

XML & JSON

CS 240 – Advanced Programming Concepts

Background

- XML and JSON are two standard, textual data formats for representing arbitrary data
 - XML stands for “eXtensible Markup Language”
 - JSON stands for “JavaScript Object Notation”
- Both are commonly used in practice
- XML came first
- JSON, which uses JavaScript syntax, became popular for representing data in web applications and services
 - If you’re using JavaScript, JSON is an obvious choice
- Both formats are reasonable choices, although some people have strong biases
- Most programming languages have libraries for parsing and generating both XML and JSON
- You should be familiar with both

Structure of XML Documents

- Header
- Root Element
- Start Tags / End Tags
- Element Contents
 - Child Elements
 - Text
 - Both (mixed contents)
- Element Attributes
- Comments
- Entity References
- Examples (your browser will interpret some of the tags, so select “View Page Source”)
 - [Element-heavy](#)
 - [Attribute-heavy](#)
 - [Hybrid](#)

Predefined Entities

Entity name	Character	Specified By
quot	"	"
amp	&	&
apos	'	'
lt	<	<
gt	>	>

Structure of JSON Documents

- Supported data types: Objects, Arrays, Numbers, Strings, Boolean, Null
- Objects delimited by { ... } with comma-separated properties in between
 - { “name”: “Bob”, “age”: 32, “alive”: true }
- Arrays delimited by [...] with comma-separated elements in between
 - [“testing”, 1, 2, 3, { “gpa”: 3.4 }]
- Examples
 - [Verbose](#)
 - [Simple](#)

Parsing XML & JSON Data

- Most languages provide both XML and JSON parsers, so there's no need to write your own
- Three Major Types of Parsers
 - DOM Parsers
 - Convert XML or JSON text to an in-memory tree data structure (the tree is called a DOM, or “document object model”)
 - After running the parser to create a DOM, traverse the DOM to extract the data you want
 - Stream Parsers
 - Tokenizers that return one token at a time from the XML or JSON data file
 - Serializers / Deserializers
 - Use a library to convert from XML or JSON to Java Objects (and vice versa)
 - Jackson for XML
 - Gson or Jackson for JSON

JSON Parsing Examples

- Source File: [cd_catalog.json](#)
- Domain Object: [CD](#)
- DOM Parser Example
 - [JsonDomParserExample.java](#)
- Stream Parser Example
 - [JsonStreamParserExample.java](#)

JSON Parsing Examples (cont.)

- Deserialization using the Gson library from Google
- Simple Deserialization Example
 - Source File: [cd_catalog_simple.json](#)
 - Domain Objects: [Catalog](#), [CD](#)
 - [JsonSimpleObjectDeserializationExample.java](#)
- Complex Deserialization Example
 - Source File: [cd_catalog.json](#)
 - Domain Objects: [Catalog](#), [CD](#)
 - [JsonObjectDeserializationExample.java](#)

XML Parsing Examples

- Source File: [cd_catalog.xml](#)
- Domain Object: [CD](#)
- DOM Parser Example
 - [XmlDomParserExample.java](#)
- Stream Parser Example
 - [XmlStreamParserExample.java](#)

XML Parsing Examples (cont.)

- Deserialization using Jackson
 - Source File: [cd_catalog.xml](#)
 - Domain Objects: [Catalog](#), [CD](#)
 - [XmlObjectDeserializationExample.java](#)

Generating XML & JSON Data

- Programs often need to generate (or create) XML and JSON data
- You can print XML or JSON data yourself (it's just text), but it's better to use a library that handles tricky special cases like escaping special characters)
- Three Ways to Generate XML or JSON
 - Create DOM tree in memory, and then tell the tree to write itself to text
 - Write the data as a stream, one token at a time
 - Use a “serializer” class that converts Java objects to XML or Json

JSON Generation Examples

- Domain Objects: [CDFactory](#) (used to generate Java objects to be converted), [Catalog](#), [CD](#)
- DOM Generator Example
 - [JsonDomGenerationExample.java](#)
- Stream Generator Example
 - [JsonStreamGenerationExample.java](#)

JSON Generation Examples (cont.)

- Serialization using Gson
- Simple Serialization Example
 - Domain Objects: [CDFactory](#), [Catalog](#), [CD](#)
 - [JsonSimpleObjectSerializationExample.java](#)
- Complex Deserialization Example
 - Domain Objects: [CDFactory](#), [Catalog](#), [CD](#)
 - [JsonObjectSerializationExample.java](#)

XML Generation Examples

- Domain Objects: [CDFactory](#), [Catalog](#), [CD](#)
- DOM Generator Example
 - [XmlDomGenerationExample.java](#)
- Stream Generator Example
 - [XmlStreamGenerationExample.java](#)

XML Generation Examples (cont.)

- Serialization using the Jackson library
- Domain Objects: [CDFactory](#), [Catalog](#), [CD](#)
- [XmlObjectSerializationExample.java](#)