Apache Commons UrlValidator Testing
CS 362 - Final Project

Winter 2016
Oregon State University

Marco Zamora
Jason Ridder
James Linnenburger

Contents:

Overview:

For this project, we were to use what we have learned this quarter to test the Apache Commons UrlValidator program. The intent of UrlValidator is to tell whether or not a given String would constitute a valid URL or not. This tool can (and is) imported into many different projects that are used for a wide variety of purposes. As such, it is critical that the UrlValidator functions as intended and should be thoroughly tested and debugged. For this particular project, we were provided a version that had intentionally been altered to introduce several different bugs. Our goal was to find as many bugs as we could using our test suite that we developed specifically for this task.

Methodology:

One of the our first objectives as a team was to figure out what exactly made up an URL(Agans' rule #1 "Understand the System"). We did this by reading the specifications for URLs , articles written about them, and other searches. Even with all the information we obtained, we still had lengthy discussions about what was and was not allowed as well as how certain parts of URLvalidator worked. It helped to bounce ideas off of each other. Once we had a stronger understanding of how URL were constructed, we formulated a plan to move forward with test construction. We decided on three phases of testing: manual testing, Input partition testing, and programmed (unit) testing. Each phase of testing helped to inform the later phases of testing, so manual testing was completed before input partitioning, and the programmed testing was done last.

Collaboration Summary:

Since the testing was split into three sections it only made sense for us to divide the work that way. Marco was in charge of manual, Jason did input partition, and finally James worked on programming based testing. Each person owned their section but all of us had a hand in each other's work double checking it. For collaboration, we used a mixture of of gmail and google hangouts. Hangouts was useful because it allowed us share our screens to talk about code, debug in eclipse, or share pictures/drawings about ideas we had.

In our testing we didn't think too much(Agans' rule #3 "Quit Thinking and Look"). After we had found out what elements composed a correct URL, we tested correct and incorrect versions of it. If we found out there was a bug we used Eclipses tools to help track it down.

Manual Testing:

For the manual testing component of our test suite, it was decided to pick several different common URLs and also include some paths and query information. Our intent for manual testing was to make sure that some 'obviously' valid and invalid URLs came back as such after being passed to the UrlValidator. In addition to obvious URLs, we also included URLs that included some basic paths and queries. Below are a few samples of the manual tests that were used:

Sample Manual Test Code:

```
System.out.println("http://www.amazon.com - Should be TRUE");
System.out.println(urlVal2.isValid("http://www.amazon.com"));

System.out.println("http://www.website.com/inner is valid? " +
                   urlVal.isValid("http://www.website.com/inner"));
assertTrue(urlVal.isValid("http://www.website.com/inner"));

System.out.println("adfgdhewr - Should be FALSE");
System.out.println(urlVal2.isValid("adfgdhewr"));

System.out.println("http://www.amazon.biz - Should be TRUE");
System.out.println(urlVal2.isValid("http://www.amazon.biz"));

System.out.println("http://sdfgsdfg.me - Should be TRUE");
System.out.println(urlVal2.isValid("http://sdfgsdfg.me"));

System.out.println("http://www.sdfgsdfg.com - Should be TRUE");
System.out.println(urlVal2.isValid("http://www.sdfgsdfg.com"));

System.out.println("http://www.website.com/inner?name=Ted is valid? " +
                   urlVal.isValid("http://www.website.com/inner?name=Ted"));
assertTrue(urlVal.isValid("http://www.website.com/inner?name=Ted"));
```

Bugs Found through Manual Testing:

Bug ID: **Validator - 401**

Summary: *UrlValidator.isValid returns incorrect value for: http://www.website.com/inner?name=Ted*

Type: *Bug*
Status: *OPEN*
Priority: *Major*
Resolution: *Unresolved*
Affects Ver: *1.5.0*
Fix Ver: *None*
Components: *Routines*
Labels: *Easy fix*

Description:
The method isValid returns false when it should return true for **http://www.website.com/inner?name=Ted** . On line 313 in the file UrlValidator.java isValidQuery returns false when it shouldn't return anything at all. Line pictured below.

```
313         if (!isValidQuery(urlMatcher.group(PARSE_URL_QUERY))) {
314             return false;
315         }
```

Actual test code:

```
assertTrue(urlVal.isValid("http://www.website.com/inner?name=Ted"))
;
```

Bug ID: **Validator - 402**

Summary: *UrlValidator.isValid returns incorrect value for: http://join.me*

Type: *Bug*
Status: *OPEN*
Priority: *Major*
Resolution: *Unresolved*
Affects Ver: *1.5.0*
Fix Ver: *None*
Components: *Routines*
Labels: *Easy fix*

Description:
The method isValid returns false when it should return true for http://join.me . On line 385 in the file UrlValidator.java isValid(hostLocation) returns false when it should be true. Line pictured below.

```
385             if (!inetAddressValidator.isValid(hostLocation)) {
386                 // isn't either one, so the URL is invalid
387                 return false;
388             }
```

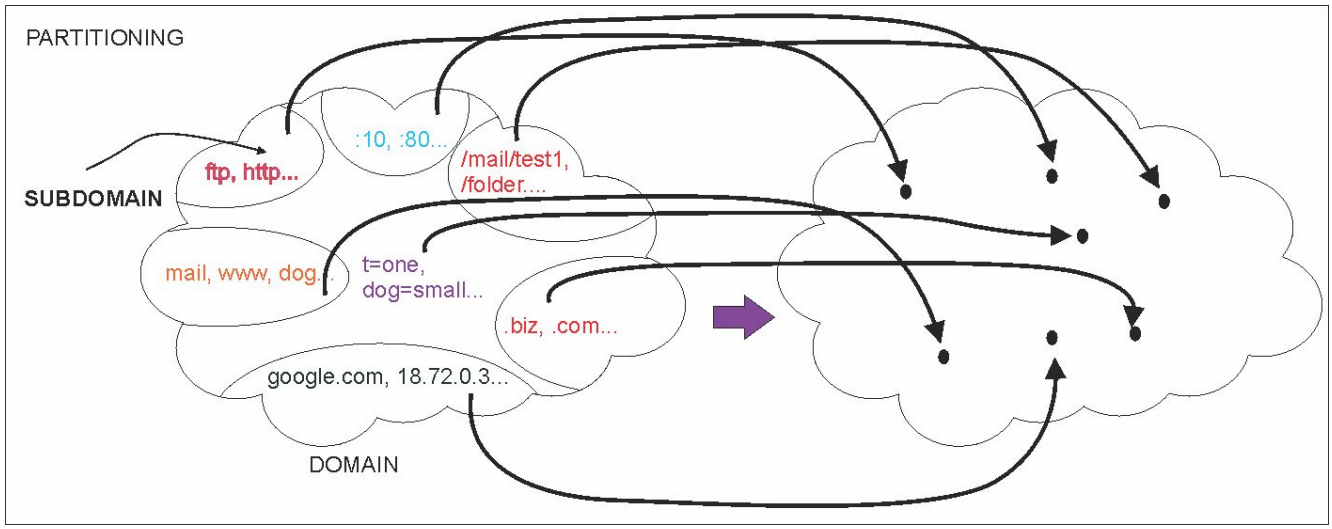
Actual test code:

```
assertTrue(urlVal.isValid("http://join.me"));
```

Input Partition Testing:

For this component of our testing suite, we examined the different components of what actually makes up a URL. A URL is made up of schemes, host, subdomain, top-level domain, second-level domain, port, URL-path, and URL-parameters. Each of these categories will be split into their own partition and tested, because there are so many test cases for each one it would be impossible to test them all. Testing includes correct and incorrect URLs for each subdomain tested.

Picking one component from each subdomain and testing it:



Default URL for testing: http://www.google.com

Sample Partition Test Code:

URL = {set of all strings}
URL1(Schemes) = Test Example:

```
assertTrue("http://www.espn.com - Should be TRUE",  urlVal.isValid("http://www.espn.com"));
 ssertFalse("http://www.?.com - Should be FALSE", urlVal.isValid("http://www.?.com"));
```

URL2(Host) = Test Example:

```
assertTrue("http://www.espn.com - Should be TRUE",  urlVal.isValid("http://www.espn.com"));
assertFalse("http://www.?.com - Should be FALSE", urlVal.isValid("http://www.?.com"));
```

URL3(Subdomain) = Test Example:

```
assertTrue("http://mail.google.com - Should be TRUE", urlVal.isValid("http://mail.google.com"));
assertFalse("http://?.google.com - Should be FALSE", urlVal.isValid("http://?.google.com"));
```

URL4(Top-level Domain) = Test Example:

```
assertTrue("http://www.google.me - Should be TRUE", urlVal.isValid("http://www.google.me"));
assertFalse("http://www.google.? - Should be FALSE", urlVal.isValid("http://www.google.?"));
```

URL5(Second-level Domain) = Test Example:

```
assertTrue("http://www.google.com.su - Should be TRUE",
            urlVal.isValid("http://www.google.com.su"));
assertFalse("http://www.google.?.su - Should be FALSE",
             urlVal.isValid("http://www.google.?.su"));
```

URL6(Port) = Test Example:

```
assertTrue("http://www.google.com:10 - Should be TRUE",
```

```
            urlVal.isValid("http://www.google.com:10"));
assertFalse("http://www.google.com:^ - Should be FALSE",
             urlVal.isValid("http://www.google.com:^"))
```

URL7(URL-Path) = Test Example:

```
assertTrue("http://www.google.com/test1 - Should be TRUE",
            urlVal.isValid("http://www.google.com/test1"));
assertFalse("http://www.google.com/*&^% - Should be FALSE",
             urlVal.isValid("http://www.google.com/*&^%"));
```

URL8(URL-Parameters) = Test Example:

```
assertTrue("http://www.google.com/test1?t=one - Should be TRUE",
            urlVal.isValid("http://www.google.com/test1?t=one"));
assertFalse("http://www.google.com/test1?t=@@@@@@ - Should be FALSE",
             urlVal.isValid("http://www.google.com/test1?t=@@@@@@")
```

Bugs Found through Partition Testing:

Bug ID: **Validator - 403**

Summary: *UrlValidator.isValid returns incorrect value for: http://www.google.me*

Type: *Bug*
Status: *OPEN*
Priority: *Major*
Resolution: *Unresolved*
Affects Ver: *1.5.0*
Fix Ver: *None*
Components: *Routines*
Labels: *Easy fix*

Description:
The method isValid returns false when it should return true for http://www.google.me, which is a top level domain URL. On line 305 in the file UrlValidator.java isValidAuthority(authority) returns false when it shouldn't return anything at all. Line pictured below.

```
304             // Validate the authority
305             if (!isValidAuthority(authority)) {
306                 return false;
307              }
```

Actual test code:

```
UrlValidator urlVal = new UrlValidator(null, null,UrlValidator.ALLOW_ALL_SCHEMES);
assertTrue("http://www.google.me - Should be TRUE",
             urlVal.isValid("http://www.google.me"));
assertFalse("http://www.google.? - Should be FALSE",
              urlVal.isValid("http://www.google.?"));
```

Bug ID: **Validator - 404**

Summary: *UrlValidator.isValid returns incorrect value for: http://www.google.com.su*

Type: *Bug*
Status: *OPEN*
Priority: *Major*
Resolution: *Unresolved*
Affects Ver: *1.5.0*
Fix Ver: *None*
Components: *Routines*
Labels: *Easy fix*

Description:
The method isValid returns false when it should return true for http://www.google.com.su, which is an second level domain URL. On line 305 in the file UrlValidator.java isValidAuthority(authority) returns false when it shouldn't return anything at all. Line pictured below.

```
304             // Validate the authority
305             if (!isValidAuthority(authority)) {
306                 return false;
307              }
```

Actual test code:

```
UrlValidator urlVal = new UrlValidator(null, null, UrlValidator.ALLOW_ALL_SCHEMES);
assertTrue("http://www.google.com.su - Should be TRUE",
            urlVal.isValid("http://www.google.com.su"));
assertFalse("http://www.google.?.su - Should be FALSE",
             urlVal.isValid("http://www.google.?.su"));
```

Bug ID: **Validator - 405**

Summary: *UrlValidator.isValid returns incorrect value for: http://www.google.com/test1?t=on*

Type: *Bug*
Status: *OPEN*
Priority: *Major*
Resolution: *Unresolved*
Affects Ver: *1.5.0*
Fix Ver: *None*
Components: *Routines*
Labels: *Easy fix*

Description:
The method isValid returns false when it should return true for http://www.google.com/test1?t=on. The URL has a valid scheme(http://),

Authority(www.google.com), path(test1) and query(?t=on). On line 314in the file UrlValidator.java, The isValidQuery function returns false when it should return true.

```
312         }
313
314         if (!isValidQuery(urlMatcher.group(PARSE_URL_QUERY))) {
315             return false;
316         }
317
318         if (!isValidFragment(urlMatcher.group(PARSE_URL_FRAGMENT))) {
319
320             return false;
321
322         }
```

Actual test code:

```
UrlValidator urlVal = new UrlValidator(null, null, UrlValidator.ALLOW_ALL_SCHEMES);
assertTrue("http://www.google.com/test1?t=one - Should be TRUE",
            urlVal.isValid("http://www.google.com/test1?t=one"));
assertFalse("http://www.google.com/test1?t=@@@@@@ - Should be FALSE",
             urlVal.isValid("http://www.google.com/test1?t=@@@@@@"));
```

Programming Based Testing:

For the programmed based testing component of our test suite, we designed a single unit test that generated a large number of different URLs based on lists of provided components. The components each were tagged with whether or not they were valid components so an overall validity of the constructed URL could be determined before sending into the UrlValidator isValid() method. The program thoroughly performed the construction of over 200,000 unique URLs and tested each one.

Sample Programmed Testing Code:

The method below shows how the URLs were automatically generated and tested. For the full source code used in the programed base you can look in the “URLValidator” folder of user “linnenbj” in the class git repository. Programming testing code files used include: UrlComponent.java, Host.java, TestURL.java and UrlValidatorTesting.java.

```
public void testIsValid()
  {
      UrlValidator urlVal = new UrlValidator(null, null, UrlValidator.ALLOW_ALL_SCHEMES);

      ArrayList<UrlComponent> protocols = new ArrayList<UrlComponent>();
      ArrayList<Host> hosts = new ArrayList<Host>();
      ArrayList<UrlComponent> ports = new ArrayList<UrlComponent>();
      ArrayList<UrlComponent> paths = new ArrayList<UrlComponent>();
      ArrayList<UrlComponent> parameters = new ArrayList<UrlComponent>();
      buildComponents(protocols, hosts, ports, paths, parameters);
```

```
        TestURL test;

        for(UrlComponent protocol : protocols)
            for(Host host : hosts)
                for(UrlComponent port : ports)
                    for(UrlComponent path : paths)
                        for(UrlComponent parameter : parameters)
                        {
                            test = new TestURL(protocol, host, port, path, parameter);
                            assertEquals(urlVal.isValid(test.toString()), test.isValid());
                        }
    }
```

Bugs Found through Programmed Testing Testing:

Bug ID: **Validator - 406**

Summary: *UrlValidator.isValid returns incorrect value for: http://video.espn.com:80/test1?action=view*

Type: *Bug*
Status: *OPEN*
Priority: *Major*
Resolution: *Unresolved*
Affects Ver: *1.5.0*
Fix Ver: *None*
Components: *Routines*
Labels: *Easy fix*

Description:
The method isValid() returns a 'false' when it should return a true for the url "http://video.espn.com:80/test1?action=view". On line 446 of the file UrlValidator.java, the return statement is returning the opposite value of what it should be.

```
441    protected boolean isValidQuery(String query) {
442        if (query == null) {
443            return true;
444        }
445
446        return !QUERY_PATTERN.matcher(query).matches();
447    }
```

Actual test code:

```
UrlValidator urlVal = new UrlValidator(null, null, UrlValidator.ALLOW_ALL_SCHEMES);
test = new TestURL(protocol, host, port, path, parameter);
//test.toString() will return "http://video.espn.com:80/test1?action=view"
assertEquals(urlVal.isValid(test.toString()), test.isValid());
```

Bug ID: **Validator - 407**

Summary: *UrlValidator.isValid returns incorrect value for: http://video.espn.com:65535/test1?action=view*

Type: *Bug*
Status: *OPEN*
Priority: *Major*
Resolution: *Unresolved*
Affects Ver: *1.5.0*
Fix Ver: *None*
Components: *Routines*
Labels: *Easy fix*

Description:
The method isValid() returns a 'false' when it should return a true for the url "http://video.espn.com:65535/test1?action=view". (In fact, this bug makes an incorrect isValid() return on any URL that has a port number that is 4 or more digits.) The problem is a faulty regex pattern that can be found on line 158 of the file UrlValidator.java.

```
158     private static final String PORT_REGEX = "^:(\\d{1,3})$";
159     private static final Pattern PORT_PATTERN = Pattern.compile(PORT_REGEX);
```

Actual test code:

```
UrlValidator urlVal = new UrlValidator(null, null, UrlValidator.ALLOW_ALL_SCHEMES);
test = new TestURL(protocol, host, port, path, parameter);
//test.toString() will return "http://video.espn.com:65535/test1?action=view
assertEquals(urlVal.isValid(test.toString()), test.isValid());
```

Bug ID: **Validator - 408**

Summary: *UrlValidator.isValid returns incorrect value for: http://video.espn.co.uk:80/test1?action=view*

Type: *Bug*
Status: *OPEN*
Priority: *Major*
Resolution: *Unresolved*
Affects Ver: *1.5.0*
Fix Ver: *None*
Components: *Routines*
Labels: *Easy fix*

Description:
The method isValid() in the file UrlValidator.java returns a 'false' when it should return a 'true' for any valid hostname that ends in '.co.uk'. The problem lies within the DomainValidator.isValid() method in the DomainVAlidator.java file

Actual test code:

```
UrlValidator urlVal = new UrlValidator(null, null, UrlValidator.ALLOW_ALL_SCHEMES);
test = new TestURL(protocol, host, port, path, parameter);
//test.toString() will return “http://www.google.co.uk”
assertEquals(urlVal.isValid(test.toString()), test.isValid());
```

Sources:

Berners-Lee, T., L. Masinter, and M. McCahill. "RFC #1738."*Www.ietf.org*. N.p., 1 Dec. 1994. Web. 14 Mar. 2016. <https://www.ietf.org/rfc/rfc1738.txt>.

Cutts, Matt. "Talk like a Googler: Parts of a Url." Matt Cutts Gadgets Google and SEO. Matt Cutts, 15 Aug. 2007. Web. 14 Mar. 2016. <https://www.mattcutts.com/blog/seo-glossary-url-definitions/>.

"UrlValidator (Apache Commons Validator 1.5.0 API)." UrlValidator (Apache Commons Validator 1.5.0 API). Apache Software Foundation, 2015. Web. 14 Mar. 2016. <https://commons.apache.org/proper/commons-validator/apidocs/org/apache/commons/validator/UrlValidator.html> .

Bennett, Sean, and John Regehr. "Software Testing | Udacity." Software Testing | Udacity. Udacity, n.d. Web. 14 Mar. 2016. <https://www.udacity.com/course/software-testing--cs258>.