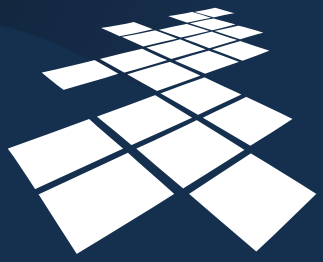
Zaawansowane aplikacje internetowe

Wykład 6

Wprowadzenie do Web Services

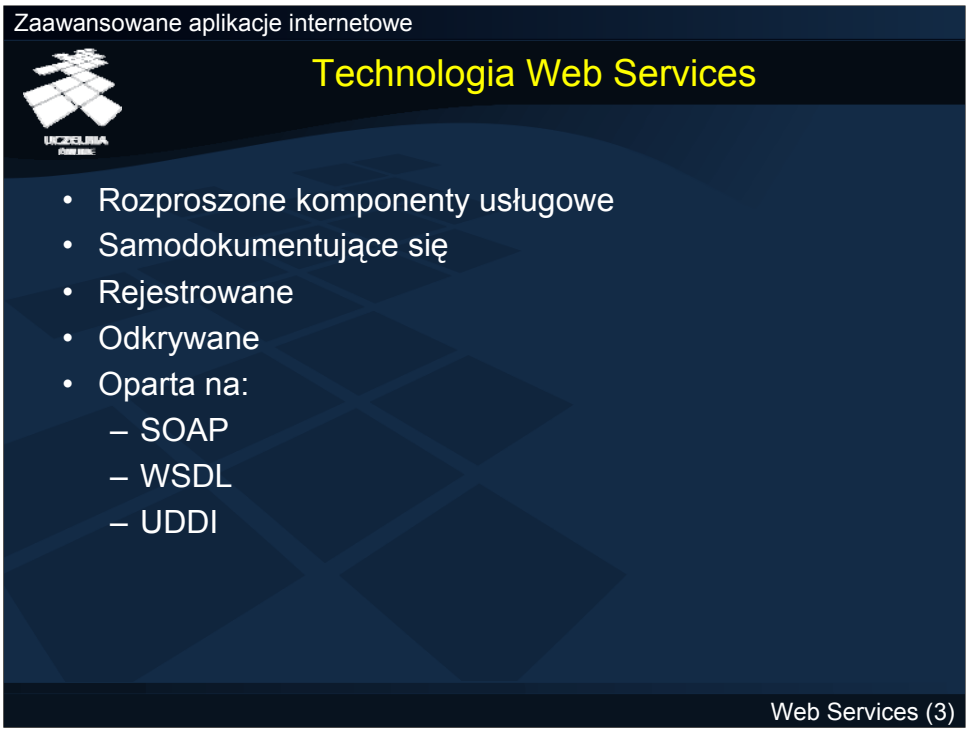
wykład prowadzi: Maciej Zakrzewicz



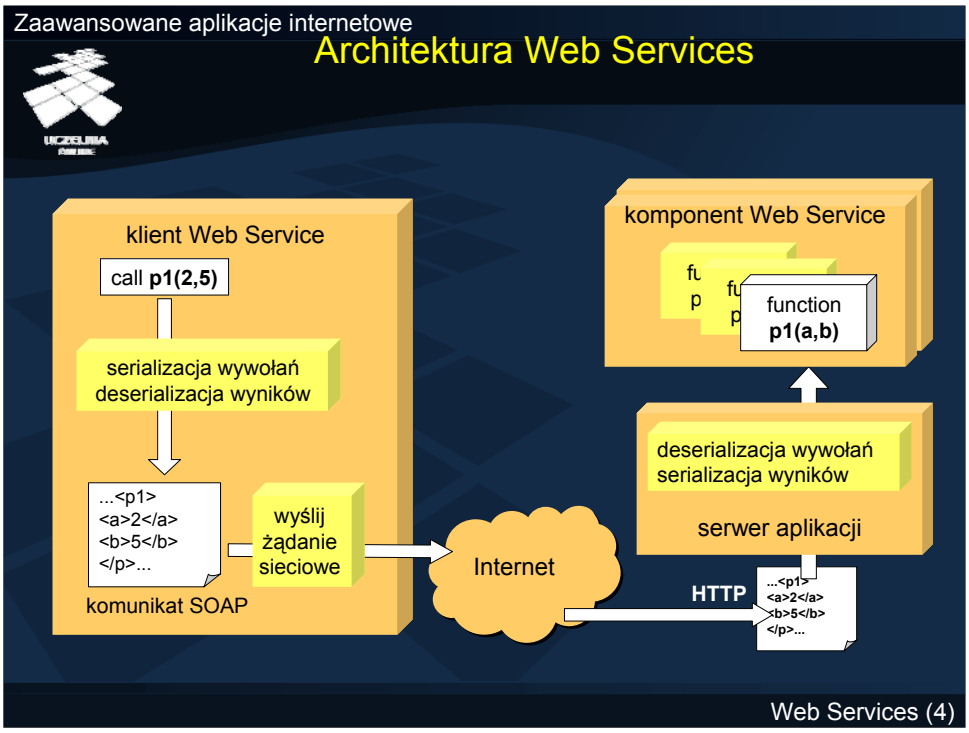
UCZELNIA ONLINE

Web Services

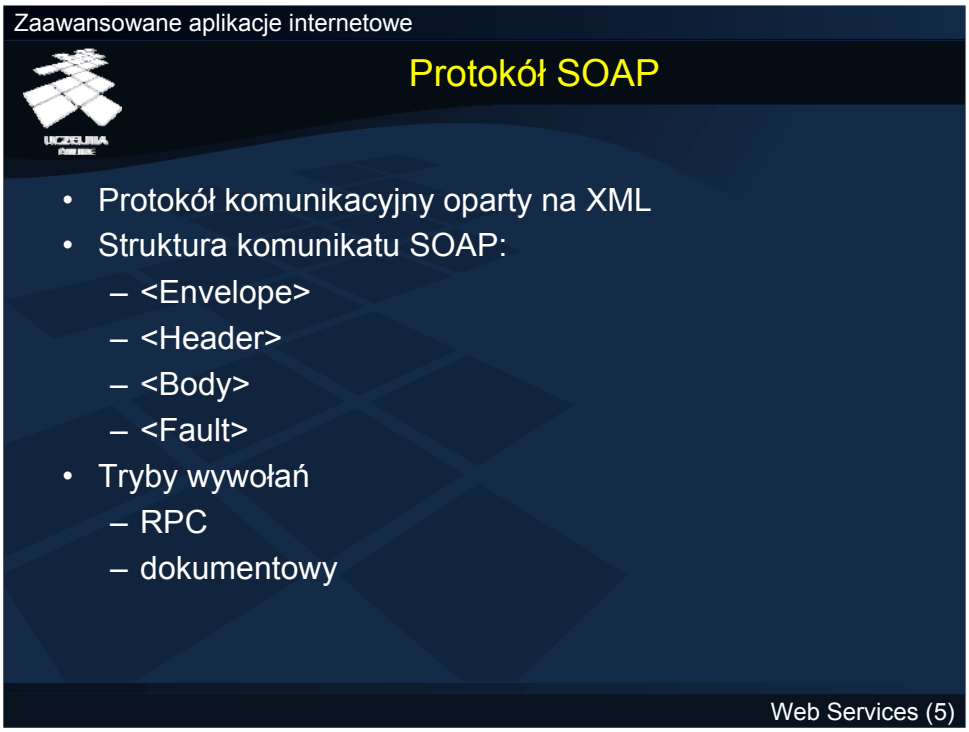
Celem wykładu jest omówienie technologii Web Services, umożliwiającej implementację rozproszonych komponentów programowych udostępnianych za pośrednictwem protokołu SOAP. Wykład obejmie przedstawienie architektury Web Services, protokołu komunikacyjnego SOAP, języka opisu interfejsu WSDL i rejestrów UDDI.



Web Services to technologia konstrukcji rozproszonych komponentów usługowych, stanowiących podstawę dla realizacji aplikacji biznesowych w architekturze zorientowanej na usługi. Zgodnie z powszechnie akceptowaną definicją, usługa Web Service to zwarty, samodokumentujący się komponent programowy, który może być przez swojego twórcę zarejestrowany w sieci komputerowej, a następnie przez twórcę aplikacji-konsumenta odkryty i wywołany w trybie zdalnego wykonania. Technologia Web Services opiera się na szeregu skorelowanych rozwiązań informatycznych, spośród których najważniejsze to: protokół komunikacyjny SOAP – służący do przekazywania zdalnych wywołań, język opisu interfejsu usługi WSDL – służący do dystrybucji parametrów połączeń sieciowych, specyfikacja bazy danych UDDI – służącej do rejestracji udostępnianych komponentów usługowych.



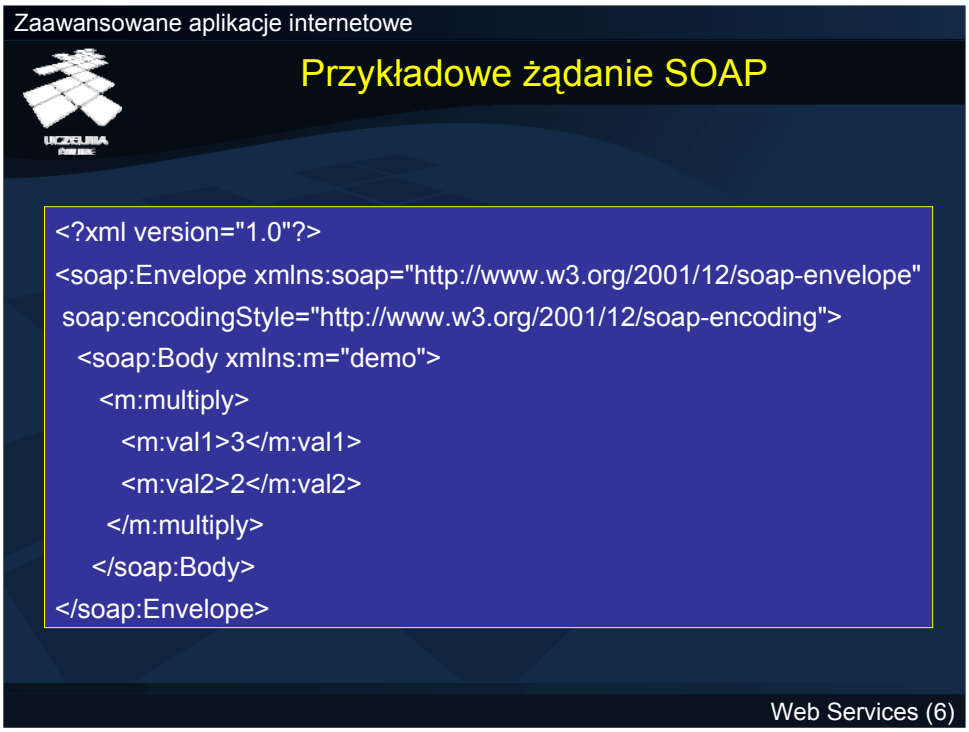
Podstawowa architektura Web Services została przedstawiona na slajdzie. Rysunek przedstawia: (1) aplikację zainteresowaną wywołaniem kodu zdalnego modułu programowego, tzw. klienta Web Service, (2) zdalny moduł programowy oferujący implementację funkcji logiki programowej, tzw. komponent Web Service, (3) serwer aplikacji odpowiedzialny za odbieranie i obsługę sieciowych żądań wywołania kodu zdalnego modułu programowego. Klient Web Service zapisuje treść wywołania komponentu w postaci komunikatu SOAP, stanowiącego rodzaj serializacji wywołania w formacie XML. Komunikat SOAP jest przekazywany za pośrednictwem wybranego protokołu sieciowego (najczęściej HTTP) do zdalnego serwera aplikacji. Serwer aplikacji dokonuje deserializacji komunikatu SOAP w celu ekstrakcji pierwotnego zapisu wywołania komponentu Web Service, a następnie lokalnie przekazuje sterowanie do właściwej jednostki programowej komponentu. Ewentualne wyniki pracy komponentu mogą być w analogiczny sposób przekazane zwrotnie do klienta – następuje wówczas serializacja wyników do komunikatu SOAP, przesłanie sieciowe, a w ostatnim kroku deserializacja wyników po stronie klienta. Komponenty usługowe Web Service mogą pracować w trybie bezstanowym (ulotne wartości zmiennych pomiędzy wywołaniami) lub stanowym (nieulotne wartości zmiennych pomiędzy wywołaniami). Mogą również korzystać z dostępnych mechanizmów sesji, np. HTTPSession, oraz uwierzytelniania, np. HTTP Basic Authentication.



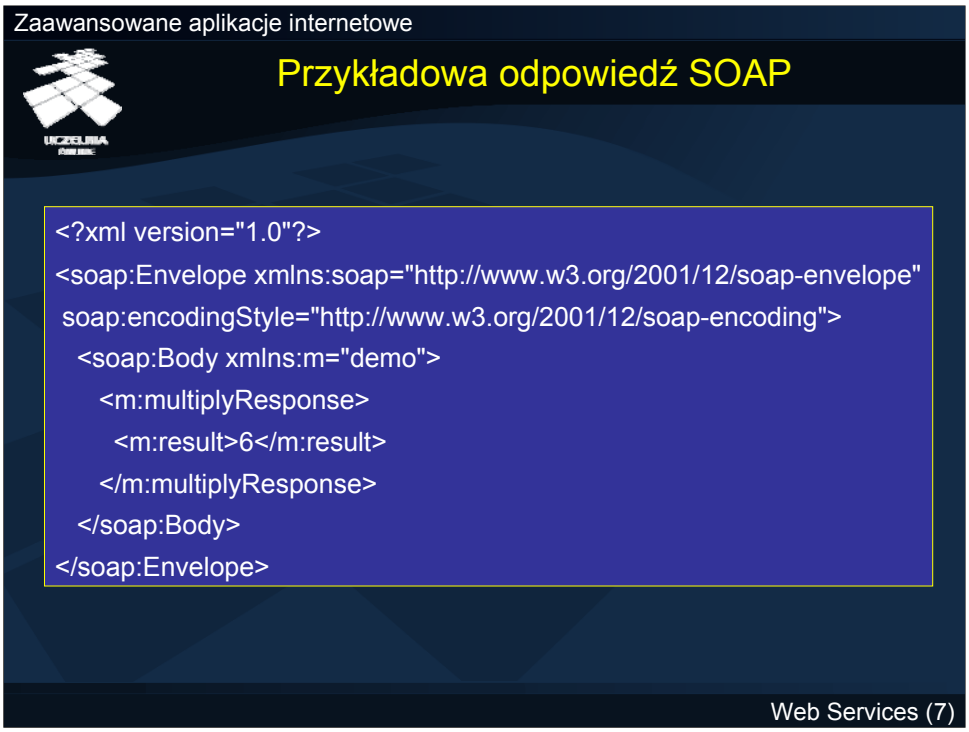
SOAP (Simple Object Access Protocol) to prosty protokół komunikacyjny oparty na języku XML, umożliwiający przekazywanie wywołań zdalnych komponentów Web Services. SOAP może współdziałać z dowolnym niskopoziomowym sieciowym mechanizmem transportowym, np. HTTP, HTTPS, SMTP, JMS, RMI. Podstawowymi znacznikami wykorzystywanymi do budowy komunikatów SOAP są: <Envelope> – otacza cały komunikat, <Header> – zawiera informacje nagłówkowe, <Body> – zawiera informacje o żądaniu i odpowiedzi, <Fault> – opisuje błędy, jakie wystąpiły podczas przetwarzania wywołania.

Protokół SOAP umożliwia wywoływanie komponentów Web Service w dwóch trybach: (1) Remote Procedure Call (RPC) i (2) dokumentowym (document-oriented). Tryby te różnią się formą przekazywania parametrów. W trybie RPC wywołanie ma charakter tradycyjny – komponentowi przekazywana jest lista parametrów formalnych wraz z ich bieżącymi wartościami. W trybie dokumentowym usługa otrzymuje tylko jeden parametr wywołania, którym jest dokument XML.

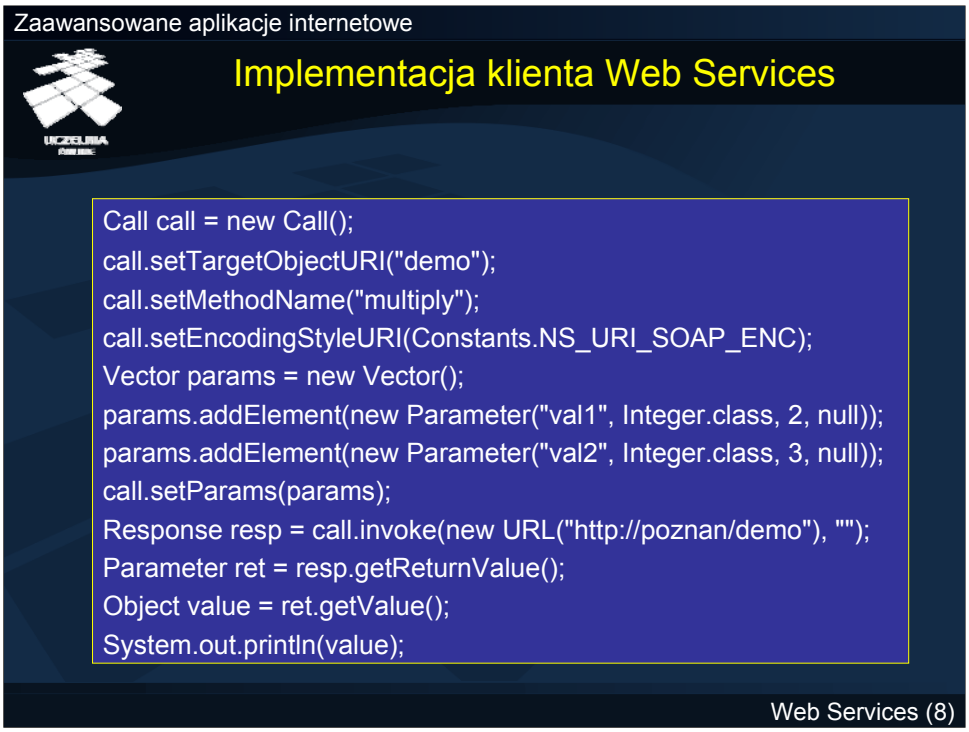
Wywołania komponentów usługowych Web Services mogą mieć charakter synchroniczny lub asynchroniczny. W trybie wywołania synchronicznego aplikacja klienta wysyła żądanie uruchomienia zdalnej funkcji biznesowej i wstrzymuje pracę aż do chwili otrzymania wyników jej realizacji. Tryb ten może opierać się na komunikacji HTTP, HTTPS, RMI/IIOP, SMTP, itp. W trybie wywołania asynchronicznego aplikacja klienta wysyła żądanie uruchomienia zdalnej funkcji biznesowej lecz nie oczekuje na jej wynik, kontynuując działanie. Tryb ten może opierać się na komunikacji HTTPR, JMS, IBM MQSeries Messaging, MS Messaging, itp. Zwykle tryb synchroniczny jest wykorzystywany przez komponenty RPC, natomiast tryb asynchroniczny – przez komponenty dokumentowe.



Przykładowy komunikat żądania SOAP został przedstawiony na slajdzie. Dotyczy on wywołania komponentu o nazwie „demo”, zawierającego funkcję „multiply(int val1, int val2)”. Komunikat zawiera sparametryzowane wywołanie funkcji „multiply(3,2)”.



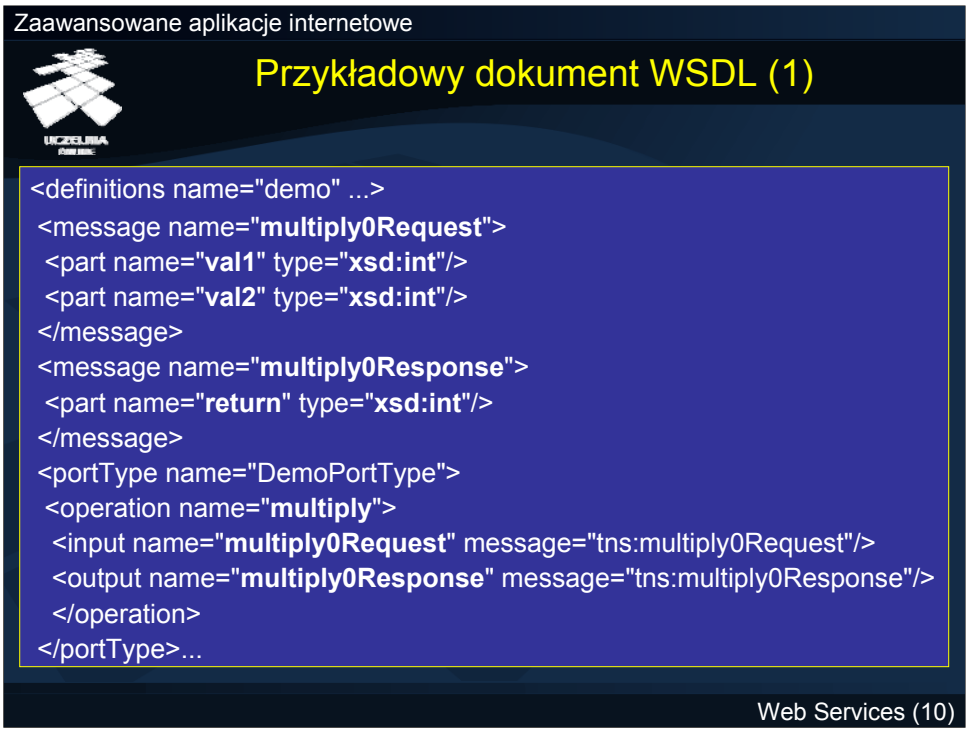
Przykładowy komunikat odpowiedzi SOAP został przedstawiony na slajdzie. Dotyczy on wywołania komponentu o nazwie „demo”, zawierającego funkcję „multiply(int val1, int val2)”. Przykład obrazuje komunikat wynikowy, przekazujący klientowi rezultat wywołania funkcji „multiply” – liczbę 6.



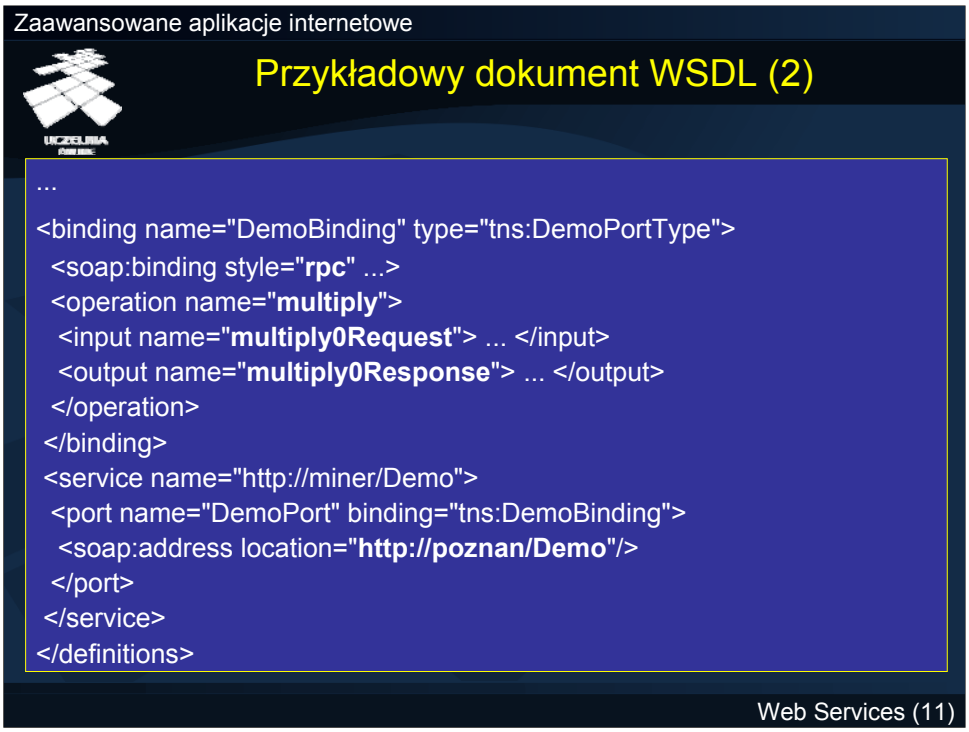
Mimo, iż implementacja środowiska aplikacyjnego w technologii Web Services jest możliwa w dowolnym z języków programowania, to jednak największą uwagę projektantów przyciąga obecnie język Java. W celu ułatwienia implementacji obsługi protokołu SOAP, programiści Java mogą korzystać z biblioteki Java API for XML-based RPC (JAX-RPC), która wyręcza ich z konieczności konstrukcji, przesyłania i parsowania XML-owych komunikatów SOAP. Warto nadmienić, że analogiczne biblioteki są dostępne dla innych popularnych języków programowania, np. SOAP::Lite dla Perla, gSOAP dla C/C++, ZSI dla Pythona, .NET. Na slajdzie przedstawiono przykład kodu aplikacji-klienta, dokonującej zdalnego wywołania funkcji „multiply(2,3)” komponentu Web Service o nazwie „demo”. Sam komponent Web Service mógłby również zostać zaimplementowany w języku Java. Byłby klasą zawierającą jedną metodę "multiply", pobierającą dwie liczby całkowite i zwracającą ich iloczyn.

Pomimo dostępności bibliotek programistycznych wspomagających implementację komunikacji w oparciu o protokół SOAP, konstrukcja i pielęgnacja aplikacji-klienta wymaga od programisty istotnego wysiłku. W celu uproszczenia tych zadań postanowiono wykorzystać koncepcję automatycznego generowania kodu komunikacyjnego w oparciu o zestaw parametrów sieciowych opisujących komponent usługowy. Przyjęto, że twórca komponentu Web Service opisuje jego interfejs w języku WSDL (Web Service Description Language), a twórca klienta Web Service dokonuje zautomatyzowanej transformacji tego opisu do kodu źródłowego obsługującego komunikację sieciową z komponentem (tzw. Web Service Proxy) w wybranym języku programowania. Transformacja ta może mieć charakter: (1) statyczny, gdy realizowana jest na etapie budowy/kompilacji aplikacji klienta, (2) dynamiczny, gdy realizowana jest na etapie wykonania aplikacji klienta.

WSDL to język znaczników XML służący do opisu technicznych parametrów połączenia sieciowego aplikacji-klienta z komponentem Web Service. Strukturę znaczników dokumentu WSDL przedstawiono na slajdzie. Role wymienionych znaczników są następujące. Znacznik <definitions> otacza całą zawartość dokumentu. Znaczniki <service> wraz ze znacznikami <port> definiują adresy punktów dostępowych dla usługi. Znaczniki <portType> służą do deklaracji funkcji biznesowych oferowanych przez usługę. Znaczniki <binding> określają metody kodowania parametrów wywołania i parametrów zwrotnych usługi.



Na slajdzie zamieszczono pierwszą część przykładowego dokumentu WSDL. Jest to dokument opisujący usługę o nazwie „demo” zawierającą jedną metodę „multiply(int val1, int val2)”.



Na slajdzie zamieszczono drugą część przykładowego dokumentu WSDL. Jest to dokument opisujący usługę o nazwie „demo” zawierającą jedną metodę „multiply(int val1, int val2)”.

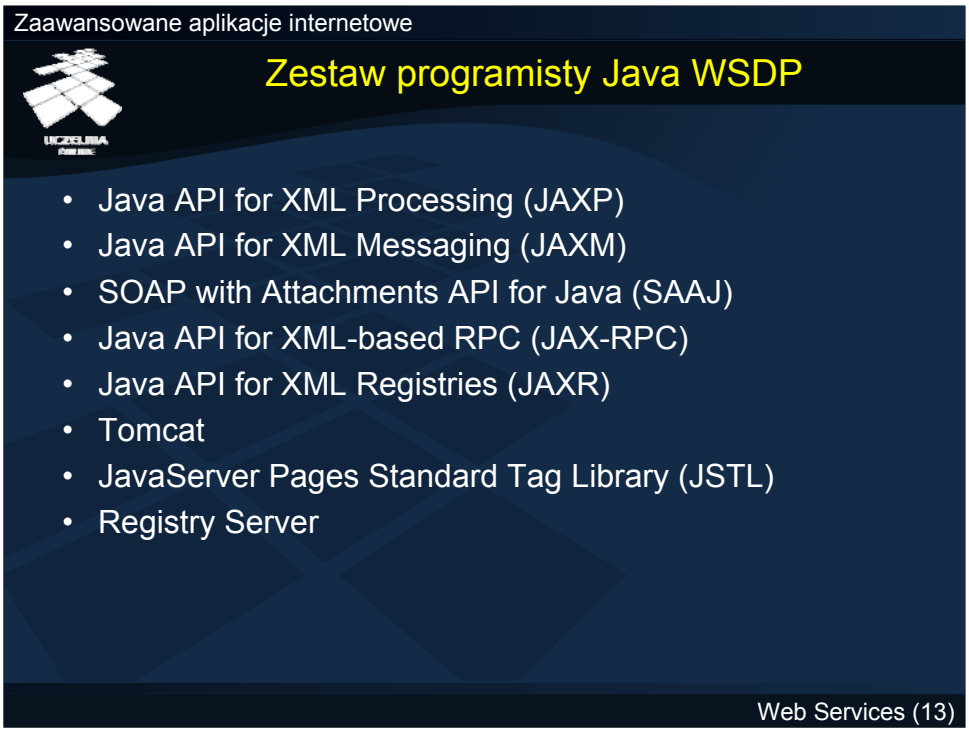
W celu uproszczenia dystrybucji dokumentów WSDL i semantyki komponentów usługowych Web Services zaproponowano, aby dostępne komponenty były rejestrowane w specjalizowanej bazie danych, która może być następnie przeszukiwana przez twórców aplikacji-klientów lub przez same aplikacje-klientów z zamiarem odkrycia potrzebnych komponentów. Opracowaną specyfikację takiej bazy danych nazwano UDDI (Universal Description, Discovery, and Integration). Bazy danych UDDI deklarują dostępne usługi posługując się trzema poziomami opisu:

•White Pages: podstawowe dane kontaktowe o dostawcy usługi, obejmujące nazwę firmy, adres, identyfikator (np. numer identyfikacji podatkowej),

•Yellow Pages: lokalizacja usługi w taksonomiach kategoryzacyjnych,

•Green Pages: techniczny opis usługi i jej semantyki, często też wskaźnik do pliku WSDL.

Dostęp do rejestru UDDI jest zwykle możliwy zarówno poprzez interfejs HTML, jak i poprzez interfejs programistyczny, np. biblioteki JAXR-API. Na slajdzie przedstawiono cykl życia usługi Web Service z punktu widzenia jej deklaracji w rejestrze UDDI: (1) twórca komponentu implementuje komponent usługowy Web Service i rejestruje go w UDDI, (2) twórca klienta wyszukuje opis komponentu usługowego Web Service w UDDI i implementuje aplikację-klienta, (3) uruchomiona aplikacja-klient dokonuje zdalnego wywołania komponentu usługowego Web Service.



Dla programistów Java zainteresowanych przede wszystkim konstrukcją aplikacji Web Services opracowano specjalny pakiet narzędziowy o nazwie Java WSDP. Pakiet ten zawiera zarówno podstawowe biblioteki programisty, jak i narzędzia uruchomieniowe. Poniżej wymieniono najważniejsze składniki pakietu:

•Java API for XML Processing (JAXP): podstawowe biblioteki przetwarzania danych XML, zgodne z rekomendacją W3C,

•Java API for XML Messaging (JAXM): biblioteka sieciowej komunikacji opartej na języku XML,

•SOAP with Attachments API for Java (SAAJ): biblioteka sieciowej komunikacji w oparciu o protokół SOAP,

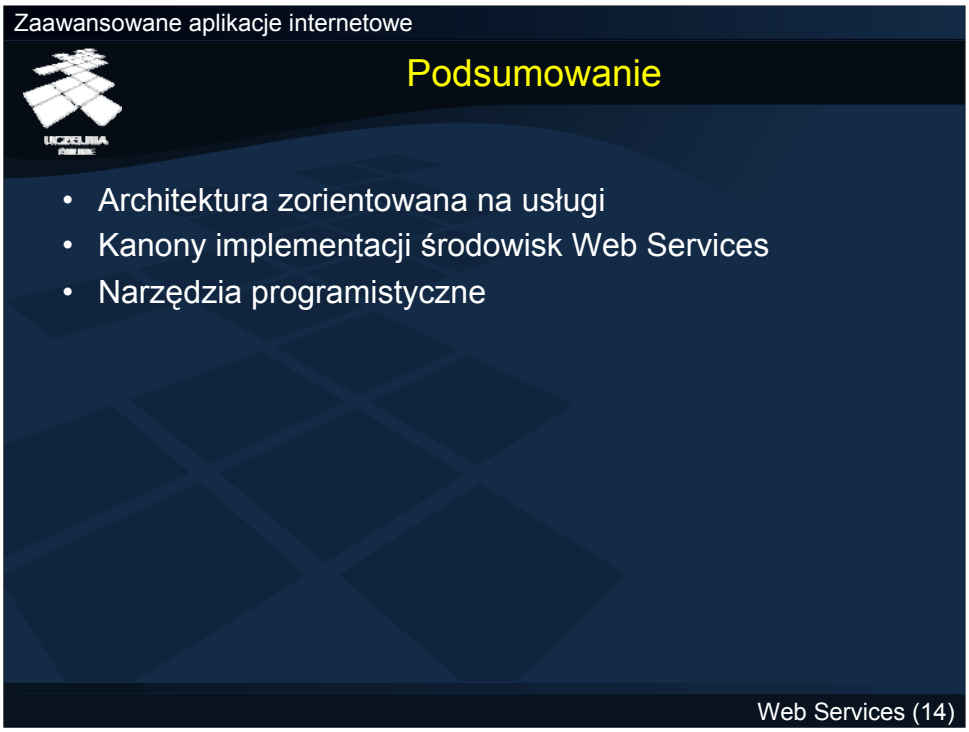
•Java API for XML-based RPC (JAX-RPC): biblioteka służąca do realizacji zdalnych wywołań komponentów usługowych w trybie RPC,

•Java API for XML Registries (JAXR): biblioteka służąca do dostępu do XML-owych rejestrów usług Web Servicesm np. UDDI,

•Tomcat – zredukowany serwer J2EE umożliwiający instalowanie i uruchamianie komponentów usługowych Web Services,

•JavaServer Pages Standard Tag Library (JSTL): standardowa biblioteka znaczników JSP,

•Registry Server: rejestr UDDI.



Technologia Web Services umożliwia efektywną konstrukcję aplikacji biznesowych zgodnych z architekturą zorientowaną na usługi, według której logika biznesowa jest rozproszona pomiędzy wiele rozproszonych komponentów, często zarządzanych przez niezależne od siebie instytucje. Podstawowe kanony implementacji środowisk Web Services obejmują: implementację komponentu usługowego w dowolnym języku programowania, sporządzenie pliku opisu interfejsu komponentu (WSDL), zgłoszenie komponentu do centralnego rejestru (UDDI), wyszukanie komponentu w centralnym rejestrze (UDDI), wygenerowanie kodu komunikacyjnego dla klienta (Web Service Proxy), implementację klienta Web Service. Dzięki rozległemu wsparciu ze strony narzędzi programistycznych, tworzenie środowisk Web Services wymaga od programisty skupienia się prawie wyłącznie na implementacji kodu logiki biznesowej.

•"Web Services Activity", http://www.w3.org/2002/ws

•"Web Services and Other Distributed Technologies", http://msdn.microsoft.com/webservices/

•"SOA and Web Services", http://www-128.ibm.com/developerworks/webservices

•"XMethods", http://www.xmethods.net/

•"Java Technology and Web Services", http://java.sun.com/webservices/

•"Web Services Project @ Apache", http://ws.apache.org/