

# 6

## Tecnologie per il Web 2.0

---

### 6.1 Introduzione

Nei sistemi software il numero di versione serve a indicare una release che comprende tutte le modifiche e i miglioramenti apportati nel tempo intercorso dalla versione precedente, congelati allo stato corrente, e fino al rilascio di una ulteriore versione. Il termine web 2.0 si rifà a questa convenzione, ma solo superficialmente, in quanto il Web è in evoluzione continua e non presenta un rilascio cadenzato di nuove caratteristiche come succede nei sistemi software tradizionali. Si prenda quindi Web 2.0 come una semplificazione atta a descrivere, in modo sommario ed a volte un po' vago, una serie di aspetti, tecnici ma soprattutto sociali, relativi al Web degli ultimi anni. In particolare, i siti web 2.0 sono caratterizzati da una partecipazione maggiore dell'utente, sia in termini di produzione di contenuti e riconfigurazione di contenuti esistenti, sia in senso più strettamente riferito all'interazione con la pagina, che non è più una pagina statica destinata solo a presentare contenuti all'utente, ma si modifica a seconda degli input forniti dall'utente stesso o da altre fonti esterne.

Questo libro tuttavia si rivolge al programmatore web che intende sviluppare siti tramite tecnologie lato server, e pertanto il contenuto di questo capitolo in particolare terrà conto esclusivamente degli aspetti tecnologici relativi al Web 2.0, che in sintesi si possono descrivere come quelle tecnologie destinate a facilitare la produzione di siti interattivi, sia dal punto di vista dell'utente che dell'interazione tra applicazioni web.

### 6.2 Distribuire il contenuto con RSS

Alcuni siti web si occupano di fornire contenuto che viene aggiornato in continuazione, secondo uno schema molto semplice, come quello tipico dei blog (dove l'unità di contenuto è costituita dal post) o dai siti di news, dove l'unità di contenuto è costituita dal singolo articolo o notizia. Questi siti sono tipicamente costituiti da sistemi CMS, in cui gli autori di contenuto inseriscono notizie o post, che poi vengono impaginati in modo che appaiano su una pagina del sito in ordine cronologico inverso (si leggono prima le ultime notizie).

Nel 1999 Netscape pensò di popolare il suo portale con titoli e brevi notizie provenienti da altri siti, incorporando il contenuto da essi prodotto o un suo sommario. Per fare ciò HTML era inadeguato, in quanto non presentava semantica sufficiente a distinguere le porzioni di pagina web relative ad una singola notizia. A questo scopo

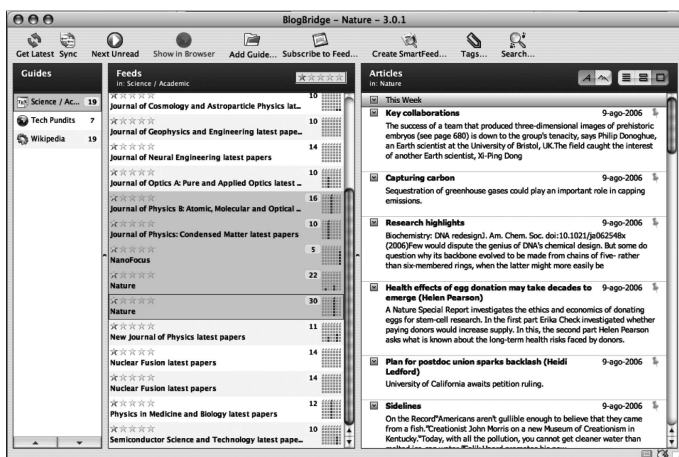


Figura 6.1: Una schermata di BlogBridge

quindi venne sviluppata la specifica di un nuovo linguaggio – RSS v. 0.9 (*RDF Site Summary*) – che permettesse proprio di identificare precisamente le singole notizie, fornendo così un estratto del sito web. Questa prima versione era un po' complessa da utilizzare poiché era basata su RDF (Resource Description Framework), un linguaggio ora in uso nel Semantic Web ed a sua volta basato su XML. Poco dopo Userland Software ne produsse una variante (0.91) basata direttamente su XML, e quindi più semplice da usare (*Really Simple Syndication*). Al momento esistono due versioni ufficiali del linguaggio: la 1.0, derivata dalla 0.9 e quindi basata su RDF, e la 2.0, basata su XML; nel seguito faremo riferimento a quest'ultima, in quanto più semplice da descrivere ed utilizzare.

Un sito che pubblichi notizie in continuazione, a fianco della tradizionale pagina web in HTML, pubblicherà anche una descrizione secondo il formato RSS, detta *feed*.

La descrizione fornita in formato RSS può essere utilizzata da almeno tre grandi categorie di software ed applicazioni web.

**Newsreader, o feedreader:** programmi in grado di effettuare il download automatico e temporizzato di più feed RSS, visualizzandone i contenuti in base alle esigenze dell'utente (per esempio, per data o per argomento). In questo modo l'utente può essere informato quasi in tempo reale quando una serie di siti vengono aggiornati, e vedere almeno i titoli delle nuove aggiunte, procedendo poi con il browser tradizionale per una lettura più approfondita. Una schermata di esempio da un newsreader (*BlogBridge*, <http://www.blogbridge.com>) è visibile in figura 6.1.

**Aggregatori:** come nelle intenzioni originali di Netscape, un sito può incorporare contenuto prodotto da altri siti, utilizzandone le descrizioni RSS e riorganizzando tali contenuti in base a criteri diversi da quelli di partenza; per esempio, in base al tema di ogni singolo item a prescindere da chi l'ha



Figura 6.2: Una schermata da Blogbabel

prodotto. Un tipico esempio è <http://news.google.it>, che raduna notizie giornalistiche provenienti da centinaia di siti, riorganizzandole per tema (politica interna ed estera, sport, tecnologia, ecc). *Technorati* è l'esempio più noto (<http://www.technorati.com>); un esempio italiano è BlogBabel (<http://it.blogbabel.com/>), visibile in figura 6.2), che riorganizza per tema post notificati autonomamente dai gestori di blog.

**Siti Mash-up:** Il feed o i feed di diversi siti, assieme ad altre fonti, contribuiscono a generare un nuovo sito web che va oltre il concetto di aggregatore di feed, secondo quanto discusso nel paragrafo 6.14.

### 6.2.1 RSS 2.0

Lo scopo di RSS è descrivere il contenuto di un sito della categoria prima descritta: pertanto conterrà una descrizione formale del *canale*, cioè del sito stesso, e poi un elenco di *item*, corrispondenti alle singole notizie (con più o meno dettaglio: titolo, autore, data, parole chiave, intero contenuto o una sua sintesi). Come già accennato, RSS 2.0 è basato su XML; vediamo quindi gli elementi che fanno parte del linguaggio.

L'elemento radice per questo formato è `<rss version="2.0">`, che conterrà uno o più elementi `<channel>` per descrivere il sito in complesso, o sue sottoparti (normalmente il canale è uno solo). Al suo interno ci saranno tre elementi obbligatori: `<title>` (il titolo del sito), `<link>` (l'URI del sito) e `<description>` (una breve descrizione del contenuto del sito). Con altri elementi opzionali è possibile esprimere ulteriori informazioni (`<language>`, `<copyright>`, `<managingEditor>`,

The image shows a screenshot of a blog page titled "Il Tao del blog". The main content area displays three posts:

- 21 - July - 2006**: A post titled "Semplice" by Alan Kay, with the text "Il miglior modo per predire il futuro è inventarlo." and 4 comments.
- 19 - July - 2006**: A post titled "Discussioni titaniche d'oltreoceano" by Peter Norvig, discussing the Semantic Web and Google's point of view. It has 3 comments.
- Il vero fattore C**: A post by Elio Vittorini, discussing the challenges of the Semantic Web. It has 0 comments.

The sidebar on the right contains an RSS 2.0 feed icon, a search box, and a list of "mini progetti" including "ditloids".

Figura 6.3: Un blog con tre post

<pubDate>, <category>, ecc). Seguono poi zero o più elementi <item>, ognuno dei quali descriverà una singola notizia del sito.

Gli elementi obbligatori per un dato item sono <title> (il titolo della notizia) e <description> ; tra gli opzionali, sono utili <link> (l'URI diretta della notizia), uno o più elementi <category> per descrivere le parole chiave (o *tag*) che ne descrivono il contenuto, e <author>, per identificare l'autore della notizia qualora non corrispondesse al managingEditor del sito.

Vediamo un esempio pratico. Un blog può avere l'aspetto visibile in figura 6.3 ed il suo feed può essere composto come segue:

### Listato 6.1: Un feed RSS 2.0

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <rss version="2.0">
3 <channel>
4 <title>Il Tao dei blog</title>
5 <link>http://www.webdomus.it/tao</link>
6 <description>quello che fa comparire una volta l'oscuro e una
  volta il chiaro</description>
7 <pubDate>Thu, 27 Jul 2006 07:55:02 +0000</pubDate>
8 <generator>http://wordpress.org/?v=1.5.3-beta1</generator>
9 <language>it</language>
11 <item>
12 <title>Semplice</title>
13 <link>http://www.webdomus.it/tao/?p=709</link>
14 <pubDate>Fri, 21 Jul 2006 09:13:23 +0000</pubDate>
15 <category>Citazioni</category>
16 <description>Il miglior modo per predire il futuro è
  inventarlo.</description>
17 </item>

```

```

18 <item>
19 <title>Discussioni titaniche d'oltreoceano</title>
20 <link>http://www.webdomus.it/tao/?p=707</link>
21 <pubDate>Wed, 19 Jul 2006 15:23:00 +0000</pubDate>
22 <category>Web Tecnica</category>
23 <category>Web Cultura</category>
24 <category>Tecnologie</category>
25 <category>Mondo</category>
26 <category>Informatica</category>
27 <description>Nel descrivere il piccolo screzio tra
    iperborei... </description>
28 </item>
29 <item>
30 <title>Il vero fattore C</title>
31 <link>http://www.webdomus.it/tao/?p=708</link>
32 <pubDate>Wed, 19 Jul 2006 15:00:59 +0000</pubDate>
33 <category>Citazioni</category>
34 <description>La cultura è la forza umana che ...</
    description>
35 </item>
36 </channel>
37 </rss>

```

Si noti come il sito includa anche altri elementi (il logo, eventuali link sulla barra laterale, altre informazioni) che però sono di contorno, oppure molto più stabili nel tempo. Ciò che viene riportato dal feed invece costituisce il contenuto variabile, riguardo il quale possiamo nutrire la necessità di notifica ed aggiornamento.

## 6.3 Caso di studio 1: generazione di feed RSS

Il blog sviluppato al paragrafo 3.7 è un ottimo punto di partenza per sviluppare uno script PHP che generi il feed RSS dei blog. Il necessario per generare codice RSS non è molto diverso da quel che serve per mostrare i post in una pagina (X)HTML: si tratterà solo di fare attenzione ad utilizzare i tag corretti secondo la specifica di RSS (in questo caso 2.0), estraendo il loro contenuto dal database in cui sono memorizzati i post. Il codice necessario è presentato nel listato seguente.

### Listato 6.2: Generazione di feed RSS

```

1 <?php
2 require("config.php");
3 require("funzioni.php");
4 header("Content-type: text/xml");
5 $conn = dbConnect();
6 $sql = "SELECT * FROM posts ORDER BY data DESC LIMIT 10";
7 $risposta = mysql_query($sql) or die("Errore nella query: "
8 . $sql . "\n" . mysql_error());
9 echo "<rss version=\"2.0\">\n",
10     "<channel>\n",

```

```

11     "<title>", $TITOLO, "</title>\n",
12     "<link>", $URL, "</link>\n",
13     "<description>Feed del blog ", $TITOLO, "</description>\n",
14     "<language>IT-it</language>\n",
15     "<generator>pwlsBlog</generator>\n";
16 while ($riga = mysql_fecch_array($risposta)) {
17     echo "<item>\n",
18         "<title>", $riga["titolo"], "</title>\n",
19         "<link>", $URL, "</link>\n",
20         "<description>", $riga["testo"], "</description>\n",
21         "<author>", $UTENTE, "</author>\n",
22         "</item>\n";
23 }
24 echo "</channel>\n</rss>\n";
25 mysql_close($conn);
26 ?>

```

Come si può vedere, lo script si limita a emettere in output i tag propri di RSS 2.0, utilizzando come contenuto quanto facente parte del blog. In particolare, l'elemento `channel` conterrà informazioni proprie del blog in generale, mentre gli elementi `item` sono ottenuti con un ciclo che estrae dal database gli ultimi 10 post. L'esito dell'esecuzione dello script sarà un file conforme alla specifica di RSS (listato 6.3), che arriverà al client che lo ha richiesto accompagnato dall'intestazione definita alla riga 4, che fa sì che il browser riconosca la risorsa come XML.

### Listato 6.3: Il feed generato

```

1 <rss version="2.0">
2 <channel>
3   <title>Quattro chiacchiere</title>
4   <link>http://www.miosito.com</link>
5   <description>Feed del blog Quattro chiacchiere</description>
6   <language>IT-it</language>
7   <generator>pwlsBlog</generator>
8   <item>
9     <title>Il blog funziona bene</title>
10    <link>http://www.miosito.com</link>
11    <description>Beh, devo dire che sono stato proprio bravo a
        realizzarlo.</description>
12    <author>CapitanBlogger</author>
13  </item>
14  <item>
15    <title>Questo &egrave; il mio primo post</title>
16    <link>http://www.miosito.com</link>
17    <description>Finalmente sono riuscito a realizzare il mio
        blog.<br/>Questo vuol dire che non vi libererete cos&
        igrave; facilmente di me, ora. :-)</description>
18    <author>CapitanBlogger</author>
19  </item>
20 </channel>
21 </rss>

```

Se il client è un browser web normale, probabilmente si limiterà a mostrare il codice XML o anche solo il contenuto dei tag. Alcuni browser però sono in grado di riconoscere e gestire opportunamente i feed; uno di questi è Safari, che mostrerà il feed del listato 6.3 ornato con un foglio di stile molto complesso, il cui esito è visibile in figura 6.4.



Figura 6.4: Un feed RSS visto con Safari

## 6.4 Caso di studio 2: un semplice aggregatore

In questo caso di studio realizzeremo un aggregatore di feed, in grado di leggere i feed di alcuni blog (configurabili), riordinare i post di tutti per data di pubblicazione, ed emettere una lista dei post in ordine cronologico inverso.

### 6.4.1 RSS è XML

Poiché in questo esercizio leggeremo i feed anziché crearli, è importante ricordare che RSS è comunque XML: pertanto useremo le funzioni fornite da PHP per gestire XML. Questo esercizio diventa così utile non solo per imparare come gestire i feed, ma anche per la gestione di XML.

Così come visto nel capitolo 3, anche in questo caso faremo uso della libreria *SimpleXML* per la manipolazione dei documenti XML in stile DOM.

### 6.4.2 Il codice per l'aggregatore

L'elenco dei feed da aggregare è fornito dall'array `$feeds`. Nella prima fase, avviene un ciclo su tale array, in modo da leggere ogni feed con la funzione `simplexml_load_file($feed)` che restituisce un oggetto di tipo

`SimpleXMLElement` contenente la radice del documento XML; in questo caso, sarà memorizzato nella variabile `$xml`.

Nello specifico, `$xml->channel` contiene l'elemento `channel` del feed RSS (cfr. listato 6.1), ed a sua volta `$xml->channel->title` contiene il titolo del blog. Inoltre, `$xml->channel->item` è un array contenente gli oggetti corrispondenti ad ogni singolo elemento `item` del feed, i cui sottoelementi sono di nuovo accessibili come proprietà (che è quel che viene effettuato nel ciclo più interno, righe 33-42, per leggere titolo, link e data di pubblicazione).

#### Listato 6.4: Il codice completo per l'aggregatore di feed

```

1 <html xmlns="http://www.w3.org/1999/xhtml">
2 <head>
3 <title>PWLS Aggregator</title>
4 </head>
5 <body>
6 <h1>PWLS Aggregator</h1>
7 <p>Questo aggregatore mostra i post di una serie di blog dotati
  di feed, ordinati in ordine cronologico inverso.</p>

9 <?php
10 //una funzione di supporto
11 function confronta($e1, $e2) {
12     $a = $e1['quando'];
13     $b = $e2['quando'];
14     if ($a == $b)
15         return 0;
16     return ($a > $b) ? -1 : 1;
17 }

19 //feed da aggregare
20 $feeds = array(
21     "http://www.apogeeonline.com/rss2/libri",
22     "http://attivissimo.blogspot.com/feeds/posts/default?alt=rss"
23     ,
24     "http://www.nazioneindiana.com/feed/",
25     "http://feeds.feedburner.com/mantellini/feed"
26 );

27 $i = 0;
28 //lettura dei feed
29 foreach($feeds as $feed) {
30     $xml = simplexml_load_file($feed);
31     $blog = $xml->channel->title;
32     $posts = $xml->channel->item;
33     foreach($posts as $post) {
34         $item['blog'] = $blog;
35         $item['titolo'] = $post->title;
36         $item['link'] = $post->link;
37         $item['quando'] = strtotime($post->pubDate);
38         if($item['quando'] > strtotime("2 weeks ago")) {

```

```
39     $lista[$i] = $item;
40     $i = $i + 1;
41 }
42 }
43 }

45 //riordino la lista
46 usort($lista, "confronta");
47 ?>
48 <ul>
49 <?php
50 //stampo l'esito
51 foreach($lista as $post) {
52     $quando = date("j/n/y H:i", $post['quando']);
53     echo "<li><a href=\"", $post['link'], "\">",
54         $post['titolo'], "</a>",
55         " (<em>", $post['blog'], "</em>, ", $quando, ")</li>\n";
56 }
57 ?>
58 </ul>

60 </body>
61 </html>
```

---

Il codice non presenta altre complessità, anche se vale la pena accennare a due funzioni che sono state utilizzate.

`strtotime(stringa)` è in grado di trasformare una stringa che descrive una data, una data ed un'ora, ecc., in un *timestamp* di Unix, cioè un numero intero. Detta funzione accetta la descrizione della data in moltissimi formati, rendendo le date facilmente confrontabili. Qui la usiamo in due momenti: per salvare nell'array la data di pubblicazione di ogni post (riga 37), e per capire se un post è più recente di due settimane, nell'istruzione condizionale immediatamente successiva (si noti l'eleganza dell'espressione "2 weeks ago", che viene correttamente convertita in un *timestamp*).

L'altra funzione interessante è `usort(array, funzione_di_confronto)` (riga 46). Essa fa parte di una famiglia di funzioni per l'ordinamento degli elementi di un array (`sort()`, `arsort()`, `asort()`, `ksort()`, ecc.), ognuna con delle caratteristiche peculiari. Nel nostro esempio usiamo `usort()` perché l'array che vogliamo riordinare ha elementi che sono a loro volta array associativi, ed il criterio di ordinamento è su una delle chiavi degli elementi stessi (`quando`, che contiene un *timestamp*): `usort()` permette di definire una funzione di confronto qualsiasi e di usarla come criterio di ordinamento. La funzione deve avere due parametri di ingresso (che saranno elementi del vettore), e deve restituire 0 se, secondo il nostro criterio sono uguali, -1 se il primo parametro viene prima del secondo, +1 se il secondo viene prima del primo. Questo è quanto viene implementato nella funzione `confronta($a, $b)`, che si limita a confrontare la data di pubblicazione dei due parametri.



**Figura 6.5:** Un aggregatore di feed RSS

Si noti infine che in questa implementazione non è previsto alcun controllo sugli errori: pertanto è necessario che i feed da aggregare siano validi e conformi, altrimenti il programma non funzionerà.

Il risultato dell'utilizzo dell'aggregatore è visibile in figura 6.5.

## 6.5 Web Service

Secondo la definizione ufficiale del Consorzio W3 (Garg et al. (2002)),

Un Web Service è un'applicazione software identificata da un URI, in cui le interfacce e le connessioni si possono definire, descrivere e ricercare attraverso artefatti XML, e supportano l'interazione diretta con altri applicativi software usando messaggi basati su XML, attraverso i protocolli di base di Internet.

In altre parole, un Web Service è una componente software che effettua delle operazioni, messe a disposizione remotamente ad altre applicazioni che risiedono su altri calcolatori tramite linguaggi basati su XML e su uno strato di trasporto costituito dai protocolli di base di Internet (principalmente HTTP).

Tramite Web Service si possono implementare applicazioni web secondo il paradigma della *Service Oriented Architecture* (SOA): le risorse computazionali sono rese disponibili come servizi indipendenti e modulari, accessibili con modalità standard e composte al fine di ottenere determinate funzioni.

Esistevano già delle tecniche di programmazione per l'esecuzione di procedure a distanza (RPC, Remote Procedure Call) o invocazione di metodi, nel caso orientato agli oggetti (RMI: Remote Method Invocation); la novità dei web service è però la totale indipendenza dal linguaggio di programmazione, per cui si potrebbe definire una funzione in Java su un server e richiamarla da un programma scritto in Fortran su un qualche client.

Le ragioni per l'utilizzo dei Web Service sono diverse:

- connessione tra sistemi informativi di aziende cooperanti: per esempio, si immagina un negozio online, che automaticamente contatta il sistema online dei fornitori quando il magazzino è vuoto;
- interfacciamento di sistemi legacy: funzioni realizzate da sistemi non sostituibili possono venire esposte ad altri sistemi aggiungendo uno strato software che implementa un web service;
- adozione di funzioni web in altre applicazioni: forse l'applicazione al momento più diffusa, permette di aggiungere ai propri programmi funzioni fornite da altri sistemi web (es. ricerche tramite motori di ricerca);
- ricerca e visualizzazione di informazioni complesse in agenti utente: per esempio, spedizioni via corriere, voli, ecc.

È possibile reperire numerosi esempi ed implementazioni di web service al sito <http://www.xmethods.net/>.

### 6.5.1 La tecnologia dei Web Service

In una struttura basata su Web Service, si possono distinguere alcuni attori principali:

- un fornitore di servizi (*service provider*) che offre una o più funzioni;
- uno o più richiedenti (*service requestor*) che necessitano di servizi remoti;
- eventualmente, un repertorio di servizi (*service registry*) ove ricercare il servizio più adeguato per un'applicazione.

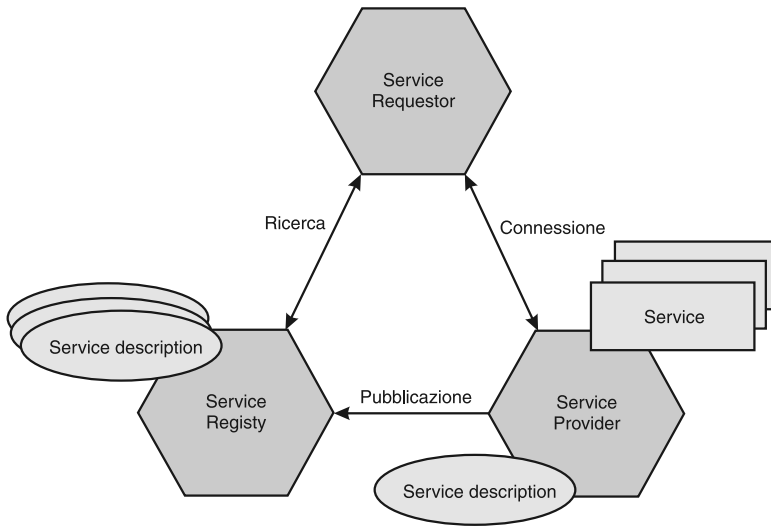
L'interazione tra gli attori appena menzionati segue lo schema visibile in figura 6.6.

Nella realizzazione dei Web Service si possono utilizzare alcuni standard o proposte di standard che identificano il formato dei messaggi, la descrizione dei servizi possibili, e la loro eventuale ricerca. In particolare, le specifiche coinvolte sono le seguenti:

**SOAP (Simple Object Access Protocol):** è il linguaggio (basato su XML) che identifica il formato dei messaggi scambiati tra service requestor e service provider.

**WSDL (Web Service Description Language):** è un altro linguaggio, di nuovo basato su XML, per descrivere le funzioni fornite da un Web Service, i parametri di ingresso e di uscita, e gli eventuali canali di comunicazione su cui il Web Service è disponibile.

**UDDI (Universal Description, Discovery and Integration):** è un framework per la realizzazione di service registry, ricercabili e aggiornabili tramite Web Service.



**Figura 6.6:** Gli attori coinvolti nei Web Service

Oltre ai web services standard, basati sulle summenzionate tecnologie, esistono anche delle tecniche di realizzazione meno standard, note con il nome di *RESTful Web Services*, ove REST significa *Representational State Transfer*. In sintesi, in questa modalità i parametri vengono passati al provider tramite URI (in maniera quindi semplice), mentre il risultato è un documento in formato XML o JSON o altro, composto secondo schemi stabiliti dal programmatore del service provider e quindi specifici di ogni provider. In questa sezione i Web Services di tipo REST non verranno trattati.

Analizziamo ora in dettaglio le due tecnologie di maggiore interesse.

## SOAP

SOAP, o *XML-Protocol*, è il linguaggio utilizzato sia dal service requestor per comporre la sua chiamata al web service, sia dal service provider per restituire il risultato. Il messaggio, similmente ad altri protocolli, è costituito da un'intestazione (**header**) ed un corpo (**body**), in questo caso racchiuso in un elemento **envelope**. L'intestazione può essere vuota; il corpo contiene nel caso della chiamata, il nome della funzione chiamata ed i suoi parametri, nel caso della risposta il risultato (o i risultati) della chiamata. Se parametri o risultati sono costituiti da file binari, è possibile allegarli al messaggio con lo stesso meccanismo degli allegati (attachment) della posta elettronica.

Il listato 6.5 mostra un possibile messaggio SOAP per la chiamata di una procedura `cambio(stato1, stato2)` che restituisce come risultato il rapporto di cambio tra le valute dei due stati.

---

**Listato 6.5: Richiesta SOAP**

---

```
1 <?xml version="1.0"?>
2 <SOAP:Envelope
3 xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope" />
4 <SOAP:Body>
5   <PROC:cambio xmlns:PROC="urn:cambiovaluta" >
6     <PROC:stato1>Italia</PROC:stato1>
7     <PROC:stato2>USA</PROC:stato2>
8   </PROC:cambio>
9 </SOAP:Body>
10 </SOAP:Envelope>
```

---

In questo caso, si passano come valori dei parametri di ingresso `stato1` e `stato2` le stringhe Italia e USA. Si noti che gli elementi XML prefissi da `SOAP:` appartengono alla definizione dello schema di SOAP<sup>1</sup>; gli elementi XML prefissi da `PROC:` sono invece specifici del web service invocato.

Il listato 6.6 mostra invece un possibile messaggio SOAP per la risposta alla chiamata di procedura effettuata con il messaggio del listato 6.5.

---

**Listato 6.6: Risposta SOAP**

---

```
1 <?xml version="1.0"?>
2 <SOAP:Envelope
3 xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope" />
4 <SOAP:Body>
5   <PROC:cambioRisultato xmlns:PROC="urn:cambiovaluta">
6     <PROC:risultato>1.28</PROC:risultato>
7   </PROC:cambioRisultato>
8 </SOAP:Body>
9 </SOAP:Envelope>
```

---

**WSDL**

La descrizione WSDL (da leggere come *wisdle*) di un web service può essere usata direttamente dal programmatore per “capire” come utilizzare il web service stesso, oppure automaticamente da un programma che permette di generare lo scheletro del codice che richiama detto web service. Entrambi sono dei supporti all’attività del programmatore.

A solo scopo illustrativo, il listato 6.7 riporta uno spezzone di possibile descrizione WSDL per il web service invocato nel messaggio del listato 6.5. Per i dettagli si suggerisce di esaminare le specifiche di WSDL.

---

<sup>1</sup>Il meccanismo utilizzato è quello dei cosiddetti *namespace*, che permettono di utilizzare più schemi nello stesso file. I dettagli di questo meccanismo sono al di fuori degli obiettivi del presente testo.