XML

XML Expressions and Patterns

By Burak Emir

This chapter describes the syntactic structure of XML expressions and patterns. It follows as closely as possible the XML 1.0 specification, changes being mandated by the possibility of embedding Scala code fragments.

XML expressions

XML expressions are expressions generated by the following production, where the opening bracket `<` of the first element must be in a position to start the lexical XML mode.

```
XmlExpr ::= XmlContent {Element}
```

Well-formedness constraints of the XML specification apply, which means for instance that start tags and end tags must match, and attributes may only be defined once, with the exception of constraints related to entity resolution.

The following productions describe Scala's extensible markup language, designed as close as possible to the W3C extensible markup language standard. Only the productions for attribute values and character data are changed. Scala does not support declarations. Entity references are not resolved at runtime.

```
Element       ::=    EmptyElemTag
                |    STag Content ETag

EmptyElemTag  ::=    '<' Name {S Attribute} [S] '/>'

STag          ::=    '<' Name {S Attribute} [S] '>'
ETag          ::=    '</' Name [S] '>'
Content       ::=    [CharData] {Content1 [CharData]}
Content1      ::=    XmlContent
                |    Reference
                |    ScalaExpr
XmlContent    ::=    Element
```

```
                |    CDSect
                |    PI
                |    Comment
```

If an XML expression is a single element, its value is a runtime representation of an XML node (an instance of a subclass of `scala.xml.Node`). If the XML expression consists of more than one element, then its value is a runtime representation of a sequence of XML nodes (an instance of a subclass of `scala.Seq[scala.xml.Node]`).

If an XML expression is an entity reference, CDATA section, processing instruction, or a comment, it is represented by an instance of the corresponding Scala runtime class.

By default, beginning and trailing whitespace in element content is removed, and consecutive occurrences of whitespace are replaced by a single space character `\u0020`. This behavior can be changed to preserve all whitespace with a compiler option.

```
Attribute  ::=    Name Eq AttValue

AttValue      ::=    '"' {CharQ | CharRef} '"'
                |    "'" {CharA | CharRef} "'"
                |    ScalaExpr

ScalaExpr     ::=    Block

CharData    ::=  { CharNoRef } $\textit{ without}$ {CharNoRef}'{'CharB {CharNoRef}
                       $\textit{ and without}$ {CharNoRef}']]>'{CharNoRef}
```

XML expressions may contain Scala expressions as attribute values or within nodes. In the latter case, these are embedded using a single opening brace `{` and ended by a closing brace `}`. To express a single opening braces within XML text as generated by CharData, it must be doubled. Thus, `{{` represents the XML text `{` and does not introduce an embedded Scala expression.

```
BaseChar, CDSect, Char, Comment, CombiningChar, Ideographic, NameChar, PI, S, Reference
              ::=  $\textit{"as in W3C XML"}$

Char1         ::=  Char $\textit{ without}$ '<' | '&'
CharQ         ::=  Char1 $\textit{ without}$ '"'
CharA         ::=  Char1 $\textit{ without}$ "'"
CharB         ::=  Char1 $\textit{ without}$ '{'

Name          ::=  XNameStart {NameChar}

XNameStart    ::= '_' | BaseChar | Ideographic
                  $\textit{ (as in W3C XML, but without }$ ':'$)$
```

XML patterns

XML patterns are patterns generated by the following production, where the opening bracket `<` of the element patterns must be in a position to start the lexical XML mode.

```
XmlPattern  ::= ElementPattern
```

Well-formedness constraints of the XML specification apply.

An XML pattern has to be a single element pattern. It matches exactly those runtime representations of an XML tree that have the same structure as described by the pattern. XML patterns may contain Scala patterns.

Whitespace is treated the same way as in XML expressions.

By default, beginning and trailing whitespace in element content is removed, and consecutive occurrences of whitespace are replaced by a single space character `\u0020`. This behavior can be changed to preserve all whitespace with a compiler option.

```
ElemPattern   ::=    EmptyElemTagP
                |    STagP ContentP ETagP

EmptyElemTagP ::=    '<'  Name [S] '/>'
STagP         ::=    '<'  Name [S] '>'
ETagP         ::=    '</' Name [S] '>'
ContentP    ::=   [CharData] {(ElemPattern|ScalaPatterns) [CharData]}
ContentP1     ::=    ElemPattern
                |    Reference
                |    CDSect
                |    PI
                |    Comment
                |    ScalaPatterns
ScalaPatterns ::=    '{' Patterns '}'
```