

[JAVA TUTORIAL](#)[#INDEX POSTS](#)[#INTERVIEW QUESTIONS](#)[RESOURCES](#)[HIRE ME](#)[DOWNLOAD ANDROID APP](#)[CONTRIBUTE](#)**Subscribe to Download Java Design Patterns eBook****DOWNLOAD NOW**[HOME](#) » [JAVA](#) » [JAVA EE](#) » [TOMCAT DATASOURCE JNDI EXAMPLE IN JAVA](#)

Tomcat DataSource JNDI Example in Java

APRIL 2, 2018 BY [PANKAJ](#) — [40 COMMENTS](#)

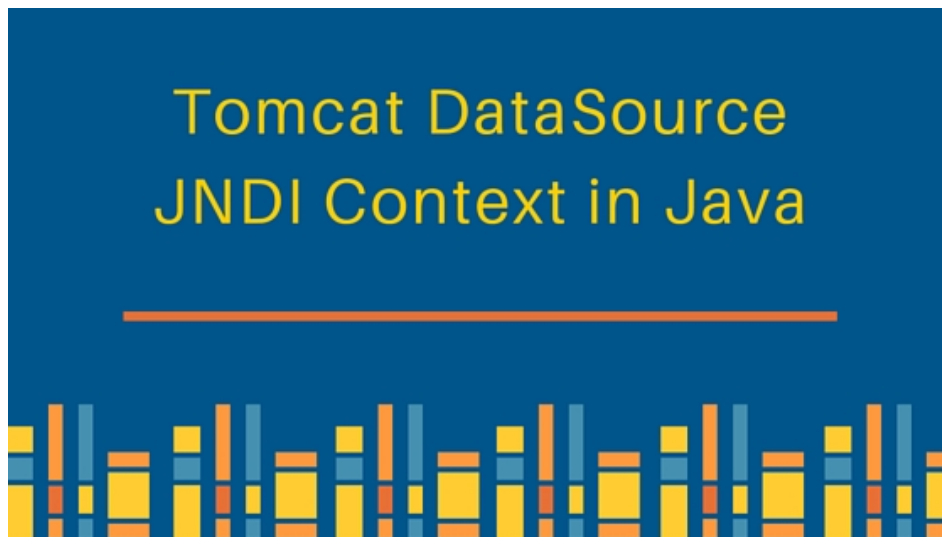
Welcome to Tomcat DataSource JNDI Example Tutorial. We looked at the [JDBC DataSource](#) in the last tutorial and learned how to use that in standalone java application.

Table of Contents [\[hide\]](#)

[1 Tomcat DataSource JNDI](#)

[1.1 Tomcat DataSource JNDI Configuration Example – server.xml](#)[1.2 Tomcat DataSource JNDI Resource Link Configuration – context.xml](#)[1.3 Tomcat DataSource JNDI Example](#)

Tomcat DataSource JNDI



Actual benefit of DataSource comes when we use it with a **JNDI Context**. For example, connection pool in a web application deployed in a servlet container. Most of the popular servlet containers provide built-in support for DataSource through Resource configuration and JNDI context. This helps us in creating and using DataSource connection pool with just few lines of configuration. This tutorial is aimed to provide Tomcat DataSource JNDI configuration example.

Apache Tomcat provide three ways to configure DataSource in JNDI context.

1. **Application context.xml** – This is the easiest way to configure DataSource, all we need is a context.xml file in META-INF directory. We have to define Resource element in the context file and container will take care of loading and configuring it. The approach is simple but it has some drawbacks;
 - Since the context file is bundled with the WAR file, we need to build and deploy new WAR for every small configuration change. Same issue comes if your application works in distributed environment or your application needs to be deployed in different testing environments such as QA, IT, PROD etc.
 - The datasource is created by container for the application usage only, so it can't be used globally. We can't share the datasource across multiple applications.
 - If there is a global datasource (server.xml) defined with same name, the application datasource is ignored.
2. **Server context.xml** – If there are multiple applications in the server and you want to share DataSource across them, we can define that in the server context.xml file. This file is located in `apache-tomcat/conf` directory. The scope of server context.xml file is application, so if you define a DataSource connection pool of 100 connections and there are 20 applications then the datasource will be created for each of the application. This will result in 2000 connections that will obviously consume all the database server resources and hurt application performance.
3. **server.xml and context.xml** – We can define DataSource at global level by defining them in the server.xml `GlobalNamingResources` element. If we use this approach, then we need to define a `ResourceLink` from context.xml file of server or application specific. This is the preferred way when you are looking to share a common resource pool across multiple applications running on the server.

Regarding resource link, whether to define it at server level context xml file or application level depends on your requirement.

Let's head over to the Tomcat DataSource JNDI example in [java web application](#).

For the test data setup, please refer to my last article about [JDBC DataSource Example](#).

Tomcat DataSource JNDI Configuration Example – server.xml

Add below code in the tomcat server.xml file. The code should be added in the `GlobalNamingResources` element. Also make sure that database driver is present in the tomcat lib directory, so in this case mysql jdbc jar have to be present in the tomcat lib.

```
<Resource name="jdbc/MyDB"
    global="jdbc/MyDB"
    auth="Container"
    type="javax.sql.DataSource"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/UserDB"
    username="pankaj"
    password="pankaj123"

    maxActive="100"
    maxIdle="20"
    minIdle="5"
    maxWait="10000"/>
```

Here we are creating JNDI context with name as `jdbc/MyDB` which is a type of `DataSource`. We are passing database configurations in url, username, password and driverClassName attribute. Connection pooling properties are defined in maxActive, maxIdle and minIdle attributes.

Tomcat DataSource JNDI Resource Link Configuration – context.xml

Add below code in the server context.xml file.

```
<ResourceLink name="jdbc/MyLocalDB"
    global="jdbc/MyDB"
    auth="Container"
    type="javax.sql.DataSource" />
```

Notice that resource link name is different than global link, we have to use this name in our java program to get the `DataSource`.

Tomcat DataSource JNDI Example

Create a dynamic web application with name **JDBCDataSourceTomcat** and then create a Servlet with below code.

```
package com.journaldev.jdbc.datasource;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;

@WebServlet("/JDBCDataSourceExample")
public class JDBCDataSourceExample extends HttpServlet {
    private static final long serialVersionUID = 1L;
```

Notice that I am using **Servlet 3 Annotation based configuration** and it will work in Tomcat 7 or higher versions. If you are using lower version of Tomcat then you need to do some modifications to the servlet code, to remove `WebServlet` annotation and configure in `web.xml` file.

The part of servlet code that we are interested in;

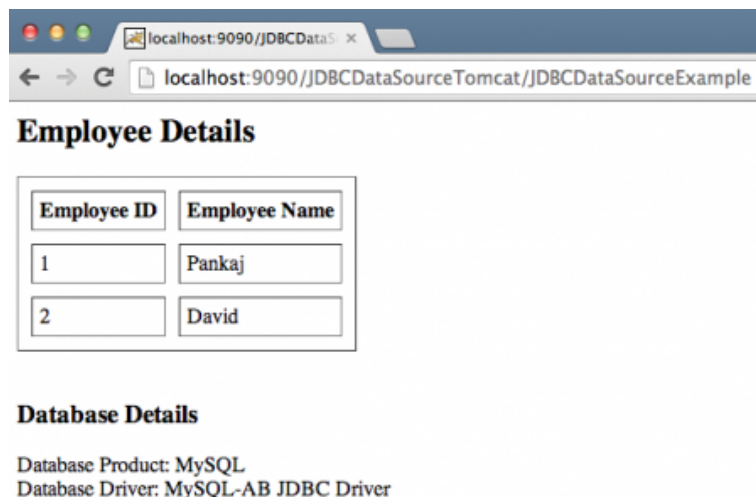
```
ctx = new InitialContext();
DataSource ds = (DataSource) ctx.lookup("java:/comp/env/jdbc/MyLocalDB");
```

This is the way to get the JNDI resources defined to be used by the application. We could have written it in this way too;

```
ctx = new InitialContext();  
Context initCtx = (Context) ctx.lookup("java:/comp/env");  
DataSource ds = (DataSource) initCtx.lookup("jdbc/MyLocalDB");
```

I am also printing some database information to check which database we are connected.

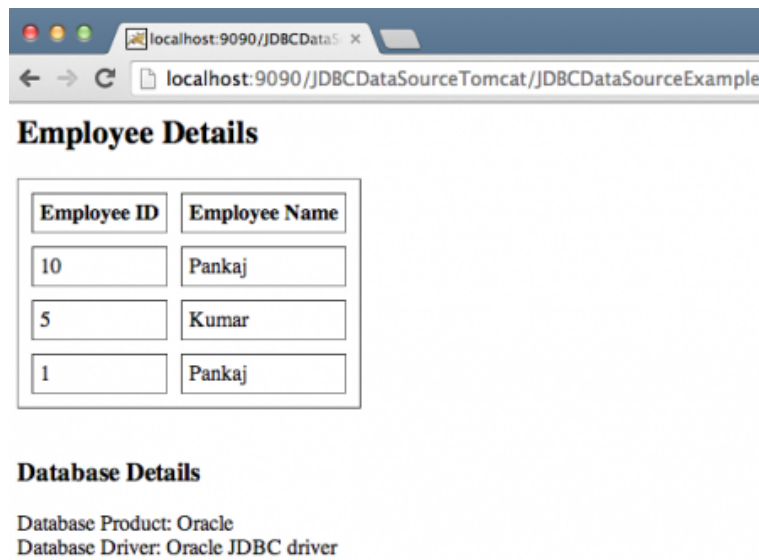
Now when you will run the application, you will see following output.



Let's see how easy it is to switch the database server because we are using Tomcat DataSource. All you need is to change the Database properties. So if we have to switch to Oracle database, my Resource configuration will look like below.

```
<Resource name="jdbc/MyDB"  
  global="jdbc/MyDB"  
  auth="Container"  
  type="javax.sql.DataSource"  
  driverClassName="oracle.jdbc.driver.OracleDriver"  
  url="jdbc:oracle:thin:@localhost:1521:orcl"  
  username="hr"  
  password="oracle"  
  
  maxActive="100"  
  maxIdle="20"  
  minIdle="5"  
  maxWait="10000"/>
```

And when we restart the server and run the application, it will connect to Oracle database and produce below result.



Employee ID	Employee Name
10	Pankaj
5	Kumar
1	Pankaj

Database Details

Database Product: Oracle
Database Driver: Oracle JDBC driver

That's all for Tomcat DataSource JNDI configuration example tutorial, you can define the resource in similar way in context.xml files too.

« PREVIOUS

Java DataSource, JDBC DataSource Example

NEXT »

Java PreparedStatement IN clause alternatives

About Pankaj

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on [Google Plus](#), [Facebook](#) or [Twitter](#). I would love to hear your thoughts and opinions on my

articles directly.

Recently I started creating video tutorials too, so do check out my videos on [Youtube](#).

FILED UNDER: [DATABASE](#), [JAVA EE](#)

Comments

Pran Sukh says

APRIL 17, 2018 AT 11:41 AM

I don't see any error or exception in your point

" If there is a global datasource (server.xml) defined with same name, the application datasource is ignored. "

I tried everything to let this happen..... but nothing happens, i need to raise this issue, why it is not ignoring the application context.xml Datasource when there is Global DataSource in server.xml with same name.

[Reply](#)

Debaranjan Ghosh says

APRIL 6, 2018 AT 9:52 PM

while running this getting following error type Status report
message /JDBCDataSourceExample/
description The requested resource is not available.

[Reply](#)

Pran Sukh says

APRIL 17, 2018 AT 11:44 AM

JDBCDataSourceExample may be your project name but you need to call your servlet class, just put /ServletClassName. Your URL will be like

<http://localhost:8080/JDBCDataSourceExample/ServletClassName>

My project name is RestWithJerrsy and ServletDataSource is servlet class, so the perfect URL is.

<http://localhost:8080/RestWithJerrsy/ServletDataSource>

[Reply](#)

rahul kumar says

JULY 16, 2017 AT 5:47 AM

nice post , each and every thing is define clearly actually i configure in context.xml due to this in my webapp there are 6 folder so it create 6 times connection . i give initialsize 10 so it create 60 connection 10 for each application . but when i conigure in server.xml it work fine and create 10 connection . tanks for this post.

[Reply](#)**Fazeela says**

NOVEMBER 14, 2016 AT 12:28 AM

Nice article.

[Reply](#)**Alberto says**

OCTOBER 26, 2016 AT 11:29 AM

Dear Mr. JournalDev,

Congrats for your job.

I have a web application with a connection pool defined in server.xml of the server (\$CATALINA_HOME/server.xml). I can use this connection pool for JPA connections but I cannot open connection from this pool directly using the context and the lookup of the context. I have tried using ResourceLinks on context.xml of the application which pointed to the dataSource globally defined but it does not work. I have created another connection pool in the context.xml application file and it works, but it is not so efficient. The question is: it is possible to use the global Data Source defined on server.xml of the web application server for connecting database trough the Web Application? Thanks in advance, Alberto.

[Reply](#)**Pankaj says**

OCTOBER 26, 2016 AT 11:49 AM

Yes, that is the topic of this tutorial. The datasource is defined in Tomcat server.xml file and our web application is using that.

[Reply](#)**Sandeep says**

APRIL 28, 2016 AT 1:00 AM

Hi,

what is difference between Resource and ResourceLink in context.xml and server.xml why you have used java:/comp/env/jdbc/MyLocalDB to get DataSource why not jdbc/MyDB. Please let me know as i am beginner

[Reply](#)

Pran Sukh says

APRIL 17, 2018 AT 11:47 AM

Me too..... Please answer.

[Reply](#)

Sandeep says

APRIL 28, 2016 AT 12:25 AM

Hi Pankaj,

what is difference between these two configurations why you are using java:/comp/env/jdbc/MyLocalDB why can't be this java:/comp/env/jdbc/MyDB to get the DataSource

[Reply](#)

Santhoshkumar says

JANUARY 20, 2016 AT 2:38 PM

Hey Everyone,

I need some help.

"java:/comp/env" without providing this how to read JNDI configuration in Tomcat Server. My application is working in Jetty server without any issues, but while migrating to Tomcat i am facing JNDI issues. if i am adding JNDI Prefix it is working fine. I have to change entire application. Without any changes in my application how to resolve this issue.

[Reply](#)

Sandy says

MAY 23, 2017 AT 7:07 AM

you can't bro....

[Reply](#)

Lavanya says

DECEMBER 23, 2015 AT 4:46 AM

what is happening in the following code. please tell me the flow of the program, please let me know what is the difference between test0001 and test003. is resource leak happening in this code ???

```
package data;
import javax.naming.InitialContext;
import javax.naming.NamingException;
public class Sample{
public static void test000(){
try {
InitialContext ctx = new InitialContext(); // jtest is able to find a resource leak here.
} catch (NamingException e) {
e.printStackTrace();
}
}
public void test001(){
try {
InitialContext ctx = new InitialContext();
call001(ctx);
} catch (NamingException e) {
e.printStackTrace();
}
}
public void call001(InitialContext ctx){
// do nothing.
}
public void test002(){
try{
InitialContext ctx = new InitialContext();
call002(ctx);
}catch(NamingException e){
e.printStackTrace();
}
}
private boolean isTrue = false;
public void call002(InitialContext ctx){
if(this.isTrue){
try{
ctx.close();
}catch(NamingException e){
e.printStackTrace();
}
}
}
```

```
}
```

Thanks in advance ☐

[Reply](#)

Pankaj says

DECEMBER 23, 2015 AT 5:34 AM

Well its a long program, but it's clear that you are not closing InitialContext object so resource leak is happening. In case of test000 method, it's clear and since there are no close() call even in any if-else condition, so jtest is able to find it easily. In callo02() method, if condition is used and you can set isTrue to "true" from some other code to close the context.

[Reply](#)

Lavanya says

DECEMBER 28, 2015 AT 2:45 AM

thank you for your response.

is resource leak happening in test001 method???

Thanks for ur time.

[Reply](#)

Sagar says

SEPTEMBER 21, 2015 AT 11:41 PM

Fantastic useful stuff.Thanks bro:)

[Reply](#)

Atelo says

AUGUST 14, 2015 AT 2:14 PM

Hi, Great article, very helpful....

But I have a doubt, which is a difference between:

java:comp/env/jdbc/name_ds

java:/comp/env/jdbc/name_ds

java:jdbc/name_ds

java:/jdbc/name_ds

where can I setup the way to call the jndi? I have this problem, in my pc an example works with "java:/comp/env/jdbc/name_ds", the same example in the my friend computer doesn't work, we changed to "java:/jdbc/name_ds" to work.

We don't know what is the reason...

Anybody can help us?

Thanks

[Reply](#)

Dmitry says

JUNE 13, 2015 AT 11:27 PM

Hello,

I have an error when I run this example.

org.apache.tomcat.dbcp.dbcp.SQLNestedException: Cannot create JDBC driver of class " for connect URL 'null'.

I can't find any solution in google ☐

Pls help me.

[Reply](#)

Yatin says

JUNE 10, 2015 AT 6:05 AM

Hi All,

I have following configuration..

— main-context.xml

–DataSource.xml

WEB-INF/database.properties

<!--

-->

Instead of above definition I have used following definition

–database.properties

jdbc.jndiName=java:comp/env/jdbc/MYAPPDB

–server.xml

I have encrypted datasource password using BASE64Encoder

Here is my EncryptedDataSource class for decrypting password which extends JndiObjectFactoryBean
package my.app.util;

```
import java.io.IOException;
```

```
import javax.naming.NamingException;
```

```
import org.apache.commons.configuration.JNDIConfiguration;
```

```
import org.springframework.jndi.JndiObjectFactoryBean;
```

```
import sun.misc.BASE64Decoder;
```

```
import sun.misc.BASE64Encoder;
```

```
public class EncryptedDataSource extends JndiObjectFactoryBean{
```

```
    /*@Override
```

```
    public String getPassword() {
```

```
        String password = super.getPassword();
```

```
password = decode(password);
return password;
}*/

private String decode(String decode) {
    BASE64Decoder decoder = new BASE64Decoder();
    try {
        decode = new String(decoder.decodeBuffer(decode));
    } catch (IOException e) {
        e.printStackTrace();
    }
    return decode;
}

private String encode(String encode) throws IOException {
    BASE64Encoder encoder = new BASE64Encoder();
    encode = new String(encoder.encodeBuffer(encode.getBytes()));
    return encode;
}

public static void main(String[] args) throws IllegalArgumentException, NamingException {
    EncryptedDataSource ed = new EncryptedDataSource();
    try {
        String password = ed.encode("password");
        System.out.println("password = "+password);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

}

I want to decrypt while connection pooling. Please help.
```

[Reply](#)**Kamil says**

APRIL 28, 2015 AT 1:36 PM

Great article, very helpful. Thanks!

[Reply](#)**Sam says**

MARCH 16, 2015 AT 12:18 AM

This is my first time i'm commenting on the web, because it really works and the article is really great.

[Reply](#)

Jeevitha says

FEBRUARY 17, 2015 AT 9:44 PM

Sir,I would like to know about my project . regarding the connection of servlet code to telegram web for bus booking through online

[Reply](#)**Jorge de Lemos says**

FEBRUARY 13, 2015 AT 5:17 AM

THANK YOU SO MUCH for this post, I had problems to understand this JNDI stuff, I didn't know anything about JNDI and now everything is clear!

Thanks Pankaj ☐

[Reply](#)**Punnoose says**

NOVEMBER 22, 2014 AT 4:42 AM

Nice article. Thank You.

[Reply](#)**Pavan says**

NOVEMBER 18, 2014 AT 8:33 PM

nice post.Very helpful

[Reply](#)**Ali Saleh says**

OCTOBER 20, 2014 AT 11:38 PM

Thank you Pankaj for this useful article,

But I would like to add regarding Eclipse

Don't forget that the Tomcat files will be under the "Server" in the eclipse project hierarchy so modifying them in tomcat alone will not help

thank you again

[Reply](#)

Oleg says

OCTOBER 15, 2014 AT 9:19 AM

Thank for great tutorials.

I have problem with this one. It doesn't show me table, it just show message "TODO write content". What can be wrong?

[Reply](#)

Ju says

OCTOBER 12, 2014 AT 1:56 AM

Thanks a lot for this really clear example.

[Reply](#)

pradeep says

JULY 18, 2014 AT 9:38 PM

Thanks for the post. I read a lot of blogs, and everywhere I saw only the configuration in Server.xml file. I was trying to following the same thing but it never worked, after looking at your explanation, I added the elements in Context.xml and it worked like a charm. Thanks again, brother.

[Reply](#)

irshad says

JUNE 25, 2014 AT 5:17 AM

I have put in server context.xml. but i have put following tag in the web.xml

DB Connection

jdbc/MoranDB

javax.sql.DataSource

Container

=====

i am able to use the Datasource connection. But i want to know the difference between using (server.xml and context.xml) and as i do(context.xml of server and web.xml of application).

PS: i am using tomcat-7 and Mysql database.

please give you insight. thank you

[Reply](#)

irshad says

JUNE 25, 2014 AT 5:18 AM

DB Connection
jdbc/MoranDB
javax.sql.DataSource
Container
[Reply](#)

lalit says

JUNE 12, 2014 AT 10:19 PM

i do not understand this sir plz explain where it is ..."java:/comp/env/jdbc/MyLocalDB"

[Reply](#)

Denn says

SEPTEMBER 30, 2014 AT 11:35 AM

Seems it's typo. Use jdbc/MyDB instead

[Reply](#)

Venkata Sriram says

JUNE 2, 2014 AT 2:04 AM

Hi sir,i created a datasource in Server.xml and writing resourcelink in context.xml but it is not considering second Database Connection Details sir.

in Server.xml under GlobalNamingResources tag i wrote 2 Resource tags for 2 different db connections and in Context.xml i wrote 2 ResourceLink for those 2 different db connections,

First Connection is working where as second one is not working sir.

[Reply](#)

Anshul Jain says

MAY 1, 2014 AT 2:47 AM

THanks for the post, but

I am getting Error: java.lang.ClassCastException: org.apache.tomcat.dbcp.dbcp.BasicDataSource cannot be cast to sun.jdbc.odbc.jdbc4.DriverDataSource

can you please help

[Reply](#)

Oleg says

OCTOBER 15, 2014 AT 2:13 PM

Wrong import lib

[Reply](#)**Zulifqar says**

MARCH 27, 2014 AT 1:58 PM

Excellent post.....Could you please explain how you get this " java:/comp/env".... Sorry I am kind of beginner in Java language. Thanking in advance.

regards,

Zulifqar

[Reply](#)**Pankaj says**

MARCH 27, 2014 AT 7:02 PM

java:comp/env is the root node of JNDI tree from where we can lookup specific JNDI contexts.

[Reply](#)**Arun says**

MARCH 1, 2014 AT 3:30 PM

good explanation...

[Reply](#)**subbareddy says**

FEBRUARY 3, 2014 AT 5:28 AM

nice post.....and where did u get this path java:/comp/env

[Reply](#)**Leave a Reply**

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

☐

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

DOWNLOAD ANDROID APP



RECOMMENDED TUTORIALS

Java Tutorials

- > [Java IO](#)
- > [Java Regular Expressions](#)
- > [Multithreading in Java](#)
- > [Java Logging](#)
- > [Java Annotations](#)
- > [Java XML](#)
- > [Collections in Java](#)
- > [Java Generics](#)
- > [Exception Handling in Java](#)
- > [Java Reflection](#)
- > [Java Design Patterns](#)
- > [JDBC Tutorial](#)

Java EE Tutorials

- > [Servlet JSP Tutorial](#)
- > [Struts2 Tutorial](#)
- > [Spring Tutorial](#)
- > [Hibernate Tutorial](#)
- > [Primefaces Tutorial](#)
- > [Apache Axis 2](#)
- > [JAX-RS](#)
- > [Memcached Tutorial](#)

XML Content Management

XML Web Services

© 2018 · Privacy Policy · Don't copy, it's Bad Karma · P

Client Server

XML Schema

Proxy Servers

Web Servers

Top IT Schools

**Online Software Engineering
Degrees**

