/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Input Partitioning Ideas

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

I decided to partition the inputs for the URL validator into five categories. These are schemes, authorities, ports, paths, and queries. I show the inputs I used for partitioning and an explanation beneath each one for why I chose the inputs I did.

private ResultPair[] testUrlSchemes = {

new ResultPair("", true),

new ResultPair("http://", true),

new ResultPair("https://", true),

new ResultPair("ftp://", true),

new ResultPair("htt://", false),

new ResultPair("http:", false),

new ResultPair("http:/", false),

new ResultPair("https/", false),

new ResultPair("///", false)

};

For Schemes I probably included more options than is strictly necessary to fully isolate different possible inputs. I wanted to first include the most common schemes (http, ftp, and http), and an empty scheme. After that, I misformed the http scheme in a few ways. I figured it was unnecessary to misform ftp and https, since this would probably cover those possible situations. Here I found that it returned as false when the scheme was missing and returned true when it was set to “htt://”. Otherwise, It worked.

private ResultPair[] testAuthorities = {

new ResultPair("www.google.com", true),

new ResultPair("amazon.com", true),

new ResultPair("uk.yahoo.com", true),

new ResultPair("www.yahoo.co.uk", true),

new ResultPair("yahoo.co.uk", true),

new ResultPair("0.0.0.0", true),

new ResultPair("192.168.0.1", true),

new ResultPair("192.174.456.1", false),

new ResultPair("223.com", true),

new ResultPair("www.google.ninja", true),

new ResultPair("www.google.net", true),

new ResultPair("www.google.gov", true),

new ResultPair("www.google.edu", true),

new ResultPair("www.google.io", true),

new ResultPair("docs.google.com", true),

new ResultPair("www.google.444", false),

new ResultPair("192.145.3.2.", false),

new ResultPair("192.168.1", false),

new ResultPair(".4.3.4.5", false),

new ResultPair("", false)};

There is quite a bit of partitioning required for the authority part of the URL. This is because it has 3 somewhat independent parts, and it can also be an IP address. I started by removing or malforming the first third and the TLD(last third). There is no reason to do that with the middle, since it can be almost anything and be valid. I also wanted to test various TLDs, since there are many available now, but there are still values that can be incorrect. Lastly I tested a few IP adddresses. Some malformed and some not. In my testing I found that it doesn’t have the newest TLD’s available, or at least it doesn’t have “.ninja”. Also, it returned an invalid IP address "192.174.456.1" as a valid possible Authority. Also, it didn’t recoginze the TLD “.co.uk”. Otherwise, it did a pretty good job.

private ResultPair[] testPorts = {

new ResultPair(":80", true),

new ResultPair(":0", true),

new ResultPair("", true),

new ResultPair(":-1", false),

new ResultPair(":65636", true),

new ResultPair(":65a", false)

};

In testing the Ports I figured the most important inputs to test were, a valid port, a missing port, an invalid port, and an unusual port. I found that the only one that cause issues was having an unusual port (“65636”). As far as I know, this should return as a valid URL and still work.

private ResultPair[] testPaths = {

new ResultPair("/mail/u/2", true),

new ResultPair("/mail", true),

new ResultPair("/$37", true),

new ResultPair("/mail/", true),

new ResultPair("/..", false),

new ResultPair("/../", false),

new ResultPair("", true),

new ResultPair("/", true),

new ResultPair("/..//mail", false),

new ResultPair("/inbox//mail", false)

};

For the path input partitioning I wanted to test having it missing or just an empty slash, one value, multiple values, and malforming it in the way you might want to do if attempting to traverse a linux file system, or just straight up incorrect malformed values. All of the examples I used above passed validation.

private ResultPair[] testQueries = {

new ResultPair("", true),

new ResultPair("?twenty=20&UnitID=439", true),

new ResultPair("?stuff=true", true)

};

Queries was the simplest to partition. As far as I know, there are really only three main partitions. An empty query, a query with one value and a query with multiple values. The query logic is very broken, and seems to return false every time it’s anything but empty.

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Bugs found via unit test

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

These appear to be the same issue. It doesn't recognize having a URL query as valid when it is.

http://www.google.com:80/mail/u/2?twenty=20&UnitID=439 isValid(): false expected: true

http://www.google.com:80/mail/u/2?stuff=true isValid(): false expected: true

Adding a weird port number should work. This says it's invalid when it's probably valid.

http://www.google.com:65636/mail/u/2 isValid(): false expected: true

It returns as false when the scheme is missing. But, it can be missing and be valid.

www.google.com:80/mail/u/2 isValid(): false expected: true

It doesn't recognize .co.uk as a valid TLD.

http://www.yahoo.co.uk:80/mail/u/2 isValid(): false expected: true

It doesn't realize that this invalid IP address is invalid. It returns true when it should be false.

http://192.174.456.1:80/mail/u/2 isValid(): true expected: false

Newer TLDs are missing. (Yes .ninja is now a valid TLD)

http://www.google.ninja:80/mail/u/2 isValid(): false expected: true

The scheme htt:// is not valid.

htt://www.google.com:80/mail/u/2 isValid(): true expected: false

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

START - Gerald - Location of bugs in debugger - URLValidator.java

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Areas in URLValidator

These appear to be the same issue. It doesn't recognize having a URL query as valid when it is.

http://www.google.com:80/mail/u/2?twenty=20&UnitID=439 isValid(): false expected: true

breaks

line 314: if (!isValidQuery(urlMatcher.group(PARSE\_URL\_QUERY))) {

http://www.google.com:80/mail/u/2?stuff=true isValid(): false expected: true

breaks

line 314: if (!isValidQuery(urlMatcher.group(PARSE\_URL\_QUERY))) {

Adding a weird port number should work. This says it's invalid when it's probably valid.

http://www.google.com:65636/mail/u/2 isValid(): false expected: true

breaks

line 393: if (!PORT\_PATTERN.matcher(port).matches()) {

It returns as false when the scheme is missing. But, it can be missing and be valid.

www.google.com:80/mail/u/2 isValid(): false expected: true

breaks

line 361: if (authority == null) {

It doesn't recognize .co.uk as a valid TLD.

http://www.yahoo.co.uk:80/mail/u/2 isValid(): false expected: true

breaks

line 385: if (!inetAddressValidator.isValid(hostLocation)) {

It doesn't realize that this invalid IP address is invalid. It returns true when it should be false.

http://192.174.456.1:80/mail/u/2 isValid(): true expected: false

NOTE: Hard to test as everything passed but these were the functions that it ran through

Returns true in:

isValidScheme()

isValidAuthority() This should have returned false

isValidPath()

isValidQuery()

isValidFragment()

Newer TLDs are missing. (Yes .ninja is now a valid TLD)

http://www.google.ninja:80/mail/u/2 isValid(): false expected: true

breaks

line 385: if (!inetAddressValidator.isValid(hostLocation)) {

The scheme htt:// is not valid.

htt://www.google.com:80/mail/u/2 isValid(): true expected: false

NOTE: Hard to test as everything passed but these were the functions that it ran through

Returns true in:

isValidScheme() This should have returned false

isValidAuthority()

isValidPath()

isValidQuery()

isValidFragment()

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

END - Gerald - Location of bugs in debugger - URLValidator.java

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Scheme bugs:

Line 295 and 337 should be return true;