(/)

A Guide to the Java UR

Last modifiedFebruary 12, 202

by baeldun(https://www.baeldung.com/author/bae

Java(https://www.baeldung.com/categor) +

1. Overview

In this article, we are going to explore low-level operations with Java network pro**We'll be taking a deeper look URLs.**

A URL is a reference or an address to a resource on the network. And simply put, Java code communicating ov can use th*java.net.URL*class to represent the addresses of res

The Java platform ships with built-in networking support, bundled *java.ne*package

```
import java.net.*
```

2. Creating a URL

Let's first create*java.net.URL*object by using its constructor and passing in a String representing the human read of the resourc

```
URL url =new URL("/a-guide-to-java-socke");
```

We've just created **absolute URL obje** The address has all the parts required to reach the desired

We can also crea**a relative URL** assuming we have the URL object representing the home page of I

```
URL home =new URL("http://baeldung.co);
```

Next, let's create a new URL pointing to a resource we already know; we're going to use another constructor, tha
existing URL and a resource name relative to that

```
URL url =new URL(home,"a-guide-to-java-socket);
```

We have now created a new URL ol*url* relative t*hom* so the relative URL is only valid within the context of the
URL.

We can see this in a te

```
@Test
public void givenBaseUrl_whenCreatesRelativeUrl_thenCor() {
    URL baseUrl =new URL("http://baeldung.co);
    URL relativeUrl =new URL(baseUrl,"a-guide-to-java-socket);

    assertEquals"http://baeldung.com/a-guide-to-java-soc,
      relativeUrl.toString()
}
```

However if the relative URL is detected to be absolute in its component part*baseURL*is ignored

```
@Test
public void givenAbsoluteUrl_whenIgnoresBaseUrl_thenCor() {
    URL baseUrl =new URL("http://baeldung.co);
    URL relativeUrl =new URL(
      baseUrl,"/a-guide-to-java-socke);

    assertEquals"http://baeldung.com/a-guide-to-java-soc,
      relativeUrl.toString()
}
```

Finally, we can create a URL by calling another constructor which takes in the component parts of the URL s
cover this in the next section after covering URL comp

3. URL Component

A URL is made up of a few components – which we'll explore in this

Let's first look at the separation between the protocol identifier and the resource – these two components ar
colon followed by two forward slashe*//*.

If we have a URL such a*http://baeldung.c* then the part before the separ*http*is the protocol identifier while the
that follows is the resource na*baeldung.co*

Let's have a look at the API that *URL* object expose

3.1. The Protocc

To retrieve**the protocc**– we use th*getProtocol* methoc

```
@Test
public void givenUrl_whenCanIdentifyProtocol_thenCorr(){
    URL url =new URL("http://baeldung.co);

    assertEquals"http", url.getProtocol())
}
```

3.2. The Port

To get**the port**– we use th*getPort(*.methoc

```
@Test
public void givenUrl_whenGetsDefaultPort_thenCorr(){
    URL url =new URL("http://baeldung.co);

    assertEquals(1, url.getPort())
    assertEquals80, url.getDefaultPort()`
}
```

Note that this method retrieves the explicitly defined port. If no port is defined explicitly, it wil

And because HTTP communication uses port 80 by default – no port i

Here's an example where we do have an explicitly defin

```
@Test
public void givenUrl_whenGetsPort_thenCorre(){
    URL url =new URL("http://baeldung.com:809);

    assertEquals8090, url.getPort())
}
```

3.3. The Host

The hosis the part of the resource name that starts right a*//*separator and ends with the domain name extensior case*com*.

We call th*getHost(*method to retrieve the hostna

```
@Test
public void givenUrl_whenCanGetHost_thenCorr(){
    URL url =new URL("http://baeldung.co);

    assertEquals"baeldung.com, url.getHost())
}
```

3.4. The File Nam

Whatever follows after the hostname in a URL is referred t**file name of the resour** It can include both path and qu parameters or just a file na

```
@Test
public void givenUrl_whenCanGetFileName_thenCorre() {
    URL url = new URL("http://baeldung.com/guidelines.);

    assertEquals("/guidelines.txt", url.getFile())
}
```

Assuming Baeldung has java 8 articles under the */articles?topic=java&version=* Everything after the hostname is file name

```
@Test
public void givenUrl_whenCanGetFileName_thenCorre() {
    URL url = new URL("http://baeldung.com/articles?topic=java&vers);

    assertEquals("/articles?topic=java&version", url.getFile())
}
```

3.5. Path Parameter

We can also only inspect **the path** parameters which in our case */article*

```
@Test
public void givenUrl_whenCanGetPathParams_thenCorr() {
    URL url = new URL("http://baeldung.com/articles?topic=java&vers);

    assertEquals("/articles", url.getPath())
}
```

3.6. Query Parameter

Likewise, we can inspect **the query parameter** which is *topic=java&version=8*

```
@Test
public void givenUrl_whenCanGetQueryParams_thenCorr() {
    URL url = new URL("http://baeldung.com/articles?topic=java<em>&version=8);

    assertEquals("topic=java<em>&version=8</e", url.getQuery())
}
```

4. Creating URL With Component Pa

Since we have now looked at the different URL components and their place in forming the complete address to t can look at another method of creating a URL object by passing in the compo

The first constructor takes the protocol, the hostname and the file name re

```
@Test
public void givenUrlComponents_whenConstructsCompleteUrl_thenCor() {
    String protocol ="http";
    String host ="baeldung.com;
    String file ="/guidelines.txt;
    URL url =new URL(protocol, host, file

    assertEquals"http://baeldung.com/guidelines., url.toString())
}
```

Keep in mind the meaning of filename in this context, the following test should mak

```
@Test
public void givenUrlComponents_whenConstructsCompleteUrl_thenCorr() {
    String protocol ="http";
    String host ="baeldung.com;
    String file ="/articles?topic=java&versior;
    URL url =new URL(protocol, host, file

    assertEquals"http://baeldung.com/articles?topic=java&versi, url.toString())
}
```

The second constructor takes the protocol, the hostname, the port number and the filenam

```
@Test
public void givenUrlComponentsWithPort_whenConstructsComplete
  thenCorrect() {
    String protocol ="http";
    String host ="baeldung.com;
    int port = 9000;
    String file ="/guidelines.txt;
    URL url =new URL(protocol, host, port, file

    assertEquals
      "http://baeldung.com:9000/guidelines., url.toString())
}
```

5. Conclusio

In this tutorial, we covered t*URL* class and showed how to use it in Java to access network resources progra

As always, the full source code for the article and all code snippets can be fGitHub projec(https://github.cc /eugenp/tutorials/tree/master/core-java-modules/core-jav).

Learning to "Build your AI
with Sprin"?

Enter your email addr

>> Get the eBoo

Comments are closed on this ar

CATEGORIES

ezoic (https://www.ezoic.com/what-is-

report this ad

SPRING (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SPRIN
REST (HTTPS://WWW.BAELDUNG.COM/CATEGORY/RES
JAVA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JAV
SECURITY (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SECURITY-
PERSISTENCE (HTTPS://WWW.BAELDUNG.COM/CATEGORY/PERSISTENC
JACKSON (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JSON/JACKSO
HTTP CLIENT-SIDE (HTTPS://WWW.BAELDUNG.COM/CATEGORY/HTT
KOTLIN (HTTPS://WWW.BAELDUNG.COM/CATEGORY/KOTLIN

SERIES

JAVA "BACK TO BASICS" TUTORIAL (/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (/JACKSON)
HTTPCLIENT 4 TUTORIAL (/HTTPCLIENT-GUIDE)
REST WITH SPRING TUTORIAL (/REST-WITH-SPRING-SERIES)
SPRING PERSISTENCE TUTORIAL (/PERSISTENCE-WITH-SPRING-SERIES)
SECURITY WITH SPRING (/SECURITY-SPRING)

ABOUT

ABOUT BAELDUNG (/ABOUT)
THE COURSES (HTTPS://COURSES.BAELDUNG.COM)
JOBS (/TAG/ACTIVE-JOB)
THE FULL ARCHIVE (/FULL_ARCHIVE)
WRITE FOR BAELDUNG (/CONTRIBUTION-GUIDELINES)
EDITORS (/EDITORS)
OUR PARTNERS (/PARTNERS)
ADVERTISE ON BAELDUNG (/ADVERTISE)

TERMS OF SERVICE (/TERMS-OF-SERVICE)
PRIVACY POLICY (/PRIVACY-POLICY)
COMPANY INFO (/BAELDUNG-COMPANY-INFO)
CONTACT (/CONTACT)