CS 240: JSON Stream Parser Transcript

[00:00:00] The first type of JSON parser is a stream parser.

[00:00:05] Now, one thing I should emphasize is that if you need to process JSON data in your applications, you should not write your own parser, you should use a library that already exists.

[00:00:17] There's, there's lots of Jason parsing libraries out there in the world.

[00:00:21] I mean, you could write your own character by character, Jason parsing algorithms if you wanted to, but that would almost certainly be a mistake.

[00:00:29] Um So I encourage you to, to use libraries that uh do that low level parsing for you in this class, we're going to use a library named JSON, which is a Jason Parsing library from Google.

Start visual description. The professor demonstrates how to use the JSON library from Google for low-level parsing. He emphasizes the importance of using existing libraries for JSON parsing instead of writing your own parser. End visual description.

[00:00:41] And some of the examples we're about to look at.

[00:00:44] Um We'll use the JSON library as well as other libraries.

[00:00:48] Now, a streaming parser is essentially a tokes.

[00:00:52] So if you've taken CS 236 you probably implemented a, as part of the, the CS 236 project, you probably implemented a tokenizer or a lexical analyzer or a scanner or whatever term they use for it.

[00:01:07] And basically what a tokenizer does is it's, it's just a class that you can connect to a JSON file.

[00:01:13] And what it will do is it will return the tokens in the Jason file.

[00:01:18] Now, what are the tokens in adjacent file? Well, tokens are just the little pieces or bits of information that are in a file. And so, in the case of Jason, uh the first token in adjacent file, at least this Jason file is a, a start object or begin object token.

[00:01:36] So that left curly brace conceptually is a begin object or start object token.

[00:01:41] So if you're using a streaming parser on this file, the first thing that the parcel would return to you would be a begin object token.

[00:01:49] And then in your program, you could do whatever you want with that information.

[00:01:52] The next token would be a property name.

[00:01:55] So the next token would be the catalog property name in this case.

[00:01:58] And so the next time you call the par it will give you back a property name and it, and the name associated with that property would be cataloged.

[00:02:06] And the next token, if you keep reading the file would be a begin array or start array token.

[00:02:12] And then as we continue on, we'd have a start object.

[00:02:15] We'd have a property name, begin object, property name, string, property name, string, property name, string, property name, string, property, name, number, property, name, number and object and object.

[00:02:32] And then after that end object, we would uh next get a begin object.

[00:02:36] And so really a, a streaming parser is a fairly simple class that just feeds you or feed your program the next bit or token of information that that's in the file. And then in your program you can, you can parse. Um, well, you can process those tokens that come out of the parser in any way you want.

[00:02:55] And I'll show you an example.

[00:02:56] Um, here in a minute now, a streaming Ure is so simple that you think, well, it's almost not even worth using because it doesn't do that much.

[00:03:04] And, and that's partially true.

[00:03:05] But one thing a streaming parser does do is it handles um strings properly, like there's different syntax for escaping or quoting characters and strings in, in JSON and so forth.

[00:03:18] And so using a parser, a streaming parser would at least handle some of the complexities of string handling and also parsing floating points.

[00:03:27] And so it, it's still worth your while to use a streaming parser even if um it doesn't seem like it does a ton for you. Now, as an example of a stream parser in the example code, there's an example called JSON stream parser example.

[00:03:47] And in this case, we're using um a library Jax dot JSON library to do some stream parsing.

Start visual description. The professor shows an example of a stream parser using the Jax. JSON library. He explains how the parser reads tokens from a JSON file and processes them to create Java CD objects. End visual description.

[00:04:02] There's different libraries and they all have stream parsers, and they have different names for their stream parsing classes in this particular library.

[00:04:09] The name of the stream parser class is Jason parser and that's their stream parser. So, in this case, uh we have the parse method, what it's going to do as input, it’s going to receive a file and that’s a file of Jason data and it’s going to return back a list of CD objects.

[00:04:27] Now, of course, these are java CD objects.

[00:04:30] So the input to this method is really going to be this, this file that we looked at earlier.

[00:04:35] So it’s got a catalog wrapper object that contains an array of CD wrapper objects.

[00:04:40] And each CD wrapper object has a CD inside of it.

[00:04:44] So let’s just look at the code that uses a streaming parser to parse that that data.

[00:04:49] So our goal here is to parse the whole file and return a list of CD java objects.

[00:04:56] And so to do that, we start out with an empty list of CD objects.

[00:05:00] Then we open the file, and we wrap that file in a buffered reader so that our, our input is more efficient.

[00:05:08] And then what we do is we go to the Jason class and call the create parser method and we pass it the, the file, the JSON file and it gives us back a Jason parser object.

[00:05:18] So now that we’ve got our streaming parser object, we can start to read the tokens out of the out of the file.

[00:05:23] Now, the basic algorithm for this is just to read through the file and collect all these attributes for each CD.

[00:05:30] So we want the title Artist Country Company price and year for each one.

[00:05:34] And once we have those attributes, we can then create a, a Java CD object with those values and then add that object to the list.

[00:05:42] And so what you'll see here is that we essentially um have a wild loop.

[00:05:48] And so what this Y loop says is while there are more tokens in the file, keep looping.

[00:05:54] And so when you use a streaming parser, you typically have a loop like that, that just reads the tokens out of the file.

[00:06:00] Now, just to make this a little more concrete, I'm going to show you the, the method interface for the JSON parser that we're using.

[00:06:08] So if we look here, the, the streaming parser class has a few essential methods.

Start visual description. The professor explains the methods of the streaming parser class, such as hasNext and next, and how they are used to read tokens from a JSON file. He demonstrates how to handle different types of tokens and their associated values. End visual description.

[00:06:13] It has the has next method and that method will tell you if there's any more tokens remaining in the file.

[00:06:19] So that's what controls our loop.

[00:06:22] So we call HAZ next to see if there's any more tokens.

[00:06:26] Now in this library, they don't call them tokens, they call them events, but it's the same thing as a token.

[00:06:30] So if we look at the event, you can see here that these are all the different kinds of tokens or events that can come out of a, a JSON file starter, a start object key name. So key name would be a property name.

[00:06:44] And then the values would be strings, numbers, true, false null, then we have end object array. So those are all the tokens that, that can be in adjacent file.

[00:06:55] And so as we sit here in our loop, we're just going to call, has next until it returns falls.

[00:07:01] And then as long as there's more tokens, we're going to call next to get the next event.

[00:07:05] And then um some events have a value associated with them.

[00:07:10] So a start object or a starter, those tokens don't have any particular value.

[00:07:15] But when we um start reading the values of properties, then we're going to get actual data values in the tokens.

[00:07:23] And so depending on the token type, um you, you get back, um you can then call these, these additional methods to get the values associated with those tokens.

[00:07:35] So for example, if it's a, a key name or a property name token, you can call, get string and that will give you the name of the property.

[00:07:44] Now, if it returns a string token, then you can call, get string to get the value of, of the string.

[00:07:52] If it's a number token, you can call, get in or get long or uh other methods that they provide here to get the value of the number, the number could also be a floating point and so on and so forth. And so, if a token has a data value associated with it, you can call these additional methods to get the data value out of it.

[00:08:14] So that way you can get the property names, and you can get the values of the properties, which could be any of the possible uh JSON data types.

[00:08:22] So that kind of gives you an idea of what a streaming parser interface looks like.

[00:08:27] So if we go back to our CD parsing code, then we create our parser, we just declare these variables for all the CD properties.

[00:08:35] And then we just go into our wild loop while there's, while there's more tokens continue parsing, we call the next method to get the next token from the parser.

[00:08:44] And then we just have a big switch.

[00:08:48] So here we call, get strained, get the, the name of the, the property and then we just do a switch on the property name.

[00:08:54] So it could be a CD property, um could be title, artist country or, or any of those things that we see in the file.

[00:09:03] Now, one thing we have to do as we parse is we have to kind of keep track of where we're at in the nested structure.

[00:09:08] So when we begin, we're up here at the, the catalog level. But once we um start parsing out the individual C DS, then we have these wrapper objects to deal with.

[00:09:16] And so we have some variables in our code that keep track of what level of nesting we're at.

[00:09:21] Um So that's what some of these variables are like, um in CD, that's a Boolean variable we use to determine whether we're inside a CD object or whether we're farther out inside the catalog or, or where we're at.

[00:09:37] But for the most part, all we're doing is we're looking for the titles and the artists in the country and in all those, those attributes and once we um get to an end object token, then we know we have all the data that we need to create a new CD object and added to our list.

[00:09:53] And um that's essentially the style that you use to, to use a streaming parser.

[00:10:01] Now, one thing you may notice is that using a stream parser is fairly low level, it’s not particularly easy or convenient.

[00:10:07] You’re still working at the token level, which is um just one level above the character level.

[00:10:13] So for most applications, you’re probably not going to use a streaming parser because it is more laborious to use than some of the other parsers.

[00:10:20] But there are use cases or applications where a streaming parser is appropriate.

[00:10:25] For example, let’s suppose you have a, a really large JSON file and most of the data that’s in that file you don’t really care about.

[00:10:32] There’s just maybe one or two pieces of information in that whole file that you need.

[00:10:38] So if, if you’re going to, if you want to plow through a big Jason file and just look for the one or two things you actually care about the streaming parcel would be a very efficient way to do that because you’re just going to throw away most of the data anyway.

[00:10:50] So for some applications, a streaming parser would be appropriate.

[00:10:54] But usually you’re going to want to use something that’s a little more convenient like a Dom parser, um which we’ll talk about next.

[00:11:03] So far, we’ve been learning how to use a streaming parser to parse the tokens out of a JSON file.

[00:11:08] Well, there’s also the problem of generating or creating JSON data. So, what if I have a data structure and I want to convert it or save it to, to JSON file? For example, if I have one of my catalog CD catalogs. So, in my program in main memory, I’ve got this catalog java object inside of it.

[00:11:28] It's got a list of CD Java objects.

[00:11:30] Let's suppose I want to save that catalog to a Jason file or export it as a Jason file.

[00:11:39] Well, how would I do that? Well, one approach is to just do it yourself with no help from a library. You could, you could easily iterate over the catalog and the, the contained C DS and you could print out the right um Jason syntax and and do that yourself.

[00:11:54] But again, it's, it's kind of hard to get some of the details, right? Especially when you're dealing with strings and strings, character, quoting and escaping and so forth.

[00:12:04] And so it's, it's even uh in this scenario, it's, it's best to use a library.

[00:12:11] So the, the output analog to uh streaming parser would be a class that lets you write JSON tokens to a file.

[00:12:21] So for example, in the go library that we're going to use um in this class, they actually have a class named um JSONWriter.

[00:12:34] So if you look in the, the Jason library, they have a class named JSONWriter.

Start visual description. The professor introduces the JSONWriter class from the Go library, which is used to write JSON tokens to a file. He shows how to use various methods of the JSONWriter class to generate JSON data and save it to a file. End visual description.

[00:12:39] And it's really just AAA class that you hook up to a file and then it lets you write Jason tokens to a file. So, for example, if I wanted to output this data, uh this example shows the code that you could use to do that.

[00:12:54] And what you can see here is that uh they're just really writing tokens one at a time.

[00:13:00] So for example, beginner a, so writer dot beginner A that writes a left score bracket to the file.

[00:13:08] Uh begin object down here that that method writes uh left curly brace to the file and so on and so forth.

[00:13:17] But they've got a method on this writer class for writing basically every kind of token that you, you can write to adjacent file.

[00:13:22] So the name method is used to write a property name.

[00:13:26] Um And, and so if you look at the documentation, you can see that really just has a bunch of methods for writing different kinds of tokens, beginner, a begin object end array and object.

[00:13:38] Um And they've got extra stuff here too. But for the most part, you can write a null value, you can uh write strings.

[00:13:48] So you can call the value method to write a billion-value method to write a number or a double um so on and so forth.

[00:13:55] So they've just got a bunch of methods for writing the various kinds of tokens.

[00:13:58] So if you need to generate some uh Jason data or save it, then this is one way that, that you can do that a little bit more easily.