**Introduction**

**WML, WML Script and WAP 2.0.**

**WML (Wireless Markup Language)**

The Wireless Markup Language is a markup language based on XML and is intended for use in specifying content and user interface for narrowband devices, including mobile phones and pagers.

**WMLScript**

A scripting language used to program a mobile device. WMLScript is an extended subset of the JavaScript scripting language.

Wireless Markup Language (WML) is a markup language that is based on the Extensible Markup Language (XML) and is used for creating content destined for display on wireless mobile devices.

A WML source file has a MIME content type of text/vnd.wap.wml. An encoded or compiled WML file has a MIME content type of application/vnd.wap.wmlc.

A WML file, whether encoded or not, is called a deck. A deck is the smallest unit of WML that a web server can send to a user agent for processing in the mobile device.

The basic constituent of a WML deck is the card, which specifies a single interaction between the device's user agent and the user. When a user agent receives a deck, it typically activates the first card in the deck, though it can be directed to any particular card in the deck.

WML is a case-sensitive language, that is, Speed and speed are different entities. WML also converts one or more contiguous newlines, carriage returns, tabs and spaces to a single space. To make the examples contained here easier to follow, we have formatted them with newlines and

tabs. However, note that this formatting is not required for WML to be valid.

WML inherits most of its syntactic constructs from XML. For detailed information on the syntactical issues of XML, refer to the XML Specification.

**Card**

A single WML navigational and user interface unit. A card may contain information to present to the user or instructions for gathering user input, for example.

**Deck**

A collection of one or more WML cards contained in a file of type wml. A WML deck is also an XML document.

**DTD (Document Type Definition)**

A document type definition defines the elements that are permitted within an XML document, where these elements may appear within the document, which attributes and attribute values are permissible, and so on. Basically, a DTD is a set of rules that an XML parser uses to create a parse tree of a document, which is required for further processing.

**Entities**

WML text can contain numeric or named character entities that specify specific characters in the document character set. Entities are used to specify characters in the document character set which must either be escaped in WML or which may be difficult to enter in a text editor. For example, the ampersand (&) is represented by the named entity &amp;. All entities begin with an ampersand and end with a semicolon.

**Tags**

A tag is a language element descriptor. A tag describes an element and contains an element type name and a unique identifier. A tag could also include attributes describing other properties.

WML consists of content surrounded by formatting tags, each enclosed in a pair of angle brackets,< and >.

<tag> This starts an element. The start tag can contain attributes.

</tag> This ends an element.

<tag/> This is an empty element, for example <br/>, indicating a line break.

**Elements**

Elements specify all markup and structural information for a WML deck. Elements may contain a start tag, content, other elements, and an end tag. Elements have one of two structures:

<tag> content </tag>

- or-

<tag/>

Elements containing content and other elements are identified by a start tag <tag> and an end tag </tag>. An empty-element tag <tag/> identifies elements with no content.

The following table lists some commonly used WML elements:

|  |  |
| --- | --- |
| **Category** | **WML elements** |
| Deck and cards  Events  Tasks  Variables  User Input  Anchor's Images  and Timers    Text Formatting | wml  card  template  head  access  meta    do  ontimer  onenterforward onenterbackward  onpick  onevent  postfield  go  prev  refresh  noop  setvar  input  select  option  optgroup  fieldset  a  anchor  img  timer  br  p  table  tr  td  pre (WAP 1.2 and 1.3) |

**Attributes**

Many WML elements allow you to include attributes in them. Attributes specify additional

information for an element. You always specify attributes in the start tag of an element, using the following syntax:

<tag attribute1="value1" attribute2="value2" attribute3="value3"...>

You separate each option-value pair by white space, which may be a tab, newline, carriage return, or space character. You must enclose the value in double quotation marks (").

Note that attribute names must be lowercase.

Some attributes are mandatory. For example, the go element requires the href attribute:

<go href="http://www.acme.com"/>

Other attributes are optional and may have default values. For example, the align attribute is optional for the img element. If you do not specify the align attribute, its value defaults to bottom.

**Comments**

WML comments follow the XML commenting style and have the following syntax:

<!-- a comment -->

Comments are intended to be used by the WML author and are not displayed to the user by the user agent. Note that WML comments cannot be nested.

**Variables**

Parameters can be set for WML cards and decks using variables. To substitute a variable into a cardor deck, the following syntaxes are used:

$identifier

$(identifier)

$(identifier:conversion)

Parentheses are required if white space does not indicate the end of a variable. Variable syntax has the highest priority in WML, that is, anywhere the variable syntax is legal, an unescaped ‘$’

character indicates a variable substitution. Variable references are legal in any PCDATA and in any attribute value identified by the vdata entity type.

**Case Sensitivity**

XML is a case-sensitive language, and WML has inherited this characteristic. No case folding is performed when parsing a WML deck. This implies that all WML tags, attributes and contents arecase sensitive. In addition, any enumerated attribute values are case sensitive.

For example, the following attribute values are all different:

id="Card1"

id="card1"

id="CARD1"

**Deck and Card Elements**

The following table briefly explains the elements you use to define the cards within a deck.

|  |  |
| --- | --- |
| ***Element*** | **Explanation** |
| *wml* | Defines a deck and encloses all the information and cards in the deck. |
| *card* | Indicates the general layout and user schemes in the user agent. |
| *template* | Declares a template for cards in the deck. |
| *head* | Contains information relating to the deck as a whole, including meta-data and access control elements. |
| *access* | Specifies access control information for the entire deck. |
| *meta* | Contains generic meta information relating the deck. |

**Card Element**

The card is the basic unit of navigation within a deck. It is possible to move from one card to another within the same deck or to move to another deck. Unless specified otherwise, upon navigation to another deck, the first card is displayed

The following is a simple card:

<card id="card\_name" title="title">

<p>

Text containing <i> information </i>.

</p>

</card>

**Event Elements**

Several WML elements can generate intrinsic events when the user interacts with them. These intrinsic events indicate state transitions inside the user agent. WML defines the four intrinsic events described in this section.

The following table briefly describes these elements.

|  |  |
| --- | --- |
| **Element** | **Explanation** |
| do | Provides a general mechanism for performing actions on the current card. |
| ontimer | Specifies an intrinsic event that occurs when a timer expires. |
| onenterforward | Specifies an intrinsic event that occurs when the user enters a card under specific circumstances. |
| onenterbackward | Specifies an intrinsic event that occurs when the user navigates into a card by using a URI retrieved from the history stack. |
| onpick | Specifies an intrinsic event that occurs when the user selects or deselects an item in which the event is specified. |
| onevent | Binds a task to a particular intrinsic event. |
| postfield | Specifies a field name and value for transmission to an origin server during a URI request. |

**Do Element**

A card element can contain do elements that assign a task to a certain key. The following is a simple example of a do element:

<do type="tasktype" label="label" name="name" optional="false">

**Task Elements**

A do element can specify tasks that the user agent is to do when the user presses a key or navigates to a card or deck. The go, prev, noop and refresh tasks of WML.

**Go Element**

The go element directs the user agent to the location specified by an href. If the href names a

WML card or deck, it is displayed.

The following is a simple example using the go element.

<go href="#card2"/>

The following table describes the elements used in the above example.

<go> Start tag of a go element, indicating navigation to the URI

specified by href.

href="URI" Specifies the destination, for example, the location of the

card to display. This attribute is required.

**Refresh Element**

The refresh element declares a refresh task, indicating an update of the user agent context

as specified by the setvar element. Side effects of the state changes that are visible to the user

(for example, a change in the screen display) occur during the processing of the refresh task.

**Noop Element**

The noop element specifies that nothing should be done, that is, “no operation”. You will find this a useful element when you want to override template action.

**Anchor Element**

The anchor and a elements specify the head of a link. The tail of a link is specified as part of other elements, for example, a card name attribute. Note that it is a WML syntax error to nest anchored links.

You can use anchors anywhere formatted text is legal, except for option elements.

An anchored link must have an associated task that specifies the behavior when the anchor is selected. You must anchor one of the following task elements to a link:

* go
* prev
* refresh

Note that it is a WML syntax error to specify more than one task in either an anchor or a element. The a element is a short form of the anchor element and is bound to a go task without variable**Experiment 1**

**Aim: Write a program to create a card and print "Hello! WML" in <p> tag.**

**Algorithm:**

1. Create a new file in text editor (notepad or notepad++).
2. Write the complete WML code.
3. Save the file with ".wml" extension.
4. Open WinWap browser.
5. Write the complete path of the saved in Address bar or drag and drop the .wml file onto the browser window to view the output.

**WML CODE:**

**<<<Write code here>>>**

**OUTPUT:**

**<<<Print colored output here>>>**

**Experiment 2**

**Aim: Write a program to create and execute external links in WML.**

**Algorithm:**

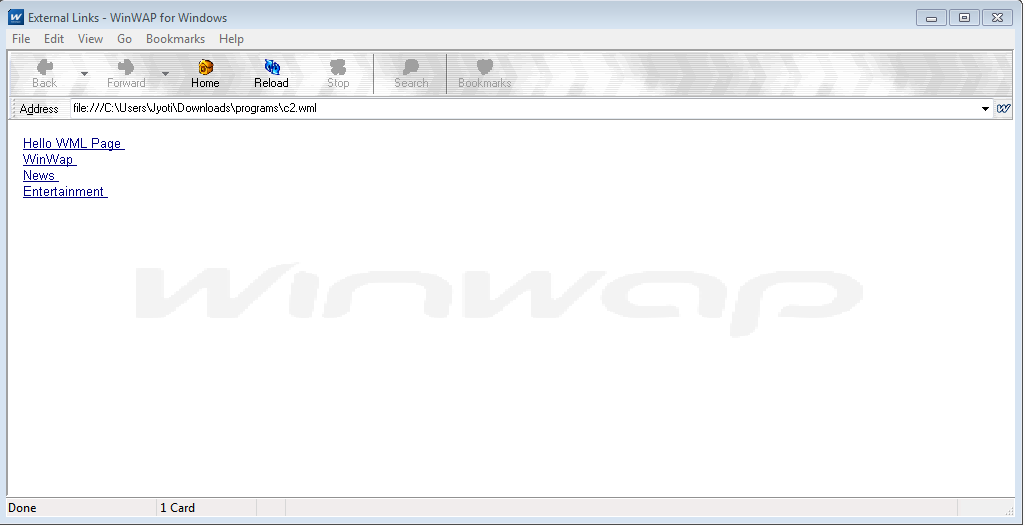
1. Create a new file in text editor (notepad or notepad++).
2. Write the complete WML code.
3. Save the file with ".wml" extension.
4. Open WinWap browser.
5. Write the complete path of the saved in Address bar or drag and drop the .wml file onto the browser window to view the output.

**WML CODE:**

**<<<Write code here>>>**

**OUTPUT:**

**<<<Print colored output here>>>**



**EXPERIMENT 3**

**Aim: Write a program to create a multiple cards in WML and perform navigation between them using do tags.**

**Algorithm:**

1. Create a new file in text editor (notepad or notepad++).
2. Write the complete WML code.
3. Save the file with ".wml" extension.
4. Open WinWap browser.
5. Write the complete path of the saved in Address bar or drag and drop the .wml file onto the browser window to view the output.

**WML CODE:**

<wml>

<card id="c1" title="Hello WML">

<p>

Hello WML!!!<br/><br/>

<do type="Card 2">

<go href="#c2"/>

</do>

<do type="Card 3">

<go href="#c3"/>

</do>

<do type="Card 4">

<go href="#c4"/>

</do>

</p>

</card>

<card id="c2" title="Card 2">

<p>

<em>Card 2</em><br/><br/>

<br/>

<a href="#c1">

Back

</a><br/>

</p>

</card>

<card id="c3" title="Card 3">

<p>

<em>Card 3</em><br/><br/>

<br/>

<a href="#c1">

Back

</a><br/>

</p>

</card>

<card id="c4" title="Card 4">

<p>

<em>Card 4</em><br/><br/>

<br/>

<a href="#c1">

Back

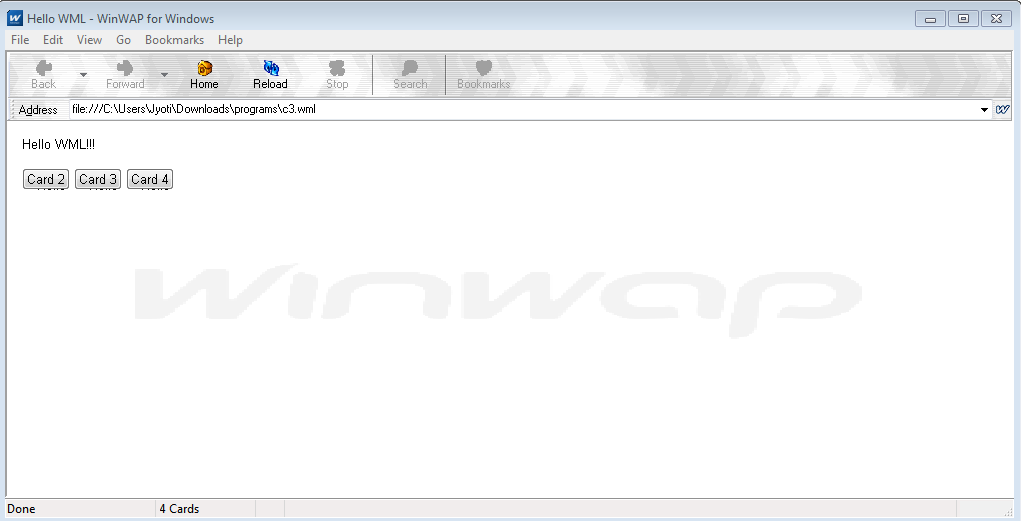
</a><br/>

</p>

</card>

</wml>

**OUTPUT:**



**EXPERIMENT 4**

**Aim: (a) Write a program to show images (.wbmp file) in WML page.**

**(b) Write a program to create a table in WML.**

**Algorithm:**

1. Create a new file in text editor (notepad or notepad++).
2. Write the complete WML code.
3. Save the file with ".wml" extension.
4. Open WinWap browser.
5. Write the complete path of the saved in Address bar or drag and drop the .wml file onto the browser window to view the output.

**(a) WML CODE:**

<wml>

<card title="Picture">

<p>

This is an ".wbmp" image.<br><br>

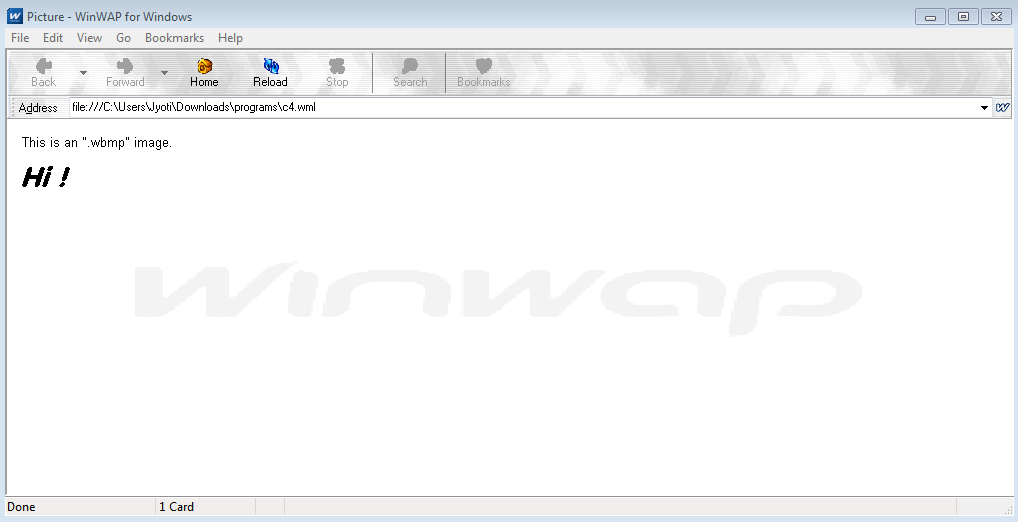
<img src="hi.wbmp" alt="WML Pic"/>

</p>

</card>

</wml>

**(a) OUTPUT:**

****

**(b) WML CODE:**

<wml>

<card title="Table">

<p>

<table columns="2">

<tr>

<td>Rollno:</td>

<td>Name:</td>

</tr>

<tr>

<td>001</td>

<td>ABC</td>

</tr>

<tr>

<td>002</td>

<td>PQR</td>

</tr>

<tr>

<td>003</td>

<td>STU</td>

</tr>

<tr>

<td>004</td>

<td>VWX</td>

</tr>

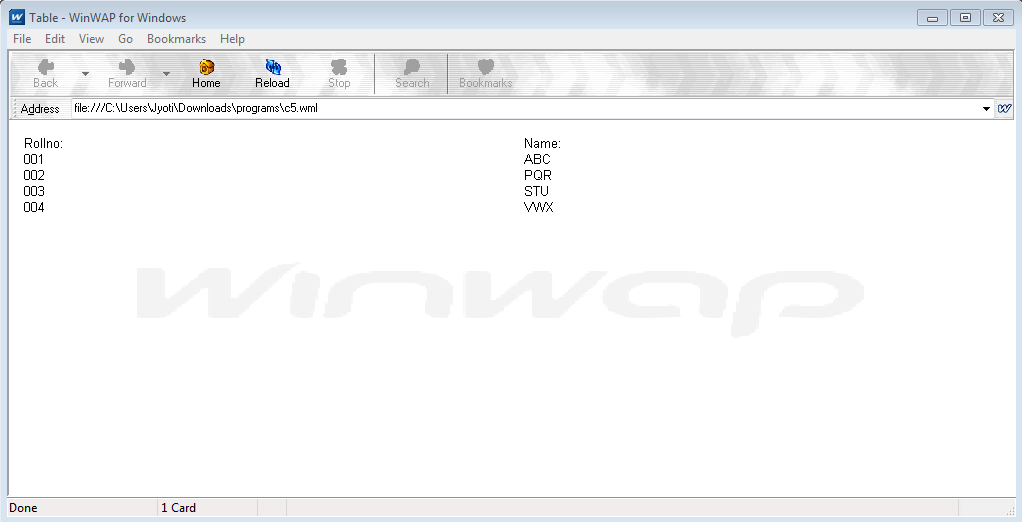
</table>

</p>

</card>

</wml>

**(b) OUTPUT:**

****