ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Εισαγωγή στην Επιστήμη και Τεχνολογία των Υπηρεσιών

Ενότητα 13: Web Services

Χρήστος Νικολάου
Τμήμα Επιστήμης Υπολογιστών

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται στην άδεια χρήσης **Creative Commons** και ειδικότερα

Αναφορά – Μη εμπορική Χρήση – Όχι Παράγωγο Έργο v. 3.0

(Attribution – Non Commercial – Non-derivatives)

- Εξαιρείται από την ως άνω άδεια υλικό που περιλαμβάνεται στις διαφάνειες του μαθήματος, και υπόκειται σε άλλου τύπου άδεια χρήσης. Η άδεια χρήσης στην οποία υπόκειται το υλικό αυτό αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

XML
Web Services
635.444

David Silberberg

Lecture 24

What are Web Services

- W3C: a **Web Service system** is a software system designed to support interoperable Machine-to-Machine interaction over a network.
- Web services are accessed via Web APIs over a network, such as the Internet, and executed on a remote system hosting the requested services.
- Web service clients and servers
  - Communicate using XML messages that follow the SOAP standard (will describe later)
  - Common in both the field and the terminology is the assumption that there is also a machine readable description of the operations supported by the server in the Web Services Description Language (WSDL)

More about Web Services

- Web Services are services offered via the Web
- A Web Service is programmable application logic accessible using standard Internet protocols
  - Not generally monolithic or course-grained services like zdnet.com or amazon.com (although they could be)
  - Each service represents black-box functionality that can be reused independent of its implementation
- Web services are usually component-based services
  - Building blocks for larger services
  - Can be aggregated in multiple ways to form an enterprise
  - The same services can be used to form other enterprises, as well
  - Individual services are usually ―light-weight‖ modules to provide a single (or closely related set of) services

Service-Oriented Architectures

- Interest in Web Services has renewed an interest in service-oriented architecture (SOA).
- An SOA is an architecture that is composed of components and interconnections that stress interoperability and location transparency
- The term service has been used for many years
  - Transaction monitoring software uses term "service"
  - Client-server development efforts use the term "service" to indicate the ability to make a remote method call
  - Web Services has given the term service more prominence
- Services and service-oriented architectures are about designing and building systems using heterogeneous network addressable software components

General Characteristics of Web Services

- Applications
  - Self-contained - complete modules usable (and reusable) by one or more applications
  - Self-describing - can discern its use from its description
  - Modular - bite-sized chunks of code
- Use of services
  - Services are published for anticipated and unanticipated clients
  - Clients can discover and locate through directory services
  - Clients can invoke the services without notifying the services a priori
- Range of use from simple server to complicated business processes

Examples

- ―Stock Quote‖ (simple) service
  - Client asks for the stock price of some specific stock
  - Web service (server) returns the current price of the specific stock
  - Request is filled almost immediately
  - Request and response are part of the same method call
- ―Efficient Route Map‖ (complex) service
  - Client sends request specifying delivery destinations
  - Web service returns an efficient route
  - Since the process is complex, the response time can be large
  - Request and response are part of two operations
- Multiple client/server calls
  - Customer asks for availability of item from retail service
  - Retail service ―turns around‖ and asks for availability of item from warehouse service
  - Warehouse service responds to retail service, which responds to the customer

Why Web Services - Why Not Use Other Platforms?

- Other middleware platforms exist to support these types of operations
  - RMI - for Java to Java routines
  - Jini - a Java-based remote discovery and invocation service
  - CORBA - the OMG standard for supporting remote services in multiple languages
  - DCOM - Microsoft's services platform
- All the platforms above are widely used
  - However, none have the ubiquity of the entire Internet
  - All have specific protocols that must be adhered to by participants
- Web Services
  - Uses XML, HTTP, SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), UDDI (Universal Description, Discovery and Integration service)
  - Significant advantage of Web Services over other platforms
    - HTTP - probably THE most widely accepted Internet protocol
    - XML - ubiquitous data exchange markup language

Remote Procedure Calls (RPC)

- Distributed components that need to communicate to complete application
  - Code on one computer needs to call code on other computers
  - Those pieces of code might need to call each other or yet other pieces of code on other computers
- Need to know a number of things for this type of platform to be successful
  - Where does the code you want to call reside?
  - Does the code need any parameters?
  - Does the code return any data?
- In addition, the following issues must be dealt with
  - Networking issues
  - Packaging data for transport
  - Data conversion (Little Endian vs. Big Endian)

Protocols

- Protocols are created to handle these issues
  - Enable different applications to talk to each other
  - Enable communication across heterogeneous platforms
- Provide ―handshaking‖ rules to enable communications
  - Describe how to address remote computers
  - Describe how to address remote applications
  - Describe how to package parameters to be sent to remote procedures
  - Describe how to package result data
  - Describe how to initiate a call
  - Describe how to handle errors
  - etc.

DCOM

- Based on COM (Components Object Model)
- Microsoft's standard for writing sharable objects
  - Objects can be discovered at run time
  - Can be shared by any application running on the computer
  - Language independent (objects can be written in Java, C, C++, etc.)
- Example of use
  - Microsoft Office Suite
  - One application can make use of another
  - Word can incorporate Excel spreadsheets
  - Spell checker can be reused
- Distributed COM (DCOM) extends this capability across computers
  - Microsoft applications can speak to each other across platforms
  - However, DCOM is generally Microsoft specific

IIOP for CORBA

- Common Object Request Broker Architecture (CORBA)
  - Solves the same problem as DCOM
  - Predated COM and DCOM
  - Platform independence
  - Language independence
  - Vendor independence
- Difference between DCOM and CORBA
  - COM/DCOM - operating system (Windows) provides communication between application and COM object
  - CORBA - Object Request Broker (ORB) provides communication between application and CORBA object
  - CORBA is operating system independent
- Internet Inter-ORB Protocol (IIOP)
  - Enables ORBs from different vendors to communicate
  - Provides ORB independence!

Java RMI

- DCOM and IIOP provide a language-independent way to enable remote objects to communicate
- IIOP provides a protocol for objects to be located on any platform
- However, Java already provides platform independence
  - But not language independence, obviously
- Java Remote Method Invocation (RMI) enables distributed Java objects to communicate
  - You can write Java objects for DCOM (with non-standard extensions) and CORBA
  - However, RMI provides a small learning curve to implement RPC
- Drawbacks
  - RMI is only for Java - no language independence
  - RMI can get difficult for complex implementations (e.g., JDBC drivers, etc.)

SOAP

- Simple Object Access Protocol (SOAP) is XML documents sent via HTTP
- SOAP specifies
  - Rules on how RPC is sent
    - Really, any network protocol can be used (e.g., IBM MQSeries, Microsoft Message Queue MSMQ, etc.)
    - However, HTTP is more common and thus the rules state how requests need to be specified
  - Structure of XML that is sent - called the envelope
  - Rule of how data is represented - called the encoding rules
- Benefit
  - XML is a ubiquitous data exchange model
  - HTTP is a ubiquitous protocol

HTTP

- Hypertext Transfer Protocol (HTTP) is a request/response protocol
- Upon request
  - Connection to HTTP server is made
  - Request is sent to server
  - Processing done by server
  - Response from server is sent back
  - Connection is closed
- HTTP message contains
  - Header - describes message and enables a server to interpret the message
  - Body - optional data to be processed by server

Example HTTP GET Method Header

```
GET /inventory.html HTTP/1.1
Accept: */*
Accept-Encoding: gzip
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: www.store.com
```

- GET - common type of request (POST is other common type of request)
- `Accept` - which MIME (Multipurpose Internet Mail Extensions) type(s) the browser accepts
- `Accept-Encoding` - specifies to server if the content can be encoded before sending to browser and which encoding(s) are acceptable
- `User-Agent` - specifies the type of browser used. In this case, it is Mozilla/4.0. In addition, it is compatible with JavaScript 1.0, IE 5.5, and was generated on a Windows NT 5.0 platform).
- `Host` - the target server machine

Example HTTP GET Method Response

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Sun, 18 Aug 2002 14:15:27 GMT
Content-Type: text/html
Last-Modified: Sun, 18 Aug 2002 12:17:20 GMT
Content-Length: 396

<html>
<head><title>inventory</title></head>
<body>
<H1>Inventory
<p>
<OL>
<LI>Ivory Soap
...
```

Web Page for POST Method

- Create an HTML page to be browsed

```
<html>
<head><title>Number of Soap Bars</title></head>
<body>
<form
   action="http://www.apl.jhu.edu/cgi-instruct/davids/SoapCGI"
   method="POST">
   Enter the brand of soap you wish to check:
   <input type="text" name="soapName"><br>
   <input type="text" name="size"><br>
   <input type="submit" value="Get Inventory">
</form>
</body>
</html>
```

HTTP Post Method Header and Body

```
POST /cgi-instruct/davids/SoapCGI HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: www.apl.jhu.com
Content-Length: 24

soapName=Ivory&size=16oz
```

- Contains a request body
- Useful for RPC

HTTP Post Method Response

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Sun, 18 Aug 2002 15:05:53 GMT
Content-Type: text/html
Content-Length: 112

<html>
<head><title>Inventory for Ivory - 16oz</title></head>
<body>
<H1>Inventory
<p>
3440 cases
</body>
</html>
```

HTTP for SOAP

- Similar to regular POST request
- Few steps must be done
  - Use POST method
  - Body of the message must be a SOAP request
  - Add extra HTTP header called **SOAPAction**, which specifies the remote procedure to be called
- Server must return a valid HTTP status code
  - 500 if the request is unsuccessful
  - 200 if the request is successful

Example SOAP Call

```
POST /inventory-cgi HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: www.apl.jhu.com
SOAPAction: “http://www.apl.jhu.edu/soap/inventorySize”
Content-Length: 241

<Envelope xmlns=“http://schemas.xmlsoap.org/soap/envelope/”>
<Body>
   <inv:inventorySize
                  xmlns:inv=“http://www.apl.jhu.edu/soap/”>
        <inv:soapName>Ivory</inv:soapName>
        <inv:size>16oz</inv:size>
   </inv:inventorySize>
</Body>
</Envelope>
```

SOAPAction Header

- Indicates the remote procedure being called
- Uses URI notation
- Sometimes the string is left empty

```
SOAPAction: “”
```

- Or it can be completely blank

```
SOAPAction:
```

- In this case, the SOAP server looks in the envelope to find the remote procedure
- SOAPAction header is used to filter the SOAP messages that pass through a firewall

Example SOAP Response

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Sun, 18 Aug 2002 16:22:52 GMT
Content-Type: text/html
Content-Length: 215

<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Body>
   <inv:inventoryResponse
                xmlns:inv="http://www.apl.jhu.edu/soap/">
        <inv:cases>3440</inv:cases>
   </inv:inventoryResponse>
</Body>
</Envelope>
```

SOAP Envelope

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <SOAP-ENV:Header>
        <head-ns:headerElt xmlns:head-ns="nsURI"
                                soap:mustUnderstand="1"
                                soap:actor="actorURI"/>
     </SOAP-ENV:Header>
     <SOAP-ENV:Body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <some-ns:someElt  xmlns:some-ns="someURI"/>
        <!-- OR -->
        <SOAP-ENV:Fault>
          <faultcode/>
          <faultstring/>
          <falutactor/>
          <detail/>
        </SOAP-ENV:Fault>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Envelope Elements

- <Envelope>
  - Root element of SOAP envelope
  - Must use namespace: http://schemas.xmlsoap.org/soap/envelope/
- <Body>
  - Contains the main SOAP message
  - RPC calls are made using the children of Body
  - Names of procedures are the names of the child elements
  - Direct children must reside in a different namespace than the SOAP namespace
  - Namespace uniquely identifies procedure

Envelope Elements (2)

- <Header>
  - Provides additional information beyond the procedure calls of the body
  - Can be used for external operations
    - Authentication (user/password)
    - Logging
    - Etc.
  - Not required
  - If used, it must be the first child of <Envelope>
  - It must exist in a namespace other than the SOAP envelope namespace

Header Attributes

- mustUnderstand
  - Specifies whether it is necessary for the SOAP server to process the header message
    - ―1‖ is yes
    - ―0‖ is optional
  - Absence of attribute is the same as optional
- actor
  - Sometimes SOAP messages are passed among multiple machines
  - actor attribute specifies the machine that must process the header
  - If a machine processes the header, it cannot pass that header on to the next machine

Fault Element

- For processing errors
  - Service unavailable
  - Version mismatch
  - Invalid parameters
  - Etc.
- <faultcode>
  - Type of error
  - VersionMismatch (server does not understand this version of protocol)
  - MustUnderstand (mustUnderstand is mandatory, but server still does not understand)
  - Client (message not properly formatted)
  - Server (server could not process even though format was fine)
- <faultstring> - human-readable error message
- <faultactor> - URI of SOAP intermediary that caused the error
- <detail> - addition application-specific error message

Web Services

- SOAP enables applications to make RPC calls
- However, to use the procedures, other information is necessary
  - Location of procedure (URI)
  - How to pass information to procedure
  - Results format
- Web Services Description Language (WSDL)
  - XML document
  - Describes Web Services
  - Used to locate Web Services
- Universal Discovery, Description, and Integration (UDDI) protocol
  - Enables registration of Web Services
  - Enables discovery of Web Services

WSDL Document Structure

- WSDL document contains these major elements
  - <portType>
    - Defines a web service, the operations that can be performed, and the messages that are involved.
    - Can be compared to a function library (or a module, or a class) in a traditional programming language.
  - <message>
    - Data elements of an operation
    - Each messages can consist of one or more parts
    - The parts can be compared to the parameters of a function call in a traditional programming language

WSDL Structure (2)

- <types>
  - Data types used by the web service
  - For maximum platform neutrality, WSDL uses XML Schema syntax to define data types.
- <binding>
  - Defines the message format and protocol details for each port

Main WSDL Structure

```
<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>
```

- Can also contain other elements
  - Extension elements
  - Service element that makes it possible to group together the definitions of several web services in one single WSDL document

WSDL Example

- This is a simplified fraction of a WSDL document:

```
<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>
```

WSDL Example (2)

- <portType> element defines
  - "glossaryTerms" as the name of a port
  - "getTerm" as the name of an operation.
- The "getTerm" operation has
  - input message called "getTermRequest"
  - output message called "getTermResponse".
- <message> elements defines the parts of each message and the associated data types.
- Compared to traditional programming
  - glossaryTerms is a function library
  - "getTerm" is a function with "getTermRequest" as the input parameter and getTermResponse as the return parameter.

WSDL Ports

- <portType> element is the most important WSDL element.
  - Defines
    - web service
    - operations that can be performed
    - messages that are involved
- The request-response type is the most common operation type, but WSDL defines four types:
  - One-way -- operation can receive a message but will not return a response
  - Request-response -- operation can receive a request and will return a response
  - Solicit-response -- operation can send a request and will wait for a response
  - Notification -- operation can send a message but will not wait for a response

WSDL Ports - One-Way Example

```
<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 >
```

- "glossaryTerms" defines a one-way operation called "setTerm".
- "setTerm" operation allows input of new glossary terms messages using a "newTermValues" message with the input parameters "term" and "value"
- No output is defined for the operation

WSDL Ports - Request-Response Example

```
<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>
```

- "glossaryTerms" defines a request-response operation called "getTerm"
- "getTerm" operation requires
  - Input message called "getTermRequest" with a parameter called "term‖
  - Returns an output message called "getTermResponse" with a parameter called "value"

WSDL Bindings to SOAP

- Regular WSDL elements

```
<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>
```

WSDL Bindings to SOAP (2)

```
<binding type="glossaryTerms" name="b1">
<soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation
     soapAction="http://example.com/getTerm"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
```

WSDL Bindings to SOAP (3)

- <binding> element has two attributes - the name attribute and the type attribute.
- name attribute
  - Can use any name you want
  - Defines the name of the binding
  - Defines the type attribute points to the port for the binding (in this case, the "glossaryTerms" port)
- <soap:binding> element has two attributes - the style attribute and the transport attribute.
  - style attribute can be "rpc" or "document" (in this case, document)
  - transport attribute defines the SOAP protocol to use (in this case we use HTTP)
- <operation> element defines each operation that the port exposes
  - For each operation the corresponding SOAP action has to be defined
  - input and output are encoding must be specified (in this case, "literal‖)

UDDI

- UDDI is a platform-independent framework for
  - Describing services
  - Discovering businesses
  - Integrating business services by using the Internet
  - Platform independent
- Directory for storing information about web services
- Directory of web service interfaces described by WSDL
- Communicates via SOAP
- Sponsored by the Organization for the Advancement of Structure Information Standards (OASIS) http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

UDDI History

- The UDDI was integrated into the Web Services Interoperability (WS-I) standard
  - Central pillar of web services infrastructure
  - By the end of 2005, it was on the agenda for use by more than seventy percent of the Fortune 500 companies in either a public or private implementation
- A UDDI business registration consists of three components:
  - White Pages — address, contact, and known identifiers
  - Yellow Pages — industrial categorizations based on standard taxonomies
  - Green Pages — technical information about services exposed by the business
- In 2006, IBM, Microsoft, and SAP closed their public UDDI nodes
- Most UDDI implementations are now found internal to a company

UDDI Concepts

- Before UDDI
  - There was no Internet standard for businesses to reach their customers and partners with information about their products and services
  - There was no method of how to integrate into each other's systems and processes.
- Problems the UDDI specification can help to solve:
  - Making it possible to discover the right business from the millions currently online
  - Defining how to enable commerce once the preferred business is discovered
  - Reaching new customers and increasing access to current customers
  - Expanding offerings and extending market reach
  - Solving customer-driven need to remove barriers to allow for rapid participation in the global Internet economy
  - Describing services and business processes programmatically in a single, open, and secure environment

Example Use

- Company X is installing a standard business software package that is capable of participating in eBusiness transactions
- At installation time, the package would ask company X whether they want to accept orders over the Web
  - If they answers yes, the package would expose the necessary Web services to accept electronic orders
  - Then it would go to UDDI to register company X and publish its new order entry service
- As company X starts using the new business package, they will start entering customer names and addresses and creating invoices to send to those customers
  - Because this business package is UDDI-aware, it will search for each new customer in the UDDI registry to determine whether that customer supports electronic invoicing
  - If the customer does support electronic invoicing, company X would have the option to send invoices electronically
  - In both of these scenarios, UDDI is used as a programmatically accessible central director of businesses and their services

Programming UDDI

- To ensure most platforms can access UDDI's services, the UDDI directory exposes a bunch of APIs in the form of a SOAP-based Web service
- There are nodes that expose the UDDI Web service
- There are also UDDI test directories
- These can be found by searching the web

UDDI APIs

FINDING THINGS	CATEGORY
find_business	Inquiry
find_service	Inquiry
find_binding	Inquiry
find_tModel	Inquiry
GETTING DETAILS ABOUT THINGS	
get_businessDetail	Inquiry
get_serviceDetail	Inquiry
get_bindingDetail	Inquiry
get_tModelDetail	Inquiry

UDDI APIs (2)

SAVE THINGS	CATEGORY
save_business	**Publishing**
save_service	**Publishing**
save_binding	**Publishing**
save_tModel	**Publishing**
DELETE THINGS	
delete_business	**Publishing**
delete_service	**Publishing**
delete_binding	**Publishing**
delete_tModel	**Publishing**
SECURITY	
get_authToken	**Publishing**
discard_authToken	**Publishing**

Invoking UDDI Procedures

- Send SOAP messages with the appropriate body content
- For example, to search for a company called ―XYZ Industries,‖ the following XML in the body of a SOAP message would be sent:

```
<find_business generic='1.0' xmlns='urn:uddi-org:api'>
   <name>XYZ Industries</name>
</find_business>
```

Conclusion

- Commercial business software could use UDDI to
  - Make it easy for software users to publish Web services
  - Find other Web services that are needed
- Now, UDDI registries are generally used internal to companies
- Enabled by sending and receiving SOAP messages
- Microsoft UDDI SDK is a useful tool for COM-aware services
  - Handles all the SOAP and XML work
  - Programmers write to a COM-based object model
  - http://msdn.microsoft.com/en-us/library/aa966237%28BTS.10%29.aspx

Τέλος Ενότητας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης