**aParse - User Manual (C#)**

Reference: <http://www.parse2.com/manual-cs.shtml>

Example: <http://www.parse2.com/examples.shtml>

This describes the steps involved in the implementation and use of an **aParse** generated Java parser.

|  |  |
| --- | --- |
| [Write Grammar](http://www.parse2.com/manual-cs.shtml#WriteGrammar) | Use the [ABNF](http://www.parse2.com/abnf.shtml) metalanguage to define the syntax of a protocol. |
| [Generate Parser](http://www.parse2.com/manual-cs.shtml#GenerateParser) | Use **aParse** to generate the parser source code. |
| [Test Parser](http://www.parse2.com/manual-cs.shtml#TestParser) | Use the parser to parse instances of the protocol and verify whether the grammar correctly describes the protocol. |
| [Produce Visitor](http://www.parse2.com/manual-cs.shtml#ProduceVisitor) | Produce a visitor for instances of the protocol that performs the desired conversion or translation. |
| [Employ Parser](http://www.parse2.com/manual-cs.shtml#EmployParser) | Make use of the parser and visitor in a program. |

Additional information can also be found in the following sections.

|  |  |
| --- | --- |
| [External Rules](http://www.parse2.com/manual-cs.shtml#ExternalRules) | Using external user defined rules. |
| [Options](http://www.parse2.com/manual-cs.shtml#Options) | **aParse** supported options. |
| [Apache Ant](http://www.parse2.com/manual-cs.shtml#Ant) | Invoking **aParse** from [Apache Ant](http://ant.apache.org/). |

**Write Grammar**

Using the [ABNF](http://www.parse2.com/abnf.shtml) metalanguage, define the syntax of a protocol in a text file.

For example, the definition of a 24 hour clock in clock.abnf.

**Generate Parser**

Use **aParse** to generate the Clock parser source code.

java -cp aparse.jar com.parse2.aparse.Parser -language cs clock.abnf

Any syntax errors or inconsistencies in the grammar will be highlighted at this point. For example, if the Separator rule was not declared then **aParse** would generate the following errors.

Once the grammar compiles cleanly, the following C# source files will have been produced.

|  |  |
| --- | --- |
| Parser.cs | The parser of instances of Clock. |
| Rule.cs | An abstract base class that is inherited by all concrete Rule\_ and Terminal\_ classes. |
| Rule\_\*.cs | Classes, one for each of the rules defined in the grammar. For example,Rule\_Clock, Rule\_Hours and Rule\_Minutes. |
| Terminal\_\*.cs | Two classes, Terminal\_StringValue and Terminal\_NumericValue, for the[ABNF](http://www.parse2.com/abnf.shtml) terminal string and numeric values. |
| ParserContext.cs | A context class that encapsulates the information required by the Rule\_and Terminal\_ parsers. |
| ParserAlternative.cs | A class that encapsulates the details of an alternative encountered during the parsing of an instance of Clock. |
| ParserException.cs | An exception that is raised by the Parser when it attempts to parse an instance of Clock that does not conform to the specified grammar. |
| Visitor.cs | An abstract base class that must be implemented by classes that want to traverse a Parser generated rule tree. |
| Displayer.cs | A class that implements the interface defined by the Visitor class and displays the terminal string and numeric values of a Parser generated rule tree. |
| XmlDisplayer.cs | A class that implements the interface defined by the Visitor class and displays the contents of a Parser generated parse tree in XML. |

Compile these source files and build an executable using the relevant tools of an appropriate C# development environment. From henceforth it is assumed an executable called parser has been built.

**Test Parser**

To test the parser, create a file, for example clock.txt, containing an instance of Clock.

Parse it and display the contents of the parse tree using the parser and Displayer.

parser -visitor Displayer -file clock.txt

Alternatively, parse it and display the contents of the parse tree in XML using the XmlDisplayervisitor.

parser -visitor XmlDisplayer -file clock.txt

See [Clock Parser](http://www.parse2.com/example-clock.shtml) for a working example.

**Produce Visitor**

Rather than use the following automatically generated XmlDisplayer it is possible to produce and use an alternative Visitor.

For example, the following Clock2Xml visitor is identical to the XmlDisplayer except it does not output the ":" separators.

See [Clock Parser](http://www.parse2.com/example-clock.shtml) for a working example.

Similarly, the following Clock24To12 visitor converts the 24 hour clock values to their 12 hour clock equivalents.

See [Clock Parser](http://www.parse2.com/example-clock.shtml) for a working example.

**Employ Parser**

Having verified the operation of the parser and visitor, they may be built into a program. The following code segment shows how the parser and Clock24To12 visitor might be used to parse and process the contents of the file clock.txt.

**External Rules**

**aParse** supports the use of external code to parse and encapsulate protocol elements that are not directly support by the [ABNF](http://www.parse2.com/abnf.shtml) metalanguage.

For example, a protocol may not use separators to identify the boundary between variable length elements. Instead, the length of an element is found within the element. The following shows how a length prefixed string may be supported.

The following [ABNF](http://www.parse2.com/abnf.shtml) grammar states that a Message is composed of any number of Stringelements. The use of the $rule directive tells **aParse** that the user defined LLString rule is to be used for String elements.

The following is the C# class for the LLString rule. This supports variable length strings where the ASCII format length of the string is located in the first two characters.

This user defined class must provide at least two methods: Parse() and Accept().

**parse()**

This is a factory method that identifies whether the next sequence of characters in the input being parsed represent a two character length prefixed string and returns an instance of the LLString rule if they do. It must return null if they do not.

All the information required by the Parse() method is contained in the supplied ParserContext. The context.text string is the stream of characters being parsed and context.index points to the start of the characters, within the context.text string, that the Parse() method must attempt to parse. If the parse is successful, the context.index must be advanced by the number of characters taken up by the LLString element. If the parse fails, context.index must not be changed.

The first and last things the Parse() method must do are to call the context.Push() andcontext.Pop() methods. The call to context.Push() tells **aParse** that the LLString parser has been called and it is the supplied rulename that would appear in the rule stack output with any ParseException thrown. The call to context.Pop() tells **aParse** that the parsing has completed and, most importantly, whether or not the parse was successful or not.

**Accept()**

This is the Accept method of the visitor pattern and it will simply pass the LLString to the specified visitor.

**Options**

**aParse** supports the following optional arguments.

|  |  |
| --- | --- |
| -destdir directory | The directory in which the generated C# files are put. |
| -includedirs directories | The directories that are scanned when **aParse** searches for files included via $include directives. This is a comma or semi-colon separated list. |
| -namespace namespace | The namespace that the generated parser belongs to. An appropriate namespace statement is added to the generated C# source files. |
| -trace | Instructs **aParse** to output a trace showing the rules being compared against the input. |

**Apache Ant**

**aParse** provides the com.parse2.aparse.AntTask class that can be invoked directly from [Apache Ant](http://ant.apache.org/).

<taskdef name="aparse" classname="com.parse2.aparse.AntTask" classpath="aparse.jar">

<target name="makeparser">  
<aparse grammar="grammar.abnf" language="cs" namespace="com.company.product"/>  
</target>

The com.parse2.aparse.AntTask class supports the setting of the following C# related attributes.

|  |  |
| --- | --- |
| grammar | The [ABNF](http://www.parse2.com/abnf.shtml) grammar file. |
| language | Must be set to cs to instruct **aParse** to generate C# files. |
| destDir | The directory in which the generated C# files are put. |
| includeDirs | The directories that are scanned when **aParse** searches for files included via $include directives. This is a comma or semi-colon separated list. |
| namespace | The C# namespace that the generated parser belongs to. An appropriate namespace directive is added to the generated C# files. |
| trace | When set to on, this instructs **aParse** to output a trace showing the rules being compared against the input. |