1.in which scenario you used multi-threading like banking or eCommerce project .please explain with example?

Ans- As a 3 year experience developer I didn’t get a chance to work on Multithreading environment .these part is developed by our senior team members. But I have aware on multithreading and I know how to work.

Or

###### Ans- Multithreading Realtime Examples

1. Background jobs like running application servers like Oracle application server, Web servers like Tomcat ctc which will come into action whenever a request comes.
2. Performing some execution while I/O blocked.
3. Gathering information from different web services running in parallel.
4. Typing MS Word document while listening to music.
5. Games are very good examples of threading. You can use multiple objects in games like cars, motor bikes, animals, people etc. All these objects are nothing but just threads that run your game application.
6. Railway ticket reservation system where multiple customers accessing the server.
7. Multiple account holders accessing their accounts simultaneously on the server. When you insert a ATM card, it starts a thread for perform your operations.
8. Servlets are multithreaded.

2.what is the difference between spring data jpa, hibernate spring jdbc? how you used in project?

Ans- Hibernate is a JPA implementation, while Spring Data JPA is a JPA Data Access Abstraction. Spring Data offers a solution to GenericDao custom implementations. It can also generate JPA queries on your behalf through method name conventions.

Spring Data JPA is not an implementation or JPA provider, it's just an abstraction used to significantly reduce the amount of boilerplate code required to implement data access layers for various persistence stores. I hope this article useful to you

Hibernate provides a reference implementation of the Java Persistence API that makes it a great choice as an ORM tool with benefits of loose coupling.

Or

**What is diff b/w spring-jdbc and hibernate?**

Ans-There is no more different because spring provided integration with both JDBC and Hibernate but based on requirement we have to choose,

If my application is internet application with having multiple users so here data operation is more so we have to choose JDBC approach. Ex: Banking Application

If my application is intranet application where the access user is not more and data operation is less then in this case we have to use hibernate Ex: Any internal application for specific organization.

3.many to many relationship example?

Ans- In our database we create many tables and many of them may be associated with each other. At higher lever, these associations can be classified into one-to-one, one-to-many and many-to-many. These associations can be either unidirectional or bidirectional mappings. Hibernate provides support to all these associations.

In this page, we see how to implement bidirectional Many-to-Many mapping between persisted objects using @ManyToMany annotation.

A many-to-many relationship occurs when multiple entities are related to multiple occurrences of another entity.

Create DB Table

Here is our DB tables. Our tables are already Many-to-Many mapped. We have created EMPLOYEES, PROJECTS, and EMP\_ASSIGNMENTS tables.

CREATE TABLE EMPLOYEES (

EMP\_ID BIGINT NOT NULL AUTO\_INCREMENT,

NAME VARCHAR(252),

DEPARTMENT VARCHAR(128),

SALARY BIGINT,

JOINED\_ON TIMESTAMP,

PRIMARY KEY (EMP\_ID)

);

CREATE TABLE PROJECTS (

PR\_ID BIGINT NOT NULL AUTO\_INCREMENT,

NAME VARCHAR(252),

OWNER VARCHAR(252),

PRIMARY KEY (PR\_ID)

);

CREATE TABLE EMP\_ASSIGNMENTS (

EA\_ID BIGINT NOT NULL AUTO\_INCREMENT,

EMP\_ID BIGINT,

PR\_ID BIGINT,

PRIMARY KEY (EA\_ID),

FOREIGN KEY (EMP\_ID) REFERENCES EMPLOYEES(EMP\_ID),

FOREIGN KEY (PR\_ID) REFERENCES PROJECTS(PR\_ID)

);

INSERT INTO EMPLOYEES (EMP\_ID, NAME, DEPARTMENT, SALARY, JOINED\_ON) VALUES (1, 'Nataraja G', 'Documentation', 10000, CURRENT\_TIMESTAMP);

INSERT INTO EMPLOYEES (EMP\_ID, NAME, DEPARTMENT, SALARY, JOINED\_ON) VALUES (2, 'Amar M', 'Entertainment', 12000, CURRENT\_TIMESTAMP);

INSERT INTO EMPLOYEES (EMP\_ID, NAME, DEPARTMENT, SALARY, JOINED\_ON) VALUES (3, 'Nagesh Y', 'Admin', 25000, CURRENT\_TIMESTAMP);

INSERT INTO EMPLOYEES (EMP\_ID, NAME, DEPARTMENT, SALARY, JOINED\_ON) VALUES (4, 'Vasu V', 'Security', 2500, CURRENT\_TIMESTAMP);

INSERT INTO PROJECTS (PR\_ID, NAME, OWNER) VALUES (1, 'Ticket Online', 'Bunty');

INSERT INTO PROJECTS (PR\_ID, NAME, OWNER) VALUES (2, 'Disco Drum', 'Chanty');

INSERT INTO PROJECTS (PR\_ID, NAME, OWNER) VALUES (3, 'Traffic Audit', 'Sonty');

INSERT INTO EMP\_ASSIGNMENTS (EA\_ID, EMP\_ID, PR\_ID) VALUES (1, 1, 1);

INSERT INTO EMP\_ASSIGNMENTS (EA\_ID, EMP\_ID, PR\_ID) VALUES (2, 1, 2);

INSERT INTO EMP\_ASSIGNMENTS (EA\_ID, EMP\_ID, PR\_ID) VALUES (3, 1, 3);

INSERT INTO EMP\_ASSIGNMENTS (EA\_ID, EMP\_ID, PR\_ID) VALUES (4, 2, 2);

INSERT INTO EMP\_ASSIGNMENTS (EA\_ID, EMP\_ID, PR\_ID) VALUES (5, 2, 3);

INSERT INTO EMP\_ASSIGNMENTS (EA\_ID, EMP\_ID, PR\_ID) VALUES (6, 3, 1);

INSERT INTO EMP\_ASSIGNMENTS (EA\_ID, EMP\_ID, PR\_ID) VALUES (7, 3, 3);

INSERT INTO EMP\_ASSIGNMENTS (EA\_ID, EMP\_ID, PR\_ID) VALUES (8, 4, 1);

INSERT INTO EMP\_ASSIGNMENTS (EA\_ID, EMP\_ID, PR\_ID) VALUES (9, 3, 2);

Hibernate Configuration File

Create hibernate xml based configuration file in your resources folder (classpath). This file includes hibernate configurations like driver class, DB connectivity URL, DB credentials and so on...

j2n-hibernate.cfg.xml

?

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/java2novice</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>

<property name="show\_sql">false</property>

</session-factory>

</hibernate-configuration>

Hibernate Mapping with Entity Class

Here is the entity classes according to our database tables along with Many-to-Many mappings.

package com.java2novice.model;

import java.io.Serializable;

import java.util.Date;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

@Entity

@Table(name="EMPLOYEES")

public class Employee implements Serializable {

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

@Column(name="EMP\_ID", unique = true, nullable = false)

private Long empId;

private String name;

private String department;

private Long salary;

@Column(name="JOINED\_ON")

private Date joinedOn;

@ManyToMany(cascade=CascadeType.ALL, fetch=FetchType.EAGER)

@JoinTable(

name = "EMP\_ASSIGNMENTS",

joinColumns = { @JoinColumn(name = "EMP\_ID") },

inverseJoinColumns = { @JoinColumn(name = "PR\_ID") }

)

private List<Project> empAssignmentList;

public Long getEmpId() {

return empId;

}

public void setEmpId(Long empId) {

this.empId = empId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

public Long getSalary() {

return salary;

}

public void setSalary(Long salary) {

this.salary = salary;

}

public Date getJoinedOn() {

return joinedOn;

}

public void setJoinedOn(Date joinedOn) {

this.joinedOn = joinedOn;

}

public List<Project> getEmpAssignmentList() {

return empAssignmentList;

}

public void setEmpAssignmentList(List<Project> empAssignmentList) {

this.empAssignmentList = empAssignmentList;

}

@Override

public String toString() {

String resp = this.empId+" | "+this.name+" | "+this.department+" | "+this.salary+" | "+this.joinedOn;

return resp;

}

}

package com.java2novice.model;

import java.util.List;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

@Entity

@Table(name="PROJECTS")

public class Project {

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

@Column(name="PR\_ID", unique = true, nullable = false)

private Long prId;

private String name;

private String owner;

@ManyToMany(mappedBy = "empAssignmentList")

private List<Employee> employees;

@Override

public String toString() {

return this.prId +" | "+ this.name +" | "+ this.owner;

}

public Long getPrId() {

return prId;

}

public void setPrId(Long prId) {

this.prId = prId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getOwner() {

return owner;

}

public void setOwner(String owner) {

this.owner = owner;

}

}

Hibernate Utility Class

This class manages hibernate session.

?

package com.java2novice.hibernate;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.Configuration;

import org.hibernate.service.ServiceRegistry;

import com.java2novice.model.Employee;

import com.java2novice.model.Project;

public class HibernateUtil {

private static SessionFactory sessionFactory = null;

static {

try{

loadSessionFactory();

}catch(Exception e){

System.err.println("Exception while initializing hibernate util.. ");

e.printStackTrace();

}

}

public static void loadSessionFactory(){

Configuration configuration = new Configuration();

configuration.configure("/j2n-hibernate.cfg.xml");

configuration.addAnnotatedClass(Employee.class);

configuration.addAnnotatedClass(Project.class);

ServiceRegistry srvcReg = new StandardServiceRegistryBuilder().applySettings(configuration.getProperties()).build();

sessionFactory = configuration.buildSessionFactory(srvcReg);

}

public static Session getSession() throws HibernateException {

Session retSession=null;

try {

retSession = sessionFactory.openSession();

}catch(Throwable t){

System.err.println("Exception while getting session.. ");

t.printStackTrace();

}

if(retSession == null) {

System.err.println("session is discovered null");

}

return retSession;

}

}

Hibernate Dao Class

?

package com.java2novice.hibernate;

import java.util.List;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.Transaction;

import com.java2novice.model.Employee;

public class EmployeesDao {

public List<Employee> getEmployeeList(){

Session session = null;

List<Employee> empList = null;

try {

session = HibernateUtil.getSession();

String queryStr = "select emp from Employee emp";

Query query = session.createQuery(queryStr);

empList = query.list();

} catch(Exception ex) {

ex.printStackTrace();

// handle exception here

} finally {

try {if(session != null) session.close();} catch(Exception ex) {}

}

return empList;

}

public static void main(String a[]) {

EmployeesDao empDao = new EmployeesDao();

List<Employee> empList = empDao.getEmployeeList();

System.out.println("emp size: "+empList.size());

System.out.println("---------------------------");

empList.stream().forEach(e -> {

System.out.println(e);

System.out.println("\n-- projects assigned to "+e.getName()+" --");

e.getEmpAssignmentList().stream().forEach(System.out::println);

System.out.println("---------------------------");

});

}

}

Output:

emp size: 4

---------------------------

1 | Nataraja G | Documentation | 10000 | 2017-12-16 11:59:59.0

-- projects assigned to Nataraja G --

1 | Ticket Online | Bunty

2 | Disco Drum | Chanty

3 | Traffic Audit | Sonty

---------------------------

2 | Amar M | Entertainment | 12000 | 2017-12-16 11:59:59.0

-- projects assigned to Amar M --

2 | Disco Drum | Chanty

3 | Traffic Audit | Sonty

---------------------------

3 | Nagesh Y | Admin | 25000 | 2017-12-16 11:59:59.0

-- projects assigned to Nagesh Y --

1 | Ticket Online | Bunty

2 | Disco Drum | Chanty

3 | Traffic Audit | Sonty

---------------------------

4 | Vasu V | Security | 2500 | 2017-12-16 12:00:21.0

-- projects assigned to Vasu V --

1 | Ticket Online | Bunty

4.How to prevent dirty checking in hibernate.?

Ans- Dirty Checking  is one of the features of hibernate. In dirty checking, hibernate automatically detects whether    an object is modified (or) not and need to be updated. As long as the object is in persistent state i.e., bound to a particular Session(org.hibernate.Session). Hibernate monitors any changes to the objects and executes sql.  
  
Note:- For dirty checking to work, the object must exist in cache.  
  
Example:-  
I used oracle database and so oracle driver.  
  
**hibernate.cfg.xml**  
  
  
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
                                         "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  
  
  <hibernate-configuration>  
     <session-factory>  
<!-- connection properties-->  
       <property name="hibernate.connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>  
       <property name="hibernate.connection.url">jdbc:oracle:thin:@localhost:1521:serviceName</property>  
       <property name="hibernate.connection.username">userName</property>  
        <property name="hibernate.connection.password">password</property>  
     <!-- limit the connection pool connections-->  
       <property name="hibernate.connection.pool\_size">10</property>  
  
  
<!-- Hibernate properties -->  
      <!-- Dialect class is used to convert hql ddl, dml etc queries to its database dependent queries. This is        developed by database vendors -->  
        <property name="hibernate.dialect">org.hibernate.dialect.OracleDialect</property>  
     <!-- this  property displays the sql query on the console -->  
        <property name="hibernate.show\_sql">true</property>  
<!-- mapping hibernate mapping files -->  
<mapping resource="Gender.hbm.xml"/>  
      </session-factory>  
   </hibernate-configuration>  
  
  
**Gender.hbm.xml**  
  
  
<?xml version="1.0"?>  
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
<hibernate-mapping>  
  <class name="packageName.Gender" table="M\_COM\_GENDER">  
    <id name="id" type="java.lang.Long" column="GNDR\_N\_ID">  
      <generator class="increment"/>  
    </id  
   <property name="name" type="java.lang.String" column="GNDR\_VC\_TYP">  
             
        </property>  
        <property name="status" type="java.lang.String" column="GNDR\_C\_STS">  
             
        </property>  
  </class>  
</hibernate-mapping>  
  
**Gender.java**  
//This is the pojo class.  
  
  
package packageName;  
public class Gender {  
  
 private Long id;  
 private String name;  
 private String status;  
 public Long getId() {  
 return id;  
 }  
 private void setId(Long id) {  
 this.id = id;  
 }  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public String getStatus() {  
 return status;  
 }  
 public void setStatus(String status) {  
 this.status = status;  
 }  
}  
  
**ClientApp.java**  
  
  
package packageName;  
  
  
  
import org.hibernate.cfg.Configuration;  
import org.hibernate.SessionFactory;  
import org.hibernate.Session;  
import org.hibernate.Transaction;  
public class ClientApp {  
  
   
 public static void main(String[] args) {  
 Configuration con = new Configuration();  
 con.configure("com/spry/hibernate/dirtychecking/NewFile.cfg.xml");  
 SessionFactory sf = con.buildSessionFactory();  
 Session session = sf.openSession();  
 Transaction trans = session.beginTransaction();  
 try  
 {  
                       //loading a gender object of genderId = 1  
 Gender gender = (Gender)session.get(Gender.class, new Long(1));  
                       //print the gender naem  
 System.out.println("gender.getName()=======>"+gender.getName());  
                          //modify the gender name  
 gender.setName("female");  
 session.getTransaction().commit();  
 session.flush();  
   
 }catch(Exception e)  
 {  
 e.printStackTrace();  
 }finally  
 {  
 session.close();  
 }  
  
 }  
  
}  
  
In the above client application, gender object is loaded first. and then object state is changed by altering the name  of the object. Here, we are not calling update() , even though object state is written to the database. This is called automatic dirty checking. Hibernate monitors whether any changes in the session objects and automatically synchronizes them to database.  
session.getTransaction.commit() is mandatory. Because, after any modifications, if commit() is not called then data will not be modified in the database eventhough update query is executed.  
  
This also works for collections.

5.What’s wrong with using HashMap’s in a multi-threaded environment? When does a get() method go into an infinite loop?

Ans- As you may know that how the HashMap works and the working of ***hashing***function. For every hashmap there is a property called ***load factor*** which is by default set to 0.75, what that means is, if the bucket list is 75% of the capacity full,***rehashing*** of the hashmap is done with new bucket with double the capacity and the values from the old bucket are then added to the new bucket.

For eg. if hashmap with capacity of 16 and load factor of 0.75, it means that hashmap will rehash and double the size of the bucket when 13th key-value pair is added ( 16\*0.75 = 12 ).

Few important points to note when ***rehashing*** is done are:

1. A new bucket list is created with double the size.
2. Key-Value pairs in old bucket is transferred to the new bucket.
3. Key-Value pairs are added to the new bucket in reverse order.  
   For eg. If in old bucket the list at 0th index is 10->12->15 then in the new bucket list at 0th index will be 15->12->10. (I have assumed the example, in real scenario it will be key value pairs). *It does this for a reason to avoid traversing the list and to keep the insertion time to constant.*

Now coming to the concurrent access of the HashMap.

Assuming there are 2 threads ***ThreadA*** and ***ThreadB***and both of them are accessing the same HashMap with 12 elements in it.

When both threads ThreadA and ThreadB want to add a new Key-Value pair in the hashmap they notice that the loadfactor of the hashmap is reached and rehashing has to be done to add new pairs (Note that both the threads are accessing it concurrently and both want to add a new distinct pair to it).

Take the example again in old bucket the nodes are arranged in the following order: 10->12->**null**  When rehashing is done the order is reversed according to point 3.

Suppose ThreadA has the cpu it sees that loadfactor is reached and calls for rehashing it creates a new bucket with double the capacity and now it is ready for the transferring of the nodes, it creates two pointer assume the pointer names to be current and **next** where current pointer will point to the node being transferred and next points to the next node to be transferred. In the above scenario current = 10 & **next**=12 . But ThreadA looses the CPU time, remember that context switching occurs and ThreadB gets the chance.

Fortunately ***ThreadB*** gets the complete CPU time to complete the transferring process now the node list in the new bucket will appear like this 12->10->**null**

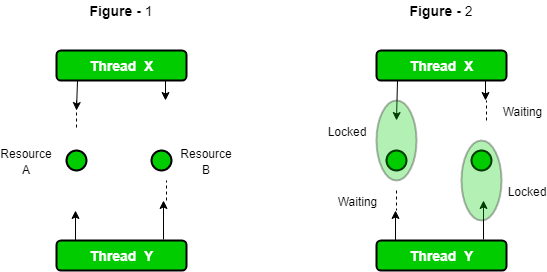
Now ***ThreadA*** gets the chance the **context** of ***ThreadA*** is loaded, now remember that before giving control to ***ThreadB*** the current and **next** pointer were pointing to node 10 and 12 respectively. Now when the ***ThreadA***continues the transferring process ***ThreadA*** adds the node pointed by the current pointer i.e node 10 to the bucket list now the current will point to node pointed by **next** i.e node 12 and **next** will point to the next node pointed by the node 12 i.e node 10 as the changes done by the ***ThreadB***.

Hence any **get()/put()** call will end up in infinite loop as ***ThreadB*** has reversed the order.

6.one class has get and put method and one counter which increment input and get return counter value.

7.What is deadlock and write deadlock code example.

Ans-[**synchronized**](http://quiz.geeksforgeeks.org/synchronized-in-java/) keyword is used to make the class or method thread-safe which means only one thread can have lock of synchronized method and use it, other threads have to wait till the lock releases and anyone of them acquire that lock.  
It is important to use if our program is running in multi-threaded environment where two or more threads execute simultaneously. But sometimes it also causes a problem which is called [**Deadlock**](http://quiz.geeksforgeeks.org/operating-system-process-management-deadlock-introduction/). Below is a simple example of Deadlock condition.



filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // Java program to illustrate Deadlock  // in multithreading.  class Util  {      // Util class to sleep a thread      static void sleep(long millis)      {          try          {              Thread.sleep(millis);          }          catch (InterruptedException e)          {              e.printStackTrace();          }      }  }    // This class is shared by both threads  class Shared  {      // first synchronized method      synchronized void test1(Shared s2)      {          System.out.println("test1-begin");          Util.sleep(1000);            // taking object lock of s2 enters          // into test2 method          s2.test2(this);          System.out.println("test1-end");      }        // second synchronized method      synchronized void test2(Shared s1)      {          System.out.println("test2-begin");          Util.sleep(1000);            // taking object lock of s1 enters          // into test1 method          s1.test1(this);          System.out.println("test2-end");      }  }      class Thread1 extends Thread  {      private Shared s1;      private Shared s2;        // constructor to initialize fields      public Thread1(Shared s1, Shared s2)      {          this.s1 = s1;          this.s2 = s2;      }        // run method to start a thread      @Override      public void run()      {          // taking object lock of s1 enters          // into test1 method          s1.test1(s2);      }  }      class Thread2 extends Thread  {      private Shared s1;      private Shared s2;        // constructor to initialize fields      public Thread2(Shared s1, Shared s2)      {          this.s1 = s1;          this.s2 = s2;      }        // run method to start a thread      @Override      public void run()      {          // taking object lock of s2          // enters into test2 method          s2.test2(s1);      }  }      public class GFG  {      public static void main(String[] args)      {          // creating one object          Shared s1 = new Shared();            // creating second object          Shared s2 = new Shared();            // creating first thread and starting it          Thread1 t1 = new Thread1(s1, s2);          t1.start();            // creating second thread and starting it          Thread2 t2 = new Thread2(s1, s2);          t2.start();            // sleeping main thread          Util.sleep(2000);      }  } |

Output : test1-begin

test2-begin

It is not recommended to run the above program with online IDE. We can copy the source code and run it on our local machine. We can see that it runs for indefinite time, because threads are in deadlock condition and doesn’t let code to execute. Now let’s see step by step what is happening there.

1. Thread t1 starts and calls test1 method by taking the object lock of s1.
2. Thread t2 starts and calls test2 method by taking the object lock of s2.
3. t1 prints test1-begin and t2 prints test-2 begin and both waits for 1 second, so that both threads can be started if any of them is not.
4. t1 tries to take object lock of s2 and call method test2 but as it is already acquired by t2 so it waits till it become free. It will not release lock of s1 until it gets lock of s2.
5. Same happens with t2. It tries to take object lock of s1 and call method test1 but it is already acquired by t1, so it has to wait till t1 release the lock. t2 will also not release lock of s2 until it gets lock of s1.
6. Now, both threads are in wait state, waiting for each other to release locks. Now there is a race around condition that who will release the lock first.
7. As none of them is ready to release lock, so this is the Dead Lock condition.
8. When you will run this program, it will be look like execution is paused.

**Detect Dead Lock condition**

We can also detect deadlock by running this program on cmd. We have to collect Thread Dump. Command to collect depends on OS type. If we are using Windows and Java 8, command is jcmd $PID Thread.print  
We can get PID by running jps command. Thread dump for above program is below:

5524:

2017-04-21 09:57:39

Full thread dump Java HotSpot(TM) 64-Bit Server VM (25.25-b02 mixed mode):

"DestroyJavaVM" #12 prio=5 os\_prio=0 tid=0x0000000002690800 nid=0xba8 waiting on condition [0x0000000000000000]

java.lang.Thread.State: RUNNABLE

"Thread-1" #11 prio=5 os\_prio=0 tid=0x0000000018bbf800 nid=0x12bc waiting for monitor entry [0x000000001937f000]

java.lang.Thread.State: BLOCKED (on object monitor)

at Shared.test1(GFG.java:15)

- waiting to lock (a Shared)

at Shared.test2(GFG.java:29)

- locked (a Shared)

at Thread2.run(GFG.java:68)

"Thread-0" #10 prio=5 os\_prio=0 tid=0x0000000018bbc000 nid=0x1d8 waiting for monitor entry [0x000000001927f000]

java.lang.Thread.State: BLOCKED (on object monitor)

at Shared.test2(GFG.java:25)

- waiting to lock (a Shared)

at Shared.test1(GFG.java:19)

- locked (a Shared)

at Thread1.run(GFG.java:49)

"Service Thread" #9 daemon prio=9 os\_prio=0 tid=0x000000001737d800 nid=0x1680 runnable [0x0000000000000000]

java.lang.Thread.State: RUNNABLE

"C1 CompilerThread2" #8 daemon prio=9 os\_prio=2 tid=0x000000001732b800 nid=0x17b0 waiting on condition [0x0000000000000000]

java.lang.Thread.State: RUNNABLE

"C2 CompilerThread1" #7 daemon prio=9 os\_prio=2 tid=0x0000000017320800 nid=0x7b4 waiting on condition [0x0000000000000000]

java.lang.Thread.State: RUNNABLE

"C2 CompilerThread0" #6 daemon prio=9 os\_prio=2 tid=0x000000001731b000 nid=0x21b0 waiting on condition [0x0000000000000000]

java.lang.Thread.State: RUNNABLE

"Attach Listener" #5 daemon prio=5 os\_prio=2 tid=0x0000000017319800 nid=0x1294 waiting on condition [0x0000000000000000]

java.lang.Thread.State: RUNNABLE

"Signal Dispatcher" #4 daemon prio=9 os\_prio=2 tid=0x0000000017318000 nid=0x1efc runnable [0x0000000000000000]

java.lang.Thread.State: RUNNABLE

"Finalizer" #3 daemon prio=8 os\_prio=1 tid=0x0000000002781800 nid=0x5a0 in Object.wait() [0x000000001867f000]

java.lang.Thread.State: WAITING (on object monitor)

at java.lang.Object.wait(Native Method)

- waiting on (a java.lang.ref.ReferenceQueue$Lock)

at java.lang.ref.ReferenceQueue.remove(Unknown Source)

- locked (a java.lang.ref.ReferenceQueue$Lock)

at java.lang.ref.ReferenceQueue.remove(Unknown Source)

at java.lang.ref.Finalizer$FinalizerThread.run(Unknown Source)

"Reference Handler" #2 daemon prio=10 os\_prio=2 tid=0x000000000277a800 nid=0x15b4 in Object.wait() [0x000000001857f000]

java.lang.Thread.State: WAITING (on object monitor)

at java.lang.Object.wait(Native Method)

- waiting on (a java.lang.ref.Reference$Lock)

at java.lang.Object.wait(Unknown Source)

at java.lang.ref.Reference$ReferenceHandler.run(Unknown Source)

- locked (a java.lang.ref.Reference$Lock)

"VM Thread" os\_prio=2 tid=0x00000000172e6000 nid=0x1fec runnable

"GC task thread#0 (ParallelGC)" os\_prio=0 tid=0x00000000026a6000 nid=0x21fc runnable

"GC task thread#1 (ParallelGC)" os\_prio=0 tid=0x00000000026a7800 nid=0x2110 runnable

"GC task thread#2 (ParallelGC)" os\_prio=0 tid=0x00000000026a9000 nid=0xc54 runnable

"GC task thread#3 (ParallelGC)" os\_prio=0 tid=0x00000000026ab800 nid=0x704 runnable

"VM Periodic Task Thread" os\_prio=2 tid=0x0000000018ba0800 nid=0x610 waiting on condition

JNI global references: 6

Found one Java-level deadlock:

=============================

"Thread-1":

waiting to lock monitor 0x0000000018bc1e88 (object 0x00000000d5d645a0, a Shared),

which is held by "Thread-0"

"Thread-0":

waiting to lock monitor 0x0000000002780e88 (object 0x00000000d5d645b0, a Shared),

which is held by "Thread-1"

Java stack information for the threads listed above:

===================================================

"Thread-1":

at Shared.test1(GFG.java:15)

- waiting to lock (a Shared)

at Shared.test2(GFG.java:29)

- locked (a Shared)

at Thread2.run(GFG.java:68)

"Thread-0":

at Shared.test2(GFG.java:25)

- waiting to lock (a Shared)

at Shared.test1(GFG.java:19)

- locked (a Shared)

at Thread1.run(GFG.java:49)

Found 1 deadlock.

As we can see there is clearly mentioned that found 1 deadlock. It is possible that the same message appears when you try on your machine.

**Avoid Dead Lock condition**

We can avoid dead lock condition by knowing its possibilities. It’s a very complex process and not easy to catch. But still if we try, we can avoid this. There are some methods by which we can avoid this condition. We can’t completely remove its possibility but we can reduce.

* **Avoid Nested Locks :**This is the main reason for dead lock. Dead Lock mainly happens when we give locks to multiple threads. Avoid giving lock to multiple threads if we already have given to one.
* **Avoid Unnecessary Locks :**We should have lock only those members which are required. Having lock on unnecessarily can lead to dead lock.
* **Using thread join :**Dead lock condition appears when one thread is waiting other to finish. If this condition occurs we can use Thread.join with maximum time you think the execution will take.

**Important Points :**

* If threads are waiting for each other to finish, then the condition is known as Deadlock.
* Deadlock condition is a complex condition which occurs only in case of multiple threads.
* Deadlock condition can break our code at run time and can destroy business logic.
* We should avoid this condition as much as we can.

8.What is live-lock?

Ans-.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A livelock is a recursive situation where two or more threads would keep repeating a particular code logic. The intended logic is typically giving opportunity to the other threads to proceed in favor of 'this' thread.  A real-world example of livelock occurs when two people meet in a narrow corridor, and each tries to be polite by moving aside to let the other pass, but they end up swaying from side to side without making any progress because they both repeatedly move the same way at the same time.  [From Oracle reference docs:](https://docs.oracle.com/javase/tutorial/essential/concurrency/starvelive.html)   A thread often acts in response to the action of another thread. If the other thread's action is also a response to the action of another thread, then livelock may result. As with deadlock, livelocked threads are unable to make further progress. However, the threads are not blocked - they are simply too busy responding to each other to resume work.  For example consider a situation where two threads want to access a shared common resource via a Worker object but when they see that other Worker (invoked on another thread) is also 'active', they attempt to hand over the resource to other worker and wait for it to finish. If initially we make both workers active they will suffer from livelock.  main The Common Resource Class public class CommonResource {  private Worker owner;  public CommonResource (Worker d) {  owner = d;  }  public Worker getOwner () {  return owner;  }  public synchronized void setOwner (Worker d) {  owner = d;  }  }   The Worker Class public class Worker {  private String name;  private boolean active;  public Worker (String name, boolean active) {  this.name = name;  this.active = active;  }  public String getName () {  return name;  }  public boolean isActive () {  return active;  }  public synchronized void work (CommonResource commonResource, Worker otherWorker) {  while (active) {  // wait for the resource to become available.  if (commonResource.getOwner() != this) {  try {  wait(10);  } catch (InterruptedException e) {  //ignore  }  continue;  }  // If other worker is also active let it do it's work first  if (otherWorker.isActive()) {  System.out.println(getName() +  " : handover the resource to the worker " +  otherWorker.getName());  commonResource.setOwner(otherWorker);  continue;  }  //now use the commonResource  System.out.println(getName() + ": working on the common resource");  active = false;  commonResource.setOwner(otherWorker);  }  }  }   The main class public class Livelock {  public static void main (String[] args) {  final Worker worker1 = new Worker("Worker 1 ", true);  final Worker worker2 = new Worker("Worker 2", true);  final CommonResource s = new CommonResource(worker1);  new Thread(() -> {  worker1.work(s, worker2);  }).start();  new Thread(() -> {  worker2.work(s, worker1);  }).start();  }  }   Output: There will be never ending recursion of the following output:  Worker 1 : handing over the resource to the worker: Worker 2  Worker 2 : handing over the resource to the worker: Worker 1  Worker 1 : handing over the resource to the worker: Worker 2  Worker 2 : handing over the resource to the worker: Worker 1  Worker 1 : handing over the resource to the worker: Worker 2  Worker 2 : handing over the resource to the worker: Worker 1  ........   Avoiding Livelock In above example we can fix the issue by processing the common resource sequentially rather than in different threads simultaneously.  Just like deadlock, there's no general guideline to avoid livelock, but we have to be careful in scenarios where we change the state of common objects also being used by other threads, for example in above scenario. the Worker object. Example Project Dependencies and Technologies Used:   * JDK 1.8 * Maven 3.0.4 | [[https://www.logicbig.com/images/amazon.png](https://www.amazon.com/seller/logicbig?&_encoding=UTF8&tag=logicbig-20&linkCode=ur2&linkId=2901371988cbe90d960ccdd6e8dac800&camp=1789&creative=9325)](https://www.amazon.com/seller/logicbig?&_encoding=UTF8&tag=logicbig-20&linkCode=ur2&linkId=2901371988cbe90d960ccdd6e8dac800&camp=1789&creative=9325" \t "_blank)  [[https://www.logicbig.com/service/amzn-cache/1.jpghttps://www.logicbig.com/service/amzn-cache/2.jpghttps://www.logicbig.com/service/amzn-cache/3.jpghttps://www.logicbig.com/service/amzn-cache/4.jpg](https://www.amazon.com/seller/logicbig?&_encoding=UTF8&tag=logicbig-20&linkCode=ur2&linkId=2901371988cbe90d960ccdd6e8dac800&camp=1789&creative=9325)](https://www.amazon.com/seller/logicbig?&_encoding=UTF8&tag=logicbig-20&linkCode=ur2&linkId=2901371988cbe90d960ccdd6e8dac800&camp=1789&creative=9325" \t "_blank)  https://ir-na.amazon-adsystem.com/e/ir?t=logicbig-20&l=ur2&o=1  [https://assets-auctionnudge.s3.amazonaws.com/feed/item/img/rnoe-166x96.png](https://www.auctionnudge.com/to_ebay/home/site_id/0/user_id/39261/tool_name/item)   |  |  |  | | --- | --- | --- | | [**Dell Latitude E5450 Ultrabook i7-5600U (2.6GHz-3.2GHz) 16GB RAM 256GB SSD Win 10**](https://www.auctionnudge.com/to_ebay/item/site_id/0/user_id/39261/tool_name/item/item_hash/ldo7ado9ld)   |  |  | | --- | --- | | [https://thumbs2.ebaystatic.com/m/mig5brlCoL6UNOap0z1mGgA/140.jpg](https://www.auctionnudge.com/to_ebay/item/site_id/0/user_id/39261/tool_name/item/item_hash/ldo7ado9ld) | **$400.00** (Buy It Now) Time Left: 23d 15h 4m [View Item »](https://www.auctionnudge.com/to_ebay/item/site_id/0/user_id/39261/tool_name/item/item_hash/ldo7ado9ld) | | | [**Dell OptiPlex 3020 SFF Intel Quad i5-4570 3.20-3.60GHz 8GB RAM 500GB HDD**](https://www.auctionnudge.com/to_ebay/item/site_id/0/user_id/39261/tool_name/item/item_hash/ldo6al7heo)   |  |  | | --- | --- | | [https://thumbs1.ebaystatic.com/m/mRaQ1NDg7ShF1h_3kc8VvfQ/140.jpg](https://www.auctionnudge.com/to_ebay/item/site_id/0/user_id/39261/tool_name/item/item_hash/ldo6al7heo) | **$185.00** (Buy It Now) Time Left: 28d 3h 49m [View Item »](https://www.auctionnudge.com/to_ebay/item/site_id/0/user_id/39261/tool_name/item/item_hash/ldo6al7heo) | | | [**CUK MSI GP72 Leopard Pro 17.3-inch i7-5700HQ 16GB 250GB SSD NVIDIA GTX 950M 2GB**](https://www.auctionnudge.com/to_ebay/item/site_id/0/user_id/39261/tool_name/item/item_hash/ldo664o67o)   |  |  | | --- | --- | | [https://thumbs1.ebaystatic.com/m/mAoj_-doFgjQPI-RYHiiKYg/140.jpg](https://www.auctionnudge.com/to_ebay/item/site_id/0/user_id/39261/tool_name/item/item_hash/ldo664o67o) | **$590.00** (Buy It Now) Time Left: 22d 10h 24m [View Item »](https://www.auctionnudge.com/to_ebay/item/site_id/0/user_id/39261/tool_name/item/item_hash/ldo664o67o) | |   [eBay Listings from Auction Nudge](https://www.auctionnudge.com/tools/your-ebay-items) Java 11 Tutorials  * [Java 11 Features](https://www.logicbig.com/tutorials/core-java-tutorial/java-11-changes.html)  Java 10 Tutorials  * [Java 10 Features](https://www.logicbig.com/tutorials/core-java-tutorial/java-10-changes.html)  Java 9 Tutorials  * [Java 9 Module System](https://www.logicbig.com/tutorials/core-java-tutorial/modules.html) * [Java 9 Misc Features](https://www.logicbig.com/tutorials/core-java-tutorial/java-9-changes.html) |
| |  |  | | --- | --- | | ui-buttonui-button Thread Livelock Example | [https://www.logicbig.com/images/view_fullscreen.png](javascript:showCodeFullScreen(1);) [Select All [https://www.logicbig.com/images/select_all.png](javascript:selectAllCode(1);)](javascript:selectAllCode(1);) [Download [https://www.logicbig.com/images/download.png](https://www.logicbig.com/tutorials/core-java-tutorial/java-multi-threading/thread-livelock/thread-livelock-example.zip)](https://www.logicbig.com/tutorials/core-java-tutorial/java-multi-threading/thread-livelock/thread-livelock-example.zip) |  * java-thread-livelock   + src     - main       * java         + com   logicbig  example  **CommonResource.java**  **Livelock.java**  **Worker.java**   * + **pom.xml**   package com.logicbig.example;  public class Livelock {  public static void main (String[] args) {  final Worker worker1 = new Worker("Worker 1 ", true);  final Worker worker2 = new Worker("Worker 2", true);  final CommonResource s = new CommonResource(worker1);  new Thread(() -> {  worker1.work(s, worker2);  }).start();  new Thread(() -> {  worker2.work(s, worker1);  }).start();  }  } | |

9.What is executor framework? Why is it needed why not creating a single thread?

Ans- Threads provide a multitasking ability to a process (process = program in execution). A program can have multiple threads; each of them provide a unit of control as one of its strands. Single threaded programs execute in a monotonous, predictable manner. But, a multi-threaded program brings out the essence of concurrency or simultaneous execution of program instruction where a subset of code executes or is supposed to execute in parallel mode. This mechanism leverages performance, especially because modern processing workhorses are multi core. So, running a single threaded process that may utilize only one CPU core is simply a waste of resources.

[Java](http://www.developer.com/java) core's APIs includes a framework called Executors Framework, which provides some relief to the programmer when working in a multi-threaded arena. This article mainly focuses on the framework and its uses with a little background idea to begin with.

## Parallel Execution

Parallel execution requires some hardware assistance, and a threaded program that brings out the essence of parallel processing is no exception. Multi-threaded programs can best utilize multiple CPU cores found in modern machines, resulting in manifold performance boost. But, the problem is that maximum utilization of multiple cores requires a program's code to be written with parallel logic from the ground up. Practically, this is easier said than done. In dealing with simultaneous operations where everything is seemingly multiple, problems and challenges are also multi-faceted. Some logics are meant to be parallel whereas some are very linear. The biggest problem is to balance between them yet keep up with maximal utilization of processing resources. Parallel logic is inherently parallel, whose implementation is pretty straightforward, but converting a semi-linear logic into an optimal parallel code can be a daunting task. For example, the solution of 2 + 2 = 4 is quite linear but the logic to solve expression such as (2 x 4) + (5 / 2) can be leveraged with parallel implementation.

## Implementation Primitive

There are many aspects to be considered before modeling a program for multi-threaded implementation. Some basic questions to ask while modeling one are:

* How is the thread to be created and submitted for execution?
* Is dependency involved in the thread for successful execution?
* Does it imply synchronous or asynchronous execution?
* How do you find out whether the error cropped up from a thread in execution?
* Who executes the thread?
* How to get the feedback from the thread after execution is complete?

When creating a task (task = individual unit of work), what we normally do is either implement an interface called Runnable or extend the Thread class:

**public class** SampleTask **implements** Runnable {

//...

**public void** run(){

//...

}

}

And, create the task as follows:

SampleTask st1=**new** SampleTask();

SampleTask st2=**new** SampleTask();

And then execute each task as follows:

Thread t1=**new** Thread(st1);

Thread t2=**new** Thread(st2);

t1.start();

t2.start();

To get a feedback from individual task, we have to write additional code. But, the point is that there are too many intricacies involved in managing a thread execution, such as creation and destruction of a thread, has a direct bearing on the overall time required to start another task. If it is not performed gracefully, unnecessary delay in the start of a task is certain. A thread consumes resources, so multiple threads may consume multiple resources. This has a propensity to slack overall CPU performance; worse, it can crash the system if the number of threads exceeds the permitted limit of the underlying platform. It also may happen that some thread consumes most of the resources leaving other threads starved, or a typical race condition. So, the complexity involved in managing thread execution is easily intelligible.

## Executor Framework

The Executor Framework attempts to address this problem and bring some controlling attributes. The predominant aspect of this framework is to state a clear demarcation between the task submission from task execution. The executor says, create your task and submit it to me; I'll take care of the rest (execution details). The mechanics of this demarcation is attributed to the interface called Executorunder the java.util.concurrent package. Rather than creating thread explicitly, the code above can be written as:

Executor executor=Executors.newFixedThreadPool(5);

and then

executor.execute(**new** SampleTask()); // or executor.execute(st1);

executor.execute(**new** SampleTask()); // or executor.execute(st2);

Calling the executor method does not ensure that the thread execution is initiated; instead, it merely refers to a submission of a task. The executor takes up the responsibility on behalf, including the details about the policies to adhere to in the course of execution. The class library supplied by the executor framework determines the policy, which, however, is configurable.

There are many static methods available with the Executors class (Note that Executor is an interface and Executors is a class. Both included in the package java.util.concurrent). A few of the commonly used are as follows:

* **newCachedThreadPool():** It creates new thread as needed, but is quick to reuse any available thread previously constructed. The thread pool can shrink and expand, depending upon the workload.
* **newFixedThreadPool(int nThreads):** The thread pool created by this method has a fixed size set by the parameter passed. At any given time, there can be maximum of nThread (number of threads) in the pool.
* **newSingleThreadExecutor():** This method ensures that there will be only one thread to execute all the tasks. If this thread dies unexpectedly, a new one is created. But, there is a guarantee that there will be a single thread at any given time.

All of these methods return an ExecutorService object.

The ExecutorService interface extends Executor and provides necessary methods to manage execution of threads, such as the shutdown () method to initiate an orderly shutdown of threads. There is another interface, called ScheduledExecutorService, which extends ExecutorService to support scheduling of threads.

## A Quick Example

Let's create a very simple program to understand the use of an executor.

**package** org.mano.example;

**public class** MyTask **implements** Runnable {

**private int** id;

**private int** counter;

**public** MyTask(**int** id, **int** counter) {

**this**.id = id;

**this**.counter = counter;

}

@Override

**public void** run() {

**for** (**int** i = 0; i < counter; i++) {

**try** {

System.out.println("Task ID: " +

id + " Iter No: " + i);

Thread.sleep(1000);

} **catch** (Exception ex) {

System.out.println("Task ID: " +

id + " is interrupted.");

**break**;

}

}

}

}

**package** org.mano.example;

**import** java.util.concurrent.ExecutorService;

**import** java.util.concurrent.Executors;

**public class** TestMyTask {

**public static void** main(String[] args) {

ExecutorService executorService =

Executors.newFixedThreadPool(2);

**for** (**int** i = 0; i < 3; i++)

executorService.submit(**new** MyTask(i, 5));

executorService.shutdown();

}

}

10.How to achieve thread safety in multi-threading?

Ans- Thread safety is a computer programming concept applicable in the context of multi-threaded programs. A piece of code is thread-safe if it functions correctly during simultaneous execution by multiple threads. In particular, it must satisfy the need for multiple threads to access the same shared data, and the need for a shared piece of data to be accessed by only one thread at any given time.

There are a few ways to achieve thread safety:

**Re-entrancy:**

Writing code in such a way that it can be partially executed by one task, reentered by another task, and then resumed from the original task. This requires the saving of state information in variables local to each task, usually on its stack, instead of in static or global variables.

**Mutual exclusion:**

Access to shared data is serialized using mechanisms that ensure only one thread reads or writes the shared data at any time. Great care is required if a piece of code accesses multiple shared pieces of data—problems include race conditions, deadlocks, livelocks, starvation, and various other ills enumerated in many operating systems textbooks.

**Thread-local storage:**

Variables are localized so that each thread has its own private copy. These variables retain their values across subroutine and other code boundaries, and are thread-safe since they are local to each thread, even though the code which accesses them might be reentrant.

**Atomic operations:**

Shared data are accessed by using atomic operations which cannot be interrupted by other threads. This usually requires using special machine language instructions, which might be available in a runtime library. Since the operations are atomic, the shared data are always kept in a valid state, no matter what other threads access it. Atomic operations form the basis of many thread locking mechanisms.

11.How do you think the user threads / Hits will be managed in a shopping site like eBay ? Will that be synchronized? If yes, how it is concurrently managed?

Ans-

13.What is the Data structure used for fixed key-size kind of HashMap?

Ans-open link--- <https://adrianmejia.com/blog/2018/04/28/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/>

13.Explain design of web application?

### Ans- **A Definition of Web Application Architecture**

Web application architecture defines the interactions between applications, [middleware](http://searchmicroservices.techtarget.com/definition/middleware) systems and databases to ensure multiple applications can work together. When a user types in a URL and taps “Go,” the browser will find the Internet-facing computer the website lives on and requests that particular page.

The server then responds by sending files over to the browser. After that action, the browser executes those files to show the requested page to the user. Now, the user gets to interact with the website. Of course, all of these actions are executed within a matter of seconds. Otherwise, users wouldn’t bother with websites.

What’s important here is the code, which has been parsed by the browser. This very code may or may not have specific instructions telling the browser how to react to a wide swath of inputs. As a result, web application architecture includes all sub-components and external applications interchanges for an entire software application.

Of course, it is designed to function efficiently while meeting its specific needs and goals. Web application architecture is critical since the majority of global network traffic, and every single app and device uses web-based communication. It deals with scale, efficiency, robustness, and security.

### **How Web Application Architecture Works**

With web applications, you have the [server vs. the client side](http://www.codeconquest.com/website/client-side-vs-server-side/). In essence, there are two programs running concurrently:

* The code which lives in the browser and responds to user input
* The code which lives on the server and responds to [HTTP requests](https://www.w3schools.com/tags/ref_httpmethods.asp)

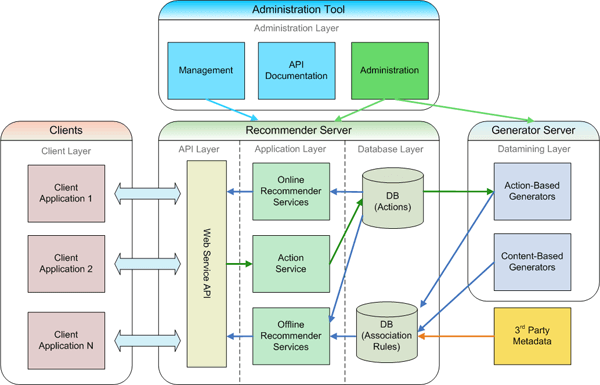


Image via [*Wikipedia*](https://en.wikipedia.org/wiki/Easyrec)

When writing an app, it is up to the web developer to decide what the code on the server should do in relation to what the code on the browser should do. With server-side code, languages include:

* Ruby on Rails
* PHP
* C#
* Java
* Python
* Javascript

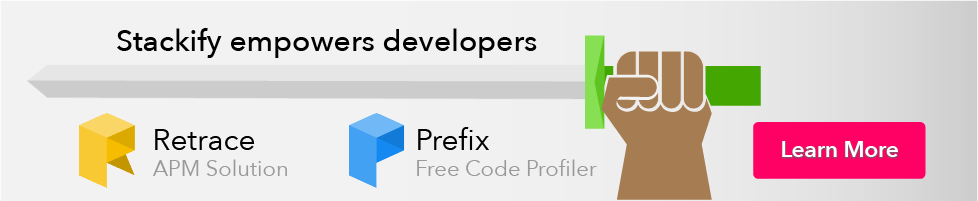
In fact, any code that can respond to HTTP requests has the capability to run on a server. Here are a few other attributes of server-side code:

* Is never seen by the user (except within a rare malfunction)
* Stores data such as user profiles, tweets, pages, etc…
* Creates the page the user requested

With client-side code, languages used include:

* CSS
* Javascript
* HTML

These are then parsed by the user’s browser. Moreover, client-side code can be seen and edited by the user. Plus, it has to communicate only through HTTP requests and cannot read files off of a server directly. Furthermore, it reacts to user input.

[](https://info.stackify.com/cs/c/?cta_guid=eb8ba9dc-2898-4c16-a893-432ae7a086a9&placement_guid=c00f5072-32fb-4332-a4d7-60765e6305d0&portal_id=207384&canon=https%3A%2F%2Fstackify.com%2Fweb-application-architecture%2F&redirect_url=APefjpFrKvqwB1A47MOyyuukd4wnvsjUjFVSGmva0Cfxx-oe_rQQ8Xp6Vg1xA29ZqwZi_OjpkNQCHdCnXSyFrFP2-5TjhBQOduFMjLU6nhV3C8HcNscBDhZq5JwMw1Utiw5TAT9qnsZEHJpo-vS4HshdI-LfcYkdoE7Bb0XIJ5k8Gl-72Ymv8oRcxtYiOlb91OxY5LflfwnSD3vrElRWlHZgUK_uhrpe_w&click=9f05cc83-f32c-428a-a38d-9f8144a0554e&hsutk=6f966995b06d2ba630261d647cca8a1a&utm_referrer=https%3A%2F%2Fwww.google.com%2F&__hstc=23835621.6f966995b06d2ba630261d647cca8a1a.1548171350732.1548171350732.1553373814889.2&__hssc=23835621.1.1553373814889&__hsfp=585454406)

### **Web Application Architecture is Important for Supporting Future Growth**

The reason why it is imperative to have good web application architecture is because it is the blueprint for supporting future growth which may come from increased demand, future interoperability and enhanced reliability requirements. Through [object-oriented programming](https://docs.oracle.com/javase/tutorial/java/concepts/), the organizational design of web application architecture defines precisely how an application will function. Some features include:

* Delivering persistent data through HTTP, which can be understood by client-side code and vice-versa
* Making sure requests contain valid data
* Offers authentication for users
* Limits what users can see based on permissions
* Creates, updates and deletes records

### **Trends in Web Application Architecture**

As technology continues to evolve, so does web application architecture. One such trend is the use of and creation of [service-oriented architecture](https://msdn.microsoft.com/en-us/library/aa480021.aspx). This is where most of the code for the entire application exists as services. In addition, each has its own HTTP API. As a result, one facet of the code can make a request to another part of the code–which may be running on a different server.

Another trend is a single-page application. This is where web UI is presented through a rich JavaScript application. It then stays in the user’s browser over a variety of interactions. In terms of requests, it uses AJAX or WebSockets for performing asynchronous or synchronous requests to the web server without having to load the page.

The user then gets a more natural experience with limited page load interruptions. At their core, many web applications are built around objects. The objects are stored in tables via an SQL database. Each row in a table has a particular record. So, with relational databases, it is all about relations. You can call on records just by listing the row and column for a target data point.

With the two above trends, web apps are now much better suited for viewing on multiple platforms and multiple devices. Even when most of the code for the apps remain the same, they can still be viewed clearly and easily on a smaller screen.

### **Best Practices for Good Web Application Architecture**

You may have a working app, but it also needs to have good web architecture. Here are several attributes necessary for good web application architecture:

* Solves problems consistently and uniformly
* Is as simple as possible
* Supports the latest standards include A/B testing and analytics
* Offers fast response times
* Utilizes security standards to reduce the chance of malicious penetrations
* Does not crash
* Heals itself
* Does not have a single point of failure
* Scales out easily
* Allows for easy creation of known data
* Errors logged in a user-friendly way
* Automated deployments

14.How can you prevent cloning of an Object?

Ans- **The only way to restrict the object cloning of a class. Follow singleton design pattern and override the clone method in which we through an exception. Following is the sample code to follow singleton design pattern and restricting the object cloning.   
  
  
public class Singleton   
{   
//Declaring singleton field as static field   
//This field has static nature and will shared among all   
//instance of this class.   
private static Singleton singleton=null;   
  
//restrict object creation to this class only   
// Private access modifier play the critical role   
// to restrict the access of the constructor to the outsider.   
private Singleton()   
{   
}   
  
// Obtain instance of Singleton class with the help of   
// Singleton.getInstance() method.   
public static Singleton getInstance()   
{   
if(singleton ==null)   
{   
singleton=new Singleton();   
}   
return singleton;   
}   
  
// additional stuff ...   
}   
  
  
  
Restrict the cloning of singleton object by overriding the clone() method of object class   
/\*   
\* Object.clone() is protected methods but it is safe to override and   
\*restrict the cloning explicitly.   
\* Restrict the cloning by overriding clone method of the object class.   
\*/   
@Override   
protected Object clone() throws CloneNotSupportedException   
{   
throw new CloneNotSupportedException();**

15.How to generate unique ID in Java in distributed machines(JVMs).

Ans- When identifiers are used solely within a database, their generation should be left to the database itself. (See [Statement.getGeneratedKeys](https://docs.oracle.com/javase/10/docs/api/java/sql/Statement.html#getGeneratedKeys()).)

Unique identifiers which are "published" in some way may need special treatment, since the identifier may need to be difficult to guess or forge. A typical example is the value of a cookie used as a session identifier - simply using a series of consecutive integers is generally unacceptable, since one user could easily impersonate another by altering the value of the cookie to some nearby integer.

### Style 1 - UUID

Starting with Java 5, the [UUID](https://docs.oracle.com/javase/10/docs/api/java/util/UUID.html) class provides a simple means for generating unique ids. The identifiers generated by UUID are actually universally unique identifiers.

**Example**

**import** java.util.UUID;

**public** **class** GenerateUUID {

**public** **static** **final** **void** main(String... args){

//generate random UUIDs

UUID idOne = UUID.randomUUID();

UUID idTwo = UUID.randomUUID();

log("UUID One: " + idOne);

log("UUID Two: " + idTwo);

}

**private** **static** **void** log(Object object){

System.out.println( String.valueOf(object) );

}

}

Example run:

>java -cp . GenerateUUID

UUID One: 067e6162-3b6f-4ae2-a171-2470b63dff00

UUID Two: 54947df8-0e9e-4471-a2f9-9af509fb5889

**If Java 5 is not available**, then there are other more laborious ways to generate unique ids (see below).

### Style 2 - SecureRandom and MessageDigest

The following method uses SecureRandom and MessageDigest:

* upon startup, initialize [SecureRandom](https://docs.oracle.com/javase/10/docs/api/java/security/SecureRandom.html) (this may be a lengthy operation)
* when a new identifier is needed, generate a random number using SecureRandom
* create a [MessageDigest](https://docs.oracle.com/javase/10/docs/api/java/security/MessageDigest.html) of the random number
* encode the byte[] returned by the MessageDigest into some acceptable textual form
* check if the result is already being used; if it's not already taken, it's suitable as a unique identifier

The MessageDigest class is suitable for generating a "one-way hash" of  arbitrary data. (Note that hash values never uniquely identify their source data, since different source data can produce the same hash value. The value of hashCode, for example, does not uniquely identify its associated object.) A MessageDigest takes any input, and produces a String which:

* is of fixed length
* does not allow the original input to be easily recovered (in fact, this is very hard)
* does not uniquely identify the input; however, similar input will produce dissimilar message digests

MessageDigest is often used as a checksum, for verifying that data has not been altered since its creation.

**Example**

**import** java.security.SecureRandom;

**import** java.security.MessageDigest;

**import** java.security.NoSuchAlgorithmException;

**public** **final** **class** GenerateId {

**public** **static** **void** main (String... arguments) {

**try** {

//Initialize SecureRandom

//This is a lengthy operation, to be done only upon

//initialization of the application

SecureRandom prng = SecureRandom.getInstance("SHA1PRNG");

//generate a random number

String randomNum = Integer.valueOf(prng.nextInt()).toString();

//get its digest

MessageDigest sha = MessageDigest.getInstance("SHA-1");

**byte**[] result = sha.digest(randomNum.getBytes());

System.out.println("Random number: " + randomNum);

System.out.println("Message digest: " + hexEncode(result));

}

**catch** (NoSuchAlgorithmException ex) {

System.err.println(ex);

}

}

/\*\*

\* The byte[] returned by MessageDigest does not have a nice

\* textual representation, so some form of encoding is usually performed.

\*

\* This implementation follows the example of David Flanagan's book

\* "Java In A Nutshell", and converts a byte array into a String

\* of hex characters.

\*

\* Another popular alternative is to use a "Base64" encoding.

\*/

**static** **private** String hexEncode(**byte**[] input){

StringBuilder result = **new** StringBuilder();

**char**[] digits = {'0', '1', '2', '3', '4','5','6','7','8','9','a','b','c','d','e','f'};

**for** (**int** idx = 0; idx < input.length; ++idx) {

**byte** b = input[idx];

result.append(digits[ (b&0xf0) >> 4 ]);

result.append(digits[ b&0x0f]);

}

**return** result.toString();

}

}

Example run:

>java -cp . GenerateId   
Random number: -1103747470   
Message digest: c8fff94ba996411079d7114e698b53bac8f7b037

### Style 3 - UID

Finally, here is another method, using a [java.rmi.server.UID](https://docs.oracle.com/javase/10/docs/api/java/rmi/server/UID.html). The Serializable identifiers generated by this class are unique on the host on which they are generated, provided that

* the host takes more than one millisecond to reboot
* the host's clock is never set to run backwards

In order to construct a UID that is globally unique, simply pair a UID with an [InetAddress](https://docs.oracle.com/javase/10/docs/api/java/net/InetAddress.html).

**Example**

**import** java.rmi.server.UID;

**public** **class** UniqueId {

/\*\*

\* Build and display some UID objects.

\*/

**public** **static** **void** main (String... arguments) {

**for** (**int** idx=0; idx<10; ++idx){

UID userId = **new** UID();

System.out.println("User Id: " + userId);

}

}

}

Example run:

User Id: 3179c3:ec6e28a7ef:-8000   
User Id: 3179c3:ec6e28a7ef:-7fff   
User Id: 3179c3:ec6e28a7ef:-7ffe   
User Id: 3179c3:ec6e28a7ef:-7ffd   
User Id: 3179c3:ec6e28a7ef:-7ffc   
User Id: 3179c3:ec6e28a7ef:-7ffb   
User Id: 3179c3:ec6e28a7ef:-7ffa   
User Id: 3179c3:ec6e28a7ef:-7ff9   
User Id: 3179c3:ec6e28a7ef:-7ff8   
User Id: 3179c3:ec6e28a7ef:-7ff7

Link is --- <https://github.com/fintx/fintx-identifier>

16.What are the best practices that are to be followed while designing RESTful web services?

Ans-

Typically we use a RESTful Web services design for our web APIs. The concept of REST is to separate the API structure into logical resources. There are used the HTTP methods GET, DELETE, POST and PUT to operate with the resources.

These are 10 best practices to design a clean RESTful Web Services:

## 1. Use nouns but no verbs

For an easy understanding use this structure for every resource:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Resource | GET read | POST create | PUT update | DELETE |
| /cars | Returns a list of cars | Create a new car | Bulk update of cars | Delete all cars |
| /cars/711 | Returns a specific car | Method not allowed (405) | Updates a specific car | Deletes a specific car |

Do not use verbs:

/getAllCars

/createNewCar

/deleteAllRedCars

## 2. GET method and query parameters should not alter the state

Use **PUT, POST** and **DELETE** methods  instead of the**GET** method to alter the state.  
Do not use **GET** for state changes:

GET /users/711?activate or

GET /users/711/activate

## 3. Use plural nouns

Do not mix up singular and plural nouns. Keep it simple and use only plural nouns for all resources.

/cars instead of /car

/users instead of /user

/products instead of /product

/settings instead of /setting

## **4. Use sub-resources for relations**

If a resource is related to another resource use subresources.

GET /cars/711/drivers/ Returns a list of drivers for car 711

GET /cars/711/drivers/4 Returns driver #4 for car 711

## **5. Use HTTP headers for serialization formats**

Both, client and server, need to know which format is used for the communication. The format has to be specified in the HTTP-Header.

Content-Type defines the request format.  
Accept defines a list of acceptable response formats.

## **6. Use HATEOAS**

**H**ypermedia **a**s **t**he **E**ngine **o**f **A**pplication **S**tate is a principle that hypertext links should be used to create a better navigation through the API.

|  |  |
| --- | --- |
| 01 | { |
| 02 | "id": 711, | |

|  |  |  |
| --- | --- | --- |
| 03 | "manufacturer": "bmw", | |
| 04 | "model": "X5", |

|  |  |
| --- | --- |
| 05 | "seats": 5, |
| 06 | "drivers": [ | |

|  |  |
| --- | --- |
| 07 | { |
| 08 | "id": "23", | |

|  |  |  |
| --- | --- | --- |
| 09 | "name": "Stefan Jauker", | |
| 10 | "links": [ |

|  |  |
| --- | --- |
| 11 | { |
| 12 | "rel": "self", | |

|  |  |  |
| --- | --- | --- |
| 13 | "href": "/api/v1/drivers/23" | |
| 14 | } |

|  |  |  |
| --- | --- | --- |
| 15 | ] | |
| 16 | } |

|  |  |  |
| --- | --- | --- |
| 17 | ] | |
| 18 | } |

## **7. Provide filtering, sorting, field selection and paging for collections**

**Filtering**:

Use a unique query parameter for all fields or a query language for filtering.

GET /cars?color=red Returns a list of red cars

GET /cars?seats<=2 Returns a list of cars with a maximum of 2 seats

**Sorting**:

Allow ascending and descending sorting over multiple fields.

GET /cars?sort=-manufactorer,+model

This returns a list of cars sorted by descending manufacturers and ascending models.

**Field selection**

Mobile clients display just a few attributes in a list. They don’t need all attributes of a resource. Give the API consumer the ability to choose returned fields. This will also reduce the network traffic and speed up the usage of the API.

GET /cars?fields=manufacturer,model,id,color

**Paging**

Use limit and offset. It is flexible for the user and common in leading databases. The default should be limit=20 and offset=0

GET /cars?offset=10&limit=5

To send the total entries back to the user use the custom HTTP header: X-Total-Count.

Links to the next or previous page should be provided in the HTTP header link as well. It is important to follow this link header values instead of constructing your own URLs.

Link: <https://blog.mwaysolutions.com/sample/api/v1/cars?offset=15&limit=5>; rel="next",

<https://blog.mwaysolutions.com/sample/api/v1/cars?offset=50&limit=3>; rel="last",

<https://blog.mwaysolutions.com/sample/api/v1/cars?offset=0&limit=5>; rel="first",

<https://blog.mwaysolutions.com/sample/api/v1/cars?offset=5&limit=5>; rel="prev",

## **8. Version your API**

Make the API Version mandatory and do not release an unversioned API. Use a simple ordinal number and avoid dot notation such as 2.5.

We are using the url for the API versioning starting with the letter „v“

/blog/api/v1

## **9. Handle Errors with HTTP status codes**

It is hard to work with an API that ignores error handling. Pure returning of a HTTP 500 with a stacktrace is not very helpful.

**Use HTTP status codes**

The HTTP standard provides over 70 status codes to describe the return values. We don’t need them all, but  there should be used at least a mount of 10.

200 – OK – Eyerything is working  
201 – OK – New resource has been created  
204 – OK – The resource was successfully deleted

304 – Not Modified – The client can use cached data

400 – Bad Request – The request was invalid or cannot be served. The exact error should be explained in the error payload. E.g. „The JSON is not valid“  
401 – Unauthorized – The request requires an user authentication  
403 – Forbidden – The server understood the request, but is refusing it or the access is not allowed.  
404 – Not found – There is no resource behind the URI.  
422 – Unprocessable Entity – Should be used if the server cannot process the enitity, e.g. if an image cannot be formatted or mandatory fields are missing in the payload.

500 – Internal Server Error – API developers should avoid this error. If an error occurs in the global catch blog, the stracktrace should be logged and not returned as response.

**Use error payloads**

All exceptions should be mapped in an error payload. Here is an example how a JSON payload should look like.

|  |  |
| --- | --- |
| 01 | { |
| 02 | "errors": [ | |

|  |  |
| --- | --- |
| 03 | { |
| 04 | "userMessage": "Sorry, the requested resource does not exist", | |

|  |  |  |
| --- | --- | --- |
| 05 | "internalMessage": "No car found in the database", | |
| 06 | "code": 34, |

|  |  |  |
| --- | --- | --- |
| 07 | "more info": "<http://dev.mwaysolutions.com/blog/api/v1/errors/12345>" | |
| 08 | } |

|  |  |  |
| --- | --- | --- |
| 09 | ] | |
| 10 | } |

## **10. Allow overriding HTTP method**

Some proxies support only **POST** and **GET** methods. To support a RESTful API with these limitations, the API needs a way to override the HTTP method.

Use the custom HTTP Header **X-HTTP-Method-Override** to overrider the POST Method.

17.How to decide which one of web service to use REST or SOAP?

Ans- SOAP (Simple Object Access Protocol) and REST (Representation State Transfer) are popular with developers working on system integration based projects. Software architects will design the application from various perspectives and also decides, based on various reasons, which approach to take to expose new API to third party applications. As a software architect, it is good practice to involve your development team lead during system architecture process.   
  
This article, based on my experience, will discuss when to use SOAP or REST web services to expose your API to third party clients.

## Web Services Demystified

Web services are part of the Services Oriented Architecture. Web services are used as the model for process decomposition and assembly. I have been involved in discussion where there were some misconception between web services and web API.

The W3C defines a [Web Service](http://en.wikipedia.org/wiki/Web_service) generally as:

**A software system designed to support interoperable machine-to-machine interaction over a network.**

**Web API also known as Server-Side Web API is a programmatic interface to a defined request-response message system, typically expressed in JSON or XML, which is exposed via the web – most commonly by means of an HTTP-based web server. (extracted from**[**Wikipedia**](http://en.wikipedia.org/wiki/Web_API)**)**

Based on the above definition, one can insinuate when SOAP should be used instead of REST and vice-versa but it is not as simple as it looks. We can agree that Web Services are not the same as Web API. Accessing an image over the web is not calling a web service but retrieving a web resources using is Universal Resource Identifier. HTML has a well-defined standard approach to serving resources to clients and does not require the use of web service in order to fulfill their request.

## Why Use REST over SOAP

Developers are passionate people. Let's briefly analyze some of the reasons they mentioned when considering REST over SOAP:

### REST is easier than SOAP

I'm not sure what developers refer to when they argue that REST is easier than SOAP. Based on my experience, depending on the requirement, developing REST services can quickly become very complex just as any other SOA projects. What is your service abstracting from the client? What is the level of security required? Is your service a long running asynchronous process? And many other requirements will increase the level of complexity. Testability: apparently it easier to test RESTFul web services than their SOAP counter parts. This is only partially true; for simple REST services, developers only have to point their browser to the service endpoints and a result would be returned in the response. But what happens once you need to add the HTTP headers and passing of tokens, parameters validation… This is still testable but chances are you will require a plugin for your browser in order to test those features. If a plugin is required then the ease of testing is exactly the same as using SOAPUI for testing SOAP based services.

### RESTFul Web Services serves JSON that is faster to parse than XML

This so called "benefit" is related to consuming web services in a browser. RESTFul web services can also serve XML and any MIME type that you desire. This article is not focused on discussing JSON vs XML; and I wouldn't write any separate article on the topic. JSON relates to JavaScript and as JS is very closed to the web, as in providing interaction on the web with HTML and CSS, most developers automatically assumes that it also linked to interacting with RESTFul web services. If you didn't know before, I'm sure that you can guess that RESTFul web services are language agnostic.

Regarding the speed in processing the XML markup as opposed to JSON, a performance test conducted by David Lead, Lead Engineer at MarkLogic Inc, find out to be a [myth](http://balisage.net/Proceedings/vol10/html/Lee01/BalisageVol10-Lee01.html).

### REST is built for the Web

Well this is true according to Roy Fielding [dissertation](https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf); after all he is credited with the creation of REST style architecture. REST, unlike SOAP, uses the underlying technology for transport and communication between clients and servers. The architecture style is optimized for the modern web architecture. The web has outgrown is initial requirements and this can be seen through HTML5 and web sockets standardization. The web has become a platform on its own right, maybe WebOS. Some applications will require server-side state saving such as financial applications to e-commerce.

### Caching

When using REST over HTTP, it will utilize the features available in HTTP such as caching, security in terms of TLS and authentication. Architects know that dynamic resources should not be cached. Let's discuss this with an example; we have a RESTFul web service to serve us some stock quotes when provided with a stock ticker. Stock quotes changes per milliseconds, if we make a request for BARC (Barclays Bank), there is a chance that the quote that we have receive a minute ago would be different in two minutes. This shows that we cannot always use the caching features implemented in the protocol. HTTP Caching be useful in client requests of static content but if the caching feature of HTTP is not enough for your requirements, then you should also evaluate SOAP as you will be building your own cache either way not relying on the protocol.

### HTTP Verb Binding

HTTP verb binding is supposedly a feature worth discussing when comparing REST vs SOAP. Much of public facing API referred to as RESTFul are more REST-like and do not implement all HTTP verb in the manner they are supposed to. For example; when creating new resources, most developers use POST instead of PUT. Even deleting resources are sent through POST request instead of DELETE.

SOAP also defines a binding to the HTTP protocol. When binding to HTTP, all SOAP requests are sent through POST request.

### Security

Security is never mentioned when discussing the benefits of REST over SOAP. Two simples security is provided on the HTTP protocol layer such as basic authentication and communication encryption through TLS. SOAP security is well standardized through WS-SECURITY. HTTP is not secured, as seen in the news all the time, therefore web services relying on the protocol needs to implement their own rigorous security. Security goes beyond simple authentication and confidentiality, and also includes authorization and integrity. When it comes to ease of implementation, I believe that SOAP is that at the forefront.

18.What are the primary security issues of web service?

Ans-When deploying a Web service, you have to think about how you will secure that service. Yes, even if you decide to open up access to the service to everyone and anyone, you still have to think about security — For example, protecting yourself against people seeking to deny access to your service. Security encompasses the following:

* Equipment deployment
* Authenticating users
* Guarding data so that users only see what they should see
* Tracking user activity

Any and all of these items may be a part of your overall security plan. In this chapter, we will take a look at all of these items and show how you can use them to make your Web service more secure.

**Equipment deployment**

One of the easiest things to do to secure your corporate data is to use hardware in an intelligent way. When deploying a publicly accessible Web service, you will want to expose as little of your internal infrastructure as necessary. There are a number of things you will want to do:

* Put your database machines behind a firewall.
* Use hardware to protect your equipment. For example, rely on routers instead of software firewalls. Hardware is typically faster at routing and is easier to lockdown. The software firewall may have unknown interactions with which to deal.
* Make use of a demilitarized zone (DMZ). In other words, only put the machine serving the Web service on the public Internet.

The basic theme in equipment deployment, as you have just seen, is that you should strive to keep the majority of your machines behind some sort of protective firewall.

**Authenticating users**

You authenticate a user to learn his or her identity. The identity information might be used to make sure a person should have access to the Web service. You may also use the identity to track the user's activities. When it comes to authenticating your users, you have several options:

* Application-level authentication — Users identify themselves via credentials supplied in the SOAP message.
* HTTP basic authentication — The username and password are sent as clear text. This is not useful for secure applications, but it can be useful in combination with other identification techniques.
* HTTP digest authentication — This tool sends a hashed version of the basic authentication credentials so that only the server can decode them.
* Client certificates — Using a certificate provided by a certificate authority, the client can prove its identity during SSL authentication.
* Windows authentication — Through HTTP basic/digest authentication or client certificates, IIS can map a user identity to a real Windows user.

All of these options have different uses.

**Guarding data**

When you make your data, you can use the user authentication mechanisms discussed to guard it. You can use Access Control Lists to guard files and SQL-based security to guard data in your database. As part of your security for the Web service, consider using a combination of user identity and other security mechanisms as a way to protect your data. For example, SQL server allows you to limit who can and cannot access various databases, tables and stored procedures. NTFS limits what files a particular user can access. Active Directory can be used to limit the network resources the user can access. An effective security plan uses a combination of methods to keep things safe. By authenticating the user using Windows Integrated Authentication and denying anonymous access to the Web service, the Web method will impersonate the caller when it executes. Any rights given to that caller will be enforced. This includes access to files, network resources and database objects.

**Tracking user activity**

Many applications require that you give users access to sensitive resources. When the users are using those resources, you will want to be able to see what was done. What did they look at? What actions did they fire? What data did they change? Most of the time, the users will not abuse their privileges. However, when the users are new and are trying to do something wrong, you want to see what they did. It will help you back out the changes the newbie made. As a forensic tool, tracking the changes will help you identify who did what and when.

All of this involves creating some sort of an auditing strategy. In the best of all worlds, you will know exactly who is calling you. Auditing was discussed as a debugging tool in chapter five, "Troubleshooting Web services/consumers." For security, you can use auditing to determine what the user did and when he or she did it. What information do you track when auditing user activity? This set of data is specific to security:

* User identity (if available) — This lets you know who executed the action.
* IP address of caller — This allows you to track the call back to a specific machine. This comes in handy, even if the user is coming through some sort of proxy server.
* Free text field — You will want some sort of generic field that allows you to put all audit data into one table. Within this field, you can enter information particular to the action the user was performing. Think of putting in information that would be helpful when capturing details about what happened.
* Time of action — By knowing when things happened, you will be able to figure out the order of the actions and string together what the user did.

19.What do you know about foundation security services?

Ans- As with any well-built system, effective cyber security begins with building a solid security foundation. Without thoughtful attention to building a good foundation, a business will remain vulnerable to hacks and breaches until they finally decide to face cyber security issues head on.

## Assessment

When first addressing an issue, one must first assess all the security issues, both the obvious and the potential ones. Assessing a company’s technology vulnerabilities must be performed by someone with a high level of expertise in evaluating the company overall for sub-standard IT practices.

## Planning

After determining all the security holes that need attention, the next step is to create an pro-active plan that will address each potential threat. The plan should include both steps that can be taken in order to avoid a security problem altogether, as well as a solid recovery plan to initiate in the event the worst should happen and a breach is discovered.

A comprehensive security plan will consist of several key components, including:

### Combating Intrusion

Sometimes there is a delicate balance between applying security measures to ward off the threats of intrusion, yet still providing enough openness in a computer system to allow employees to actually complete their daily tasks. A good security team will assess what their client needs in terms of effective security, but in a balanced manner, thus preventing their security efforts from hindering daily operations.

### Addressing Mobile and Remote Access

Even if all employees in a given company work on-site, it is a rare company that does not allow at least some off-site mobile and/or remote access to corporate systems by their staff members. As work life becomes more and more intertwined with personal time, if not addressed, the security risks associated with off-site access means a company will remain vulnerable to hacks and data breaches.

### Securing Endpoints

In the past, ensuring that all corporate PCs had anti-virus software installed might have been enough to combat threats. With every passing year however, hackers and data thieves become more ingenious in their behavior. A good security plan will include securing endpoints through configuration and patch management, solid firewalls, application controls, data loss protection, and in some areas, data encryption.

### Enforcing Multi-Factor Authentication

More and more businesses are enacting multi-factor authentication in order for their employees to access corporate systems and data. Many breaches have occurred from the use of woefully inadequate passwords such as “1234567” or “ABC”. Of course, companies should also support their multi-factor authentication measures by enforcing the use of strong passwords.

20.How is Security mechanism implemented using Spring?

Ans- Spring Security is a framework that enables a programmer to impose security restrictions to Spring-framework–based Web applications through JEE components. In short, it is a library that can be used, extended to customize as per the programmer's needs. Because it is a member of the same Spring family, it goes smoothly hand in hand with the Spring Web MVC. Its primary area of operation is to handle authentication and authorization at the Web request level as well as the method invocation level. Perhaps. The greatest advantage of this framework is that it is powerful yet highly customizable in its implementation. Although it follows Spring's convention over configuration, programmers can choose between default provisions or customizing it according to their needs.

## Security in a Spring MVC Application

Because Spring Security tightly integrates with the Spring Framework and other commonly used authentication mechanisms, such as HTTP basic authentication, X.509 certificate, form-based login, and so on, it has comprehensive support for both Web applications as well as method-level security. Securing Web applications is perhaps the most common concern and here we'll glimpse that aspect of Spring Security.

While employing Spring Security, one can completely eliminate the need of an XML security configuration and instead apply an annotation to configure it. We can write a simple configuration class such as follows:

package org.mano.security.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web

.configuration.\*;

@EnableGlobalMethodSecurity(prePostEnabled=true)

@EnableWebSecurity

@Configuration

public class SecurityConfiguration extends

WebSecurityConfigurerAdapter {

// ...

}

The annotation @EnableWebSecurity enables Web security; otherwise, it remains disabled by default. Now, to configure the security, we can either implements the interface called WebSecurityConfigurer or extend the more convenient class called WebSecurityConfigurerAdapter. The advantage of extending the adapter class is that we can configure Web security by overriding only those parts that we are interested in; others can remain their default form. There are three variations of the configure method that we can override to configure and secure the application:

* void configure( AuthenticationManagerBuilder auth): To configure user details services
* void configure( HttpSecurity http): To configure how requests are secured by interceptors
* void configure( WebSecurity web): To configure Spring Security's filter chain

The default filter chain is fine for most needs. So, we may configure the other two in the following manner.

public class SecurityConfiguration extends

WebSecurityConfigurerAdapter {

@Autowired

private CustomUserDetailsService userDetailsService;

@Override

protected void configure(AuthenticationManagerBuilder auth)

throws Exception {

auth.userDetailsService(userDetailsService)

.passwordEncoder(getPasswordEncoder());

}

private PasswordEncoder getPasswordEncoder() {

return new PasswordEncoder() {

@Override

matches(CharSequence rawPassword, String

encodedPassword) {

// Verifies whether encoded password obtained matches

// with the rawPassword CharSequence

}

@Override

public String encode(CharSequence rawPassword) {

// Code for password encoding

}

};

}

// ...

}

The PasswordEncoder is a service interface provided by the Spring Security framework for encoding passwords.

The interceptor's secure method configuration may look like this:

public class SecurityConfiguration extends

WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws

Exception {

http.authorizeRequests()

.antRequest().authenticated()

.and().formLogin().and().httpBasic();

}

// ...

}

This simple configuration specifies how HTTP requests are secured. The chain of method enforces that all HTTP requests are authenticated via a login form. The form-based login is a predefined login page provided by Spring Security.

When overriding the configure(AuthenticationManagerBuilder auth) method, we can use in-memory user storage as follows:

public class SecurityConfiguration extends

WebSecurityConfigurerAdapter {

@Override

protected void configure(AuthenticationManagerBuilder auth)

throws Exception {

auth.inMemoryAuthentication()

.withUser("user").password("password").roles("USER").and()

.withUser("admin").password("password").roles("USER",

"ADMIN");

}

// ...

}

21.Design webservice for shorten URL and expanded URL?

## Ans- **Overview**

This is a very hot article that you can’t leave without checking it first. This article is talking about the most popular and powerful URL shortening service ever, bit.ly.

Today, we are going to talk about bit.ly API, its functions, and how you can access them from your .NET application.

Don’t forget to download the sample code at the end of the article.

Let’s go!

## **Introduction**

Today we are going to talk about the most popular yet most powerful URL shortening service ever in the web. Yes you guessed, it is [**bit.ly**](http://bit.ly/).

Of course this service is well-known enough and we don’t need to describe it or talk about its features, so we’ll dig into the API immediately.

## **API**

### **Overview**

There’re few things that you should keep in mind about bit.ly [**API**](http://code.google.com/p/bitly-api/wiki/ApiDocumentation):

* **REST API**:

The bit.ly API is a [REST](http://justlikeamagic.com/2010/08/15/consuming-url-shortening-services-introduction/) web service; means that it’s a collection of HTTP endpoints each accessed by a simple address filled with arguments the function (endpoint) requires the way you fill up query strings of a web page. Actually, you don’t need to care about REST web services or HTTP endpoints; you just need to know how to call those functions (simply, to access those addresses programmatically.)

Yet, the current and only version of the API is version 3 and that can be accessed using the following address:

Hide   Copy Code

http://api.bit.ly/v3/[function\_name]?[args]

Simply, substitute function\_name with the name of the function you need to call, and replace args with function arguments.

Notice that you can try any function by just typing the address into the browser and navigating to the results.

* **Functions:**

The API provides you with lots of functions that satisfy your application needs. The following list contains the functions available:

* Shorten:  
  Used to shorten a long URL.
* Expand:  
  Used to expand the URL; to get the original long URL from the short one.
* Validate:  
  Used to validate a username and his API key.
* Clicks:  
  Used to retrieve stats (number of clicks) about the short URL specified.
* Lookup:  
  Used to check a long URL if exists in the database, i.e., if it has been shortened before.
* Info:  
  Used to retrieve information about the URL (e.g. the user created it, page title, etc.)
* Authenticate:  
  Used to check if a username and password are valid. **Access restricted, more information available later at the end of this article.**

As you see, the API of bit.ly is the most sophisticated yet powerful API compared to the APIs of the other URL shortening services.

* **Input Arguments:**

Any URL passed to a function must be encoded first to eliminate the ‘#’, ‘?’, ‘=’ and other problematic characters in the URL. For encoding a URL, the function System.Net.Uri.EscapeUriString() is very sufficient.

There’re three main required arguments that are used by all functions:

* login:  
  Username.
* apiKey:  
  The key used to authenticate the user access to the API.
* format:  
  The format (type) of returned data from functions.

Those three are required by all functions and you cannot work without one of them.

* **Authentication:**

All functions require user authentication. The user can prove his identity using his login name (username) and his API key (not his password.) You can get your API key by accessing the page <http://bit.ly/a/account> (after logging on to your account) or directly from <http://bit.ly/a/your_api_key>.

One of the hot features of the API is that it provides you a demo user that can be used in your API training, the information of that user is as follows:

Hide   Copy Code

Username: bitlyapidemo

API Key: R\_0da49e0a9118ff35f52f629d2d71bf07

You might face problems with this account like violation of rate limits and many other problems, and that because it’s used by many users in the same time. Therefore, it’s recommended that you use another account.

* **Supported Formats:**

The API supports two formats of its returned data, XML and JSON (the default.) Yes it supports plain text too, but it’s not supported by all functions. XML data is easily manipulated by XML, so we’ll concentrate on XML besides the plain text format of course.

* **Handling Errors:**

If the function failed and you have specified the format as Plain Text (txt) in the call, you get an exception thrown in your code. If the format was XML, you can check the returned data for whether the function succeeded or not.

The XML data returned from functions must follow this schema:

Hide   Copy Code

<?xml version="1.0" encoding="UTF-8"?>

<response>

<status\_code />

<status\_txt />

...

</response>

Here we have status\_code set to the value 200 if the function succeeded and to the error code if the function failed. The status\_txt describes the status of the function, it’s set to ‘OK’ if the function succeeded and to the error description if the function failed.

The rest of the XML data is defined based on the function.

* **Preferred Domain:**

You have the option to use one of two domains, [http://bit.ly](http://bit.ly/) and [http://j.mp](http://j.mp/) (new,) both offer you the same functionality and the same flexibility, however, the first counts to 20 characters while the other counts to only 18. (The domain can be set in the shortening function in the argument domain.)

Keep in mind that the code just after the domain name (e.g. bnPuEX of <http://bit.ly/bnPuEX>) is called **Hash** and it is exactly 6 characters (case-sensitive.)

There’re two types of hash, each short URL has many hashes:

* User Hash: That hash of the URL generated for a given user shortened the URL. That means that a long URL might have more than one user hash equals to the number of users shortened that URL. (More than one hash means more than one short URL.)
* Global Hash: A hash that is shared by all users for the same short URL.

Thus, a short URL has only one global hash, but it might have more than one user hash (for each user shortened the same long URL.)

* **Rate Limits:**

You cannot think about making thousands of function calls every hour, access to the API is limited for each user on an hourly base. Limits are very sufficient for your application, but it’s going not to be sufficient if you are willing to spam the service or to drop it!

### **Function Explanation**

Now we are going to talk about each function and how you can call it.

First, get your [API key](http://bit.ly/a/your_api_key) that will be used to authenticate your calls to the API. If you need to bother yourself and to clog your application use the demo API user **bitlyapidemo** that have the API key **R\_0da49e0a9118ff35f52f629d2d71bf07**.

### **Shortening Function**

The first function we are going to talk about today is the shortening function, shorten. This function has the following address <http://api.bit.ly/v3/shorten> (as you expected) and is used to shorten long URLs. Besides main arguments key, apiKey, and format, it takes two more:

* longUrl:  
  The long URL to be shortened.
* domain:  
  Optional. The preferred domain, bit.ly or j.mp.

You can get hands on this function and try it simply by navigating to the results of the following URL:

[http://api.bit.ly/v3/shorten?login=bitlyapidemo&apiKey=R\_0da49e0a9118ff35f52f629d2d71bf07 &format=txt&longUrl=http://JustLikeAMagic.com](http://api.bit.ly/v3/shorten?login=bitlyapidemo&apiKey=R_0da49e0a9118ff35f52f629d2d71bf07&format=txt&longUrl=http://JustLikeAMagic.com)

This call simply tries to shorten the URL [http://JustLikeAMagic.com](http://justlikeamagic.com/) by using credentials of the demo API user (substitute the current information with your own.) The format is set to plain text.

You can also use change the format to XML and get output like this:

Hide   Copy Code

<?xml version="1.0" encoding="utf-8"?>

<response>

<status\_code>200</status\_code>

<status\_txt>OK</status\_txt>

<data>

<url>http://bit.ly/bO9TgE</url>

<hash>bO9TgE</hash>

<global\_hash>9gSDEU</global\_hash>

<long\_url>http://JustLikeAMagic.com</long\_url>

<new\_hash>0</new\_hash>

</data>

</response>

Notice that the status code is 200 that means that everything went ‘OK’. Notice that we have 5 elements:

* url:  
  The long URL generated.
* hash:  
  The user hash string.
* global\_hash:  
  The globally shared hash. Can be used to browse to the URL too, it would be counted in the global statistics but not in user’s.
* long\_url:  
  The original URL.
* new\_hash:  
  Equals to 1 if this is the first time that URL being shortened (using the bit.ly service of course,) or 0 otherwise.

Now, let’s code! The following function accepts a long URL and user API credentials and tries to shorten the URL using our shortening function.

Don’t forget to add using statements to namespaces System.IO, System.Net, and System.Xml to that code and to the other code demonstrated in this article.

Hide   Shrink Image 3 for Consuming URL Shortening Services – bit.ly   Copy Code

*// C#*

string Shorten(string url, string login, string key, bool xml)

{

url = Uri.EscapeUriString(url);

string reqUri =

String.Format("http://api.bit.ly/v3/shorten?" +

"login={0}&apiKey={1}&format={2}&longUrl={3}",

login, key, xml ? "xml" : "txt", url);

HttpWebRequest req = (HttpWebRequest)WebRequest.Create(reqUri);

req.Timeout = 10000; *// 10 seconds*

*// if the function fails and format==txt throws an exception*

Stream stm = req.GetResponse().GetResponseStream();

if (xml)

{

XmlDocument doc = new XmlDocument();

doc.Load(stm);

*// error checking for xml*

if (doc["response"]["status\_code"].InnerText != "200")

throw new WebException(doc["response"]["status\_txt"].InnerText);

return doc["response"]["data"]["url"].InnerText;

}

else *// Text*

using (StreamReader reader = new StreamReader(stm))

return reader.ReadLine();

}

Take notice of the mechanism used to check for errors.

### **Expanding Function**

The next function we have is the function that is used to expand a URL, i.e., to get the long URL from the short one. Obviously, this function is called expand and it accepts the short URL shortUrl besides the three main arguments.

Likewise, calling this function generate a data based on the function format, txt, xml, or json. The following XML data is generated when the function is called while the format is set to xml:

Hide   Copy Code

<?xml version="1.0" encoding="utf-8" ?>

<response>

<status\_code>200</status\_code>

<status\_txt>OK</status\_txt>

<data>

<entry>

<short\_url>http://bit.ly/bnPuEX</short\_url>

<long\_url>http://justlikeamagic.com</long\_url>

<user\_hash>bnPuEX</user\_hash>

<global\_hash>bdE96m</global\_hash>

</entry>

</data>

</response>

Now you can see the two hashes, user\_hash and global\_hash, and the two lend the user to your page (although the access is counted differently.)

Now, let’s code! The following function retrieves the long URL from the short one:

Hide   Shrink Image 4 for Consuming URL Shortening Services – bit.ly   Copy Code

*// C#*

string Expand(string url, string login, string key, bool xml)

{

url = Uri.EscapeUriString(url);

string reqUri =

String.Format("http://api.bit.ly/v3/expand?" +

"login={0}&apiKey={1}&format={2}&shortUrl={3}",

login, key, xml ? "xml" : "txt", url);

HttpWebRequest req = (HttpWebRequest)WebRequest.Create(reqUri);

req.Timeout = 10000; *// 10 seconds*

*// if the function fails and format==txt throws an exception*

Stream stm = req.GetResponse().GetResponseStream();

if (xml)

{

XmlDocument doc = new XmlDocument();

doc.Load(stm);

*// error checking for xml*

if (doc["response"]["status\_code"].InnerText != "200")

throw new WebException(doc["response"]["status\_txt"].InnerText);

return doc["response"]["data"]["entry"]["long\_url"].InnerText;

}

else *// Text*

using (StreamReader reader = new StreamReader(stm))

return reader.ReadLine();

}

### **Validation Function**

The function validate is used to check if another username and API key pair is valid. For this function to work, you should use valid API credentials to check for the other credentials if they are valid or not. Therefore, you are going to use two additional arguments for the additional credentials, x\_login and x\_apiKey.

This function doesn’t support plain text format. If XML was used, the function returns data like the following:

Hide   Copy Code

<?xml version="1.0" encoding="UTF-8"?>

<response>

<status\_code>200</status\_code>

<status\_txt>OK</status\_txt>

<data>

<valid>0</valid>

</data>

</response>

The valid element is set to 1 if the credentials were OK or 0 otherwise.

And this is our C# function that validates user API credentials:

Hide   Shrink Image 5 for Consuming URL Shortening Services – bit.ly   Copy Code

*// C#*

string Validate(string login, string key,

string xLogin, string xKey, bool xml)

{

string reqUri =

String.Format("http://api.bit.ly/v3/validate?" +

"login={0}&apiKey={1}&x\_login={4}&x\_key={5}&format={2}" +

login, key, xLogin, xKey, xml ? "xml" : "txt");

HttpWebRequest req = (HttpWebRequest)WebRequest.Create(reqUri);

req.Timeout = 10000; *// 10 seconds*

*// if the function fails and format==txt throws an exception*

Stream stm = req.GetResponse().GetResponseStream();

if (xml)

{

XmlDocument doc = new XmlDocument();

doc.Load(stm);

*// error checking for xml*

if (doc["response"]["status\_code"].InnerText != "200")

throw new WebException(doc["response"]["status\_txt"].InnerText);

return int.Parse(doc["response"]["data"]["valid"]) == 1 ? true : false;

}

else *// Text*

using (StreamReader reader = new StreamReader(stm))

return int.Parse(reader.ReadLine()) == 1 ? true : false;

}

### **Stats Function**

This function is used to get stats about the short URL; the stats are represented in two values, user clicks and global clicks. User clicks value is the number of access times made to that user link. Global clicks value is the number of access times made from all short URLs (from all users) refer to the same address (almost like user hashes and global hash.)

The function only accepts the short URL, shortUrl, besides the three main arguments. The data returned from the function is almost like this (in XML):

Hide   Copy Code

<?xml version="1.0" encoding="utf-8" ?>

<response>

<status\_code>200</status\_code>

<data>

<clicks>

<short\_url>http://bit.ly/bnPuEX</short\_url>

<global\_hash>bdE96m</global\_hash>

<user\_clicks>0</user\_clicks>

<user\_hash>bnPuEX</user\_hash>

<global\_clicks>0</global\_clicks>

</clicks>

</data>

<status\_txt>OK</status\_txt>

</response>

The following C# function is used to retrieve number of access times for the current user and for all users:

Hide   Copy Code

*// C#*

int GetClicks(string url, string login, string key, out int globalClicks)

{

url = Uri.EscapeUriString(url);

string reqUri =

String.Format("http://api.bit.ly/v3/clicks?" +

"login={0}&apiKey={1}&shortUrl={2}&format=xml" +

login, key, url);

HttpWebRequest req = (HttpWebRequest)WebRequest.Create(reqUri);

req.Timeout = 10000; *// 10 seconds*

Stream stm = req.GetResponse().GetResponseStream();

XmlDocument doc = new XmlDocument();

doc.Load(stm);

*// error checking for xml*

if (doc["response"]["status\_code"].InnerText != "200")

throw new WebException(doc["response"]["status\_txt"].InnerText);

XmlElement el = doc["response"]["data"]["clicks"];

globalClicks = int.Parse(el["global\_clicks"].InnerText);

return int.Parse(el["user\_clicks"].InnerText);

}

### **Lookup Function**

This function is used with long URLs to check whether they have been shortened before, and if so, the function returns the short URLs.

If the long URL was found in service database, XML data like this is returned:

Hide   Copy Code

<?xml version="1.0" encoding="utf-8" ?>

<response>

<status\_code>200</status\_code>

<status\_txt>OK</status\_txt>

<data>

<lookup>

<url>http://JustLikeAMagic.com</url>

<short\_url>http://bit.ly/9gSDEU</short\_url>

<global\_hash>9gSDEU</global\_hash>

</lookup>

</data>

</response>

Otherwise, you get another form of XML data:

Hide   Copy Code

<?xml version="1.0" encoding="utf-8" ?>

<response>

<status\_code>200</status\_code>

<status\_txt>OK</status\_txt>

<data>

<lookup>

<url>http://JustLikeAMagic.com/books</url>

<error>NOT\_FOUND</error>

</lookup>

</data>

</response>

The following C# function looks-up a URL and returns its short URL if found:

Hide   Copy Code

*// C#*

string Lookup(string url, string login, string key)

{

url = Uri.EscapeUriString(url);

string reqUri =

String.Format("http://api.bit.ly/v3/lookup?" +

"login={0}&apiKey={1}&url={2}&format=xml" +

login, key, url);

HttpWebRequest req = (HttpWebRequest)WebRequest.Create(reqUri);

req.Timeout = 10000; *// 10 seconds*

Stream stm = req.GetResponse().GetResponseStream();

XmlDocument doc = new XmlDocument();

doc.Load(stm);

*// error checking for xml*

if (doc["response"]["status\_code"].InnerText != "200")

throw new WebException(doc["response"]["status\_txt"].InnerText);

if (doc["response"]["data"]["lookup"]["error"] == null)

return null; *// not found*

return doc["response"]["data"]["lookup"]["short\_url"].InnerText;

}

### **Info Function**

The info function is used to retrieve information about the current short URL. This function returns XML data like the following:

Hide   Copy Code

<?xml version="1.0" encoding="utf-8" ?>

<response>

<status\_code>200</status\_code>

<status\_txt>OK</status\_txt>

<data>

<info>

<short\_url>http://bit.ly/bnPuEX</short\_url>

<global\_hash>bdE96m</global\_hash>

<user\_hash>bnPuEX</user\_hash>

<created\_by>elsheimy</created\_by>

<title>Just Like a Magic</title>

</info>

</data>

</response>

Besides link hashes, it returns the name of user who created it and the page title. And this is our C# function that retrieves that information:

Hide   Copy Code

*// C# Code*

string GetInfo(string url, string login, string key, out string createdBy)

{

url = Uri.EscapeUriString(url);

string reqUri =

String.Format("http://api.bit.ly/v3/info?" +

"login={0}&apiKey={1}&shortUrl={2}&format=xml" +

login, key, url);

HttpWebRequest req = (HttpWebRequest)WebRequest.Create(reqUri);

req.Timeout = 10000; *// 10 seconds*

Stream stm = req.GetResponse().GetResponseStream();

XmlDocument doc = new XmlDocument();

doc.Load(stm);

*// error checking for xml*

if (doc["response"]["status\_code"].InnerText != "200")

throw new WebException(doc["response"]["status\_txt"].InnerText);

XmlElement el = doc["response"]["data"]["info"];

createdBy = el["created\_by"].InnerText;

return el["title"].InnerText;

}

### **Authentication Function**

This is the last function today, the authenticate function. This function is used to check whether a username and a password are valid. Although this function and the validation function work the same way, there’s a big difference. The validation function checks for API credentials, the username and the API key, while this function checks for login information, the username and the password. Another big difference is that this function is currently **access-restricted** and you cannot use it before asking for permission from api@bit.ly.

This function accepts two addition parameters, the username x\_login and the password x\_password.

This function is called in a very specific way. You add arguments in the body of your request. In addition, the request is made by the method POST.

If the function succeeded you get the API key for that user. For example:

Hide   Copy Code

<?xml version="1.0" encoding="UTF-8"?>

<response>

<status\_code>200</status\_code>

<data>

<authenticate>

<username>bitlyapidemo</username>

<successful>1</successful>

<api\_key>R\_0da49e0a9118ff35f52f629d2d71bf07</api\_key>

</authenticate>

</data>

<status\_txt>OK</status\_txt>

</response>

Otherwise, the successful element is set to 0 and no other information is available:

Hide   Copy Code

<?xml version="1.0" encoding="UTF-8"?>

<response>

<status\_code>200</status\_code>

<data>

<authenticate>

<successful>0</successful>

</authenticate>

</data>

<status\_txt>OK</status\_txt>

</response>

The next C# function tries to authenticate a given user and retrieve his API key (Notice how to set information in the body of the request):

Hide   Shrink Image 6 for Consuming URL Shortening Services – bit.ly   Copy Code

*// C#*

string Authenticate(string login, string key, string xLogin, string xPassword)

{

string reqUri = "http://api.bit.ly/v3/authenticate";

string body =

string.Format("login={0}&apiKey={1}&x\_login={2}" +

"&x\_password={3}&format=xml",

login, key, xLogin,xPassword);

HttpWebRequest req = (HttpWebRequest)WebRequest.Create(reqUri);

req.Timeout = 10000;

req.Method = "POST";

StreamWriter writer = new StreamWriter(req.GetRequestStream());

writer.WriteLine(body);

Stream stm = req.GetResponse().GetResponseStream();

XmlDocument doc = new XmlDocument();

doc.Load(stm);

*// error checking for xml*

if (doc["response"]["status\_code"].InnerText != "200")

throw new WebException(doc["response"]["status\_txt"].InnerText);

XmlElement el = doc["response"]["data"]["authenticate"];

if (el["successful"].InnerText == "1")

return el["api\_key"].InnerText;

else

return null;

}

22.How to maintain security and protocols in Web services.

Ans-open link-- <https://www.guru99.com/security-web-services.html>

23.Two problems you faced in the current project and how you resolved it.

Ans- we have faced many problems ,some of them are

1. Special character issue :

In our project we will send then response data in the form of json. One of the

element value is getting special character in production,where as dev and test

environment it is not getting. As a provider i am send as it is response to

consumer.Consumer end unable to perform deserializae at their end.i.e. unable convert

json to java object.

2. DB Issue : DB team while move the data from test to prod they did misake one of the

db element is configured is wrong.

So in production always we are getting SQLInvalidGrammerError.

DB team has fix the issue

For this issue PSG(Production Support Group) team has raised the Sev-2 IMR. We have

resolved the issue with the help of DB team.

Note : IMR -- Incident Management Request Sev – Seviority

Sev-1 : very high priority Sev-2 : High Sev-3 : medium Sev- 4 : Low

3. Certificate Issues : Provider production certificates got expired

In dev and test it was working fine.We assume that it will work on production also,but

after 2 days of prod deployment then we are getting "SSLHnadShakeException"--Certificate Expired.

We have informed to provider and get the renewal certificate from provider team and moved our changes in prod

24.What do you mean by Interface Driven Programming ?

Ans- 1

I have been doing a little thinking about inheritance vs. realization vs. composition. I am not about to post the whole detail here. So I was wondering, when we are not talking about creating interfaces to facilitate unit testing: Why does interface-based programming seem to focus upon the grouping of common behaviour, e.g., IPettable (for an animal), IEditable (for a user control), ISubmitable (for a form), etc. Why does the use of interface-based programming appear to be limited to behaviour and not essences. We could pragmatically use an interface, not so much realize behaviour but, to realize commonsense physical similarities which could have nothing to do with behaviour? It is not that there is some limiting feature within interface-based programming that we do not think to do this. This means that there is a tendency to use an interface in a fraction of its possible ways; so how come?