"><u>field</u></a>	Specifies an XPath expression that specifies the value used to define an identity constraint

# XML Schema

## Summary

<a href="#"><u>group</u></a>	Defines a group of elements to be used in complex type definitions
<a href="#"><u>import</u></a>	Adds multiple schemas with different target namespace to a document
<a href="#"><u>include</u></a>	Adds multiple schemas with the same target namespace to a document
<a href="#"><u>key</u></a>	Specifies an attribute or element value as a key (unique, non-nullable, and always present) within the containing element in an instance document
<a href="#"><u>keyref</u></a>	Specifies that an attribute or element value correspond to those of the specified key or unique element
<a href="#"><u>list</u></a>	Defines a simple type element as a list of values
<a href="#"><u>notation</u></a>	Describes the format of non-XML data within an XML document
<a href="#"><u>redefine</u></a>	Redefines simple and complex types, groups, and attribute groups from an external schema
<a href="#"><u>restriction</u></a>	Defines restrictions on a simpleType, simpleContent, or a complexContent
<a href="#"><u>schema</u></a>	Defines the root element of a schema
<a href="#"><u>selector</u></a>	Specifies an XPath expression that selects a set of elements for an identity constraint
<a href="#"><u>sequence</u></a>	Specifies that the child elements must appear in a sequence. Each child element can occur from 0 to any number of times
<a href="#"><u>simpleContent</u></a>	Contains extensions or restrictions on a text-only complex type or on a simple type as content and contains no elements
<a href="#"><u>simpleType</u></a>	Defines a simple type and specifies the constraints and information about the values of attributes or text-only elements
<a href="#"><u>union</u></a>	Defines a simple type as a collection (union) of values from specified simple data types
<a href="#"><u>unique</u></a>	Defines that an element or an attribute value must be unique within the scope

# Next Time

Web Services, SOAP 1.2, and WSDL 1.1

**Amazon Exclusive!!**  
**Order a Segway now!**  
**It's only at Amazon**



# Computer Science E-259

XML with Java

## Lecture 10: XML Schema, Continued

11 April 2007

David J. Malan

`malan@post.harvard.edu`