XML Technologies Report

SA1032A: XML Technologies

Nicolas Lanquetin
0604918@abertay.ac.uk

8th January 2007

University of Abertay Dundee
School of Computing & Creative Technologies

Contents

1. XML Site 1
   - 1.1. About the Site 1
   - 1.2. Architecture 2
     - 1.2.1. Filesystem Organisation 2
     - 1.2.2. HTTP Request Parameters 2
     - 1.2.3. Handling the HTTP Request 3
     - 1.2.4. Generating RSS Feeds 4
     - 1.2.5. Notes to the DTD and XSL Files 5

2. Disussion 7
   - 2.1. Some Problems with the championship DTD 7
     - 2.1.1. Types 7
     - 2.1.2. Context Insensitivity 7
     - 2.1.3. Enumerations 8
   - 2.2. Web Service 8

A. XHTML Website Structure and Layout 10

B. Mobile Site Structure and Layout 12

Useful Links 17

Chapter 1.

XML Site

1.1. About the Site

The website is realised with PHP 5 using the PHP 5 Object Model (PHP, 2007*a*). It requires the Libxslt extension (XMLSoft, 2007) compiled by default in PHP 5, as well as PHP 5 DOM and PHP 5 XSL functions (PHP, 2007*b*,*c*).

Links

- The HTTP Website:
  http://psbase.com/uad/xmltech/
- The WAP Website:
  http://psbase.com/uad/xmltech/index.php5?type=wml
- Classes Documentation and source code:
  http://psbase.com/uad/xmltech/docs/
- Subversion Repository (if online):
  http://guest:guest@home.psbase.com/svn/xmltech/trunk/
- Subversion Change View (if online):
  http://home.psbase.com/viewsvn/?do=browse&project=xmltech&path=/trunk/

1.2. Architecture

1.2.1. Filesystem Organisation

```
trunk  (Subversion Trunk)
⋮ ⋯ emu  (Free WML emulator setups)
⋮ ⋯ specification  (Coursework specification)
⋮ ⋯ www  (Root folder of the web server)
⋮  ⋮ ⋯ configs  (Configuration files)
⋮  ⋮ ⋯ docs  (Classes Documentation)
⋮  ⋮ ⋯ includes  (All PHP Classes)
⋮  ⋮  ⋮ ⋯ exceptions  (Exceptions Classes)
⋮  ⋮  ⋱ · utils  (Utility Classes)
⋮  ⋮ ⋯ libs  (PHP libraries (i.e. PEAR))
⋮  ⋮ ⋯ resources  (Contains all resources needed for the application)
⋮  ⋮  ⋮ ⋯ css  (CSS files)
⋮  ⋮  ⋮ ⋯ images  (Images)
⋮  ⋮  ⋮ ⋯ xml  (XML files)
⋮  ⋮  ⋮ ⋯ xsl  (XSL files)
⋮  ⋮  ⋮  ⋮ ⋯ rss  (XSL files for RSS output)
⋮  ⋮  ⋮  ⋮ ⋯ wml  (XSL files for WML output)
⋮  ⋮  ⋮  ⋱ · xhtml  (XSL files for XHTML output)
```

1.2.2. HTTP Request Parameters

To handle a request, the website needs the following information:

1. the type of page requested (Parameter `display`, can be '`localpage`' or '`rssfeed`')
2. which page is requested (Parameter `page`, in this application only '`championship`' exists so far)
3. the ouput type (Parameter `type`, can be '`xhtml`', '`wml`', '`rss`', '`pdf`', etc.)
4. the session id, in order to save data over requests (Parameter `sid`)

Note that these parameters are also defined in the Class `ParamNames`. A typical URL would look as follow with the parameters explained above:

```
.../index.php5?display=localpage&type=xhtml&page=championship&sid=<session_id>
```

1.2.3. Handling the HTTP Request

The UML Sequence Diagram in figure 1.1 shows the processing of a typical page request. There are three main steps to consider:

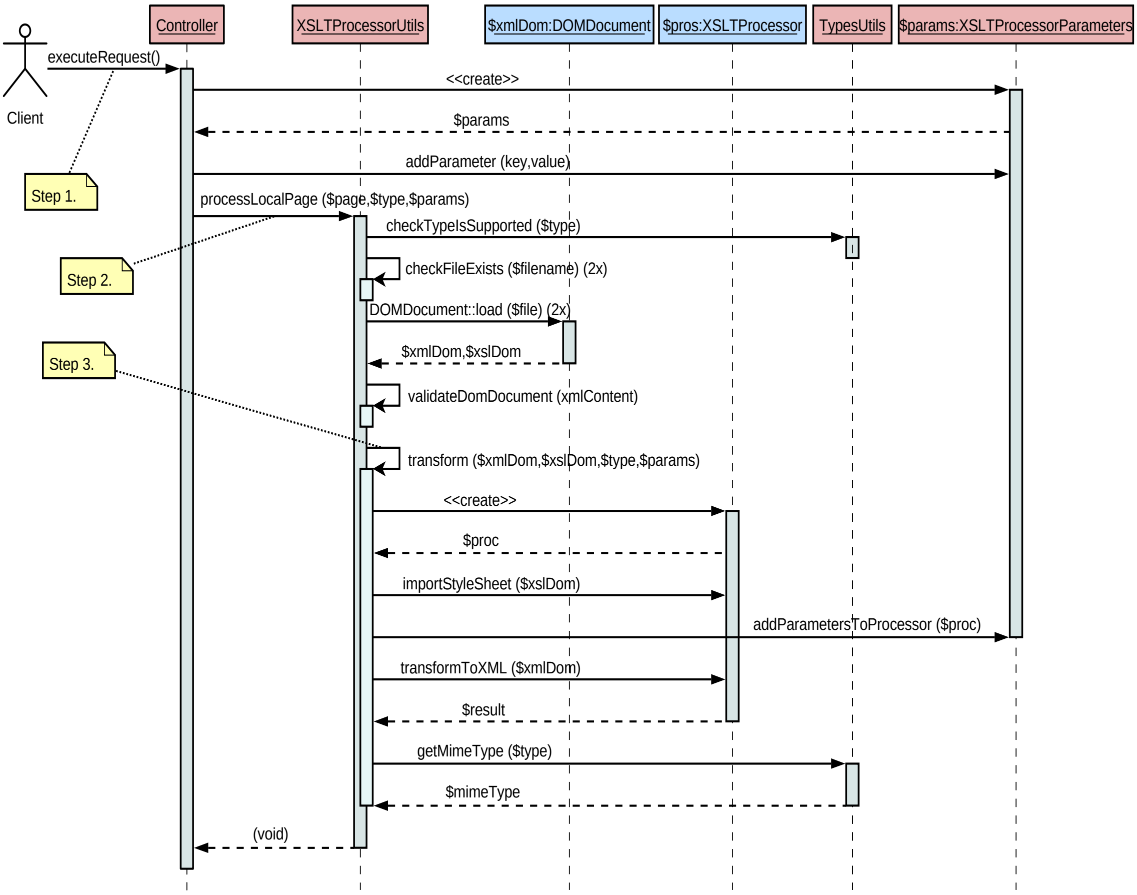

Controller
XSLTProcessorUtils
$xmlDom:DOMDocument
$pros:XSLTProcessor
TypesUtils
$params:XSLTProcessorParameters
Client
executeRequest()
<<create>>
$params
addParameter (key,value)
Step 1.
processLocalPage ($page,$type,$params)
checkTypeIsSupported ($type)
Step 2.
checkFileExists ($filename) (2x)
DOMDocument::load ($file) (2x)
Step 3.
$xmlDom,$xslDom
validateDomDocument (xmlContent)
transform ($xmlDom,$xslDom,$type,$params)
<<create>>
$proc
importStyleSheet ($xslDom)
addParametersToProcessor ($proc)
transformToXML ($xmlDom)
$result
getMimeType ($type)
$mimeType
(void)

Figure 1.1.: **UML Sequence Diagram of a Page Request**: Classes included in PHP 5 are marked in blue, and Classes specially developed for this project are marked in red.

Step 1 The static method `Controller::executeRequest()` is called by the index.php file. It gets all the HTTP Request Parameters and decides which page to display. It also creates a `XSLTProcessorParameters` which holds all parameters necessary for the XSL files. Furthermore, any Exception thrown by the application will be catched and handled by the `Controller`.

Step 2 Next, the static method `XSLTProcessorUtils::processLocalPage( .. )` is called by the Controller with the parameters `$page`, `$type` and the `$params`[1]. The UML Class Diagram in figure 1.2 shows the outline of that Class. There are two public methods: one for a normal page request, and another for RSS feeds. The `$page` tells the function which XML and XSL file to open. If the output type and the files for that particular output type exist, a `DOMDocument` is created for each file and the XML `DOMDocument` is validated against its DTD.

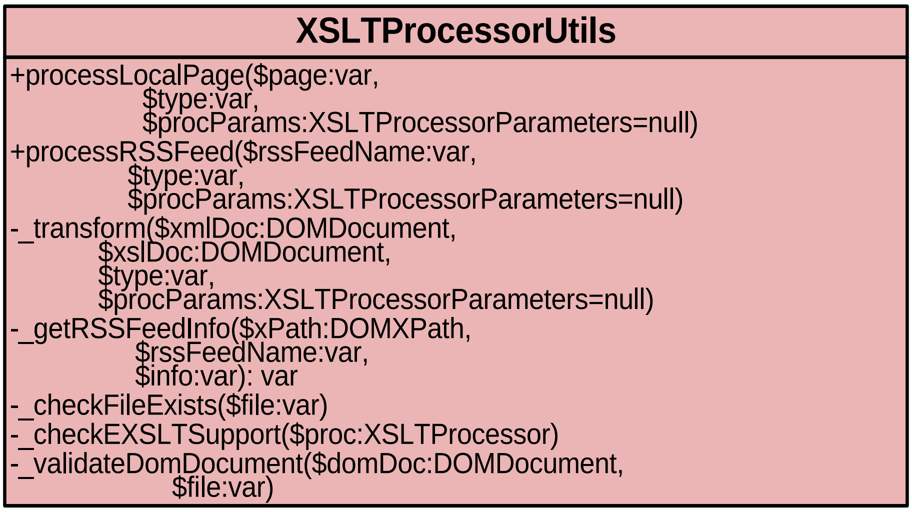

XSLTProcessorUtils
+processLocalPage($page:var,
$type:var,
$procParams:XSLTProcessorParameters=null)
+processRSSFeed($rssFeedName:var,
$type:var,
$procParams:XSLTProcessorParameters=null)
-_transform($xmlDoc:DOMDocument,
$xslDoc:DOMDocument,
$type:var,
$procParams:XSLTProcessorParameters=null)
-_getRSSFeedInfo($xPath:DOMXPath,
$rssFeedName:var,
$info:var): var
-_checkFileExists($file:var)
-_checkEXSLTSupport($proc:XSLTProcessor)
-_validateDomDocument($domDoc:DOMDocument,
$file:var)

Figure 1.2.: UML Class Diagram of `XSLTProcessorUtils`

Step 3 Finally the private method `_transform( .. )` is called with the two `DOMDocuments`, the `$type` and the `$params` (see figure 1.2). An `XSLTProcessor` Object `$proc` is instantiated and the XSL `DOMDocument` attached to that Object. If some `$params` were set, they will also be added to `$proc`. Now the XML `DOMDocument` is passed to `$proc` and the transformation starts. The resulting XML is saved into the variable `$result`. In a last step, the header is set by requesting the mime type of the `TypesUtils` Class with the `$type` as parameter, and the `$result` is displayed.

1.2.4. Generating RSS Feeds

In Step 2 of section 1.2.3, we saw how to handle a simple page request. To generate a RSS feed, the procedure is different: Instead of calling the method `processLocalPage( .. )` of the `XSLTProcessorUtils` Class, the static method `processRSSFeed( .. )` must be called. The difference is that the XML data is streamed from a website rather than read

[1] Instantiated Classes or variables begin with a dollar sign in PHP. See section 1.2.2 for the meaning of these variables.

from a local file. A XML configuration file has been created for this purpose (see Listing 1.1 and 1.2).

Listing 1.1: rssfeeds.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rssfeeds SYSTEM "rssfeeds.dtd">
<rssfeeds>
  <rssfeed name="skieuropenews">
    <description>ski-europe news feed</description>
    <xml>http://www.ski-europe.com/news/serfeed.xml</xml>
    <xsl>defaultfeed.xsl</xsl>
  </rssfeed>
  <!-- possibility to add more rssfeeds here -->
</rssfeeds>
```

Listing 1.2: rssfeeds.dtd

```
<!ELEMENT rssfeeds (rssfeed)* >
<!ELEMENT rssfeed (description?, xml, xsl) >
<!ATTLIST rssfeed name CDATA #REQUIRED >
<!ELEMENT description (#PCDATA) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT xml (#PCDATA) >
<!ELEMENT xsl (#PCDATA) >
```

The HTTP Request parameter ‘`rssfeed`’ (by default ‘`skieuropenews`’) tells the application which XML URL and XSL file to pick for the transformation. This functionality has been realised in the method `_getRSSFeedInfo` of `XSLTProcessorUtils`. It uses a `DOMXPath` Object and a XPath query to get the necessary information from the XML configuration file. Once the XML and XSL `DOMDocuments` created, the processing is resumed as explained in Step 3 of section 1.2.3.

1.2.5. Notes to the DTD and XSL Files

Changes to the championship.dtd

Listing 1.3: championship.dtd

```
<!ELEMENT championship (event)+ >
<!ELEMENT event (summary, result*) >
<!ATTLIST event id CDATA #REQUIRED >
<!ATTLIST event gender ( men | women ) #REQUIRED >
<!ELEMENT summary (name, location, finaltime, description ) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT location (#PCDATA) >
<!ELEMENT finaltime (#PCDATA) >
<!ELEMENT description (#PCDATA) >
<!ELEMENT result (competitor, country, time) >
<!ATTLIST result position (gold | silver | bronze) #REQUIRED >
<!ELEMENT competitor (#PCDATA) >
```

```
13 <!ELEMENT country (#PCDATA) >
14 <!ELEMENT time (#PCDATA) >
```

1. **Line 3**: An `id` attribute has been added to each `event` to make each truly unique, regardless of their `names`. It is used to access the events directly with XPath queries.

2. **Lines 5–6**: The `summary name` is now an element inside the `summary` element rather than an attribute of it. It is used as short title for each `event`.

3. **Line 2**: An `event` can now have more than one `result`. This is one possible solution to have a ranking of `competitors` for one `event` as shown in figure A.2 on page 10.

4. **Line 11**: The `position` is now an attribute of `result`, instead of an element. This way we can define the three possibilities '`gold`', '`silver`' and '`bronze`'. This would not have worked with an element definition, due to limitations of DTDs.

XSL Files

For every output type (xhtml, wml and rss) exist a XSL file (see section 1.2.1). To ensure the principle of code reuse, a file (bib.xsl) has been created holding any common functionality for the different types. Furthermore, it has been ensured that only templates are used. The 'championship.xsl' in the 'rss' folder certainly demonstrates this the best.

Chapter 2.

Disussion

2.1. Some Problems with the championship DTD

2.1.1. Types

The `event` `id` Attribute as Integer: The `id` attribute of the `event` element (see line 3 of Listing 1.3 on page 5) should be a an integer following the pattern `YYYYMMDDHHMM` (year, month, day, hours and minutes). It is not possible for DTDs to define such a rule. An XML Schema could resolve this problem by applying some restrictions, such as a new defined type which state that the `id` attribute should be a number following a special pattern. Listing 2.1 shows a possible definition for the `id`.

Listing 2.1: XML Schema definition for the `id` type

```
<xs:simpleType name="id">
  <xs:restriction base="xs:string">
    <xs:pattern value="
         ([0-9][0-9][0-9][0-9][0-1][0-9][0-3][0-9][0-5][0-9][0-5][0-9])"/>
  </xs:restriction>
</xs:simpleType>
```

Limitation of XML Schema: The 'date' for the `id` could be false (i.e. January 32th 2007)

2.1.2. Context Insensitivity

The `time` Elements are Context Insensitive: In the current DTD, there exists two elements for the time: the `time` element and the `finaltime` element. Because DTDs are context insensitive, we had to use two different element names. It would be much better to define each `time` element in its context. This is possible with a XML Schema by defining the hierarchy of elements and their order.

2.1.3. Enumerations

Enumerations for Elements: In point 4 of section 1.2.5 we saw that it is not possible to have enumerations for elements using a DTD. However, this can be done with a XML Schema as shown in listing 2.2.

Listing 2.2: XML Schema definition for the `position` type

```
<xs:simpleType name="position">
  <xs:restriction base="xs:string">
    <xs:enumeration value="gold"/>
    <xs:enumeration value="silver"/>
    <xs:enumeration value="bronze"/>
  </xs:restriction>
</xs:simpleType>
```

2.2. Web Service

PEAR, a framework and distribution system for reusable PHP components, proposes many so called *packages* to use web services. Many of these packages provides access to different web service providers such as Amazon, Ebay, Google and many more (PEAR, 2007*b*). It is also possible to create an own web service using the Services_Webservice package (PEAR, 2007*a*) which in turn uses the SOAP PHP extension supporting the SOAP 1.1, SOAP 1.2 and WSDL 1.1 specifications (PHP, 2007*d*).

The `Services_Webservice` Class allows the creation of the web service from an own class. We could for instance create a class with methods returning the events, a specific event or event results. Our Class only needs to extend `Services_Webservice` and can then use the many `createWSDL` methods of the latter to create the SOAP Server, the WSDL file for the access to our methods and even a DISCO file (see figure 2.1 for an UML Class diagram of the `Services_Webservice` Class).

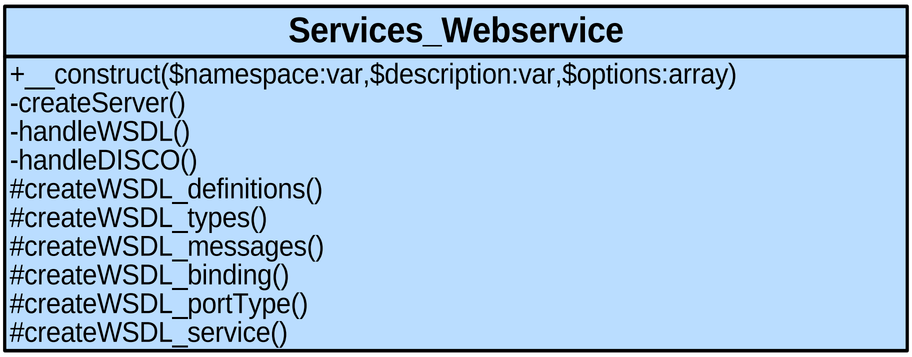


Figure 2.1.: UML Class Diagram of `Services_Webservice`. The Class attributes and some methods were omitted to make the Class more clear.

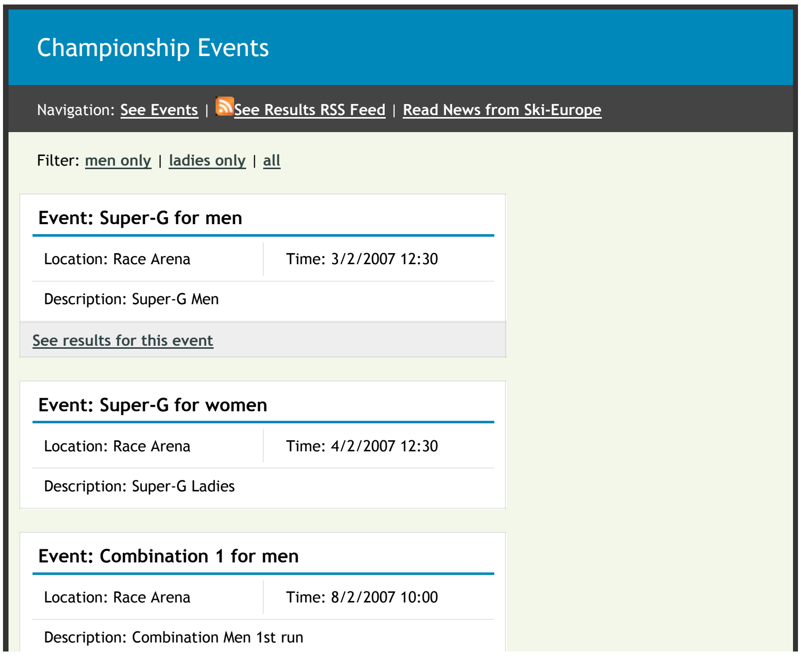
The interface to our web service methods could look as shown in table 2.1.

Method name	**Input type**	**Output type**	**Description**
`getEvents`	`void`	`SOAP-ENC:Array`	Gets a list of all events. Each event of the `SOAP-ENC:Array` is a `Complex Type`
`getEventById`	`SOAP-ENC:string`	`Complex Type`	Gets a specific event by id
`getResults`	`void`	`SOAP-ENC:Array`	Gets a list of all results. Each result of the `SOAP-ENC:Array` is a `Complex Type`
`getResultsByEventId`	`SOAP-ENC:string`	`SOAP-ENC:Array`	Gets a list of all results for the given event id. Each result of the `SOAP-ENC:Array` is a `Complex Type`

Table 2.1.: The interface to the web service of the XML website.

Appendix A.

XHTML Website Structure and Layout



Championship Events

Navigation: See Events | See Results RSS Feed | Read News from Ski-Europe

Filter: men only | ladies only | all

Event: Super-G for men

Location: Race Arena Time: 3/2/2007 12:30

Description: Super-G Men

See results for this event

Event: Super-G for women

Location: Race Arena Time: 4/2/2007 12:30

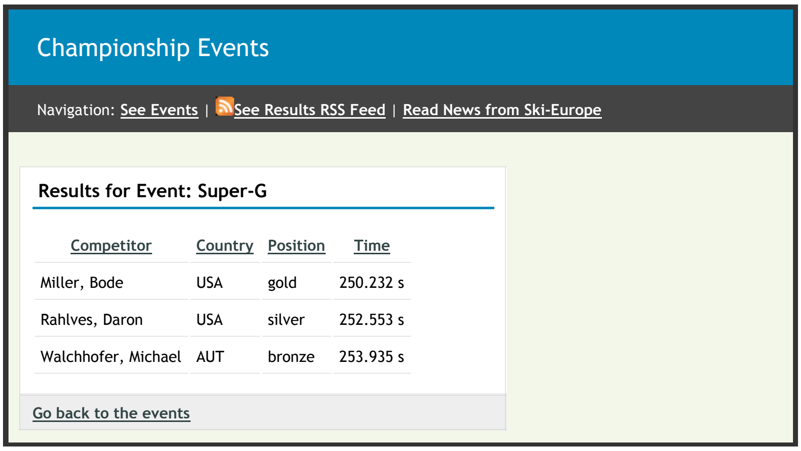
Description: Super-G Ladies

Event: Combination 1 for men

Location: Race Arena Time: 8/2/2007 10:00

Description: Combination Men 1st run

Figure A.1.: **Website Homepage**: Possibility to filter by men or ladies results. A ‘results’ link is displayed for events with results.



Championship Events

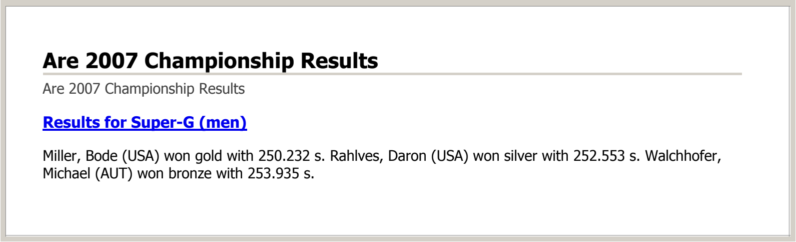
Navigation: See Events | See Results RSS Feed | Read News from Ski-Europe

Results for Event: Super-G

Competitor	Country	Position	Time
Miller, Bode	USA	gold	250.232 s
Rahlves, Daron	USA	silver	252.553 s
Walchhofer, Michael	AUT	bronze	253.935 s

Go back to the events

Figure A.2.: **View of a single result**: Possibility to sort the results by competitor, country, position or time.



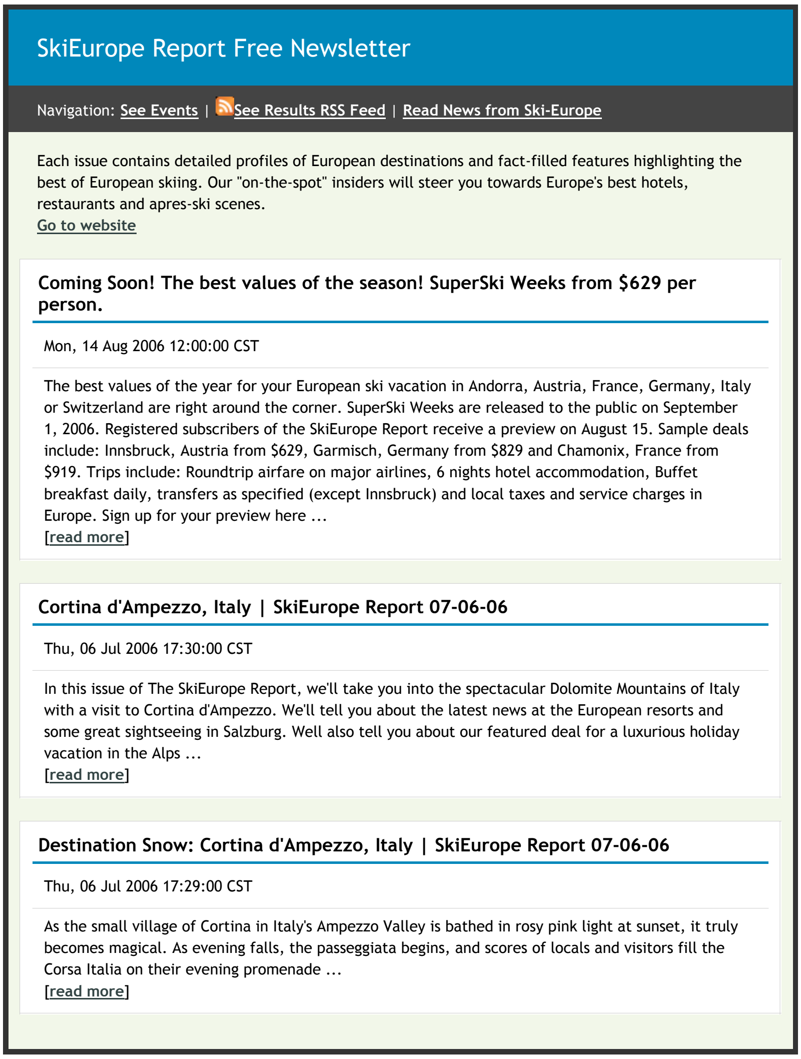
Are 2007 Championship Results

Are 2007 Championship Results

Results for Super-G (men)

Miller, Bode (USA) won gold with 250.232 s. Rahlves, Daron (USA) won silver with 252.553 s. Walchhofer, Michael (AUT) won bronze with 253.935 s.

Figure A.3.: **The results as RSS feed**: Rendered by Firefox 2.0



SkiEurope Report Free Newsletter

Navigation: See Events | See Results RSS Feed | Read News from Ski-Europe

Each issue contains detailed profiles of European destinations and fact-filled features highlighting the best of European skiing. Our "on-the-spot" insiders will steer you towards Europe's best hotels, restaurants and apres-ski scenes.
Go to website

Coming Soon! The best values of the season! SuperSki Weeks from $629 per person.

Mon, 14 Aug 2006 12:00:00 CST

The best values of the year for your European ski vacation in Andorra, Austria, France, Germany, Italy or Switzerland are right around the corner. SuperSki Weeks are released to the public on September 1, 2006. Registered subscribers of the SkiEurope Report receive a preview on August 15. Sample deals include: Innsbruck, Austria from $629, Garmisch, Germany from $829 and Chamonix, France from $919. Trips include: Roundtrip airfare on major airlines, 6 nights hotel accommodation, Buffet breakfast daily, transfers as specified (except Innsbruck) and local taxes and service charges in Europe. Sign up for your preview here ...
[read more]

Cortina d'Ampezzo, Italy | SkiEurope Report 07-06-06

Thu, 06 Jul 2006 17:30:00 CST

In this issue of The SkiEurope Report, we'll take you into the spectacular Dolomite Mountains of Italy with a visit to Cortina d'Ampezzo. We'll tell you about the latest news at the European resorts and some great sightseeing in Salzburg. Well also tell you about our featured deal for a luxurious holiday vacation in the Alps ...
[read more]

Destination Snow: Cortina d'Ampezzo, Italy | SkiEurope Report 07-06-06

Thu, 06 Jul 2006 17:29:00 CST

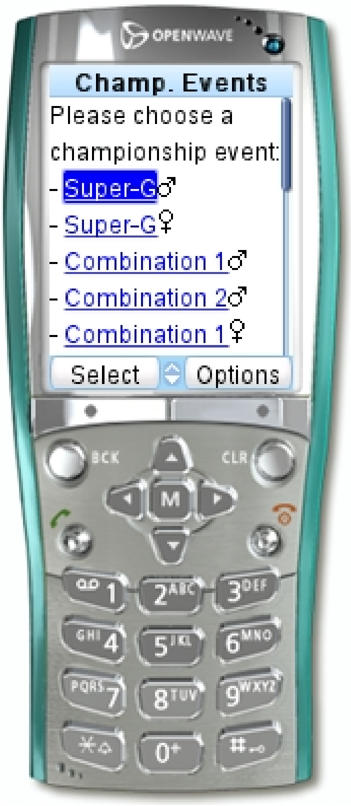
As the small village of Cortina in Italy's Ampezzo Valley is bathed in rosy pink light at sunset, it truly becomes magical. As evening falls, the passeggiata begins, and scores of locals and visitors fill the Corsa Italia on their evening promenade ...
[read more]

Figure A.4.: **External RSS feed**: Displayed using defaultfeed.xsl

Appendix B.

Mobile Site Structure and Layout

Screenshots taken from Openwave$^{\mathrm{TM}}$Phone Simulator 7.0 (Openwave, 2007)

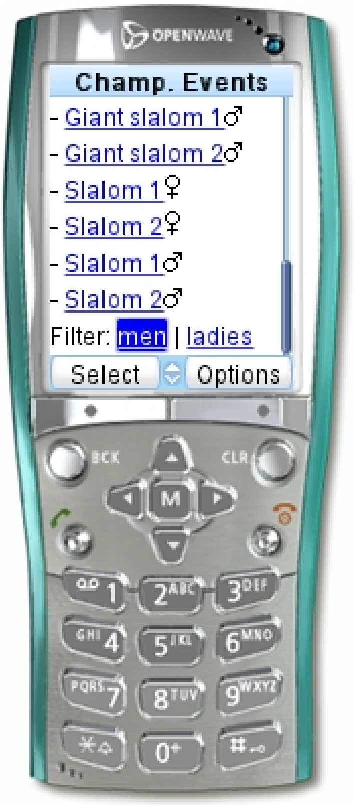


Champ. Events

Please choose a championship event:

- Super-G♂
- Super-G♀
- Combination 1♂
- Combination 2♂
- Combination 1♀

Figure B.1.: **Mobile Site Homepage** (screen 1): Displays all Event Title and a gender icon.

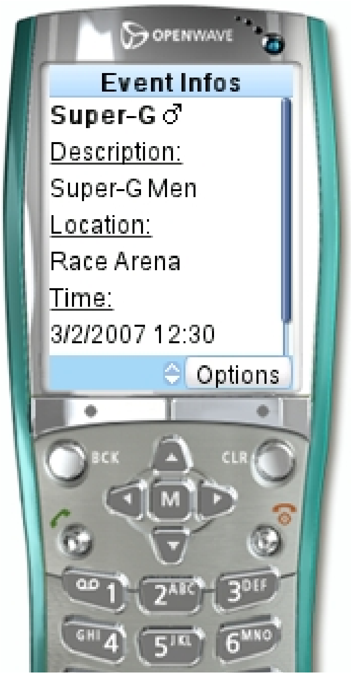


Champ. Events

- Giant slalom 1♂
- Giant slalom 2♂
- Slalom 1♀
- Slalom 2♀
- Slalom 1♂
- Slalom 2♂

Filter: men | ladies

Figure B.2.: **Mobile Site Homepage** (screen 2): Possibility to filter men and ladies.



Event Infos

Super-G ♂

Description:

Super-G Men

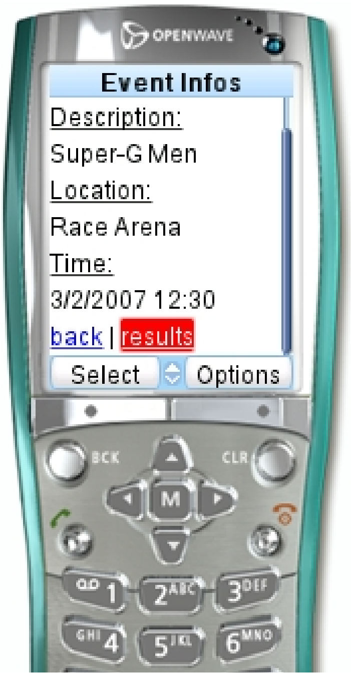
Location:

Race Arena

Time:

3/2/2007 12:30

Figure B.3.: **View of a single Event** (screen 1): Displays Event information.



Event Infos

Description:

Super-G Men

Location:

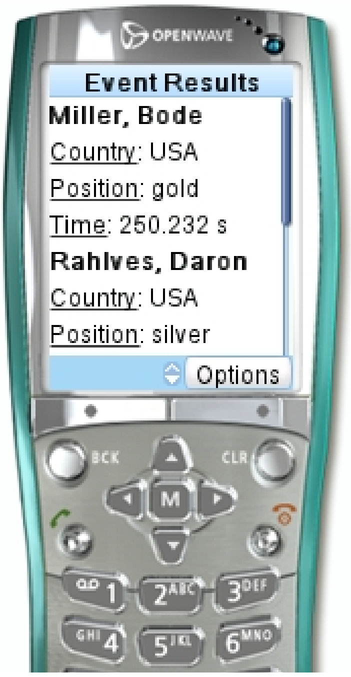
Race Arena

Time:

3/2/2007 12:30

back | results

Figure B.4.: **View of a single Event** (screen 2): Possibility to go back to the events or see results (if available).



Event Results

Miller, Bode

Country: USA

Position: gold

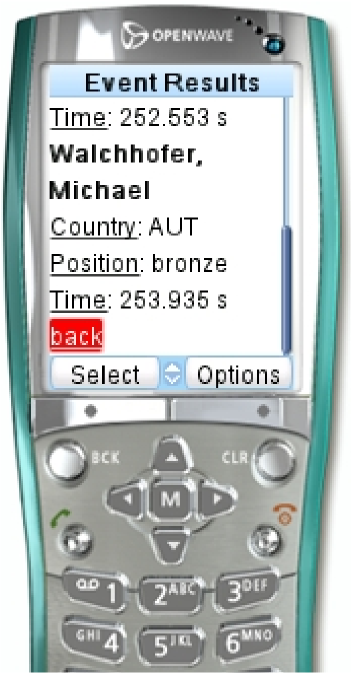
Time: 250.232 s

Rahlves, Daron

Country: USA

Position: silver

Figure B.5.: **View of Event Results** (screen 1): Displays the results for the chosen event.



Event Results

Time: 252.553 s

Walchhofer, Michael

Country: AUT

Position: bronze

Time: 253.935 s

back

Figure B.6.: **View of Event Results** (screen 2): Possibility to go back to the event view.

List of Figures

1.1. UML Sequence Diagram of a Page Request 3
1.2. UML Class Diagram of `XSLTProcessorUtils` 4

2.1. UML Class Diagram of `Services_Webservice` 8

A.1. XHTML Website: Homepage 10
A.2. XHTML Website: View of a single result 10
A.3. XHTML Website: The results as RSS feed 11
A.4. XHTML Website: Inclusion of an external RSS feed 11

B.1. Mobile Site: Homepage (screen 1) 12
B.2. Mobile Site: Homepage (screen 2) 12
B.3. Mobile Site: View of a single Event (screen 1) 13
B.4. Mobile Site: View of a single Event (screen 2) 13
B.5. Mobile Site: View of Event Results (screen 1) 13
B.6. Mobile Site: View of Event Results (screen 2) 13

List of Tables

2.1. The interface to the web service of the XML website. 9

Listings

1.1. rssfeeds.xml 5
1.2. rssfeeds.dtd 5
1.3. championship.dtd 5
2.1. XML Schema definition for the `id` type 7
2.2. XML Schema definition for the `position` type 8

Useful Links

Achour, Mehdi, Friedhelm Betz, Antony Dovgal, Nuno Lopes, Philip Olson, Georg Richter, Damien Seguy and Jakub Vrana. 2007. *PHP Manual.* Available at: http://uk.php.net/manual/en/.

Lanquetin, Nicolas. 2006*a*. *Subversion Change View.* [if online] Available at: http://home.psbase.com/viewsvn/?do=browse&project=xmltech&path=/trunk/.

Lanquetin, Nicolas. 2006*b*. *Subversion Repository.* [if online] Available at: http://guest:guest@home.psbase.com/svn/xmltech/trunk/.

Lanquetin, Nicolas. 2006*c*. *XML Technologies Classes Documentation and Source Code.* Available at: http://psbase.com/uad/xmltech/docs/.

Lanquetin, Nicolas. 2006*d*. *XML Technologies Project (HTTP).* Available at: http://psbase.com/uad/xmltech/.

Lanquetin, Nicolas. 2006*e*. *XML Technologies Project (WAP).* Available at: http://psbase.com/uad/xmltech/?type=wml.

Openwave. 2007. *Openwave Phone Simulator.* Available at: http://developer.openwave.com/dvl/tools_and_sdk/phone_simulator/index.htm.

PEAR. 2007*a*. *PEAR Package Services_Webservice.* Available at: http://pear.php.net/package/Services_Webservice.

PEAR. 2007*b*. *PEAR Webservice Packages.* Available at: http://pear.php.net/packages.php?catpid=23.

PHP. 2007*a*. *Classes and Objects (PHP 5).* Available at: http://uk.php.net/manual/en/language.oop5.php.

PHP. 2007*b*. *PHP 5 DOM Functions.* Available at: http://uk.php.net/manual/en/ref.dom.php.

PHP. 2007*c*. *PHP 5 XSL Functions.* Available at: http://uk.php.net/manual/en/ref.xsl.php.

PHP. 2007*d*. *PHP SOAP Functions.* Available at: http://uk.php.net/manual/en/ref.soap.php.

XMLSoft. 2007. *Libxslt Library.* Available at: http://xmlsoft.org/XSLT/.