<https://slideplayer.com/slide/12291470/>

<https://freecomputerbooks.com/Elements-of-Android-Jetpack.html>

* A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. As all communication is in XML, web services are not tied to any one operating system or programming language—Java can talk with Perl; Windows applications can talk with Unix applications.
* Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
* Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.
* A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

To summarize, a complete web service is, therefore, any service that −

* Is available over the Internet or private (intranet) networks
* Uses a standardized XML messaging system
* Is not tied to any one operating system or programming language
* Is self-describing via a common XML grammar
* Is discoverable via a simple find mechanism

Components of Web Services

The basic web services platform is XML + HTTP. All the standard web services work using the following components −

* SOAP (Simple Object Access Protocol)
* UDDI (Universal Description, Discovery and Integration)
* WSDL (Web Services Description Language)

All these components have been discussed in the [Web Services Architecture](https://www.tutorialspoint.com/webservices/web_services_architecture.htm) chapter.

How Does a Web Service Work?

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of −

* XML to tag the data
* SOAP to transfer a message
* WSDL to describe the availability of service.

You can build a Java-based web service on Solaris that is accessible from your Visual Basic program that runs on Windows.

You can also use C# to build new web services on Windows that can be invoked from your web application that is based on JavaServer Pages (JSP) and runs on Linux.

Example

Consider a simple account-management and order processing system. The accounting personnel use a client application built with Visual Basic or JSP to create new accounts and enter new customer orders.

The processing logic for this system is written in Java and resides on a Solaris machine, which also interacts with a database to store information.

The steps to perform this operation are as follows −

* The client program bundles the account registration information into a SOAP message.
* This SOAP message is sent to the web service as the body of an HTTP POST request.
* The web service unpacks the SOAP request and converts it into a command that the application can understand.
* The application processes the information as required and responds with a new unique account number for that customer.
* Next, the web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP request.
* The client program unpacks the SOAP message to obtain the results of the account registration process.

## What is Web Service?

A web service is a standard for exchanging information between different types of applications irrespective of language and platform. For example, an android application can interact with java or .net application using web services.

What is an example of a web service?

As an example, **Amazon provides a web service that provides prices for products sold online via amazon.com**. The front end or presentation layer can be in . Net or Java but either programming language would have the ability to communicate with the web service.

## Presentation on theme: "Consuming Web Services Using SOAP and REST Apps"— Presentation transcript:

[1](https://slideplayer.com/slide/12291470/72/images/1/Consuming+Web+Services+Using+SOAP+and+REST+Apps.jpg) **Consuming Web Services Using SOAP and REST Apps**  
Lesson 16Consuming Web Services Using SOAP and REST AppsVictor MatosCleveland State UniversityPortions of this page are reproduced from work created and shared by Google and used according to termsdescribed in the Creative Commons 3.0 Attribution License.

[2](https://slideplayer.com/slide/12291470/72/images/2/Android+%26+WebServices+Overview.jpg) **Android & WebServices Overview**  
A WebService is a Consumer\_Machine-to-Provider\_Machine collaboration schema that operates over a computer network.The data exchanges occur independently of the OS, browser, platform, and programming languages used by the provider and the consumers.A provider may expose multiple EndPoints (sets of WebServices), each offering any number of typically related functions.WebServices expect the computer network to support standard Web protocols such as XML, HTTP, HTTPS, FTP, and SMTP.Example: Weather information, money exchange rates, world news, stock market quotation are examples of applications that can be modeled around the notion of a remote data-services provider surrounded by countless consumers tapping on the server’s resources.

[3](https://slideplayer.com/slide/12291470/72/images/3/Android+%26+WebServices+Advantages+of+Using+the+WebService+Architecture.jpg) **Android & WebServices Advantages of Using the WebService Architecture**  
Under the WebService strategy the invoked functions are implemented once (in the server) and called many times (by the remote users).Some advantages of this organization are:Elimination of redundant code,Ability to transparently make changes on the server to update a particular service function without clients having to be informed.Reduced maintenance and production costs.

[4](https://slideplayer.com/slide/12291470/72/images/4/Android+%26+WebServices+Why+should+the+Android+developer+learn+how+to+create+a+WebService.jpg) Android & WebServicesWhy should the Android developer learn how to create a WebService?Simple apps are usually self-contained and do not need to collaborate with other parties to obtain additional data or services (for instance, think of a scientific calculator)However, there are many cases in which the data needed to work with is very extensive, or changes very often and cannot (should not) be hard-coded into the app. Instead, this kind of data should be requested from a reliable external source (for instance, what is the Euro-to-Dollar rate of change right now?)Another class of apps requires a very high computing power perhaps not available in the mobile device (think of problems such as finding the shortest/fastest route between to mapped locations, or best air-fare & route selection for a traveler)It is wise for an Android developer to learn how to solve typical problems that exceed the capacities of the handheld devices. Understanding the possibilities offered by the client-server computing model will make the developer be a more complete and better professional.

[5](https://slideplayer.com/slide/12291470/72/images/5/Android+%26+WebServices+WebService+Architecture.jpg) **Android & WebServices WebService Architecture**  
An ideal Webservice provider is designed around four logical layers which define the ways in which data is to be transported, encoded, exposed and discovered by the users.LayersResponsibilityTransportMove messages through the network, using HTTP, SMTP, FTP, …MessagingEncoding of data to be exchanged (XML)DescriptionWSDL (Web Service Desc. Lang) used for describing public methods available from the endpointDiscoveryUDDI (Universal Description & Discovery Integration) facilitates location and publishing of services through a common registry

[6](https://slideplayer.com/slide/12291470/72/images/6/Android+%26+WebServices+The+Client+Side+-+Consuming+WebServices.jpg) **Android & WebServices The Client Side - Consuming WebServices**  
There are two widely used forms of invoking and consuming WebServices:Representational State Transfer (REST)Closely tie to the HTTP protocol by associating its operation to the common methods: GET, POST, PUT, DELETE for HTTP/HTTPS.This model has a simple invocation mode and little overhead. Service calls rely on a URL which may also carry arguments. Sender & receiver must have an understanding of how they pass data items from one another. As an example: Google Maps API uses the REST model.Remote Procedure Call (RPC).Remote services are seen as coherent collections of discoverable functions (or method calls) stored and exposed by EndPoint providers. Some implementations of this category include: Simple Object Access Protocol (SOAP), Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM) and  Sun Microsystems's Java/Remote Method Invocation (RMI).6 6 6

[7](https://slideplayer.com/slide/12291470/72/images/7/Android+%26+WebServices+Consuming+WebServices.jpg) **Android & WebServices Consuming WebServices**  
Example: Using REST.The following URL is used to make a call to the Google Search service asking to provide links to the subject “Cleveland State University”Transport Provider Action ArgumentsFigure 1. Example of a REST web-service called with a URL that includes arguments7 7 7

[8](https://slideplayer.com/slide/12291470/72/images/8/Android+%26+WebServices+REST+vs.+SOAP.jpg) **Android & WebServices REST vs. SOAP**  
Although SOAP and REST technologies accomplish the same final goal, that is request and receive a service, there are various differences between them.REST users refer to their remote services through a conventional URL that commonly includes the location of the (stateless ) server, the service name, the function to be executed and the parameters needed by the function to operate (if any). Data is transported using HTTP/HTTPS.SOAP requires some scripting effort to create an XML envelop in which data travels. An additional overhead on the receiving end is needed to extract data from the XML envelope. SOAP accepts a variety of transport mechanisms, among them HTTP, HTTPS, FTP, SMTP, etc.SOAP uses WSDL (WebService Description Language) for exposing the format and operation of the services. REST lacks an equivalent exploratory tool.8 8 8

[9](https://slideplayer.com/slide/12291470/72/images/9/Android+%26+WebServices+Figure+2.+A+WebClient+consuming+services+using+REST+%26+SOAP.+Android.+Web-Client..jpg) Android & WebServicesFigure 2. A WebClient consuming services using REST & SOAPAndroidWeb-ClientSOAPSOAP Request: XML envelope holding function-name, arguments.Response: XML formatted resultsWSDL Exploration ToolRESTUsing common URL Requestarg2=val2ResponseFree format. Options include: Plain-text, HTML, XML, JSON…9 9 9 9

[10](https://slideplayer.com/slide/12291470/72/images/10/Android+%26+WebServices+Examples+of+Android+Apps+Using+REST+and+SOAP.jpg) **Android & WebServices Examples of Android Apps Using REST and SOAP**  
In the next sections we will present three examples showing how an Android web-client typically interacts with a remote server requesting and consuming WebServices.Example 1. SOAP client / .NET providerAn Android app uses a XML KSOAP envelope to call a Windows IIS server. WebServices are implemented as a set of C#.NET functions.Example 2. REST client / PHP providerA REST Android client invokes remote PHP services which consult a database on behalf of the client. The response is formatted using JSON.Example 3. REST client / Servlet providerOur Android app communicates with an Tomcat Server in which its WebServices are implemented as Java Servlets. As in the previous example, the results of a database query are returned as a JSON string.10 10 10

[11](https://slideplayer.com/slide/12291470/72/images/11/Android+%26+WebServices+Windows+Communication+Foundation+%28WCF%29.jpg) **Android & WebServices Windows Communication Foundation (WCF)**  
BACKGROUNDWCF is a Microsoft technology that provides a framework for writing code to communicate across heterogeneous platforms [1, 2].An IIS WebServer may host various EndPoints (WebServices).Each of those EndPoints uses WSDL to provide a way of exposing its composition and behavior to clients wishing to find and communicate with the services.Each endpoint includes:address (URL - where to send messages),binding (how to send messages ), and acontract (an explanation of what messages contain)References:[1][2]11 11 11

[12](https://slideplayer.com/slide/12291470/72/images/12/Android+%26+WebServices+WSDL+Service+Contracts.jpg) **Android & WebServices WSDL Service Contracts**  
Example: The link takes us to a WCF EndPoint useful for finding locations in the US based on zip code (only a few lines are shown). This view –written in WSDL- is known as the service contract.<wsdl:definitions xmlns:soap=" targetNamespace="<wsdl:types><s:schema elementFormDefault="qualified" targetNamespace="<s:element name="GetInfoByZIP"><s:complexType><s:sequence><s:element minOccurs="0" maxOccurs="1" name="USZip" type="s:string"/></s:sequence></s:complexType></s:element><s:element name="GetInfoByZIPResponse"><s:element minOccurs="0" maxOccurs="1" name="GetInfoByZIPResult"><s:complexType mixed="true">. . .Method’s name, Argument, TypeReturned result12 12 12

[13](https://slideplayer.com/slide/12291470/72/images/13/Android+%26+WebServices+WSDL+Service+Contracts.jpg) **Android & WebServices WSDL Service Contracts**  
Remove the fragment ?WSDL from the previous link. The shorter URLexposes the endpoint service functions as shown in the figure belowFigure 3. A .NET WebService providing USA ZIP-code information13 13 13

[14](https://slideplayer.com/slide/12291470/72/images/14/Figure+4.+WSDL+Service+Contracts+%26+SOAP+Envelopes+Outgoing+Envelop.jpg) **Figure 4. WSDL Service Contracts & SOAP Envelopes Outgoing Envelop**  
(Request)Incoming Envelop(Response)14 14 14

[15](https://slideplayer.com/slide/12291470/72/images/15/Android+%26+WebServices+WSDL+Service+Contracts+Figure+5..jpg) **Android & WebServices WSDL Service Contracts Figure 5.**  
The Response datais sent by theWebService tothe client as an XML encoded string.15 15 15

[16](https://slideplayer.com/slide/12291470/72/images/16/Android+%26+WebServices+Example1%3A+Android+SOAP+Web-Client.jpg) **Android & WebServices Example1: Android SOAP Web-Client**  
This example consists of two parts. First we construct a server-side EndPoint offering a set of Windows IIS web-services, in the second part we build an Android SOAP client that requests and consumes the previous web-services.Server Side Software:The documents:(VS2015) (VS2010) provide a step-by-step tutorial describing how to create a web-service running on a Windows IIS-Server. We will create three methods: addValues, getPersonList, and rejuvenatePerson.Client Side:KSOAP2 [1] facilitates sending requests and receiving results to/from an IIS server.KSOAP2 includes various access methods such as: .getProperty(…), .getPropertyAsString(…), .getPropertyCount() to dissect the data tokens returned to the Android app inside the composite response object.[1] KSOAP download link:16 16 16

[17](https://slideplayer.com/slide/12291470/72/images/17/Windows+Machine+IIS+Server.jpg) **Windows Machine IIS Server**  
Android & WebServicesExample1: Android SOAP Web-Client & .NET WebServicesWindows Machine IIS ServerC#.NETWeb- ServicesSOAPMySql DatabaseAndroidClient+KSOAPAPIEnvelope holding REQUESTXML Package containing: URL, Namespace, Method,<Argument, Value> setEnvelope with RESPONSEXML <tag>…..</tag>17 17 17 17

[18](https://slideplayer.com/slide/12291470/72/images/18/Android+%26+WebServices+Example1%3A+KSOAP+API+KSOAP+API.jpg) **Android & WebServices Example1: KSOAP API KSOAP API**  
BACKGROUND Android does not supply a native mechanism to handle SOAP exchanges. Consequently we will use an external library such as the KSOAP2 API.KSOAP is designed for limited hardware devices.KSOAP can exchange simple Java types (SoapPrimitive), as well as serialized complex objects (SoapObject).KSOAP APISoapEnvelopeHolds the encoded object’s head and body. Includes a parse() method to pull the supplied XML data.SoapObjectGeneric container that travels inside the XML envelope transporting app’s data. Custom objects must implement the KvmSerializable interface.HttpTransportThis method uses an URL to place an HTTP call and allow SOAP exchange using the JME generic connection framework.KSOAP download link:

[19](https://slideplayer.com/slide/12291470/72/images/19/Android+SOAP+Web-Client+Consuming+.NET+WebServices.jpg) **Android SOAP Web-Client Consuming .NET WebServices**  
Example1:Android SOAP Web-ClientConsuming .NET WebServicesThis Windows IIS EndPoint offers three remote methods. Click on any entry to see contract details regarding its requests & responses

[20](https://slideplayer.com/slide/12291470/72/images/20/Android+%26+WebServices+Example1%3A+A+Sample+of+C%23.NET+Webservices+Page+1+of+3.+namespace+WebServiceDemo1..jpg) Android & WebServicesExample1: A Sample of C#.NET Webservices Page 1 of 3namespace WebServiceDemo1{[WebService(Namespace = "[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)][System.ComponentModel.ToolboxItem(false)][System.Web.Script.Services.ScriptService]public class Service1 : System.Web.Services.WebService// add two numbers[WebMethod]public double addValues(int var1, float var2)return (var1 + var2);} 1The c#.NET methods acting as webservices are marked with the [WebMethod] annotation.Public WebMethods are added to the end-point’s WSDL contract, so their input/output structures can be exposed.

[21](https://slideplayer.com/slide/12291470/72/images/21/Android+%26+WebServices+Example1%3A+A+Sample+of+C%23.NET+Webservices+Page+2+of+3..jpg) Android & WebServicesExample1: A Sample of C#.NET Webservices Page 2 of 3// Return a list of Person objects[WebMethod]public List<Person> getPersonList(String home){List<Person> lst = new List<Person>();//partial list of the "Winterfell Castle" charactersif (home.Equals("Winterfell Castle")) {lst.Add(new Person("Catelyn Stark", 40));lst.Add(new Person("Sansa Stark", 14));lst.Add(new Person("Jon Snow", 22));}return lst;// accept name & age, return a 'younger' Person objectpublic Person rejuvenatePerson(string name, int age)return new Person(name, age - 1);}//class}//namespace2

[22](https://slideplayer.com/slide/12291470/72/images/22/Android+%26+WebServices+Example1%3A+A+Sample+of+C%23.NET+Webservices+Page+3+of+3.+namespace+WebServiceDemo1..jpg) Android & WebServicesExample1: A Sample of C#.NET Webservices Page 3 of 3namespace WebServiceDemo1{[Serializable]public class Personprivate string \_personName;private int \_personAge;public Person( String personNameValue, int personAgeValue )this.\_personName = personNameValue; this.\_personAge = personAgeValue;}public Person( )this.\_personName = "NA"; this.\_personAge = 0;public string personNameget { return this.\_personName; }set { this.\_personName = value; }public int personAgeget { return this.\_personAge; }set { this.\_personAge = value; }3Person (C#.NET)+ Person(string, int)+ personName: String+ personAge: intThere is an equivalent Java (POJO) version of this class in the Android’s app space.

[23](https://slideplayer.com/slide/12291470/72/images/23/Android+%26+WebServices+Example1%3A+C%23+Webservices+-+Comments.jpg) **Android & WebServices Example1: C# Webservices - Comments**  
Our IIS webservice is implemented in C#.Net; however any .NET language could be used . The entry called Namespace identifies the IIS workspace hosting the WebMethods to be called by the Android client app. The literal value of this namespace is important as it will be referenced later by the client app as part of its request object.In our example the .NET service getPersonList accepts a string naming a location (such as ‘Winterfell Castle’) and returns a list of its fictional inhabitants (Person objects made according to the definition shown by Bullet 3). An answer encoded in XML format is sent back to the client. For instance in our example, the returned string is as follows (only a few lines are shown):<ArrayOfPerson xmlns:xsd="<Person><personName>Catelyn Stark</personName><personAge>40</personAge></Person><personName>Sansa Stark</personName><personAge>14</personAge></ArrayOfPerson>

[24](https://slideplayer.com/slide/12291470/72/images/24/Android+%26+WebServices+Example1%3A+C%23+Webservices+-+Comments.jpg) **Android & WebServices Example1: C# Webservices - Comments**  
Important Note: Remember to modify the .NET application’s properties as follows: On the ‘Solution Explorer’ look for the application, Right-Click > Properties > Web > Use Local IIS server > Create Virtual Directory.

[25](https://slideplayer.com/slide/12291470/72/images/25/Android+%26+WebServices+Example1%3A+Android+SOAP+Web-Client+%2B+.NET+Webservices.+Figure+4.+Android+WebClient+App..jpg) Android & WebServicesExample1: Android SOAP Web-Client + .NET WebservicesFigure 4. Android WebClient AppThe figure shows the decoded response for the request: getPersonList ( home) which returns a partial list of the fictional inhabitants of a given home location (eg. “Winterfell Castle”).NOTE: You need to place a copy of the KSOAP jar file in the app’s /libs folder. The KSOAP download link is:

[26](https://slideplayer.com/slide/12291470/72/images/26/Android+%26+WebServices+Example1%3A+Android+SOAP-Client+Consuming+.NET+Webservices.+public+class+SoapTestActivity+extends+Activity+%7B.jpg) Android & WebServicesExample1: Android SOAP-Client Consuming .NET Webservicespublic class SoapTestActivity extends Activity {TextView result;// use handler to keep GUI update on behalf of background tasksHandler handler = new Handler() {@Overridepublic void handleMessage(Message msg) {// TODO Auto-generated method stubsuper.handleMessage(msg);String text = (String) msg.obj;result.append("\n" + text);} };public void onCreate(Bundle savedInstanceState) {super.onCreate(savedInstanceState);setContentView(R.layout.activity\_soap\_test);result = (TextView) findViewById(R.id.result);1

[27](https://slideplayer.com/slide/12291470/72/images/27/Android+%26+WebServices+Example1%3A+Android+SOAP-Client+Consuming+.NET+Webservices.+%2F%2Fdo+slow+calls+to+remote+server+in+a+background+thread..jpg) Android & WebServicesExample1: Android SOAP-Client Consuming .NET Webservices//do slow calls to remote server in a background threadThread slowJob = new Thread() {@Overridepublic void run() {// IP address at homefinal String URL = "final String NAMESPACE = "final String METHOD\_NAME = "getPersonList";String resultValue = "";try {//prepare SOAP REQUEST (namespace, method, arguments)SoapObject request = new SoapObject(NAMESPACE, METHOD\_NAME);//passing primitive (simple) input parametersrequest.addProperty("home", "Winterfell Castle");//prepare ENVELOPE put request insideSoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);envelope.dotNet = true;envelope.setOutputSoapObject(request);//tell the type of complex object to be returned by serviceenvelope.addMapping( NAMESPACE, METHOD\_NAME,new ArrayList<Person>().getClass());2 3

[28](https://slideplayer.com/slide/12291470/72/images/28/Android+%26+WebServices+Example1%3A+Android+SOAP-Client+Consuming+.NET+Webservices.+%2F%2F+TRANSPORT+envelope+to+destination+set+by+URL+%28call+%26+wait%29.jpg) Android & WebServicesExample1: Android SOAP-Client Consuming .NET Webservices// TRANSPORT envelope to destination set by URL (call & wait)HttpTransportSE androidHttpTransport = new HttpTransportSE( URL);androidHttpTransport.call(NAMESPACE + METHOD\_NAME, envelope);// receiving a complex response object (list of Person objects)SoapObject response = (SoapObject) envelope.getResponse();if (response == null) {resultValue = "NULL response received";} else {// get ready to show rows arriving from the serverresultValue = "RESPONSE\n" + response.toString();resultValue += "\n\nPERSON-LIST";//use KSOAP access methods to parse and extract data from responsefor (int i = 0; i < response.getPropertyCount(); i++) {resultValue += "\nRow-" + i;resultValue += "\n\tKSOAP\n\t" + response.getProperty(i);SoapObject personObj = (SoapObject)response.getProperty(i);Person p = new Person(personObj);resultValue += "\n\tJAVA:\n\t" + p.toString();} 4 5 6

[29](https://slideplayer.com/slide/12291470/72/images/29/Android+%26+WebServices+Example1%3A+Android+SOAP-Client+Consuming+.NET+Webservices.+%7D+catch+%28Exception+e%29+%7B.jpg) Android & WebServicesExample1: Android SOAP-Client Consuming .NET Webservices} catch (Exception e) {resultValue = "\nERROR: " + e.getMessage();}// send message to handler so it updates GUIMessage msg = handler.obtainMessage();msg.obj = (String) resultValue;handler.sendMessage(msg);};slowJob.start();}// onCreate7

[30](https://slideplayer.com/slide/12291470/72/images/30/Android+%26+WebServices+Example1%3A+Java+Person+Class+%26+KSOAP+Serialization.+import+org.ksoap2.serialization.SoapObject%3B.jpg) Android & WebServicesExample1: Java Person Class & KSOAP Serializationimport org.ksoap2.serialization.SoapObject;public class Person {private String name;private int age;// constructorspublic Person() {this.name = "na"; this.age = -1;}public Person(String name, int age) {this.name = name; this.age = age;// create a local Java Person object using the KSOAP response (C#) objectpublic Person(SoapObject obj) {this.name = obj.getProperty("personName").toString();this.age = Integer.parseInt(obj.getProperty("personAge").toString());// accessors (get/set) ommitted for brevity...@Overridepublic String toString() {return "Person [name=" + name + ", age=" + age + "]";Person- name: String- age: integer+ Person(SoapObject)+ toString(): String8 30

[31](https://slideplayer.com/slide/12291470/72/images/31/Android+%26+WebServices+Example1%3A+Android+SOAP+Client+App+-+Comments.jpg) **Android & WebServices Example1: Android SOAP Client App - Comments**  
The Android webclient uses a background Thread to offload the task of communicating with the (possibly slow) webserver while keeping the app’s GUI responsive. Updates to the app’s GUI are made by the main thread from the messages sent by the worker thread through a Handler object.The client prepares its request SoapObject indicating the Namespace and WebMethod in that workspace that needs to be executed. Then, each parameter (and its value) sent to the invoked method is added with the .addProperty clause.A SoapSerializationEnvelope is made to carry the user’s request to the server. The envelope’s dotNet property indicates that it expects results to be generated by the Windows server. The .addMapping method is used to identify the Java type of the data returning inside the SOAP envelope. In this example, the clausenew ArrayList<Person>().getClass()tells the response consists of a list of user-defined Person objects.

[32](https://slideplayer.com/slide/12291470/72/images/32/Android+%26+WebServices+Example1%3A+Android+SOAP+Client+App+-+Comments.jpg) **Android & WebServices Example1: Android SOAP Client App - Comments**  
The SOAP envelope is finally transported to the server designated by the given URL. After that, the Android app waits for a response.The envelope.getResponse() clause eventually sends to the Android client a collection of Person objects encoded in XML format (the formal structure of the XML dataset is available in the endpoint’s WSDL contract). The statement response.toString() converts the XML string into equivalent KSOAP notation. For instance, in our example we obtain:anyType{ Person=anyType{personName=Catelyn Stark; personAge=40; }; Person=anyType{personName=Sansa Stark; personAge=14; }; Person=anyType{personName=Jon Snow; personAge=22; }; }The method response.getPropertyCount() is used to determine the size of the returned array. The response.getProperty(i) extracts the i-th row from the response object. Each row (representing a .NET serialized person) is passed to the local Java Person-class constructor (see Bullet 8).

[33](https://slideplayer.com/slide/12291470/72/images/33/representing+serialized+object.jpg) **representing serialized object**  
Android & WebServicesExample1: Android SOAP Client App - CommentsA message is sent from the background worker thread to the main app so its GUI could be updated with the data received from the webservice.The Java Person constructor is given (Bullet 5) a SoapObject holding encoded Person data, for instance, obj looks like the following string:Person=anyType{personName=Catelyn Stark; personAge=40; }This string is dissected using the KSOAP parsing methods obj.getProperty("personName") and obj.getProperty("personAge"). These values are them used to make the equivalent Java objectPerson[name=Catelyn Stark, age=40]JavaPerson class- name : Stringage : intC#.NETPerson class- personName : string- personAge : intXML stringrepresenting serialized object

[34](https://slideplayer.com/slide/12291470/72/images/34/Windows+or+Linux%2FUnix+Machine.jpg) **Windows or Linux/Unix Machine**  
Android & WebServicesExample 2:Using an Android REST-based Client + PHP WebservicesIn this second example an Android client uses REST protocol to interact with a MS-Windows IIS Server. WebServices are implemented as a set of PHP programs. We use JSON encoding for the client-server data exchange.Windows or Linux/Unix MachinePHPJsonMySqlAndroidClient+JSONAPIREQUEST string (Post / Get)?key1=value1&key2=value2RESPONSE formatted as:JSON {……}XML <tag>…..</tag>HTML<html>….</html>CSV “aaa”,111,,333Plain-Text …

[35](https://slideplayer.com/slide/12291470/72/images/35/Android+%26+WebServices+Example+2%3A+Using+an+Android+REST-based+Client+%2B+PHP+services..jpg) Android & WebServicesExample 2: Using an Android REST-based Client + PHP servicesThe remote PHP WebService allows the retrieving of people from a given castleResponse from the server arrives in JSON notation, later it is translated to an equivalent Java collection35

[36](https://slideplayer.com/slide/12291470/72/images/36/Android+%26+WebServices+Example+2%3A+App%E2%80%99s+Layout+36.jpg) **Android & WebServices Example 2: App’s Layout 36**  
<?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android=" android:layout\_width="match\_parent"android:layout\_height="match\_parent" android:padding="7dp" android:orientation="vertical" > <TextView android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Client Request (URL)"/> <TextView android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="URL goes here..." /> <TextView android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginTop="10dp" android:text="Server Response (JSON)" /> <TextView android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Wait - JSON string goes here ..." /> <TextView android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:layout\_marginTop="10dp" android:text="Decoded Response (Java)" /> <TextView android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Wait - decoded JSON here ..." /> </LinearLayout>Example 2: App’s Layout36

[37](https://slideplayer.com/slide/12291470/72/images/37/Android+%26+WebServices+Example+2%3A+Android+REST-based+Client.jpg) **Android & WebServices Example 2: Android REST-based Client**  
import android.app.Activity; import android.app.ProgressDialog; import android.os.AsyncTask; import android.os.Bundle; import android.util.Log; import android.widget.TextView;import com.google.gson.Gson; import com.google.gson.JsonArray; import com.google.gson.JsonElement; import com.google.gson.JsonObject; import com.google.gson.JsonParser; import com.google.gson.JsonSyntaxException; import com.google.gson.reflect.TypeToken;import java.net.HttpURLConnection;import java.net.URI;import java.net.URL; import java.io.BufferedReader; import java.io.InputStream; import java.io.InputStreamReader; import java.lang.reflect.Type; import java.net.URI; import java.util.ArrayList; import java.util.Date;This app needs to import a support library. For details on how to setup external JARS, see Lesson 14 - Example 7.Gson downloaded from

[38](https://slideplayer.com/slide/12291470/72/images/38/Android+%26+WebServices+Example+2%3A+Android+REST-based+Client.jpg) **Android & WebServices Example 2: Android REST-based Client**  
public class MainActivity extends Activity {TextView txtRequestUrl, txtResponseJson, txtResponseJava;@Overridepublic void onCreate(Bundle savedInstanceState) {super.onCreate(savedInstanceState);setContentView(R.layout.activity\_main);txtRequestUrl = (TextView) findViewById(R.id.txtRequestUrl);txtResponseJson = (TextView) findViewById(R.id.txtResponseJson);txtResponseJava = (TextView) findViewById(R.id.txtResponseJava);// REQUEST object consists of "URL?ARGUMENTS" (spaces replaced by +)// each argument is a KEY=VALUE pair.//// SAMPLE1: Calling a webserver offering services implemented as PHP functions// arguments offered as ?castle=value// where possible castle values are: Winterfell, Kings\_Landing, DragonStone// Testing on local Windows machine: IIS + PHP// You should get your machine’s real IP address using c> ipconfig /allString SERVER\_URL ="1

[39](https://slideplayer.com/slide/12291470/72/images/39/Android+%26+WebServices+Example+2%3A+Android+REST-based+Client.jpg) **Android & WebServices Example 2: Android REST-based Client**  
// String SERVER\_URL =//"// Host is a Unix machine at CSU + PHP services// "// "?castle=Winterfell";//// Host is a commercial UNIX site running PHP services + MySQL databases//"//"//// SAMPLE2: This solution in based on Java code running on the server and the// client. Java-servlet service running on a local Tomcat server(port :8080)// "// "

[40](https://slideplayer.com/slide/12291470/72/images/40/Android+%26+WebServices+Example+2%3A+Android+REST-based+Client.jpg) **Android & WebServices Example 2: Android REST-based Client**  
txtRequestUrl.setText(new Date() + "\n" + SERVER\_URL);// Use AsyncTask to execute potential slow task without freezing GUInew LongOperation().execute(SERVER\_URL);}// mainprivate class LongOperation extends AsyncTask<String, Void, Void> {private String jsonResponse;private ProgressDialog dialog = new ProgressDialog(MainActivity.this);protected void onPreExecute() {dialog.setMessage("Please wait..");dialog.show();}protected Void doInBackground(String... urls) {try {// WARNING// You must use actual IP addresses, do not enter "localhost:8080/..."// try something like:// solution uses Java.Net class (Apache.HttpClient is now deprecated)2

[41](https://slideplayer.com/slide/12291470/72/images/41/Android+%26+WebServices+Example+2%3A+Android+REST-based+Client.jpg) **Android & WebServices Example 2: Android REST-based Client**  
// STEP1. Create a HttpURLConnection object releasing REQUEST to given siteURL url = new URL(urls[0]); //argument supplied in the call to AsyncTaskHttpURLConnection urlConnection = (HttpURLConnection)url.openConnection();urlConnection.setRequestProperty("User-Agent", "");urlConnection.setRequestMethod("POST");urlConnection.setDoInput(true);urlConnection.connect();// STEP2. wait for incoming RESPONSE stream, place data in a bufferInputStream isResponse = urlConnection.getInputStream();BufferedReader responseBuffer = new BufferedReader(new InputStreamReader(isResponse)));// STEP3. Arriving JSON fragments are concatenate into a StringBuilderString myLine = "";StringBuilder strBuilder = new StringBuilder();while ((myLine = responseBuffer.readLine()) != null) {strBuilder.append(myLine);}//show response (JSON encoded data)jsonResponse = strBuilder.toString();Log.e("RESPONSE", jsonResponse);} catch (Exception e) { Log.e("RESPONSE Error", e.getMessage()); }return null; // needed to gracefully terminate Void method3

[42](https://slideplayer.com/slide/12291470/72/images/42/Android+%26+WebServices+Example+2%3A+Android+REST-based+Client.jpg) **Android & WebServices Example 2: Android REST-based Client**  
protected void onPostExecute(Void unused) {try {dialog.dismiss();// update GUI with JSON ResponsetxtResponseJson.setText(jsonResponse);// Step4. Convert JSON list into a Java collection of Person objects// prepare to decode JSON response and create Java listGson gson = new Gson();Log.e("PostExecute", "content: " + jsonResponse);// set (host) Java type of encoded JSON responseType listType = new TypeToken<ArrayList<Person>>() { }.getType();Log.e("PostExecute", "arrayType: " + listType.toString());// decode JSON string into appropriate Java containerArrayList<Person> personList = gson.fromJson(jsonResponse, listType);Log.e("PostExecute", "OutputData: " + personList.toString());// Step5. Show results (update GUI with Java version of retrieved list)txtResponseJava.setText(personList.toString());// OPTIONAL.The following strategy shows an alternative mechanism to// interpret the returned JSON response. Here you parse the nodes of the4

[43](https://slideplayer.com/slide/12291470/72/images/43/Android+%26+WebServices+Example+2%3A+Android+REST-based+Client.jpg) **Android & WebServices Example 2: Android REST-based Client**  
// underlying data structure using GSON element classes. A JsonElement could// be a: JsonObject, JsonArray, or JsonPrimitiveString result = "\n";try {JsonElement jelement = new JsonParser().parse(jsonResponse);JsonArray jarray = jelement.getAsJsonArray();for (int i = 0; i < jarray.size(); i++) {JsonObject jobject = jarray.get(i).getAsJsonObject();result += jobject.get("name").toString() + " --- "+ jobject.get("age").toString() + "\n";}} catch (Exception e) {Log.e("PARSING", e.getMessage());txtResponseJava.append(result);} catch (JsonSyntaxException e) {Log.e("POST-Execute", e.getMessage());}// asyncktask}// class4

[44](https://slideplayer.com/slide/12291470/72/images/44/Android+%26+WebServices+Example+2%3A+Android+REST-based+Client+-+Comments.jpg) **Android & WebServices Example 2: Android REST-based Client - Comments**  
The client app uses an AsyncTask object to call the remote WebService. This practice is functionally equivalent to running a background thread but offers more options (such as showing progress messages).doInBackground takes the supplied URL to reach the server and request its assistance. The Java.Net class is responsible for establishing the asynchronous HTTP client-server exchange. In our example an HTTP GET operation is invoked using the URL:the called PHP method getPersonMySQL extracts its result from a MySql database. Observe that URL arguments appear after the ? symbol. You may also replace spaces with + symbols as in castle=King’s+LandingThe client app collects the JSON encoded result in a StringBuilder. In our example the returned string is: [{"name":"Tiryion Lannister", "age":30},{"name":"Cersei Baratheon","age":32}]HTTPClient Ref:

[45](https://slideplayer.com/slide/12291470/72/images/45/Android+%26+WebServices+Example+2%3A+Android+REST-based+Client+-+Comments.jpg) **Android & WebServices Example 2: Android REST-based Client - Comments**  
The onPostExecute portion of the AsyncTask decodes the JSON response string into a Java ArrayList<Person> collection.We have chosen Google’s GSON API for processing the incoming JSON string. The example shows two approaches for decoding JSON data:(a) The first invokes the fromJson(data, type) method,(b) the second alternative parses the JSON string looking for the occurrence of JsonElement, JsonArray, and JsonObject tokens.For more details on this issue see Lesson 14Google’s GSON API Download:

[46](https://slideplayer.com/slide/12291470/72/images/46/Android+%26+WebServices+Example+2%3A+Android+REST-based+Client+-+Comments.jpg) **Android & WebServices Example 2: Android REST-based Client - Comments**  
Final details: Your Gradle Scripts/build.gradle(Module:app) file needs to be modified to include references to the GSON API. In our example we have:...dependencies { compile 'com.android.support:support-v4:19.1.0' compile files('libs/gson jar') }Imported JARS should be stored in the apps/lib folderThe Manifest must include request to use Internet.<uses-permission android:name="android.permission.INTERNET"/>

[47](https://slideplayer.com/slide/12291470/72/images/47/Android+%26+WebServices+Json.jpg) **Android & WebServices Json**  
Example 3: Android REST consuming ServLet WebservicesThis problem is similar to Example-2. An Android REST-based client supplies a location and a webservice retrieves all inhabitants of that place.As before we divide the problem in two parts, first we discuss step-by-step how to create the Tomcat hosted Servlet, then we build an Android client using REST to communicate with it.AndroidClient+JSONAPIREQUEST string (Post / Get)?key1=value1&key2=value2Apache-Tomcat ServerJava ServletsJsonMySqlRESPONSE formatted as:JSON {……}XML <tag>…..</tag>HTML<html>….</html>CSV “aaa”,111,,333Plain-Text …Tomcat Server download:

[48](https://slideplayer.com/slide/12291470/72/images/48/Android+%26+WebServices+Example+3%3A+Android+REST+consuming+ServLet+Swevices..jpg) Android & WebServicesExample 3: Android REST consuming ServLet SwevicesAn HttpServlet is a Java class used to extend the functionality of a web server. It is commonly used to provide dynamic web content in a manner similar to the ASP, .NET and PHP technologies.A Servlet primarily operates on two distinct objects:HttpServletRequest: Represents a client's request. This object gives a servlet access to incoming information such as HTML form data and HTTP request headers.HttpServletResponse: Represents the servlet's response. The servlet uses this object to return data to the client such as HTTP errors, response headers (Content-Type, Set-Cookie, and others), and output data by writing to the response's output stream or output writer.Reference:48 48 48

[49](https://slideplayer.com/slide/12291470/72/images/49/Android+%26+WebServices+Example+3%3A+Create+a+ServLet.jpg) **Android & WebServices Example 3: Create a ServLet**  
We assume you are using Eclipse EE and Apache Tomcat 7.0 (or newer).From Eclipse - create a new Dynamic Web project.Set project name to: GameOfThrones.On ‘Target runtime’ pane, click ‘New runtime’. From the drop-down list choose Tomcat Server v7.0 (or a newer version). Click ‘Next’ button.In textbox ‘Tomcat Installation directory’ enter the location where your Tomcat software is placed (usually: C:\Program Files\Apache Software Foundation\Tomcat 7.0). Click button ‘Finish’ > ‘Finish’.Add new package csu.yourlastname to the folder /JavaResources/srcPlace inside the package a new Servlet. In the ‘Class name’ box enter: GetHeroes. Click the button ‘Finish’.Add to the package the Person class .Copy to the folder: WebContent/WEB-INF/lib the GSON and MySQL-JConnector jars. They are needed to support JSON encoding and access to a MySql database.49 49 49

[50](https://slideplayer.com/slide/12291470/72/images/50/Android+%26+WebServices+Example+3%3A+Create+a+ServLet.jpg) **Android & WebServices Example 3: Create a ServLet**  
Create inside the folder WebContent/META-INF/ a XML file called context.xml indicating how to access your MySql database (code will be shown shortly).Create inside the folder WebContent/WEB-INF a XML file called web.xml explaining the parts of the service and its mapping of symbolic names to Java classes (shown ahead).The structure of your Java Servlet should be as indicated in the figure. Add pending code, and test your app.Make sure that only the Eclipse developer’s web server is running.50 50 50

[51](https://slideplayer.com/slide/12291470/72/images/51/Android+%26+WebServices+Example+3%3A+Create+a+ServLet+-+Test+Results.jpg) **Android & WebServices Example 3: Create a ServLet - Test Results**

[52](https://slideplayer.com/slide/12291470/72/images/52/Android+%26+WebServices+Example+3%3A+Create+a+ServLet+context.xml.jpg) **Android & WebServices Example 3: Create a ServLet context.xml**  
This file indicates how to connect to a MySql database. It includes the user’s name, password, driver name, connection parameters, and database connection string.It must be placed in the WebContent/META-INF/ folder.<?xml version="1.0" encoding="UTF-8"?><Context><!-- comments removed --><Resource name="jdbc/mysqlresourcegameofthrones"auth="Container"type="javax.sql.DataSource"maxActive="100" maxIdle="30" maxWait="10000"username="csuperson"password="euclid"driverClassName="com.mysql.jdbc.Driver"url="jdbc:mysql:// :3306/gameofthrones"/></Context>

[53](https://slideplayer.com/slide/12291470/72/images/53/Android+%26+WebServices+Example+3%3A+Create+a+ServLet+1+of+2+web.xml.jpg) **Android & WebServices Example 3: Create a ServLet 1 of 2 web.xml**  
This file lists the parts of the servlet, its resources, and indicates how to map the Servlet’s java classes to webService names passed in the request object. It must be placed in the WebContent/WEB-INF folder.<?xml version="1.0" encoding="UTF-8"?><web-app xmlns:xsi=" xmlns=" xmlns:web=" xsi:schemaLocation=" id="WebApp\_ID" version="3.0"><display-name>GameOfThrones</display-name><welcome-file-list><welcome-file>index.html</welcome-file><welcome-file>index.htm</welcome-file></welcome-file-list><servlet><description></description><display-name>GetHeroes</display-name><servlet-name>GetHeroes</servlet-name><servlet-class>csu.matos.GetHeroes</servlet-class></servlet>

[54](https://slideplayer.com/slide/12291470/72/images/54/Android+%26+WebServices+Example+3%3A+Create+a+ServLet+2+of+2+web.xml+cont..jpg) **Android & WebServices Example 3: Create a ServLet 2 of 2 web.xml cont.**  
<servlet-mapping><servlet-name>GetHeroes</servlet-name><url-pattern>/GetHeroes</url-pattern></servlet-mapping><resource-ref><description>MySQL Datasource – GameOfThrones Example</description><res-ref-name>jdbc/mysqlresourcegameofthrones</res-ref-name><res-type>javax.sql.DataSource</res-type><res-auth>Container</res-auth></resource-ref></web-app>

[55](https://slideplayer.com/slide/12291470/72/images/55/Android+%26+WebServices+Example+3%3A+Create+a+ServLet+-+GetHeroes+Servlet+1+of+5..jpg) Android & WebServicesExample 3: Create a ServLet - GetHeroes Servlet 1 of 5GetHeores is the actual servlet. It does all its work inside the doGet method. The incoming request string is examined looking for the argument=value pairs it may carry. Once the castle value is know a list of its associated people is assembled. The servlet redundantly extracts its data from two sources, first it uses an in-memory collection of datasets, then it repeats the same type of retrieval querying a mySql database.@WebServlet("/GetHeroes")public class GetHeroes extends HttpServlet {private static final long serialVersionUID = 1L;public void doGet(HttpServletRequest request, HttpServletResponse response)throws ServletException, IOException {PrintWriter output = response.getWriter();output.println(" [0] NEW TEST: " + new Date());Map<String, String[]> requestMap = request.getParameterMap();if (requestMap.isEmpty()) {output.println(" [1] <<<< SERVLET CALLED - Empty Request >>>> ");}for (String key : requestMap.keySet()) {String[] value = requestMap.get(key);output.println(" [2] >>> " + key + " " + value[0]);1

[56](https://slideplayer.com/slide/12291470/72/images/56/Android+%26+WebServices+Example+3%3A+Create+a+ServLet+-+GetHeroes+Servlet+2+of+5.+%2F%2F+setup+memory+resident+datasets+%28fake+core+database%29.jpg) Android & WebServicesExample 3: Create a ServLet - GetHeroes Servlet 2 of 5// setup memory resident datasets (fake core database)ArrayList<Person> winterfellPeople = new ArrayList<Person>();winterfellPeople.add(new Person("Arya Stark", 11, "Winterfell Castle"));winterfellPeople.add(new Person("Jon Snow", 20, "Winterfell Castle"));winterfellPeople.add(new Person("Hodor", 40, "Winterfell Castle"));ArrayList<Person> dragonPeople = new ArrayList<Person>();dragonPeople.add(new Person("Daenerys Targaryen", 18, "Dragonstone"));ArrayList<Person> kingsPeople = new ArrayList<Person>();kingsPeople.add(new Person("Tiryion Lannister", 30, "King's Landing"));kingsPeople.add(new Person("Cersei Baratheon", 32, "King's Landing"));String castle = "";castle = request.getParameter("castle");output.println(" [3]PARAMETER castle: " + castle);// Part1. add to peopleResult data from memory-held lists// if no castle is supplied, include all charactersArrayList<Person> peopleResult = new ArrayList<Person>();if (castle==null){castle = "";peopleResult.addAll(winterfellPeople);peopleResult.addAll(dragonPeople);peopleResult.addAll(kingsPeople);2 3

[57](https://slideplayer.com/slide/12291470/72/images/57/Android+%26+WebServices+Example+3%3A+Create+a+ServLet+-+GetHeroes+Servlet+3+of+5.+%7D+else+if+%28+winterfell+castle+.startsWith%28castle%29%29+%7B.jpg) Android & WebServicesExample 3: Create a ServLet - GetHeroes Servlet 3 of 5} else if ("winterfell castle".startsWith(castle)) {peopleResult = winterfellPeople;} else if ("dragonstone".startsWith(castle)) {peopleResult = dragonPeople;} else if ("king's landing".startsWith(castle)) {peopleResult = kingsPeople;}// prepare to do GSON encoding of selected peopleGson gson = new Gson();String jsonData = gson.toJson(peopleResult);output.println(" [4] TAKEN FROM MEMORY-DB: " + jsonData);// Part2. Redo peopleResult now retrieving from mySQL db.try {peopleResult = getDbRecord(castle);jsonData = gson.toJson(peopleResult);output.println(" [5] TAKEN FROM MYSQL-DB: " + jsonData);} catch (Exception e) {output.println("NO DATA");output.flush();}// doGet4 5

[58](https://slideplayer.com/slide/12291470/72/images/58/Android+%26+WebServices+Example+3%3A+Create+a+ServLet+-+GetHeroes+Servlet+4+of+protected+void+doPost%28+HttpServletRequest+request%2C.jpg) Android & WebServicesExample 3: Create a ServLet - GetHeroes Servlet 4 of 5@Overrideprotected void doPost( HttpServletRequest request,HttpServletResponse response)throws ServletException, IOException {doGet(request, response);}// Retrieve designated people this time from a mySql databaseprivate ArrayList<Person> getDbRecord(String castle) throws Exception {ArrayList<Person> result = new ArrayList<Person>();try{javax.naming.Context initContext = new InitialContext();Context envContext = (Context)initContext.lookup("java:/comp/env");javax.sql.DataSource ds = (DataSource) envContext.lookup("jdbc/mysqlresourcegameofthrones");java.sql.Connection cnn = ds.getConnection();String mySQL = " select name, age, castle "+ " from person where castle LIKE '%" + castle + "%'";java.sql.Statement stm = cnn.createStatement();java.sql.ResultSet rs = stm.executeQuery(mySQL);6

[59](https://slideplayer.com/slide/12291470/72/images/59/Android+%26+WebServices+Example+3%3A+Create+a+ServLet+-+GetHeroes+Servlet+5+of+5.+while+%28+rs.next%28%29+%29%7B.jpg) Android & WebServicesExample 3: Create a ServLet - GetHeroes Servlet 5 of 5while ( rs.next() ){String pname = rs.getString("name");int page = Integer.parseInt(rs.getString("age"));String pcastle = rs.getString("castle");Person pers = new Person(pname, page, pcastle);result.add(pers);}catch (java.sql.SQLException e) {throw new Exception (" [\*[Problems1 ]\*] " + e.getMessage());} catch (javax.naming.NamingException e) {throw new Exception ( " [\*[Problems2 ]\*] " + e.getMessage());return result;}//getDbRecord}// class7

[60](https://slideplayer.com/slide/12291470/72/images/60/Android+%26+WebServices+Example+3%3A+Create+a+ServLet+-+Person+class.jpg) **Android & WebServices Example 3: Create a ServLet - Person class**  
public class Person {private String name;private int age;private String castle;public Person(String name, int age, String castle) {this.name = name; this.age = age; this.castle = castle;}public Person() {this.name = "na"; this.age = -1; this.castle = "na";public String getName() { return name; }public void setName(String name) { this.name = name; }public int getAge() { return age; }public void setAge(int age) { this.age = age; }@Overridepublic String toString() {return "Person [name=" + name + ", age=" + age + ", castle=" + castle + "]";8

[61](https://slideplayer.com/slide/12291470/72/images/61/Android+%26+WebServices+Example+3%3A+ServLet+-+Comments.jpg) **Android & WebServices Example 3: ServLet - Comments**  
This servlet only implements its doGet method in which a request URL string is accepted and a response object is sent back to the caller.For demonstration purposes the servlet returns twice the result-set. First data is selected from an in-memory (ArrayList-based) collection, the second time data comes from a relational database.The incoming castle argument is extracted from the request URL, it indicates what group of people should be retrieved.After a resultset consisting of Person objects held in an ArrayList is assembled, the collection is converted to a JSON string (we use GSON library, see Lecture 14).The method getDbRecord redundantly retrieves a version of the resultset from a MySql database. DataSource specs for reaching the database are taken from the global server’s contex file.A select-statement adds the castle argument to its where clause to find the tuples satisfying the search condition.Database rows are scanned one at the time. From each selected tuple a Person object is created and added to a growing ArrayList. After all rows are processed the ArrayList is returned.

[62](https://slideplayer.com/slide/12291470/72/images/62/Android+%26+WebServices+Example+3%3A+Transfer+your+Servlet+to+the+Production+Server.+Eclipse+Workspace..jpg) Android & WebServicesExample 3: Transfer your Servlet to the Production ServerEclipse WorkspaceTomcat Server1 2 3 4

[63](https://slideplayer.com/slide/12291470/72/images/63/Android+%26+WebServices+Example+3%3A+Transfer+your+Servlet+to+the+Production+Server.+Preliminary+Steps..jpg) Android & WebServicesExample 3: Transfer your Servlet to the Production ServerPreliminary StepsStop the Eclipse Tomcat server instance (no other webserver should be running).Locate the Tomcat production server in your system ( usually at c:\Program Files\Apache Software Foundation\Tomcat 7.0 ). If needed create a folder called/webapps/GameOfThrones. Add the subdirectories META-INF and WEB-INF.Transfer Files From Eclipse Workspace to the Tomcat Production ServerModify the production server’s file \conf\context.xml. Add to it a copy of the <resource… /> XML entry from your Eclipse WebContent\META-INF\context.xml file. Observe that you may end-up with more than one global <resource> entry.Use your Windows Explorer to find and copy the Eclipse’s build\classes folder. Paste this copy in the server’s webapps\GameOfThrone\WEB-INF\ folder.Copy each of the jars used by the servlet into the server’s \lib\ folder.Place a copy of the \WEB-INF\web.xml file in the corresponding \WEB-INF\ server’s folder.Restart the server.63

[64](https://slideplayer.com/slide/12291470/72/images/64/Android+%26+WebServices+Example+3%3A+Transfer+your+Servlet+to+the+Production+Server..jpg) Android & WebServicesExample 3: Transfer your Servlet to the Production ServerRun/Restart the Tomcat service (Control Panel > Administrative Tools > Services > Locate Apache Tomcat > Start). Run a browser with the following URL:The returned page is shown below. Now, you may want to remove all the debugging commentary and only return the encoded JSON data that appears at the end [{…]]64

[65](https://slideplayer.com/slide/12291470/72/images/65/Android+%26+WebServices+Example+3%3A+Android+Client.jpg) **Android & WebServices Example 3: Android Client**  
Use the same Android client from the previous example. Change the URL to the desired destination. For this example, test the client-app with the URL values:65

[66](https://slideplayer.com/slide/12291470/72/images/66/Producing+%26+Consuming+Web+Services.jpg) **Producing & Consuming Web Services**  
< Questions />66 66

[67](https://slideplayer.com/slide/12291470/72/images/67/Appendix+A.+Connecting+to+Oracle+DBMS.jpg) **Appendix A. Connecting to Oracle DBMS**  
REST Protocol – Android & Apache's Tomcat ServerThis relaces Step 8 in Example 2C. Multitier ApplicationAdd the following DataSource to the application’s context.xml file. Add the file to the /WebControl/META-INF/ folder of your Eclipse workspace solution (later, this fragment will be copied to the Tomcat’s /conf/context.xml file)<?xml version="1.0" encoding="UTF-8"?><context><Resource name="jdbc/myoracle" auth="Container"type="javax.sql.DataSource" driverClassName="oracle.jdbc.OracleDriver"username="CSUPERSON" password="EUCLID" maxActive="20" maxIdle="10"maxWait="-1"/></context>The above DataSource helps the JDBC connection identify the involved server and user. Other users could reuse the data-source and provide individual credentials.Reference:67 67 67 67

[68](https://slideplayer.com/slide/12291470/72/images/68/Appendix+B.+JDBC+Architecture.jpg) **Appendix B. JDBC Architecture 68 68 68 68**  
JDBC™ 4.0 Specification. JSR 221Sun MicrosystemsLance Andersen, Specification LeadNovember 7, 200668 68 68 68

