



Tamino

X-Application

WHITE PAPER

Contents

Executive Summary	4
Introduction	5
Tamino XML Server - Made for Electronic Business	5
Architecture and Structure of the Framework	5
The Choice for JSP Tchnology	6
The Tag Library Layer	7
Dreamweaver and the Tag Library	7
Business Document Module	8
Store Module	8
The Real Estate Demo Application	9
Real Estate Behind the Scenes	9
Tamino X-Application Generator	10
Generator in Use	10
Tamino X-Application and Web Services	10
The Backbone of Web Services	10
Web Service Operations	11
Protocols and Communication	11
Support from Tamino X-Application	11
Distribution and Availability	13
Summary	13

Executive Summary

In the age of the Internet, electronic business is setting new standards for the design and development of Web-bound applications. Countless new products and services are pouring onto the market to pave the way to the future of business. However, all this does not make it any easier to create Web applications. Creative thinking and skills when combining tools are needed, because Web applications are organized in a multitude of layers:

- In most cases, the product presentation is created by an external design agency, and therefore exists in the form of static HTML pages.
- The internal workings of the application are formed by the application logic. The logic is implemented using conventional programming languages or through scripting, and it is integrated into the Web using, for example, CGI.
- A further layer is the database layer, which is linked to the application logic and must be capable of meeting the information-structure requirements of the Web.

The biggest hurdle in developing e-business applications is to neatly separate these three components while combining them into a coherent whole: presentation layer and application logic are frequently merged into a single block of programming code, which makes it impossible for designers to further manipulate the presentation layer.

Tamino X-Application provides a way out of this dilemma and, with Tamino XML Server, gives developers all the tools they need to create user-friendly and robust Web applications. Implementing this framework successfully strikes a balance between separating and integrating the presentation layer and the application logic, and it builds a bridge between the two. The generic application logic is neatly encapsulated in a JSP tag library, so that the HTML code can be tweaked directly by the designers at any time. The designers don't have to forgo using the usual productivity tools, such as Macro-media Dreamweaver, in their day-to-day work.

With Tamino X-Application, it is possible to connect static HTML documents to Tamino XML Server in a very short time. Developers can fall back on a robust and expandable architecture, which does the majority of the work for them.

Also with Tamino X-Application, organizations can save on a number of resources in creating applications for electronic business. The amount of time needed is greatly reduced and the team's productivity increases significantly. Thanks to the stability of the framework and the clear separation of the layers, workflow processes can be carried out in parallel. There is no loss of quality in the resulting product. Applications that were developed using Tamino X-Application are based on a robust architecture and have intuitive controls, thanks to the JSP Tag Library.

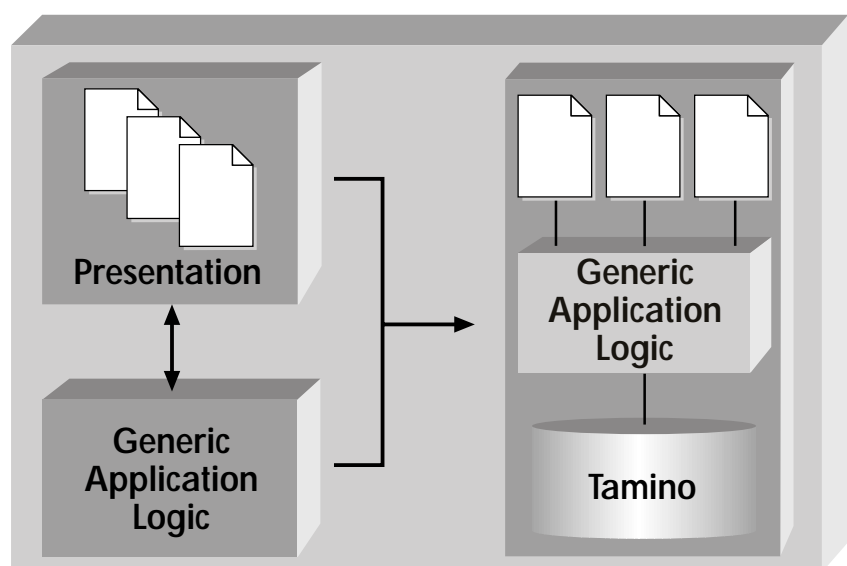


Fig. 1: Tamino X-Application is an engine that links static HTML pages to Tamino XML Server through a link with generic application logic, which makes the pages dynamic.

Introduction

With Tamino X-Application, Software AG has created a tool that simplifies the development process for Web applications considerably. In order to remove as much of the burden as possible from the developer, and at the same time meet the requirements of today's applications, the newest technologies are being implemented:

- XML is the language of the Web. Tamino X-Application takes this into account, and uses the XML format to exchange information of any kind.
- JSP is a scripting language derived from Java, which is used to create server-side presentation logic.
- Tamino XML Server is the first XML server system that can store XML data natively, that is without conversion to other data structures.

The Tamino X-Application framework consists of components that are based on this technology and connected to one another. Developers fall back on the prefabricated building blocks, and can thus dynamically link HTML pages to data from Tamino XML Server.

Another component of Tamino X-Application is Tamino X-Application Generator, which is a generic Web application that was created from a schema in Tamino. It offers the user a convenient graphic interface to search the database and to add, delete, and process new and existing information.

In addition, Tamino X-Application provides numerous support options, such as tutorials, demos, and documentation, to help developers who are embedding their applications in a Web service. Web services represent the next step in the development of distributed software components and are transforming the Web into a network of interacting programs.

Tamino XML Server-Made for Electronic Business

Tamino X-Application is based on a three-tiered architecture. Tamino XML Server, the first system that can store information in native XML format, is used for data storage. This has an enormous advantage over relational database management systems (RDBMSs), because RDBMSs are capable of storing XML only when retrofitted with special extensions:

- Conventional RDBMSs do not have the inherent hierarchical and semi-structured form that XML does. Instead, they use a transformation process to put the nested XML data into shallow tables and relations.
- SQL (Structured Query Language) is suited for tables that are subject to a fixed schema. It is not designed for the dynamic and complex nature of XML.

Because Tamino, in contrast, uses XML to store data natively, keeping its own structure, the essence is maintained. And Tamino's query language, X-Query, which is based on the familiar XML Path Language (XPath), is tailored exactly to the singularities of this format.

Architecture and Structure of the Framework

Software systems are growing ever more complex, while production cycles are growing shorter. Great care must be taken to make sure that applications are built on a robust and broad foundation of complete components. These components must eliminate significant time-consuming tasks from the developer's workload, and at the same time allow the developer sufficient scope for development.

In developing Tamino X-Application, this balance was struck perfectly: a neat architecture that is organized in modular layers lets developers decide to what extent they intend to use the framework's prefabricated building blocks or if they want to expand it by writing their own code. This underlying structure guarantees a clear division between the presentation layer and the application logic. Applications created with the help of Tamino X-Application can be changed and modified by designers, without them having to worry about the technical details that are underneath.

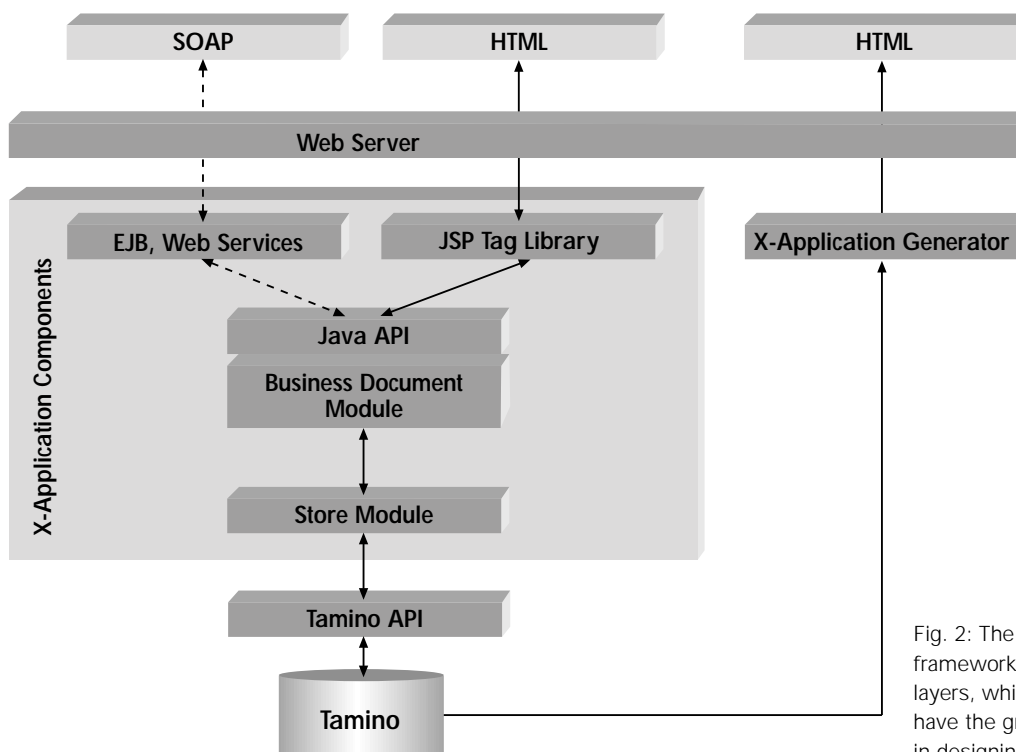


Fig. 2: The Tamino X-Application framework is divided into several layers, which allow developers to have the greatest possible flexibility in designing Web applications.

The functionality of the framework is created entirely in Java and is encapsulated in libraries. A series of JSP pages forms the server-side presentation layer. These pages embody the actual Web application that takes care of creating HTML code and is run by a JSP-capable Web server, for example the Tomcat server from the Apache Project. The JSP tag library from Tamino X-Application, which packs the complex Java classes into easily manageable JSP tags, is what builds the bridge between the presentation layer and the generic application logic.

THE CHOICE FOR JSP TECHNOLOGY

JSP (Java Server Pages) is a server-side scripting language that was developed by Sun Microsystems. Pages that have been coded in JSP can be understood as HTML that is pushed with Java commands, where the Java commands are carried out on the server and in turn take care of creating hypertext markup.

JSP has many competitors, such as Microsoft Active Server Pages and Allaire ColdFusion, which nevertheless have a crucial disadvantage: for almost all well-known servers, freely available modules can be acquired to run JSP, and this is generally not the case for the competitors.

The tag libraries are another important factor that makes JSP the server-side scripting language of choice. Tag libraries form an interface between business components and JSP. The purpose of this is to reduce the percentage of Java code in JSP pages as much as possible, in order to prevent the mixing of the presentation layer and the application logic. This is realized by associating Java classes with XML-compatible parameterizable elements. Additional tags are created that conceal certain application-specific behavior. Tag libraries are like sets of building blocks that can expand the functionality of predefined tools. In this example, a new tag that changes the tagged text to capital letters is defined through a library.

```
<%@page language="java" content-type="text/html" %>
<%@taglib uri="http://www.sag.com/test1" prefix="test" %>
<test:toUpper>this text will appear in uppercase</test:toUpper>
```

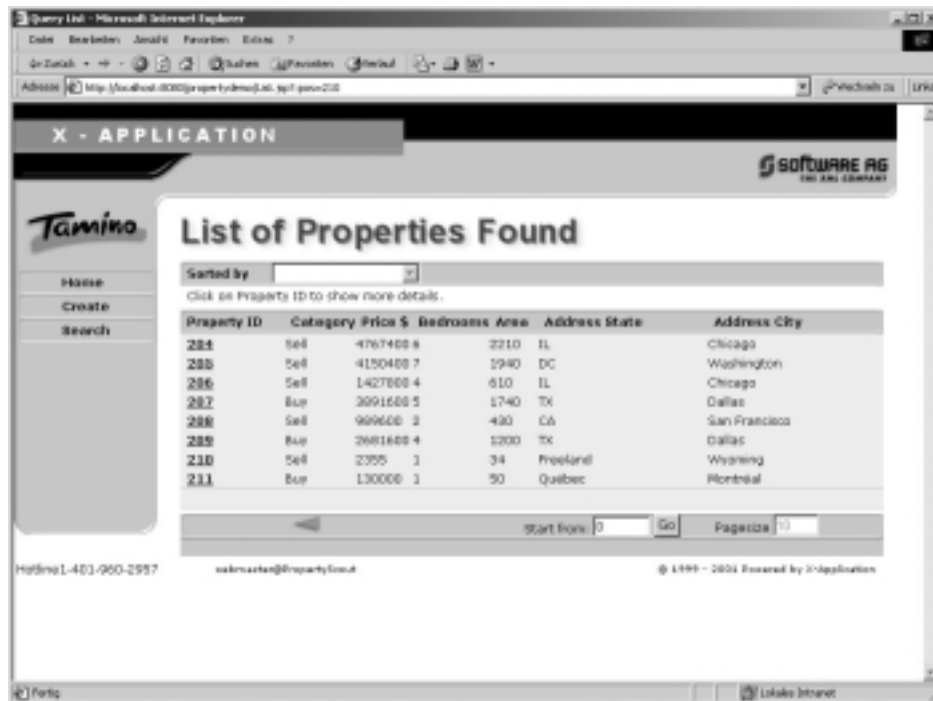


Fig. 3: Thanks to the JSP tag library, Tamino X-Application creates convenient graphic controls, such as the list that browses databases, shown here.

THE TAG LIBRARY LAYER

In the X-Application framework, the outer layer consists of a tag library that was created for the special requirements of a database-supported Web application. With the available tags, graphic controls can be created in a short time that allow the user to browse, change, delete, and create databases (see Figure 3). It is not necessary to know the details of how to control Tamino's data store when creating JSP pages, because the elements of the tag library relieve the developer of this heavy burden. Only path expressions that are related to XPath are transferred to the tags as an attribute in order to create an association between the control panel and the pertinent data in Tamino.

DREAMWEAVER AND THE TAG LIBRARY

WYSIWYG tools such as Macromedia Dreamweaver and Microsoft Frontpage have become popular for creating HTML pages. They became widely used because it is quicker and more efficient to compose Web pages graphically than to process them in a text editor. In a draft window, HTML elements can be placed on the page like building blocks and parameterized, and the results of the changes are immediately visible. In order for this convenience to be available for the JSP tag library in Tamino X-Application as well, the framework is shipped with an add-on for Macromedia Dreamweaver. When the user double-clicks on the file with the .mxp extension, the Dreamweaver Extension Manager is opened and installs the add-on automatically.

After the installation process is complete, an additional set of objects is available in the graphic editor (see Figure 4). This set contains graphic symbols for all the tags in the Tamino X-Application tag library, and each of them has its own dialog for setting attributes. From that point on, Dreamweaver recognizes all the framework's tags and can present them in design view.

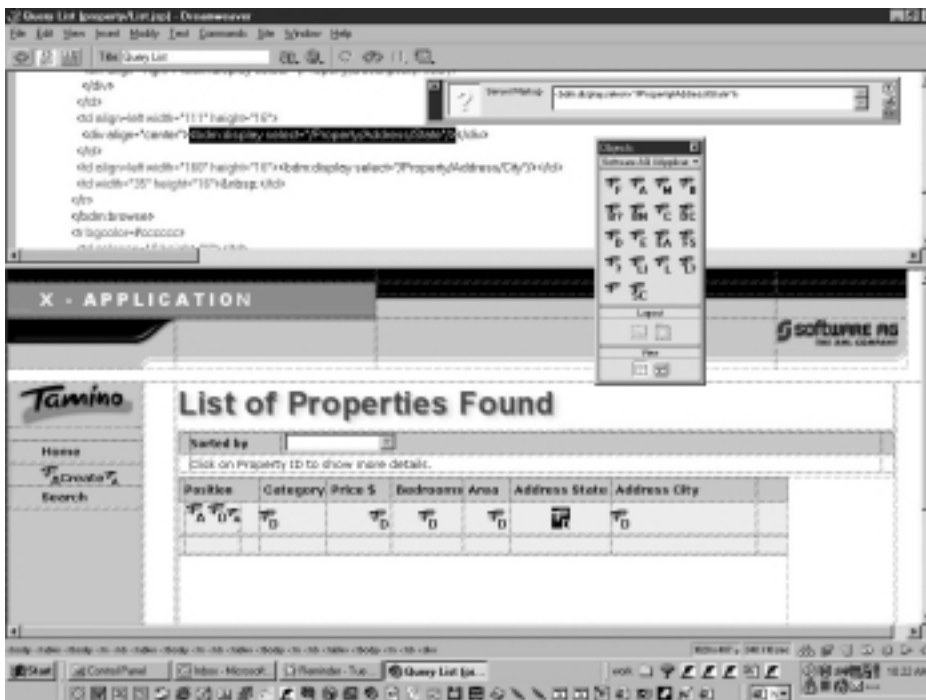


Fig. 4: The JSP library tags can be integrated into Macromedia Dreamweaver seamlessly using a plug-in module that comes with Tamino X-Application.

BUSINESS DOCUMENT MODULE

While the presentation layer of the framework is formed by the JSP tag library, the underlying Java API forms the actual application logic. Developers can access the Java API directly if they consider the functionalities provided by JSP tags to be insufficient or if, for example, the application will be included in a Web service. The Java API is subdivided into two layers: the first is the Business Document Module (BDM) and the second is the Store Module.

The BDM is responsible for maintaining XML documents and can keep several documents in its memory at the same time. In the context of relational systems, the functionality of this layer is somewhat comparable to operating on data sets via a cursor in a host language.

The following are a selection of the methods offered:

- Maintenance of several documents:
 - Object-oriented queries
 - Document browsing
 - Workspace management
- Maintenance of a single document:
 - Creation, reading, updating, and deletion of a document
 - Browsing of a single element
 - Access via XPath
 - State maintenance
 - Optimistic locking
 - Creation of copies of a document
 - Linking to schemas

The BDM is the middle layer of Tamino X-Application, because it provides the interface for accessing information in the data source. In doing so, it conceals the technical details for controlling Tamino and dealing with XML documents.

STORE MODULE

While the Business Document Module is the main support for the application logic, the Store Module represents the storage logic. This inner layer forms an abstraction level similar to ODBC, which abstracts the singularities of access to the Tamino database API (see Figure 2). Its functionality is used by the BDM, but can also be controlled directly by the application developer.

The modularity and separation of the application logic and the storage logic make it possible to interchange Tamino XML Server with other data sources, such as a file system. To do this, one need only substitute the appropriate Store Module.

The Real Estate Demo Application

Among the sample applications is the Real Estate Demo, which gives the user an idea of the functionalities of Tamino X-Application through an example from the field of real estate.

With a multitude of graphically appealing dialogs, the underlying database can be searched for real estate that is for sale or for homes that are being sought. It is possible to refine the search by entering various criteria. And it is possible to use various available forms to publish an offer or a search with all the relevant information. After the form is sent, the data is immediately transferred to the Tamino data pool and is then ready to be called up right away.

REAL ESTATE BEHIND THE SCENES

The sample application was created with the help of Tamino X-Application. Only the available tags from the JSP tag library were used to create the application; it was not necessary to extend their functionality by directly controlling the Java API. The framework of the database structure is formed by the property schema, which is shown in Figure 5.

A designer who had no programming knowledge was hired to give the interface, which was somewhat plain at first, the typical Tamino look. It was no problem for the designer to carry out this "facelift" and give the application its final, attractive appearance. This test clearly demonstrated the advantages of the Tamino X-Application architecture.

Not only is the developer relieved of a large part of the work, but also implementing the framework solidifies the clear separation of the presentation layer and the application logic.

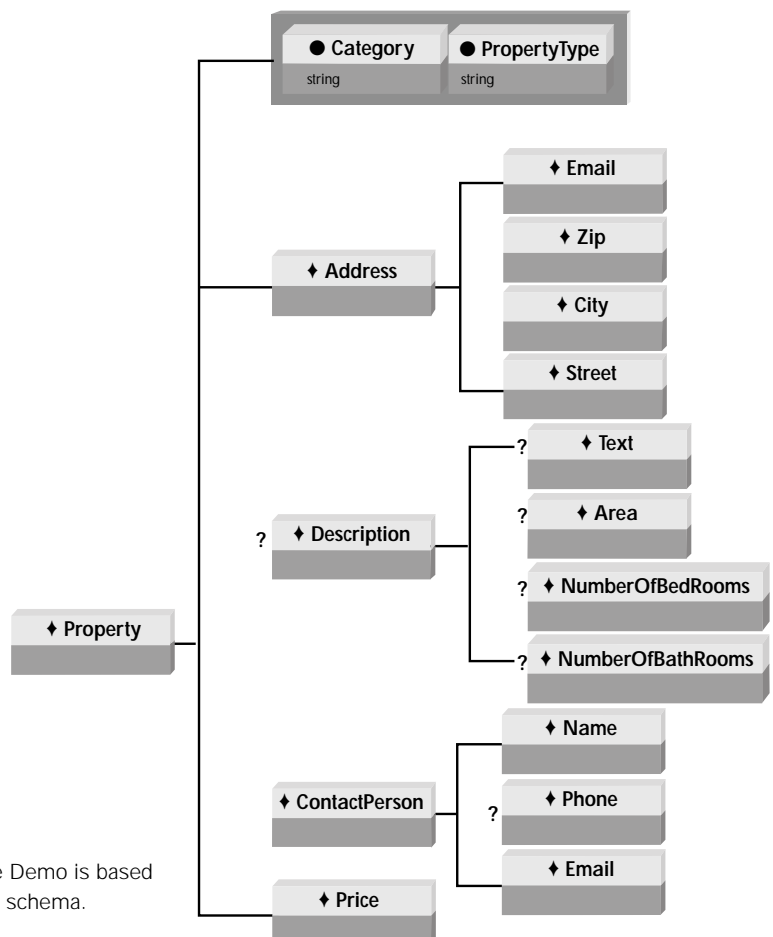


Fig. 5: The Real Estate Demo is based on the property schema.

Tamino X-Application Generator

The JSP tag library and the Java API aid developers in creating an application. In addition, with X-Application Generator, Tamino X-Application offers an even simpler way of creating Web applications. Executable JSP pages can be created completely automatically using a schema that is stored in Tamino. In most cases, not a single line of code is necessary for this (see Figure 6). The application that is generated in this way can have two purposes:

- Firstly, it can be used as a front end for Tamino, to enrich its data pool with information through a Web interface and to retrieve the information.
- Secondly, the automatically generated application can also be used as a starting point for creating a full Web application.

In the second case, some modifications and extensions are needed. Although the result of the automated process can only be of a generic nature, it is easy to make the necessary changes. These usually confine themselves to the generated JSP pages: the application logic underlies the Java API and conceals confusing technical details.

GENERATOR IN USE

The result of every automated generation by means of Tamino X-Application Generator is a set of six JSP pages. These include menus for searching, browsing, and viewing the database, and for adding, modifying, and deleting new or existing information. Furthermore, it is possible to change the GUI in

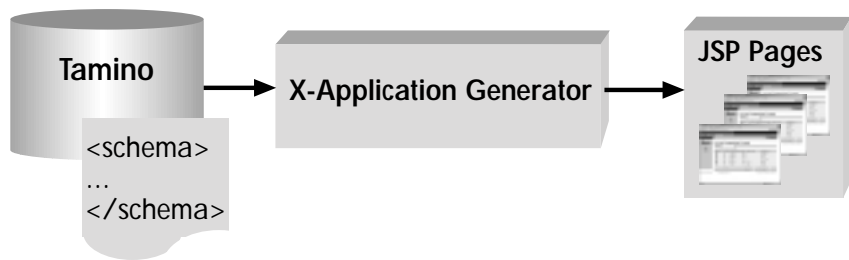


Fig. 6: Tamino X-Application Generator builds a fully functional Web application that is based solely on a Tamino schema.

which the generated application is wrapped and to modify it to meet individual requirements. The Generator requires that stylesheets be entered, which set the structure and the appearance of the pages. Instead of predefined standards, users can choose to use their own style.

Additional tuning is possible with XML files that determine, among other things, how the structure of the underlying schemas will be mapped. The resulting application can be adapted specifically to the user's unique needs when these files are modified.

Tamino X-Application and Web Services

Web services are the next big thing in information technology and serve to integrate distributed applications. Microsoft, Sun Microsystems, IBM, and Hewlett Packard have played a very important role in the evolution of Web services, and IBM and Hewlett Packard are strategic partners of Software AG. The approach that was followed has not simply risen like the phoenix from its ashes: Web services are in the tradition of well-known models such as CORBA, COM, and DCE.

Until now, the Web was reserved for browsing, but now it is opening up to all kinds of programs. The consumers of the services do not have to have any knowledge of the hardware or the operating system that the services are running on. Nor is it necessary to know anything about the object model or the programming language in which the service is implemented. Instead of using proprietary protocols including a proprietary infrastructure, as in the past, the new technology is based on an array of standard Internet technologies, such as XML and HTTP.

THE BACKBONE OF WEB SERVICES

Web services are realized through the interaction of various software components, which play certain roles within the Web services architecture and have a relationship with each other (see also Service Provider and Service Requestor in Figure 7). In essence, there are three types of involved parties that must be defined:

- Service Providers make services available that can be used by any other application. In addition, they offer descriptive information on the type of services that are offered.

- Service Requestors are applications that one or more Web services intend to make use of. Service Requestors get the help from Service Brokers to locate a suitable Service Provider.
- Service Brokers are intermediaries that make sure that an appropriate Service Provider is offered to a Service Requestor. To this end, they maintain a registry database, which includes all relevant information about the Service Providers and their available services.

WEB SERVICE OPERATIONS

The parties described here typically carry out several operations in order to guarantee interaction and communication among themselves:

- Publish is the operation that the Service Provider uses to declare the availability of a service to one or more Service Brokers. The opposite operation is called Unpublish, which asks a Service Broker to remove from the registry database a service that was previously declared available.
- The Find operation is used by the Service Requestors. They address this query to the Service Brokers to find a Service Provider that has access to their desired functionalities.
- The Bind operation represents the creation of a contract between a Service Requestor and a Service Provider. If this operation is completed successfully, the Service Requestor can call up the Service Provider and take advantage of its services.

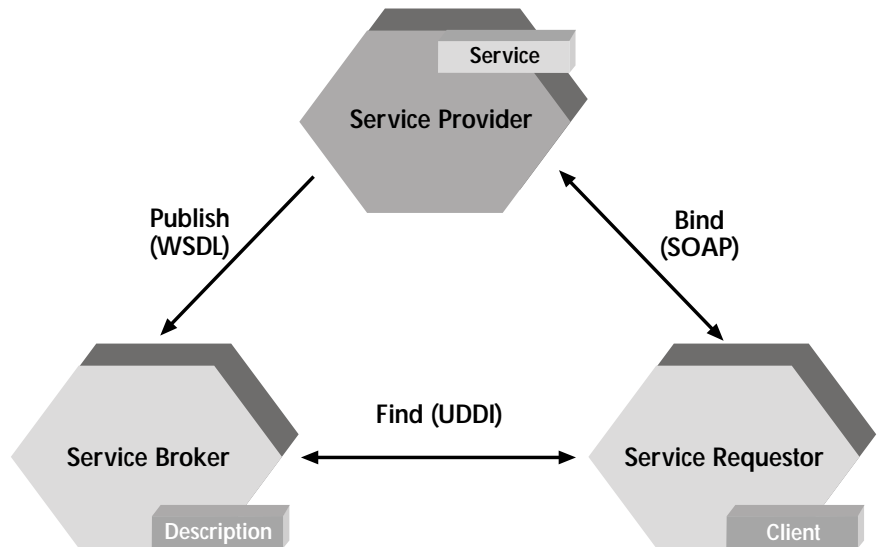


Fig. 7: The architecture of Web services is based on the interaction of various components that communicate with one another via a set of protocols.

PROTOCOLS AND COMMUNICATION

Web services run on all kinds of platforms and can be implemented in any programming language. Common standards are needed so that the individual participants can still communicate with one another:

- XML is the basis for understanding. All protocols for Web services are based on the syntax of this meta language.
- SOAP (Simple Object Access Protocol) uses XML to wrap remote method invocations and send them to the recipient. The recipient is usually also an application.
- UDDI stands for Universal Discovery, Description, and Integration. This refers to a specification for describing information and registering services. Data that is coded in UDDI can be searched and analyzed, and a result volume is then extracted from it.

- Services that have been made available are described using WSDL (Web Services Description Language). WSDL is used by both Service Providers and the Service Requestors. It is a contract language that gives the Service Requestor specific information on the methods that are being offered, and therefore makes links to the providers possible.

SUPPORT FROM TAMINO X-APPLICATION

It is assumed that the application logic will be programmed when a proprietary Web service is developed. Currently, there are no special requirements to be met when coding this logic; the creation process is similar to that of conventional applications. It is primarily changes to the outer interfaces that are needed to transform the generic application logic into a Web service. The application must simply be modified to fit the previously mentioned protocols to allow communication with the other involved parties.

Tamino X-Application is particularly suited to creating unique Web services, because the majority of the work, the programming of the application logic, is simplified through the use of the Java API. Developers need only embed the application in a Web service. And they are supported in this by Tamino X-Application as well. The framework comes with detailed documentation and tutorials that use examples to demonstrate the development and publishing of a Web service and the service clients. Among them is an example that is very similar to the Real Estate Demo application. The Web service presents an electronic marketplace for buying and selling real estate and offers its service requestors the following services:

- A potential customer can show interest in an item. The notice is stored persistently in the data pool.

- A seller can call up a list of all notices and respond to them.
- In addition, the seller can lower the price of an object that is for sale. An electronic notice is then sent to all interested parties.

The demo application is based on the Apache SOAP framework. Programs of any type that have access to an appropriate SOAP Client API could conceivably be Service Requestors (see Figure 8). In the tutorial, both a Java client and a .NET client are used; however, both of them could be replaced by any other.

When conventional applications are transformed into Web services, they can be accessed via the Web. They can be accessed by a virtually unlimited number of different types of clients, which, as Service Requestors, need only offer a suitable interface.

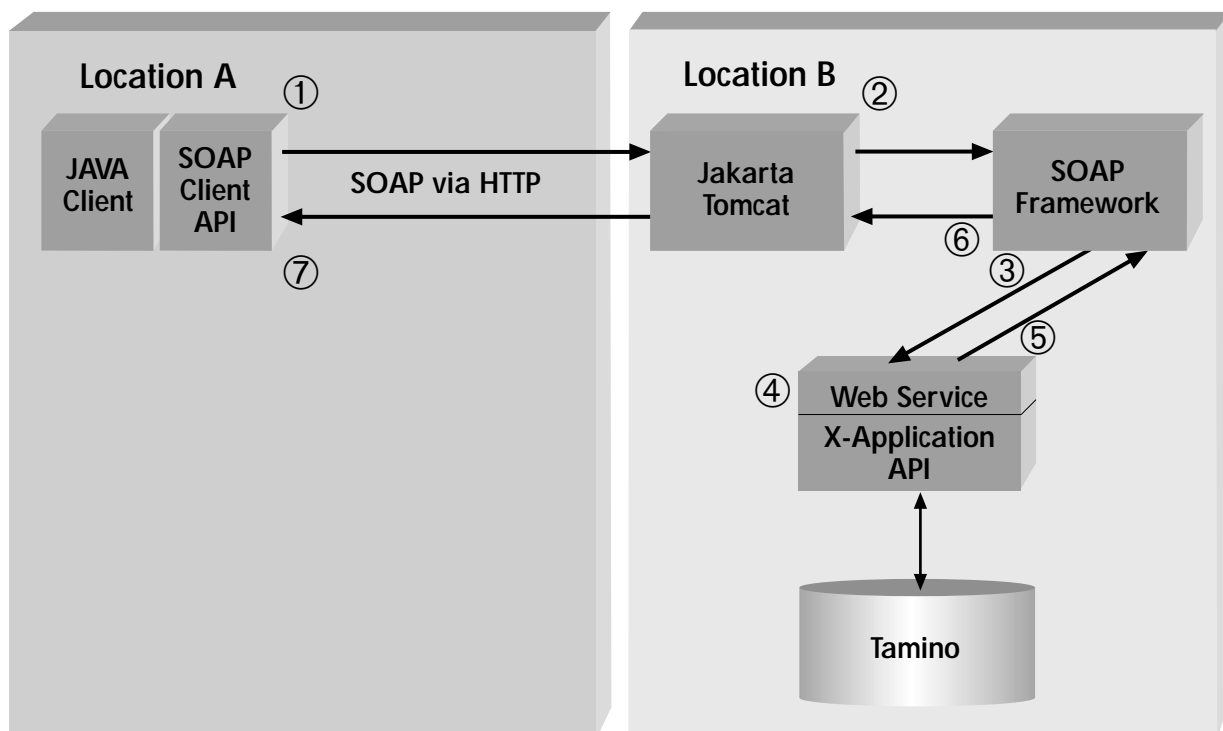


Fig. 8: The Service Provider in Location B is accessed through the Java client in Location A that is acting as a Service Requestor.

Distribution and Availability

The complete framework is freely available to Tamino developers via the Developer Community Web. Not only are all source texts, detailed documents, and numerous examples included, but there is also a test suite in JUnit and scripts for building via Ant. JUnit is a tool that is used to easily set up test cases for Java programs and thus evaluate the correctness of the programs' behavior. Ant is a complex tool for compiling and linking Java source texts.

The project is subject to a controlled open-source process: Suggestions from the Tamino Developer Community for improvements and expansions will be collected, assessed, and finally worked into the new release by the Research and Development department at Software AG.

Summary

Tamino X-Application allows Tamino developers to create their own applications within a very short time-frame. The clear organization of the framework into layers demonstrates that the reduction in effort required for development does not necessarily lead to a lower quality of the software:

- The JSP tag library forms the outer layer of Tamino X-Application and is a part of the server-side presentation layer.
- The Business Document Module implements the framework's application logic and allows various types of access to XML documents in Tamino.
- The inner layer is the Store Module, which abstracts the specifics of controlling connected data sources, usually Tamino.

With the help of Tamino X-Application Generator, generic applications can be created without any programming. The Generator produces an executable Web application

based on a Tamino schema. The implementation possibilities are many: the results can be used as a front end to manipulate data, and as a starting point for developing applications that can be adapted to meet the developer's needs.

Web services are the next step in the evolution of Web-based applications and are transforming the Internet into a network of interacting programs. Tamino X-Application supports this development and Tamino developers in embedding their applications in Web services through extensive documentation and examples that have been hammered out in detail.

For more information on Tamino XML Server:
<http://www.softwareag.com/tamino>

Tamino Community:
<http://www.softwareag.com/developer>

To download the XML Starter Kit:
<http://www.xmlstarterkit.com>

Software AG
Corporate Headquarters
Uhlandstraße 12
64297 Darmstadt/Germany
Tel: +49-61 51-92-0
Fax: +49-61 51-92-11 91
www.softwareag.com

INS/WP07E1001 249