به نام خدا

مقاله وب سرویس

Web Service

تقدیم به

شما خواننده عزیز،

که لحضات خود را در راه یادگیری و تعلیم وقف نموده اید

فهرست:

مقدمه

یک وب سرویس به معنای ساده نوعی کامپوننت تحت وب است. این کامپوننت به برنامه‌هایی که از آن استفاده می‌کنند این امکان را می‌دهد که بتوانند از متدهای این وب سرویس استفاده کنند.

وب سرویس یک تکنولوژی است که امکان می‌دهد نرم‌افزارهای کاربردی، مستقل از نوع سیستم عامل و زبان برنامه‌نویسی با یکدیگر ارتباط برقرار کنند. یک وب سرویس، واسطه‌ای نرم‌افزاری است که مجموعه‌ای از عملیات را تعریف می‌نماید، که می‌توانند بر روی یک شبکه و از طریق پیام‌رسانی استاندارد شده XML مورد دسترسی قرار گیرند.

وب سرویس‌ها فراخوانی اشیاء و یا نرم‌افزارهای کاربردی را در محیط‌های گوناگون آسانتر می‌سازند و یک تکامل تطبیقی در محاسبات توزیع‌شده بحساب می‌روند.

دو رویکرد اصلی وب سرویس‌ها عبارتند از وب سرویس‌های .NET و وب سرویس‌های Java. از آنجایی که وب سرویس‌ها مستقل از پلاتفورم هستند، این دو نوع می‌توانند بدون اشکال با یکدیگر به تبادل داده‌ها بپردازند. چهار سرنام اصلی که در بحث از وب سرویس‌ها زیاد به گوش می‌خورند عبارتند از: UDDI، WSDL، SOAP و XML.

وب سرویس‌ها شامل خانواده‌ای از پروتکل‌ها هستند که عمل توصیف، تحویل و عمل متقابل با سرویس‌ها را انجام می‌دهند. این پروتکل‌ها می‌توانند به دو زیرگروه تقسیم شوند. زیرگروه اول با موضوعاتی چون پیام‌رسانی، توصیف واسطه و پاسخگویی به تحویل سرو کار دارند. زیرگروه دوم نیز پروتکل‌ها و مشخصاتی هستند که نحوه معرفی و یافتن وب سرویس‌ها در سطح وب را تعریف می‌کنند. در این بخش می‌خواهیم به وب سرویس و پروتکل‌های آن بپردازیم و نحوه ارتباط و کارکردهای هر کدام از این پروتکل‌ها را بیان کنیم.

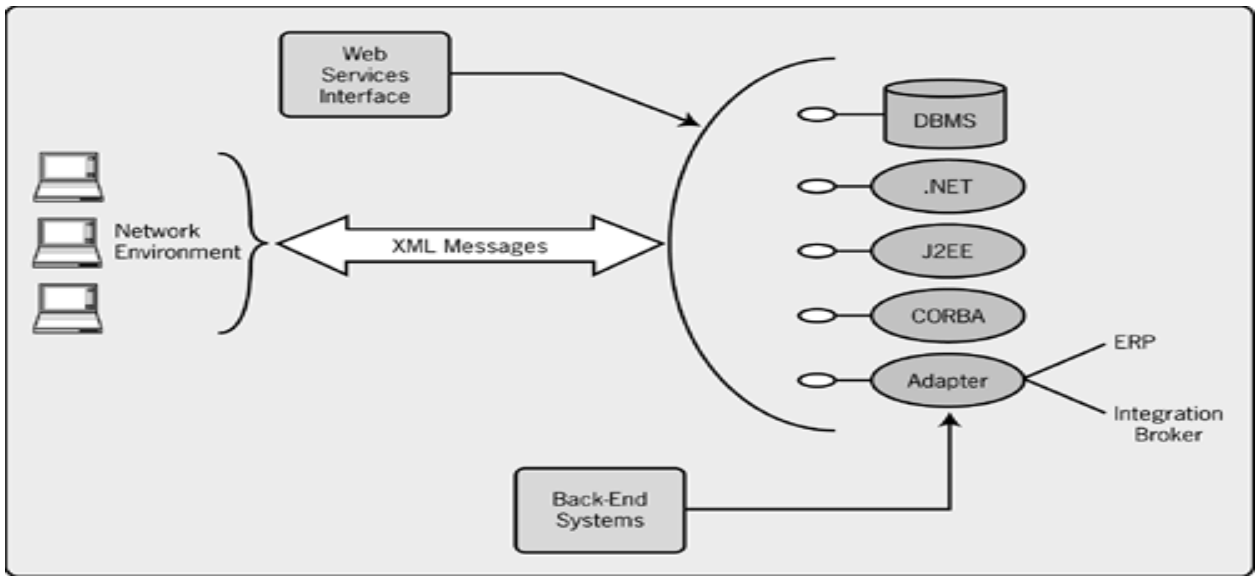
معرفی وب سرویس

تعریف وب سرویس

وب سرویس واسطی برای توصیف مجموعه‌ای از عملیات در سطح شبکه است که از طریق پیام‌هایی مبتنی بر XML، در دسترس می‌باشند. وب سرویس با استفاده از استانداردی (که بر پایه‌ی XML است) شرح داده شده است که توصیف سرویس (Service Description) نامیده می‌شود. این توصیف، شامل تمام جزییات لازم برای تعامل با سرویس، از جمله فرمت‌های پیام، پروتکل‌های انتقال و موقعیت آن می‌باشد. رابط، جزییات پیاده‌سازی سرویس را پنهان می‌کند و اجازه می‌دهد که سرویس به طور مستقل از سخت‌افزار و بستر نرم‌افزاری که روی آن پیاده شده است و همچنین مستقل از زبان برنامه‌نویسی که در آن نوشته شده است، مورد استفاده قرار گیرد. وب سرویس یک کار خاص و یا مجموعه‌ای از وظایف را انجام می‌دهد. آن‌ها می‌توانند به تنهایی مورد استفاده قرار گیرند و یا با وب سرویس‌های دیگر برای انجام مجموعه پیچیده از کارها همکاری داشته باشند.

یک برنامه، با استفاده از پیامی که مبتنی بر XML ایجاد شده است، درخواستی را به وب سرویسی که در شبکه وجود دارد ارسال می‌کند و پاسخی را در قالب پیام XML دریافت می‌کند. این فناوری می‌تواند در بسیاری از زمینه‌ها مورد استفاده قرار گیرد. وب سرویس می‌تواند در یکپارچه‌سازی B2B برای اتصال برنامه‌هایی که در سازمان‌های مختلف اجرا می‌شود، استفاده شود. وب سرویس می‌تواند مشکل توزیعی بودن EAI (Enterprise Application Integration) را با اتصال چند برنامه کاربردی که در یک سازمان در حال اجراست، به برنامه‌های کاربردی متعددی که در داخل یا خارج از دیواره آتش قرار دارد، حل کند. در همه این موارد، فناوری وب سرویس یک چسب استانداردی فراهم می‌کند که قطعه‌های مختلف نرم‌افزار را به هم وصل می‌کند.

همانطور که در شکل 1-1 زیر نشان داده شده است، وب سرویس وظیفه دریافت پیامی به صورت XML، تبدیل این پیام به شکل قابل فهم برای سیستم نرم‌افزاری خاص موجود در back-end و برگرداندن پیام پاسخ را دارد



شکل 1-1 : رابط وب سرویس با سیستم‌های back-end

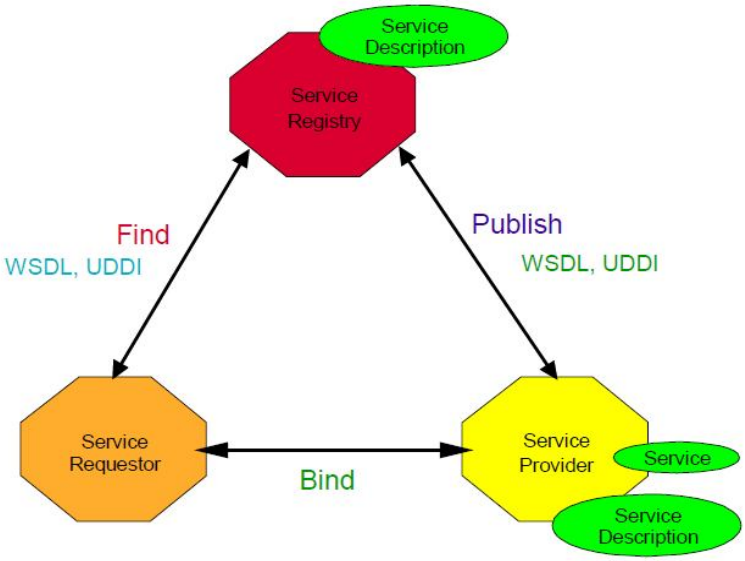
برنامه‌های تحت وب، بسته‌های نرم‌افزاری هستند که از طریق مرورگر قابل دسترسی می‌باشند. نرم‌افزار و پایگاه داده‌ی آن به جای نصب در سیستم کاربر، در یک سرور مرکزی قرار دارد و از طریق شبکه می‌توان به آن دست یافت. تفاوت بین وب سرویس و نرم‌افزارهای تحت وب، در جدول 1-1 زیر آمده است:

جدول 1-1: تفاوت وب سرویس با نرم افزارهای تحت وب

مشخصات وب سرویس	مشخصات نرم‌افزارهای تحت وب
از XML برای انتقال داده استفاده می‌کنند.	از HTML برای انتقال داده استفاده می‌کنند.
به هیچ سکو یا سیستم عاملی وابسته نیست.	وابسته به فناوری است (ASP، PHP و ...)
توسط برنامه‌های کاربردی فراخوانی می‌شوند.	توسط کاربران و با استفاده از مرورگر استفاده می‌شوند.

ساختار وب سرویس

معماری وب سرویس برپایه تعامل بین سه نقش قرار دارد: مهیاکننده سرویس، ثبت‌کننده سرویس (Service Registry) و درخواست‌کننده سرویس. عملیات شامل انتشار، یافتن و اتصال می‌باشد. این نقش‌ها و عملیات بر روی اشیا (Artifact) وب سرویس فعالیت می‌کنند (یعنی ماژول‌های نرم‌افزاری وب سرویس و توصیفات آن‌ها). مهیاکننده سرویس، یک توصیفی از سرویس برای وب سرویس ارائه می‌کند و آن را برای درخواست‌کننده سرویس یا ثبت‌کننده سرویس انتشار می‌دهد. درخواست‌کننده سرویس، توصیف سرویس را به صورت محلی یا از طریق ثبت‌کننده سرویس استخراج می‌کند و از آن توصیف برای تعامل با وب سرویس استفاده می‌کند.



شکل 1-2: ساختار وب سرویس

ثبت‌کننده سرویس (Service Registry)، یک ثبات قابل جستجو برای توصیفات سرویس است و مکانی است که مهیاکنندگان سرویس، توصیفات خود را در آن منتشر می‌کنند. عملیات در معماری وب سرویس عبارتند از:

انتشار (Publish):

برای اینکه سرویسی در دسترس باشد، توصیف آن باید طوری انتشار یابد که درخواست‌کننده سرویس بتواند آن را پیدا کند.

یافتن (Find):

در این عملیات، درخواست‌کننده سرویس، توصیف سرویس مورد نظر را به طور مستقیم و یا از طریق درخواست از ثبت‌کننده سرویس، استخراج می‌کند.

اتصال (Bind):

در نهایت، از یک سرویس باید استفاده کرد. در این عملیات، درخواست‌کننده سرویس از طریق جزییات لازم الاجرا در توصیف سرویس، شروع به ارتباط با سرویس می‌کند.

مزایا و معایب

وب سرویس‌ها همانند تکنولوژی‌های دیگر، دارای مزایا و معایبی هستند. از مزایای وب سرویس می‌توان به موارد زیر اشاره کرد:

قابلیت همکاری:

قابلیت همکاری بین برنامه‌های کاربردی متفاوت که روی سیستم‌های عامل غیریکسان در حال اجرا می‌باشند.

یکپارچگی خارجی:

تجمع آسان برنامه‌ها و سرویس‌ها از شرکت‌هایی با موقعیت جغرافیایی متفاوت برای ارائه یک سرویس یکپارچه.

استفاده مجدد از کد:

شکستن منطق و ایجاد توابع ایستا که می‌تواند توسط برنامه‌های کاربردی زیادی فراخوانی شود.

استقلال:

کدها می‌تواند توسط برنامه‌های کاربردی متفاوتی مورد استفاده قرار گیرد، برای مثال یک وب سرویس که با ASP.NET نوشته شده است، می‌تواند توسط یک صفحه JSP استفاده شود.[4]

از معایب وب سرویس نیز می‌توان موارد زیر را نام برد:

در دسترس نبودن :

هر کس که با اینترنت سروکار دارد می‌داند که هیچ سایتی 100 درصد در دسترس نیست. وب سرویس‌ها نیز زیرساختی مشابه وب‌سایت دارند، بنابراین وب سرویس‌ها هم این مشکل را خواهند داشت. حتی اگر سرور نیز در حال اجرا باشد، ممکن است ISP در دسترس نباشد. به علت وجود این مشکلات، لازم است مکانیزمی وجود داشته باشد که در صورت وقوع این اتفاق، درخواست را برای بار دوم ارسال نماید. برخی از پروتکل های جدیدی که توسط وب سرویس پشتیبانی می‌شوند، به صورت خودکار این عمل را انجام می‌دهند (مانند JMS) ولی اکثریت آن‌هایی که بر مبنای HTTP می‌باشند، از این مکانیزم پشتیبانی نمی‌کنند.

تطبیق نیازمندی‌ها:

زمانی که شما یک سرویس ایجاد می‌کنید و آن را در دسترس عموم قرار می‌دهید، می‌تواند مورد استفاده انواع مشتریان قرار گیرد، ولی این سرویس الزامات خاصی را فراهم می‌کند. ممکن است بعضی از مشتریان سرویسی را بخواهند که کمی با سرویس شما متفاوت است و فقط مورد نیاز اندکی از مشتریان باشد. وب سرویس یک تکنولوژی از نوع "اندازه ای متناسب برای استفاده حداکثر مشتریان" است. اگر این سرویس جوابگوی نیاز شما نیست، باید راه‌حل دیگری بیابید.

غیرقابل تغییر بودن:

اگر شما برای ایجاد یک وب سرویس سرمایه‌گذاری کنید، باید از هرگونه تغییر در پارامترها و متدهای آن اجتناب کنید. شما می‌توانید تغییرات را در قالب متدهای جدید به سرویس اضافه کنید، اما اگر تغییری در متدهای سرویس خود بدهید، برنامه‌های مشتریان را از بین خواهید برد. اگر متوجه شوید که سرویس شما پاسخ اشتباه به کاربران می‌دهد، شما بازهم چاره‌ای جز ایجاد یک سرویس جدید ندارید. اشتباهاتی از این قبیل در همه سیستم‌ها اتفاق می‌افتد که وب سرویس نیز مانند بقیه سیستم‌ها از این قاعده مستثنی نخواهد بود. ولی در سایر سیستم‌ها، یک راه ارتباطی بین ارائه‌دهنده خدمات و مصرف‌کنندگان آن وجود دارد که در صورت تغییر، به اطلاع کاربر می‌رسد.

ضمانت اجرا:

کل ایده وب سرویس این است که یک برنامه کامپیوتری به جای وارد کردن دستی کدها، به نیازمان پاسخ دهد که این برنامه بدون مراقبت اجرا می‌شود. HTTP یک پروتکل unreliable است و هیچ تضمینی از تحویل صحیح

داده‌ها ندارد. اگر شما به این ضمانت نیاز دارید، یا خودتان باید این مکانیزم را به صورت دستی بنویسید که تلاش مجدد را انجام دهد، و یا اینکه از طریق یک واسط (که دارای این ویژگی است) درخواست خود را ارسال نمایید.
تا اینجا مفاهیم اولیه وب سرویس را گفتیم. همانطور که اشاره شد، یک برنامه‌ی در حال اجرا برای فراخوانی یک روال که در ماشین دیگر قرار دارد، باید پیغامی مبتنی بر آنچه که در توصیفات سرویس مورد نظر آمده است، ارسال کند. در بخش بعدی به پروتکل‌های این ارتباط خواهیم پرداخت و اشاره‌ای به نحوه‌ی برقراری ارتباط با این روال‌ها خواهیم کرد.

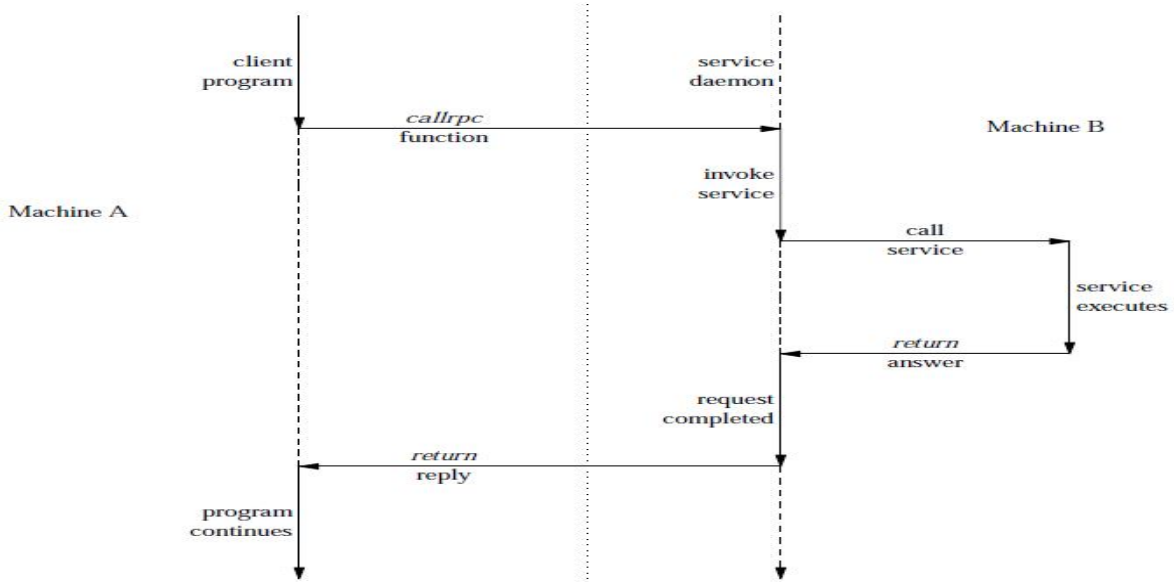
RPC (Remote Procedure Call)

تعریف RPC

یک تکنیک قوی برای شکل دادن به برنامه‌های *مبتنی بر Client/Server* و *توزیع شده* است. RPC یک متد است و مبتنی بر فراخوانی روال محلی (local procedure calling) است که در آن نیازی نیست روال فراخوانی‌شده در همان آدرس روال فراخوانی‌کننده ، قرار گرفته باشد. دو فرآیند ممکن است روی یک سیستم یکسان ویا روی سیستم‌های مختلف همراه با شبکه‌ای که آن‌ها را به هم وصل می‌کند، قرار گرفته باشند. با استفاده ازRPC ، برنامه-نویسان برنامه‌های توزیع‌شده از جزییات رابط با شبکه دور خواهند بود.

نحوه کار RPC

RPC مانند فراخوانی یک تابع است. وقتی RPC ایجاد می‌شود، آرگومان‌های فراخوانی‌کننده به روال دوردست ارسال می‌شوند و فراخوانی‌کننده تا بازگشت نتیجه، منتظر می‌ماند. شکل 1-3 جریان فعالیت را که در طول فراخوانی RPC اتفاق می‌افتد، نشان می‌دهد.



شکل 1-3: جریان فعالیت ها در طول فراخوانی RPC

Client یک فراخوانی روال را ایجاد می‌کند و آن را به Server می‌فرستد و منتظر پاسخ می‌ماند. Thread به حالت بلاک می‌رود تا زمانی که یا پاسخ دریافت شود و یا اینکه timeout اتفاق بیافتد.
زمانی که یک درخواست (به سرور) می‌رسد، سرور یک روال را برای اجرای سرویس درخواست‌شده فراخوانی می‌کند و پاسخ را به Client برمی‌گرداند. زمانی که فراخوانی RPC کامل شد، برنامه Client به اجرای خود ادامه می‌دهد.
در شکل بالا، روال موجود در سرور به طور خاص با این 3 پارامتر شناسایی می‌شود : شماره برنامه، شماره نسخه (ورژن)، شماره روال.

شماره برنامه، گروهی از روال‌ها را مشخص می‌کند که هرکدام دارای یک شماره‌ی روال منحصر به فرد هستند. یک برنامه ممکن است دارای یک یا چند نسخه مختلف باشد. هر نسخه شامل مجموعه‌ای از روال‌هاست که برای فراخوانی از راه دور در دسترس می‌باشند. شماره نسخه، باعث می‌شود که چندین نسخه از یک پروتکل RPC به طور همزمان در دسترس باشند. هر روال، یک شماره دارد.
در دنیای XML دو روش برای فراخوانی روال از راه دور (Remote Procedure Call) وجود دارد.

- XML-RPC
- SOAP

XML-RPC یک پروتکل RPC است که در سرتاسر اینترنت کار می‌کند. درخواست XML-RPC یک درخواست HTTP-POST است که بدنه این درخواست از XML می‌باشد. روالی که روی Server در حال اجراست، پاسخ درخواست را بر اساس XML می‌فرستد. پارامترهای روال می‌تواند از نوع عددی، رشته، آرایه و ... باشد.
در شکل 1-4 یک درخواست XML-RPC نشان داده شده است.

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
   <methodName>examples.getStateName</methodName>
   <params>
      <param>
         <value><i4>41</i4></value>
      </param>
   </params>
</methodCall>
```

شکل 1-4: درخواست XML-RPC

پاسخ این درخواست در شکل 1-5 آمده است.

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
<methodResponse>
   <params>
      <param>
         <value><string>South Dakota</string></value>
      </param>
   </params>
</methodResponse>
```

شکل 1-5: پاسخ مبتنی بر XML-RPC

انواع داده‌ها در XML-RPC

نوع داده‌های متداولی که به معادل XML آن تبدیل شده است، همراه با مثالی در جدول 1-2 آمده است:

جدول 1-2: نوع داده ها در XML-RPC

NO	Name	Tag Example
1	array	<array> <data> <value><i4>1404</i4></value> <value><string>Something here</string></value> <value><i4>1</i4></value> </data> </array>
2	base64	<base64>eW91IGNhbid0IHJlYWQgdGhpcyE=</base64>
3	boolean	<boolean>1</boolean>
4	date/time	<dateTime.iso8601>19980717T14:08:55</dateTime.iso8601>
5	double	<double>-12.53</double>
6	integer	<i4>42</i4> or <int>42</int>
7	string	<string>Hello world!</string>

8	struct	<struct> <member> <name>foo</name> <value><i4>1</i4></value> </member> <member> <name>bar</name> <value><i4>2</i4></value> </member> </struct>
9	nil	<nil/>

XML-RPC در سال 1998 توسط گروهی از افراد در ماکروسافت توسعه داده شد و پروتکل جدید با نام SOAP معرفی شد. SOAP مانند XML-RPC است ولی شامل ویژگی‌های بسیاری است که در XML-RPC ارائه نشده بود. [7]

از تفاوت‌های بین SOAP و XML-RPC می‌توان به موارد زیر اشاره کرد:

- XML-RPC ساده‌تر است.
- بسیاری از Web Serviceها بر مبنای XML-RPC هستند.
- SOAP انتقال اسناد پیچیده‌تر را ساده می‌کند. پس نیاز به تعریف تگ‌های بیشتر و پیچیده‌تری دارد.
- SOAP روی پرتکل‌های زیادی اجرا می‌شود، از جمله : HTTP,SMTP,FTP و ...
- SOAP سربار بیشتری دارد.
- XML-RPC با استفاده از خاصیت methodName خود، متدها را فراخوانی می‌کند که " ممکن است شامل حروف کوچک و بزرگ A-Z، اعداد، کالن و Slash باشد" که برای بسیاری از اهداف مناسب است ولی برای زمانی که بخواهیم یک شی را به عنوان آرگمان ارسال کنیم، با مشکل مواجه خواهیم شد.
- آرایه‌ها و structها همیشه بی‌نام هستند. [12],[11],[10]

سایر روش های RPC

XML-RPC تنها روش ایجاد RPC نیست. پروتکل‌های شناخته شده دیگری نیز وجود دارد از جمله CORBA، DCOM و ... . هر کدام از این پروتکل‌ها مزایا و معایبی نیز دارند.

CORBA

CORBA یک پروتکل معروف برای نوشتن برنامه‌های شی‌گرا به صورت توزیع شده است. این پروتکل به طور معمول در برنامه‌های کاربردی چندلایه استفاده می‌شود. CORBA توسط بسیاری از سازندگان و برنامه‌های کاربردی رایگان پشتیبانی شده است و با جاوا و C++ به خوبی کار می‌کند و نیز برای زبان‌های برنامه نویسی دیگر در دسترس است.

متاسفانه CORBA بسیار پیچیده است. بنابراین این پروتکل برای شرکت‌ها و برنامه‌های کاربردی کامپیوتری مناسب‌تر از برنامه‌های توزیع شده است.

DCOM

DCOM پاسخ ماکروسافت برای CORBA بود. این پروتکل زمانی که از مولفه‌های COM استفاده می‌کنید و نیز نیازی به ارتباط با سیستم‌های غیرماکروسافتی ندارید، بسیار مناسب خواهد بود، در غیر این صورت کمک چندانی به شما نمی‌کند.

لازم به ذکر است که تکنولوژی COM در خانواده سیستم عامل‌های ویندوز شرکت ماکروسافت، قابلیت ارتباط را برای مولفه‌های نرم‌افزاری فراهم می‌کند. COM توسط توسعه‌دهندگان برای "ایجاد مولفه‌های نرم‌افزاری با قابلیت استفاده مجدد (Re-usable)"، "پیوند مولفه‌های نرم‌افزاری برای ایجاد برنامه کاربردی" و نیز برای "بهره‌مندی از سرویس‌های ویندوز" مورد استفاده قرار می‌گیرد. اشیا COM را می‌توان با استفاده از زبان‌های برنامه‌نویسی متعددی ایجاد کرد. زبان‌های برنامه‌نویسی شی‌گرا، از جمله C++، دارای مکانیسم‌های برنامه‌نویسی هستند که می‌توان به سادگی اشیا COM را ایجاد کرد.

+COM، Distributed COM (DCOM)، و کنترل‌های Activex از خانواده COM هستند.

فصل دوم

XML

XML(eXtensible Markup Language) *و کاربردهای آن*

تعریف XML

XML سیستم گرامری ایجاد زبان‌های علامت‌گذاری دلخواه است. یعنی می‌توان آن را برای انواع داده‌های ریاضی، شیمی، تجاری و غیره به‌کار می‌رود. این زبان در سال 1996 توسط کنسرسیوم WWW ارائه شد و به سرعت مورد استقبال قرار گرفت. ساختار زبان XML اصول و قوانین قابل فهمی دارد که کارآیی آن را افزایش می‌دهد. اگر یک سند را بر اساس این قوانین ایجاد کنیم به آن سند، سند خوش‌فرم گفته می‌شود.

با گسترده شدن استفاده از وب در دهه 90 کم کم محدودیت‌های HTML مشخص شد. ضعف HTML در توسعه‌پذیری (قابلیت اضافه و کم‌کردن خواص) و ضعف آن در توصیف دیتاهایی که درون خود نگهداری می‌کند برنامه‌نویسان را از آن ناامید کرد. همچنین مبهم بودن تعاریف آن باعث شد از توسعه یافتن بازبماند. در پاسخ به این اشکالات W3C یک سری امکانات را در جهت توسعه HTML به آن افزود که امکان تغییر ساختار متن‌های HTML مهم‌ترین آن است. این امکان را CSS یا Cascade Style Sheet می‌نامند. تگ‌های تعریف شده در HTML ثابت هستند در حالی که در XML می‌توان تگ‌های خود تعریف ایجاد کرد.

اعتبار یک سند XML را با سندهای DTD یا shema می‌توان سنجید. این نوع سندها مشخص می‌کنند در سند XML چه تگ‌هایی را می‌توان استفاده کرد، این تگ‌ها چه صفت‌هایی دارند و ساختار تودرتویی آنها چگونه است.

برای نوشتن اسناد XML می‌توان از ویرایشگرهای ساده متن استفاده کرد مانند notepad. اگرچه ابزارهای کاملتری نیز مانند Visual studio تولید شرکت مایکروسافت نیز وجود دارد. در اصطلاح به هر زبانی که توسط XML ایجاد می‌شود، XML Application می‌گویند که به معنی کاربرد XML است نه برنامه کاربردی.

XML نیز مانند HTML از سه جزء اصلی عناصر، ویژگی‌ها و مقادیر تشکیل می‌شود. عنصر، اصلی‌ترین قسمت سند XML است و هر چیزی می‌تواند باشد؛ ازجمله، عناصر دیگر و یا متن. هر عنصر یک برچسب شروع دارد و یک برچسب پایان(شکل 2-1).[19]

شکل 2-1: تعریف یک عنصر

ویژگی‌ها در تگ شروع آورده می‌شوند و نشان‌دهنده یک صفت از آن تگ هستند. مقادیر متناظر با این ویژگی‌ها درون علامت کوتیشن در جلوی همان ویژگی نوشته می‌شود(شکل 2-2).

شکل 2-2: تعریف ویژگی

مقدار هر تگ بعد از اتمام تگ شروع و قبل از شروع تگ پایان قرار می‌گیرد و می‌تواند شامل تگ‌های دیگر یا یک متن ساده باشد که این‌ها را بر اساس shema یا DTD متناظر مشخص می‌کنیم.[19]

قوانین نگارش در XML

ساختار زبان XML اصول و قوانین قابل فهمی دارد که کارآیی آن را افزایش می‌دهد و استفاده از آن را بسیار آسان می-سازد که در ادامه به مهمترین‌های آن اشاره می کنیم.

- عنصر ریشه (Root element)

هر سند یا فایل XML باید یک عنصر ریشه داشته باشد که تمام عناصر سند را شامل می‌گردد. تنها توضیحات و دستورات پردازشی می‌توانند در خارج از عنصر ریشه قرار بگیرند. هر عنصر باید یک برچسب شروع و یک برچسب پایان داشته باشد.

- عناصر تودرتو

اگر قرار است عنصر A ابتدا و عنصر B بعد از آن بیاید، ابتدا عنصر B بسته می‌شود و سپس عنصر A.

- حروف کوچک و بزرگ

XML بین حروف کوچک و بزرگ فرق می‌گذارد. یعنی عناصر animal، ANIMAL و AnImaL با هم متفاوت هستند و می‌توانند با هم ارتباطی نداشته باشند.

در شکل2-3 نمونه‌ای از یک سند XML آمده است. این سند اطلاعات پزشکی یک فرد را در خود ذخیره کرده است.

```
<PatientRecord xmlns="http://www.medical.org/" xmlns:lab="http://www.lab.org/">
   <Name>John Doe</Name>
   <Account>123456</Account>
   <Visit date="10pm March 10, 2002">
      <Diagnosis>Broken second metacarpal</Diagnosis>
      <lab:Diagnosis><lab:Xray>encoded xray image</lab:Xray></lab:Diagnosis>
   </Visit>
</PatientRecord>
```

شکل 2-3: نمونه‌ای از یک سند XML

امنیت

داشتن امنیت در قابلیت اعتماد، حریم شخصی و یکپارچگی و امانت یک مساله حیاتی در کسب و کار آنلاین است. با رشد استقبال از تکنولوژی XML در اسناد و پروتکل‌ها، منطقی است که باید امنیت نیز با XML همراه شود. استاندارد امن XML لغت نامه و قوانین اجرایی را تعریف کرده است که نیازهای امنیت را برطرف کند. با پدیدار شدن XML، این استاندارد از نتیجه رمزنگاری و تکنولوژی‌های امنیت استفاده کرد تا یک راه‌حل انعطاف‌پذیر، توسعه‌پذیر و انجام‌پذیر را برای پاسخ دادن به نیازهای امنیتی ارائه دهد.

تکنولوژی‌های قدیمی‌تر امنیت یک مجموعه از الگوریتم‌های پایه‌ای امنیت را ارائه می‌دهند که می توانند در امنیت XML نیز استفاده شوند اما قالب های پیاده شده در آنها برای ایجاد امنیت برای بسیاری از برنامه‌های XML مناسب نیست. یکی از دلایل این عدم سازگاری این است که این استانداردها از شکل دودویی استفاده می کنند که نیاز به نرم افزار خاصی برای تفسیر پیغام ها است. دلیل دیگر این است که این استانداردها برای استفاده در XML طراحی نشده اند و از روش های معمول XML برای انتقال مفاهیم حمایت نمی کنند. مانند معلوم کردن محتوا با URI ها. علاوه بر این بعضی از تکنولوژی های موجود امنیت نیاز دارند تا با برنامه کامپیوتری یک پارچه شوند تا بتونند امنیت را تامین کنند.

امنیت برای کسب وکار آنلاین بسیار حیاتی است. تکنولوژی هایی که نیازمندیهای امنیت را برطرف کنند ایجاد شده اند. اما نیازها همچنان باقی هستند. این نیازمندی ها شامل احراز هویت ، اختیارت دسترسی، جامعیت ، امضای دجیتالی، محرمانگی، حریم خصوصی و مدیریت حقوق دجیتالی در زیر توضیح داده شده اند

- احراز هویت:

بدین معنی که چه کسی تقاضای دسترسی را دارد. مشخصات و قوانین یک گروهی را که یک فعالیت را انجام می دهد، مشخص میکند. مانند دسترسی به منابع و یا شرکت در یک انتقال.

- اختیارات دسترسی:

مشخص میکند کسی که احراز هویت شده است چقدر حق دسترسی دارد. مانند اینکه اجازه دارد یک صفحه اینترنتی خاص را مشاهده کند یا رمز عبور را تغییر دهد یا کمپانی را به پرداخت 10 میلیون دلار متعهد کند.

- جامعیت:

بدین معنی که از دست نخوردگی اطلاعات اطمینان حاصل کنیم. مطمئن شویم که اطلاعات تغییر نکرده اند چه تصادفی چه با نیت سوء. این اطلاعات می تواند یک صفحه وب باشد یا اطلاعات ذخیره شده در یک پایگاه داده و ...

- امضای دیجیتال:

ایجاد و تصدیق امضای دیجیتالی بجای امضای دستی. یک چنین امضایی برای مقاصد مختلف استفاده می شود مانند تصدیق، تایید وصول، پذیرش و توافق.

- محرمانگی:

اینکه محتوا را برای افراد غیر مجاز غیر قابل خواندن کنیم. مطمئن شویم که محتوا تنها توسط فرد مجاز قابل رویت است. محرمانگی به طور کلی با تکنولوژی های رمزنگاری همراه است، این درحالی است که دیگر روش ها مانند استگانوگرافی کمتر مورد استفاده قرار می گیرند.

- حریم خصوصی:

محدود کردن استفاده از اطلاعاتی که هویت یک شخص را معلوم می سازد. اطلاعات شخصی معمولا برای سازمان ها و ارگان ها نیاز است تا بتواند سرویس های مورد نیاز یک فرد را فراهم کنند. مثلا اطلاعات پزشکی فرد نیاز است تا دکتر شخص بتواند نظر درستی را ارائه دهد.

XML Security

تعریف

امنیت XML مجموعه ای از استانداردهای تکنیکی را معرفی می کند که نیازهای امنیت را برطرف می سازند. این استانداردها به گونه ای طراحی شده اند که با XML مطابقت داشته باشند. امنیت XML استانداردهای آن را در سطحی بالاتر قرار می دهند و به ترتیب زیر به XML کمک می کند:

1- استاندارد های امنیت XML یک مجموعه لغات را برای ارائه امنیت با استفاده از تکنولوژی های XML مانند شِما ،معرفی می کند. مثلا عنصر <KeyInfo> که در امضای دیجیتالی برای انتقال امضا یا اطلاعات رمزشده کلید استفاده می شود.

2- استانداردهای امنیت XML از استانداردهای دیگر XML بهر می برند تا کارایی را بالاتر ببرند. مانند XPath برای بازیابی بخشی از یک سند.

3- استانداردهای امنیت XML به گونه ای طراحی شده اند که انعطاف پذیری و توسعه پذیری XML را در خود داشته باشند. آنها امنیت را وارد اسناد، بخشی از یک سند، یک عنصر یا محتوای یک عنصر میکنند.

4- تکنولوژی های امنیت تقریبا به دوسر یک ارتباط وارد می شود. که این امر وقتی که پیغام های XML مسیر یابی می شوند بسیار مهم است. در طول انتقال امنیت باید همراه داده بماند. امنیت XML را می توان در ترکیب با امنیت های انتقال استفاده کرد مانند X.509 V3. الگوریتم های موجود مانند SHA1 را می توان با آدرس دادن وارد سند کرد.

جامعیت پایدار به کاربران داده اجازه می‌دهد که تغییرات اتفاق افتاده در داده را شناسایی کنند چه این تغییرات ناخواسته باشد چه با نیت سوء. در اینجا از checksum استفاده نمی‌کنیم، بلکه به جای آن، امضای دیجیتالی، یک اطلاعات را با یک خلاصه از آن همراه می کند که در واقع این خلاصه به نوعی اثر انگشت متن می‌باشد. این خلاصه یک ارزش با طول ثابت است که برای یک متن به طور یکتا تولید می‌شود و غیر ممکن است که بدون داشتن متن به آن دست پیدا کرد. استفاده از الگوریتم‌های رمزنگاری باعث شده است که تغییر کوچکی در متن اصلی این اثر انگشت بدهد. این قابلیت باعث می‌شود نه تنها در حین انتقال محتوا از جامعیت آن اطمینان حاصل کنیم بلکه در حین ذخیره و پردازش نیز این اطمینان باشد

ویژگی ها

امضای دیجیتالی یک مکانیزم برای حمایت از تولید و تایید انواع مختلفی از امضا ها را دارد که این قابلیت ها را به ما ارائه می دهد:

- تصدیق همه سند، بخشی از آن، یک عنصر یا محتوای یک عنصر
- تصدیق هر نوع سند حتی اسناد باینری
- تصدیق خواصی که به امضا اضافه میشود
- تصدیق اسناد مرکب
- تصدیق امضاهای مرکب(امضاهایی که خود شامل امضاهای دیگری هستند)

به علاوه، پیشنهادات امضای XML ، از برنامه هایی با چند امضا برای یک سند یا بخش های مختلف آن حمایت می کند.

یک عنصر <Signature> با توجه به برنامه کاربردی در چندین نقش ظاهر میشود. می تواند در سند قرار داده شود صرف نظر از اینکه چه چیزی امضا شده است. که در این حالت به آن امضای "منفصل"می گویند و این هنگامی که یک سند غیرXML را امضا می کنیم اتفق می افتد. هنگامی که یک سند XML امضا می شود، این عنصر می تواند به سند اضافه شود. این یک راحتی را ایجاد می کند به این خاطر که می توان عنصر را درون سند هرجا که نیاز است قرار دهیم. ومی توان به راحتی آن را پیدا کرد.در این حال به این عنصر امضای "پوشیده شده" می گوییم.

مفاهیم زیر مفاهیم اصلی برای درک امضای دیجیتالی XML است:

1- یک امضا تنها وقتی معتبر است که محتوای امضا شده تغییر نکند. برای درک این یک خلاصه کوتاه با طول ثابت از سند ایجاد می کنیم. بنابراین یک سند هنگامی معتبر است که خلاصه تولید شده در مبدا و مقصد برابر باشند.
2- یک عنصر <signature> یک ساختار XML است که شامل یک ارزش رمزشده است. که در عنصر <signature Value> می آید. محتوای <signedInfo> نباید تغییر کند تا سند معتبر باشد.
3- امضا کنند یک <Reference> برای هر عنصر اضافه شده ایجاد می کند. هر <reference> شامل یک خلاصه و یک URI برای هر آیتم است. همچنین توضیح می دهد که چگونه خلاصه را ایجاد کنیم ، الگوریتم را مشخص می کند و دیگر اطلاعات مورد نیاز. هر <reference> بخشی از ساختار <signedInfo> است

برای اعتبار سنجی امضا، گیرنده باید هر <reference> مستقلانه اعتبار سنجی کند با تولید خلاصه مشابه از هر آیتم.باید از URI ها برای رسیدن به مکان آیتم ها استفاده کند. و اطلاعات الگوریتم برای اینکه بداند چطور خلاصه را ایجاد کند. اگر عناصر تغییر نکرده باشند باید دو خلاصه برابر باشند

4- یک مرجع به هر چیزی توسط URI اشاره می کند که می تواند یک سند غیر XML باشد مانند یک عکس یا یک متن. این لازم نیست که آیتم را توسط URI بدست بیاوریم اما اغلب مفید فایده است. یک فرم خاصی از URI می تواند به یک عنصر خاص درون سند امضا شده XML اشاره کند.
5- یک <Reference> می تواند یک یا چندین بر تغییر کند تا برای یک عنصر مفید شود قبا از خلاصه سازی.یکی از استفاده های آن امضای بخشی از یک سند است که تغییر نمیکند.

هنگامی که یک امضا ایجاد شد می تواند به عنوان بخشی از سند یا جدای از آن قرار بگیرد.فرض کنید که دکتر در مثال قبل بخواهد که سند را امضا کند و امضا بخشی از سند شود(شکل 2-4)

```
<PatientRecord xmlns="http://www.medical.org/">
  <Name>John Doe</Name>
  <account id="acct">123456</Account>
  <Visit date="10pm March 10, 2002">
    <Diagnosis>Broken second metacarpal</Diagnosis>
    <lab:Diagnosis><lab:Xray>xhzhez</lab:Xray></lab:Diagnosis>
  </Visit>
  <Signature xmlns='http://www.w3.org/2000/09/xmldsig#'>
    <!-- the SignedInfo element and all it contains is what is signed -->
    <SignedInfo>

      <!-- Canonicalization is used to ensure that XML is handled consistently by
        different XML processors in light of white space and other variations. -->
      <CanonicalizationMethod algorithm="URI for algorithm" />

      <!-- the SignatureMethod is protected by the signature, avoiding substitution attacks
        and defines how the signature is created -->
>/      <SignatureMethod      Algorithm=http://www.w3.org/2000/07/xmldsig#rsa-sha1

      <!-- each item to be signed, XML document, portion of XML document
        or arbitrary content is represented using a Reference. Each Reference contains
        a digest of the item, a URI to refer to the item, and possibly
        transforms to apply to the item before creating the digest -->
      <Reference URI="">
<Transforms Algorithm="http://www.w3.org/TR/2000/WD-xml-c14n20000710"/>
        <DigestMethod Algorithm="http://www.w3.org/2000/07/xmldsig#sha1" />
        <DigestValue>
          Short, fixed-length "fingerprint" of referenced item
        </DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>
      encoded output of signature algorithm
    </SignatureValue>

    <!-- Optional KeyInfo used to convey key information needed to verify signature -->
    <KeyInfo>
      <KeyName>Sally Smith's Integrity Key</KeyName>
    </KeyInfo>

    <!-- optional Object to allow additional information to be associated with signature,
```

شکل 2-4: نمونه‌ای از یک سند امضا شده

محرمانگی : رمزنگاری XML

هدف و فواید

رمزنگاری XML یک مجموعه از عناصر XML و قواعد پردازش را تعریف میکند که محرمانگی را ایجاد میکند و آن را انواع مختلفی از محتوا ها اضافه میکند. رمزنگاری XML محرمانگی را در انتقال و ذخیره سازی فراهم می کند. تکنولوژی های دیگر محرمانگی نظیر SSL، TLS، یا VPN تنها محرمانگی را در سطح انتقال ایجاد می کند. صاحب اطلاعات می تواند آنها را رمز کند به طوری که کسی اطلاعات را متوجه نشود تا وقتی که رمزگشایی شود. رمزنگاری به طور کلی ترجیح می دهد که از کلید متقارن استفاده کند به این خاطر که این روش یک تکنولوژی مقرون به صرفه است حتی برای ایناد بزرگ. در این روش برای رمزنگاری و رمزگشایی از یک کلید استفاده می شود. برای ارسال یک اطلاعات رمز شده از فرستنده به گیرنده باید کلید را بین این دوطرف به اشتراک گذاشت اما دیگران از کلید اطلاعی نداشته باشند و این کار هنگامی که ملاقات حضوری نباشد، سخت میشود.

برای جلوگیری از این مشکل و اینکه بتوان اطلاعات محرمانه را بین افراد مختلف رد و بدل کرد، کلید متقارن یا کلید عمومی ایجاد شد. در این روش از دو کلید عمومی و خصوصی استفاده میشود که هر کدام اطلاعاتی را که دیگری رمز کرده است میگشاید. کلید عمومی در اختیار همه قرار میگیرد درحالی که کلید خصوصی فقط در اختیار شخص است. برای ارسال یک پیام، فرستنده پیام را با کلید عمومی گیرنده رمز می کند و گیرنده با کلید خصوصی خود آن را رمزگشایی می کند. به خاطر اینکه کارایی کلید عمومی نسبت به رمزنگاری کلید متقارن کارایی کمتری دارد برای ارسال کلید متقارن از رمزنگاری کلید عمومی استفاده می کنیم و اطلاعات را با کلید متقارن رمز می کنیم. سپس متن رمز شده و کلید متقارن رمز شده را با هم ارسال می کنیم.

خصوصیات

در رمزنگاری XML یک بستر و قوانین پردازش برای رمزنگاری و رمزگشایی تعریف شده است. یک مجموعه از عناصر XML را تعریف می کند که تمام اطلاعات لازم برای پردازش داده های رمز شده مانند الگوریتم استفاده شده برای رمز نگاری را در برمی گیرد. تمام خصوصیت های قابل پشتیبانی در زیر آمده است:

- محتواهای XML و غیر XML را می توان رمز کرد.
- نتیجه رمز نگاری یک سند خوش تریف XML باز هم یک سند خوش تعریف است. پس می توان بخشی از یک سند XML را رمز کرد و باز هم سند قابل پردازش باشد.
- رمزنگاری با امضای دیجیتالی در XML سازگار است و می توانند با هم استفاده شوند.
- از تعداد زیادی از الگوریتم ها پشتیبانی می کند

مفاهیم کلیدی

1- هنگامی که یک عنصر XML یا محتوای آن رمز میشود، با عنصر <EncryptedData> جایگذین می شود.

2- وقتی یک محتوای غیر XML رمز می شود، نتیجه یک سند XML جدید است که حاوی عنصر <EncryptedData> است.

3- یک عنصر <EncryptedData> حاوی صفت هایی است که به گیرنده کمک می کند تا بتواند متن را رمز گشایی کند. این صفت ها می تواند به عنصر یا محتوایی که رمز شده اشاره کند مثلا یک عکس. این فرآیند برای ضمیمه های ایمیل انجام می شود.

4- <EncryptedData> الگوریتم رمز نگاری را مشخص می کند، متن رمز شده را درون خود دارد و اطلاعاتی را که لازم است تا بتوان کلید رمزگشایی را بدست آورد.

5- کلید متقارنی که برای رمزنگاری کتن استفاده شده است در <EncryptedKey> قرار میگیرد.

6- XML از الگوریتم های انتخابی مناسب حمایت می کند و یک مشخصه برای حالات معمولی تعریف می کند که قابل توسعه برای دیگر حالات هم هست

7- تعریف اطلاعات کلید شاخص برپایه امضای دیجیتالی است

8- خصوصیت هایی که کاربر تعریف می کند می تواند با عنصر رمز شده ترکیب شود مانند.

9- متن واقعی رمز شده، نتیجه رمزنگاری، با استفاده از <cipherData> مشخص میشود. این عنصر حاوی عنصر دیگری است بنام <CipherValue> که مقدار رمز شده در آن است. یا حاوی یک اشاره به جایی که مقدار رمزشده در آنجا ذخیره شده است. که این مرجع درون <CipherReference> ذخیره می شود. یک مرجع هنگامی که متن رمز شده زیاد است مفید می باشد.

یکی از مسائل مهم در امنیت XML ارتباط بین امضای دیجیتالی و رمزنگاری است. فرض کنید که یک سند با امضای دیجیتالی و عنصر <EncryptedData> دریافت کرده اید مانند نمونه زیر:

```
<PatientRecord xmlns="http://www.medical.org/" xmlns:lab="http://www.lab.org/">
  <Name> John Doe </Name>
  <Account> 123456 </Account>
  <EncryptedData Type='element'>
    ...
  </EncryptedData>
  <Signature>
    <SignedInfo>
      <Reference URI="">
        ...
      </Reference>
    </SignedInfo>
  </Signature>
</PatientRecord>
```

شکل 2-5: ساختار یک سند امضا شده

در این نمونه، امضا بر روی کل سند اعمال شده است . برای اینکه URL درون عنصر <Reference>خالی است. در اینجا یک سوال پیش می آید: کدام یک زودتر باید بیاید، امضا یا رمزنگاری دانستن این نکته مهم است زیرا که امضا در صورتی معتبر است که متن تغییر نکند. اگر بخشی از متن بعد از امضا رمزنگاری شود، امضا دیگر معتبر نخواهد بود مگر اینکه بخش رمز شده رمزگشایی شود . دانستن ترتیب اعمال رمزنگاری و رمزگشایی مهم است زیرا که باید با تصدیق امضا تداخل ایجاد نکند. راه حل این مشکل این است که هنگامی که یک سند را امضا می کنیم، امضاکننده باید مشخص کند که چه بخشی از سند رمزنگاری شده است. این کار باعث می شود که قبل از تصدیق امضا بدانیم که چه بخشی از سند باید رمز گشایی شود.

فصل سوم

استانداردهای وب سرویس

WSDL

وب سرویس چهره جدید نرم‌افزاری از کسب و کار را نمایان کرده است که می‌تواند در سطح شکبه با هم تعامل کند. برای رسیدن به این تعامل باید وب سرویس به مشتریان بالقوه خود معرفی و شناسانده شود. علاوه بر این، کاربران باید بتوانند نحوه ارتباط با وب سرویس را پیدا کنند، اینکه چه داده‌هایی نیاز دارد و چگونه پاسخ خود را برمی‌گرداند و از چه پروتکل‌هایی استفاده می‌کند. برای این منظور استانداردی به نام WSDL وجود دارد. یکی از خواص وب سرویس‌ها توصیف خود آن‌هاست به این معنی که وب سرویس دارای اطلاعاتی است که نحوه استفاده از آن را توضیح می‌دهد. این توضیحات در WSDL نوشته می‌شود، متنی به صورت XML که به برنامه‌ها می‌گوید این وب سرویس چه اطلاعاتی لازم دارد و چه اطلاعاتی را برمی‌گرداند.

وقتی که سازندگان نرم‌افزار برای اولین بار SOAP و دیگر تکنولوژی‌های وب سرویس را ساختند دریافتند که برنامه‌ها قبل از اینکه شروع به استفاده از یک وب سرویس بکنند باید اطلاعاتی درباره آن را داشته باشند. اما هر کدام از آن سازندگان برای خودشان روشی برای ایجاد این توضیحات ابداع کردند و باعث شد که وب سرویس‌ها با هم هماهنگ نباشد. وقتی IBM و ماکروسافت تصمیم گرفتند تا استانداردهای خود را یکسان کنند WSDL بوجود آمد. در ماه مارس سال Microsoft 2001، IBM و Ariba نسخه 1.1 را به W3C ارائه کردند. گروهی از W3C بر روی این استاندارد کار کردند و آن را پذیرفتند. هم‌اکنون این تکنولوژی در دست ساخت است و هنوز کامل نشده. ولی اکثر سازندگان وب سرویس از آن استفاده می‌کنند.

هر وب سرویسی که بر روی اینترنت قرار می‌گیرد دارای یک فایل WSDL است که مشخصات، مکان و نحوه استفاده از وب سرویس را توضیح می‌دهد. یک فایل WSDL نوع پیغام‌هایی که وب سرویس می‌فرستد و می‌گیرد را توضیح می‌دهد مانند پارامترهایی که برنامه صدازننده برای کار با وب سرویس باید به آن بفرستد. در تئوری یک برنامه در وب برای یافتن وب سرویس مورد نظر خود از روی توضیحات WSDLها جستجو می‌کند. در WSDL اطلاعات مربوط به چگونگی ارتباط با وب سرویس بر روی HTTP یا هر پروتکل دیگر نیز وجود دارد.

این مهم است که بدانیم WSDL برای برنامه‌ها طراحی شده است نه برای خواندن آن توسط انسان. شکل فایل-های WSDL پیچیده به نظر می‌آید ولی کامپیوترها می‌توانند آن را بخوانند و تجزیه و تحلیل کنند. خیلی از نرم-افزارهایی که وب سرویس می‌سازند فایل WSDL مورد نیاز وب سرویس را نیز تولید می‌کنند بنابراین وقتی برنامه-نویس وب سرویس خود را ساخت به شکل خودکار WSDL موردنیاز با آن نیز ساخته می‌شود و احتیاجی به آموزش دستورات WSDL برای ساختن و استفاده از وب سرویس نیست. WSDL مخفف زبان توصیف سرویس‌های وب است و یک سند معمولی XML هستند. برنامه کاربردی می‌تواند یک سند WSDL را بخواند تا بفهمد چه عملیاتی بر روی سرور فعال هستند. WSDL قابلیت توسعه را دارد تا به سیستم انتهایی شبکه اجازه توصیفات و پیغام‌هایشان را بدهد صرف نظر از اینکه از چه ساختار پیغامی یا از چه پروتکلی برای ارتباط استفاده می‌کنند.

ساختار اسناد WSDL

یک سند را با استفاده از این عناصر کلیدی تعریف می‌کنیم:

<Types> : نوع داده‌ای که توسط وب سرویس استفاده می‌شود.

<messages> : پیغام‌هایی که توسط وب سرویس استفاده می‌شود.

<portType>: عملیاتی که توسط وب سرویس انجام می‌شود.

<binding>: پروتکل ارتباطی که توسط وب سرویس استفاده می‌شود.

<Port>: آدرسی را برای اتصال مشخص می‌کند.

<Service>: برای اجتماع مجموعه‌ای از پورت‌های مرتبط استفاده می‌شود.

شکل اصلی یک سند بدین صورت است (شکل3-1):

```
<definitions>

<types>
 definition of types.....
</types>

<message>
 definition of a message....
</message>

<portType>
 definition of a port.....
</portType>

<binding>
 definition of a binding....
</binding>

</definitions>
```

شکل3-1:ساختار یک WSDL

عنصر binding دو صفت دارد: نام و نوع. صفت نام، نام پوشش را مشخص می‌کند که می‌توان هر نامی برای آن گذاشت. و نوع، به پورت مورد استفاده برای پوشش اشاره می‌کند. در این مثال glossaryTerms است. در عنصر soap:binding دو صفت داریم. صفت style می‌تواند rpc یا document باشد. صفت transport نوع پروتکل انتقالی را مشخص می‌کند که در اینجا HTTP است.

عنصر <PortType> بیشترین اهمیت را در یک سند دارد. این عنصر وب سرویس، کاری که انجام می‌دهد و پیغام-هایی که درگیر هستند را توصیف می‌کند. این عنصر نقطه اتصال به وب سرویس را مشخص می‌کند. می‌توان آن را شبیه توابع کتابخانه‌ای در نظر گرفت. درخواست-پاسخ معمولی‌ترین روش عملیات است اما WSDL چند نوع عملیات را تعریف کرده است.

One-Way: عملیات می تواند پیامی را دریافت کند اما پاسخ نمی‌دهد.

Request-response: عملیات می‌تواند پیامی را دریافت کند و حتما پاسخی برگرداند.

Solicit-response: عملیات می‌تواند پیامی را ارسال کند و منتظر جواب بماند.

Notification: عملیات می‌تواند پیغامی را ارسال کند اما منتظر پاسخ نمی‌ماند.[17]

شکل 3-2 یک مثال از عملیات یک‌طرفه است. زیرا عملیات ورودی دارد اما خروجی ندارد.

```
<message name="newTermValues">
 <part name="term" type="xs:string"/>
 <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
 <operation name="setTerm">
  <input name="newTerm" message="newTermValues"/>
 </operation>
</portType>
```

شکل 3-2: عملیات یک‌طرفه

شکل 3-3 نیز یک مثال از درخواست- پاسخ است. زیرا عملیات هم ورودی دارد و هم خروجی

```
<message name="getTermRequest">
 <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
 <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
 <operation name="getTerm">
  <input message="getTermRequest"/>
  <output message="getTermResponse"/>
 </operation>
</portType>
```

شکل 3-3: درخواست- پاسخ

در اینجا نام پورت glossaryTerms و نام عملیات getTerm است. این عملیات یک ورودی و خروجی دارد که ورودی آن getTermRequest و خروجی آن getTermResponse است. message، بخش هر پیغام و نوع آن را مشخص کرده است.

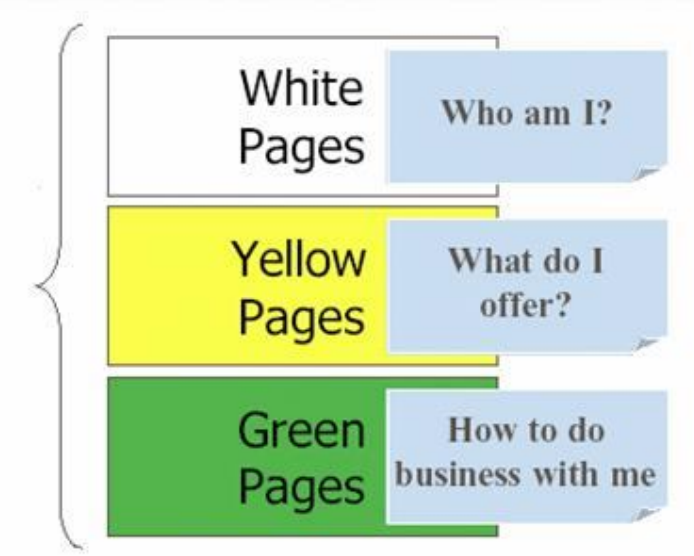
UDDI

(*Universal Description, Discovery and Integration*)

بعد از آنکه وب سرویس ایجاد و برای استفاده در سرور قرار گرفت، کاربران باید بتوانند آن را پیدا کنند. این هدف UDDI است. بایگانی UDDI اطلاعات توصیفی یک کسب و کار را دریافت و ذخیره می‌کند، که شامل اطلاعات وب سرویس‌هایی که در اختیار می‌گذارد نیز می‌شود. این بایگانی همچنین امکان جستجو برای اطلاعات خاصی را نیز در اختیار می‌گذارد. برای برقراری ارتباط با شرکت، باید اطلاعاتی از آن را داشته باشیم، مانند شماره تلفن، شهر و خیابان آن شرکت، وب سایت آن یا آدرس وب سرویس آن و .... UDDI. یک متن مبتنی بر XML را تعریف می‌کند که در آن شرکت‌ها توضیحاتی درباره چگونگی کار وب سرویس شرکت‌شان و امکانات خود می‌دهند. برای تعریف این اطلاعات از شکل خاصی که در UDDI توضیح داده شده استفاده می‌شود. شرکت‌ها می‌توانند این اطلاعات را در UDDIشرکت خود نگهداری کنند و تنها به شرکت‌های مورد نظرشان اجازه دست‌یابی به آن‌ها را بدهند یا آن‌ها را در مکان عمومی و در اینترنت قرار دهند.

بزرگ‌ترین و مهم‌ترین پایگاه UDDI پایگاه UDDI Business Registry یا UBR نام دارد و توسط کمیته UDDI طراحی و اجرا شده است. اطلاعات این پایگاه در چهار نقطه نگهداری می‌شود: مایکروسافت، IBM، SAP، HP. اطلاعاتی که در یکی از چهار پایگاه تغییر کند در سه تای دیگر نیز اعمال می‌شود.

اطلاعات درون این پایگاه‌ها شبیه دفترچه تلفن است که در آنها اطلاعات تماس شرکت‌ها و توضیحات متنی آنهاست، Yellow Pages حاوی اطلاعات طبقه‌بندی شده شرکت‌ها و اطلاعات درباره نوع فعالیت، شماره کسب و کار، نوع تولیدات آنها و ... می‌باشد. Green Pages حاوی اطلاعات تکنیکی درباره سرویس‌های آنها و نحوه پردازش اطلاعات شرکت آنها می‌باشد. White Pages حاوی اطلاعاتی در مورد خود شرکت است مانند نام شرکت و آدرس آن، اطلاعات لازم برای برقراری ارتباط، آدرس وب سایت و اطلاعات دیگر شناسایی.



شکل 3-4: اطلاعات درون یک UDDI

SOAP

(*Simple Object Access Protocol*)

هدف اصلی SOAP ایجاد روشی جهت فرستادن دیتا بین سیستم‌هایی است که بر روی شبکه پخش شده‌اند. وقتی یک برنامه شروع به ارتباط با وب سرویس می‌کند، پیغام‌های SOAP وسیله‌ای برای ارتباط و انتقال دیتا بین آن دو هستند. یک پیغام SOAP به وب سرویس فرستاده می‌شود و یک تابع یا سابروتین را در آن به اجرا در می‌آورد به این معنی که این پیغام از وب سرویس تقاضای انجام کاری را دارد. وب سرویس نیز از محتوای پیغام SOAP استفاده کرده و عملیات خود را آغاز می‌کند. در انتها نیز نتایج را با یک پیغام SOAP دیگر به برنامه اصلی می‌فرستد.

SOAP یک راه ارتباطی بین برنامه‌ها متفاوت بر روی سیستم عامل‌های متفاوت و با تکنولوژی‌های متفاوت و زبان‌های متفاوت است.

به عنوان یک پروتکل مبتنی بر XML، پروتکل SOAP تشکیل شده از یک سری الگوهای XMLی است. این الگوها شکل پیغام‌های XML را که بر روی شبکه منتقل می‌شود را مشخص می‌کند. مانند نوع دیتاها و اطلاعاتی که برای طرف مقابل تفسیر کردن متن را آسان کند. در اصل SOAP برای انتقال دیتا بر روی اینترنت و از طریق پروتکل HTTPطراحی شده است ولی از آن در دیگر مدلها مانند LAN نیز می‌توان استفاده کرد. وقتی که وب سرویس‌ها از HTTPاستفاده می‌کنند به راحتی می‌توانند از Firewall عبور کنند.

یک پیغام SOAP از سه بخش مهم تشکیل شده است: پوشش یا Envelope ، سرآیند یا Header، بدنه یا Body. قسمت پوشش برای بسته‌بندی کردن کل پیغام به کار می‌رود. این بخش محتوای پیغام را توصیف و گیرنده آن را مشخص می‌کند. بخش بعدی پیغام‌های SOAP ، Header آن است که یک بخش اختیاری می‌باشد و مطالبی مانند امنیت و مسیریابی را توضیح می‌دهد. بدنه پیغام SOAP بخشی است که دیتاهای مورد نظر در آن جای می‌گیرند. دیتاها بر مبنای XML هستند و از یک مدل خاص که الگوها (Schemas) آن را توضیح می‌دهند تبعیت می‌کنند. این الگوها به گیرنده کمک می‌کنند تا متن را به درستی تفسیر کند. پیغام‌های SOAP توسط سرورهای SOAP گرفته و تفسیر می‌شود تا در نتیجه آن، وب سرویس‌ها فعال شوند و کار خود را انجام دهند.[2]

شکل 3-5 قالب پیغام SOAP را نشان می‌دهد:

```
< ?xml version="1.0"?>
< soap:Envelope  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    < soap:Header>
        ...
    < /soap:Header>
    < soap:Body>
        ...
        <soap:Fault>
            ...
        </soap:Fault>
    < /soap:Body>
< /soap:Envelope>
```

شکل 3-5: قالب پیغام SOAP

عنصر Envelope ریشه یک پیغام است. این عنصر سند را به عنوان یک پیغام SOAP مشخص می‌کند. فضای نامی Xmlns:soap همیشه باید مقدار http://www.w3.org/2001/12/soap-envelope را داشته باشد. اگر فضای نامی دیگری استفاده شود برنامه یک پیغام خطا تولید می‌کند. و پیغام را نادیده می‌گیرد.
صفت encodingStylr برای مشخص کردن نوع داده استفاده می‌شود. این صفت در هر عنصری می‌تواند ظاهر شود و می‌تواند به یک عنصر و تمام فرزندان آن اعمال شود. یک پیغام SOAP هیچ کدگذاری پیش فرضی ندارد.
همانگونه که در شکل3-6 ملاحظه می‌شود، درخواست‌کننده و فراهم‌آورنده وب سرویس هر دو باید توانایی تولید و همچنین ترجمه پیام‌های SOAP و نیز توانایی برقراری ارتباط از طریق شبکه را داشته باشند . معمولا سرویس‌دهنده SOAP که بر روی سرویس‌دهنده وب اجرا می‌گردد این وظایف را انجام میدهد.

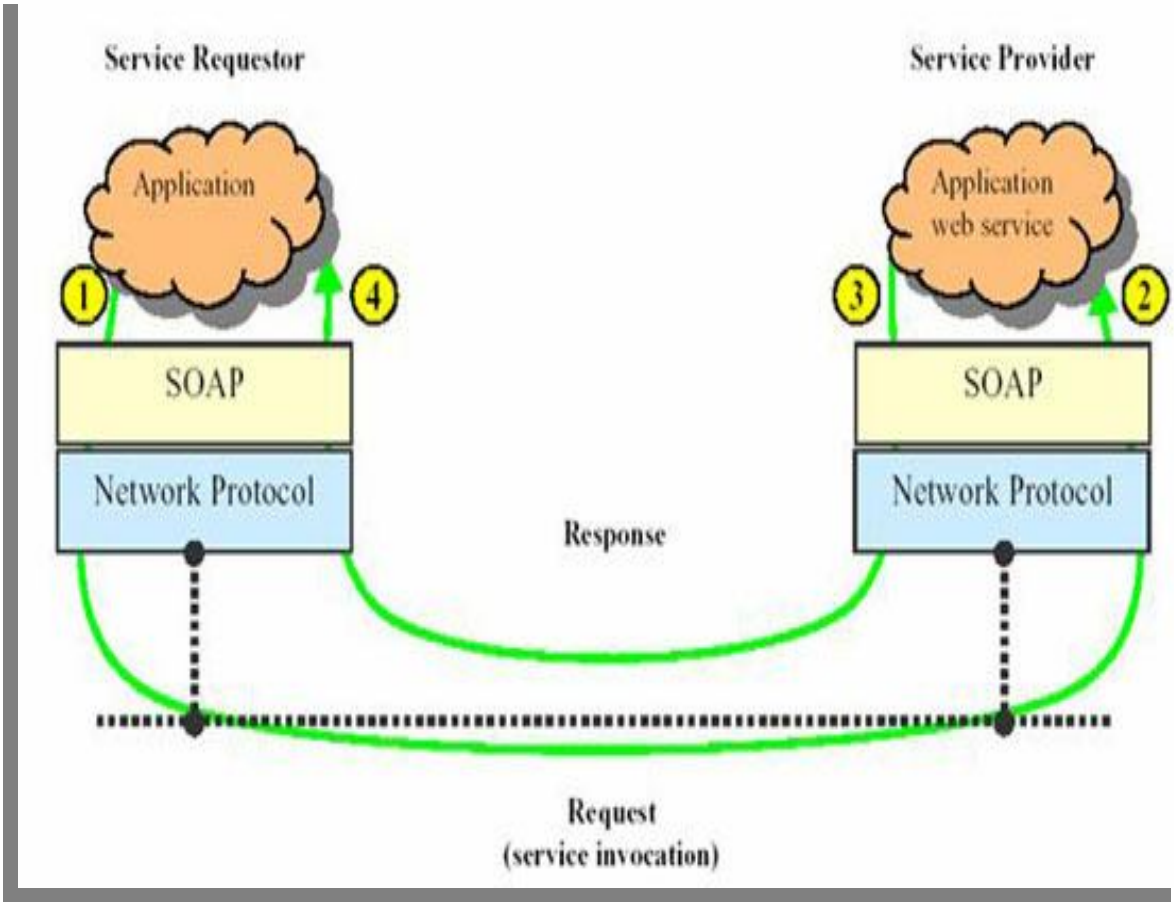
برقراری این ارتباط به چهار مرحله زیر تقسیم می‌شود:

1 – درخواست‌کننده یک وب سرویس ابتدا یک پیام SOAP مبنی بر درخواست آن از ارائه دهنده وب سرویس مورد نظر تولید می‌نماید. ارائه دهنده خدمات SOAP (برای مثال یک اجراکننده سمت کاربر SOAP) با استفاده از این پیام SOAP در کنار آدرس شبکه ارائه‌دهنده سرویس با برقراری ارتباط با پروتکل سطح شبکه (مثل HTTP) اقدام به ارسال پیام بر روی شبکه می‌کند.

2- این پیام بر روی شبکه (اینترنت یا اینترانت) به ارائه‌دهنده خدمات SOAP سمت ارائه‌دهنده سرویس (برای مثال یک سرویس‌دهنده SOAP) انتقال می‌یابد. سرویس‌دهنده SOAP بعد از تبدیل پیام، آن را به سرویس‌دهنده وب ارسال می‌کند. (سرویس‌دهنده SOAP توانایی تبدیل پیام‌های XML به اشیاء قابل استفاده تحت یک زبان خاص برای استفاده یک برنامه کاربردی را دارا می‌باشد)

3- سرویس‌دهنده وب با پردازش درخواستی که توسط پیام مطرح گردیده، پاسخ را تولید می‌کند . قطعا پاسخ برای انتقال باید به یک پیام SOAP تبدیل شود که این عمل نیز توسط اجرا کننده سمت سرویس‌دهنده SOAP انجام شده و در نهایت پیام SOAP تولیدی بر روی شبکه برای درخواست کننده ارسال می‌گردد.

4- بعد از دریافت پیام SOAP توسط گره درخواست کننده سرویس در شبکه، پیام SOAP رسیده، به اشیاء قابل استفاده توسط زبان برنامه‌نویسی مبداء تبدیل می‌گردد و نهایتا نتیجه توسط برنامه کاربردی استفاده کننده از وب سرویس نمایش داده می‌شود.



شکل 3-6: برقراری ارتباط در SOAP

فصل چهارم

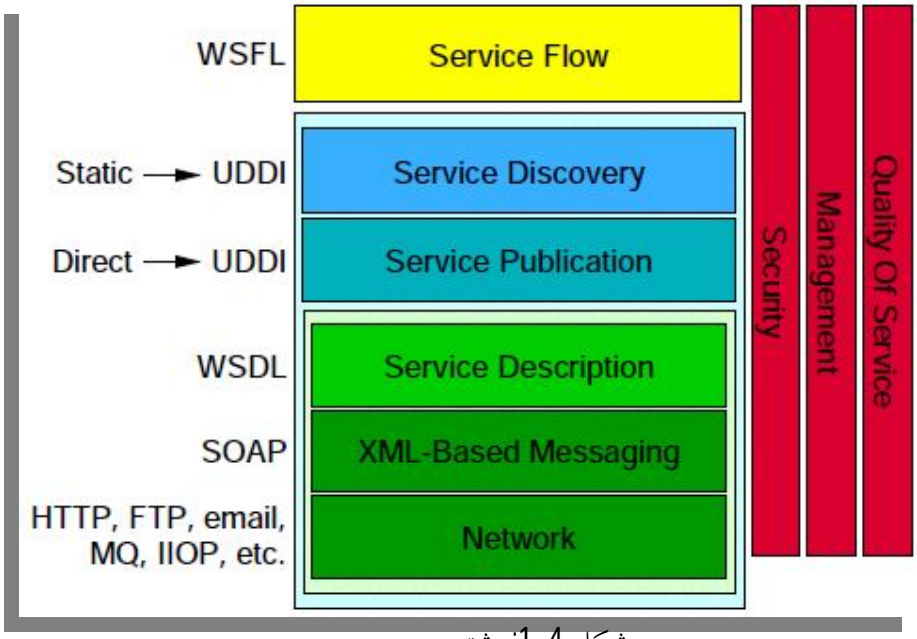
معماری وب سرویس

معماری وب سرویس

ابتدا نگاهی به پشته مفهومی وب سرویس می‌کنیم و جزئیات آن را بررسی می‌کنیم. سپس به موضوع پروتکل‌های شبکه و انتخاب آنها می‌پردازیم. علاوه بر استانداردهای پایه‌ای که قبلا توضیح داده شد، تکنولوژی‌های اضافه‌تری نیاز است که معماری وب سرویس را کامل کند تا از انواع مختلفی از برنامه‌ها را پشتیبانی کند. اگرچه استانداردهای پایه‌ای برای اهداف زیادی مناسب هستند، کسب‌وکارهای پیچیده‌تر نیاز به قابلیت‌ها و توابع اضافه‌تری دارند

پشته وب سرویس

برای به‌جا آوردن سه عملیات منتشر کردن، پیدا کردن و پیوستن، باید یک پشته وب سرویس وجود داشته باشد که استانداردهایی را برای هر سطح در برداشته باشد. شکل 4-1 پشته وب سرویس را مشخص می کند.



شکل 4-1: پشته وب سرویس

لایه بالاتر بر روی قابلیت‌هایی که لایه پایین‌تر ارائه می‌دهد بنا شده‌است. قسمت عمودی سمت راست، نشان‌دهنده نیازمندی‌هایی است که باید در هر لایه پشته مورد توجه قرار بگیرد. متن‌های سمت چپ نشان‌دهنده تکنولوژی‌های استانداردی است که در آن سطح عمل می‌کنند. پایه پشته وب سرویس شبکه است. وب سرویس باید به شبکه دسترسی داشته باشد که قابلیت در دسترس بودن را داشته باشد و بتواند توسط درخواست‌های دیگران فراخوانی شود. وب سرویس‌هایی که آشکارا در اختیار عموم قرار دارد از پروتکل‌های معمولی شبکه استفاده می کنند. به خاطر حضور

همیشگی وب سرویس پروتکل HTTP عملا پروتکل مورد استفاده آن است. پروتکل‌های دیگر اینترنت نظیر STMP و FTP نیز می‌توانند پشتیبانی شوند.
لایه بعدی، پیام نگاری مبتنی بر XML، استفاده XML را به عنوان پایه پروتکل پیام‌نگاری معرفی می‌کند. SOAP برای این لایه انتخاب شده است به دلایل زیر:

- یک روش استاندارد پوششی برای پیغام های ارتباطی و RPC است.
- ساده است.
- بهتر از ارسال یک متن ساده XML است زیرا دارای ساختار مشخص است.
- SOAP از عملیات انتشار، پیدا کردن و اتصال پشتیبانی می کند.

لایه توصیف سرویس در واقع یک پشته توصیف سند است. اول، WSDL استاندارد توصیف وب است. این حداقل استانداردهای توصیف سرویس است. WSDL رابط و مکانزیزم ارتباط با وب سرویس را توصیف می کند.
چون وب سرویس توسط SOAP از طریق شبکه در دسترس قرار می گیرد، سه لایه اول پشته نیاز است تا بتوان وب سرویس را فراهم کرد یا مورد استفاده قرار داد. در ساده‌ترین حالت پشته پورتکل HTTP برای لایه شبکه، SOAP برای لایه پیام‌نگاری مبتنی بر XML، و WSDL برای لایه توصیف سرویس نیاز است. در پشته وب سرویس بنا بر نیاز می‌توان از پورتکل‌های دیگر نیز استفاده کرد.

لایه شبکه

در پایین‌ترین قسمت پشته وب سرویس لایه شبکه قرار دارد. این لایه همه پروتکل‌های لایه شبکه معرفی می‌کند: HTTP، SMTP، FTP، MQ، RMI، IIOP، e-mail و نظایر این‌ها. پروتکل‌های شبکه بسته به برنامه‌هایی که نوشته می‌شوند مورد استفاده قرار می‌گیرند.
برای وب سرویس، پروتکلی که در همه جا موجود است، استفاده می‌شود: HTTP.

XML Messaging-Based Distributed Computing

پایه اصلی معماری وب سرویس پیام‌نگاری XML است. در حال حاضر استاندارد مورد استفاده در SOAP است. SOAP یک روش پیام‌نگاری ساده و سبک برای مبادله ساخت‌یافته پیام‌ها است. از سه بخش تشکیل شده است که در فصل گذشته به آن پرداخته شد. SOAP با همه پروتکل‌های شبکه سازگاری دارد و می‌تواند از طریق آنها ارسال شود. در اکثر مواقع با این بخش‌ها مستقیما سر و کار نداریم.

مراجع

- [١] http://www.microsoft.com/com/
- [٢] **Getting Started with XML Security**, in: http://www.sitepoint.com/
- [٣] www.w3schools.com
- [٤]