Blaze

Tuesday, May 28, 2019 11:20 AM

D:\Users\tmppv4\Projects\react-example-master>npm config set https-proxy http://tmppv4:H5BMfLLY@proxy.sg.uobnet.com:8080/

Bitbucket URL: https://bitbucket.sg.uobnet.com/ (tmppv4::tmppv4)

Network Drive: \\ntststsg99\QC11\Temp\Jake\Notepad++Portable\App\Notepad++

```
INSERT INTO TEDS_B_ROLE_PERMISSION (ROLE, PERMISSION, READ_WRITE, ENABLED, UPDATED_ON, UPDATED_BY)
VALUES ('Finance', 'contractquery', 'READ', 1, SYSTIMESTAMP, 'V1.0.0');
```

To Approve Pull Request: **admin:n!mda**

WAS Console - https://lxedttsgv99:9043/ibm/console/login.do

SIT URL: http://lxedttsgv99:9080/blaze-web/home
UAT URL: http://lxedtusgv01:9080/blaze-web/login

JIRA

Wednesday, June 19, 2019 9:50 AM

JIRA URL: https://jiraagile.sg.uobnet.com/browse/EDT-60

JIRAs worked
EDT-44: Trade Query Ext API
EDT-66: Tables should have horizontal scroll
EDT-79 : Rates viewer screen for Channel User
EDT-77 : PFS Wealth - PFS specific fields
EDT-60 : GM Login Permissions
EDT-165: Quotes Viewer navigation link should have same title
EDT-166: Sales Blotter page title is wrong
EDT-227: PFS Wealth Trade Entry - remove transaction type

Pull Requests
EDT-44 : http://lxspmtsgv99:7990/projects/FR2/repos/fr2/pull-requests/338/overview

EDT-66 : http://lxspmtsgv99:7990/projects/FR2/repos/fr2/pull-requests/344/overview

EDT-79 : http://lxspmtsgv99:7990/projects/FR2/repos/fr2/pull-requests/374/overview
http://lxspmtsgv99:7990/projects/FR2/repos/fr2/pull-requests/378/overview

EDT-77 : http://lxspmtsgv99:7990/projects/FR2/repos/fr2/pull-requests/383/overview
http://lxspmtsgv99:7990/projects/FR2/repos/fr2/pull-requests/420/overview
https://bitbucket.sg.uobnet.com/projects/ED/repos/edist-repo/pull-requests/7
https://bitbucket.sg.uobnet.com/projects/ED/repos/edist-repo/pull-requests/8
https://bitbucket.sg.uobnet.com/projects/ED/repos/edist-repo/pull-requests/11

EDT-60 : http://lxspmtsgv99:7990/projects/FR2/repos/fr2/pull-requests/422/overview
http://lxspmtsgv99:7990/projects/FR2/repos/fr2/pull-requests/432/overview

EDT-165 , EDT-166: https://bitbucket.sg.uobnet.com/projects/ED/repos/edist-repo/pull-requests/45/overview

EDT-227: https://bitbucket.sg.uobnet.com/projects/ED/repos/edist-repo/pull-requests/215/overview

SQLs

Tuesday, July 09, 2019 10:10 AM

<u>CU User Info:</u>

```
SELECT A.CHANNEL_USER_ID AS USER_ID, A.NAME AS USER_NAME, A.LOGIN,
B.GROUP_ID, B.GROUP_NAME, B.IS_DELETED,
C.ROLE, C.PERMISSION, C.READ_WRITE, C.ENABLED
FROM TEDS_B_CHANNEL_USER A, TEDS_B_CHANNEL_USER_GROUPS B, TEDS_B_ROLE_PERMISSION C
WHERE A.CHANNEL_USER_GROUP = B.GROUP_ID AND B.ROLE = C.ROLE AND A.LOGIN = 'xxxpfs'
```

<u>GM User Info:</u>

```
SELECT A.GM_USER_ID as USER_ID, A.NAME as USER_NAME, A.LOGIN,
B.GROUP_ID, B.GROUP_NAME, B.IS_DELETED,
C.ROLE, C.PERMISSION, C.READ_WRITE, C.ENABLED
FROM TEDS_B_GM_USER A, TEDS_B_GM_USER_GROUP B, TEDS_B_ROLE_PERMISSION C
WHERE A.GM_USER_GROUP = B.GROUP_ID
  AND B.ROLE = C.ROLE  AND A.LOGIN = 'xxxalf'
```

ARM

Wednesday, May 29, 2019 9:37 AM

try-with-resources Statement for the Impatient

Java application manipulates several **types of resources such as files, streams, sockets, and database connections**. Such resources must be handled with great care, because they acquire system resources for their operations. Thus, you need to ensure that they get freed even in case of errors.

A lot of boilerplate code is needed to ensure that resources are properly closed. Such boilerplate code makes it harder to read the business logic of a method. Even worse, it requires discipline from developers because it is easy to write the error handling and resource closing logic incorrectly.

Even, other programming languages have introduced constructs for simplifying the handling of such cases.

Ruby Example:
`def writing_in_ruby`
`    File.open('rdata', 'w') do |f|`
`        f.write(666)`
`        f.write("Hello")`
`    end`
`end`
The File.open method takes a block of code to be executed, and ensures that the opened file is closed even if the block's execution emits an exception.
	Python Example:
	`def writing_in_python():`
	`    with open("pdata", "w") as f:`
	`        f.write(str(666))`
	`        f.write("Hello")`
	
	The special with statement takes an object that has a close method and a code block. Again, it ensures proper resource closing no matter if an exception is thrown or not.

Java introduced a similar language construct called **try-with-resources for automatic resource management** (ARM in short)

```
try (DataOutputStream out = new DataOutputStream(new FileOutputStream("data"))) {
    out.writeInt(666);
    out.writeUTF("Hello");
}
```

Such a try-with-resources statement **may be followed by catch and finally blocks**, just like regular try statements prior to Java SE 7
A new interface called java.lang.AutoCloseable was introduced. All it does is provide a void method named close() that may throw a checked exception (java.lang.Exception). **Any class willing to participate in try-with-resources statements should implement the java.lang.AutoCloseable interface**.

The benefits of the try-with-resources statement in Java SE 7 are self-explanatory for such an example:
You have **less code to write**, the code is **easier to read**, and, last but not least, the **code does not leak resources**!

The close() methods have been retro-fitted into many classes of the standard Java SE run-time environment , including the java.io, java.nio, javax.crypto, java.security, java.util.zip, java.util.jar, javax.net, and java.sql packages

Auto-Closeable that throws Exception
This is to demonstrate the multiple exception scenario and how Exception is suppressed by the Auto-closeable.

```
public class AutoClose implements AutoCloseable {
  @Override
  public void close() {
        System.out.println(">>> close()");
        throw new RuntimeException("Exception in close()");
  }
  public void work() throws MyException {
        System.out.println(">>> work()");
        throw new MyException("Exception in work()");
  }
  public static void main(String[] args) {
        try (AutoClose autoClose = new AutoClose()) {
              autoClose.work();
        } catch (MyException e) {  e.printStackTrace(); }
  }
}
class MyException extends Exception {
  public MyException() { super(); }
  public MyException(String message) { super(message);}
}
```

The AutoClose class implements AutoCloseable and can thus be used as part of a try-with-resources statement, as illustrated in the main() method. Intentionally, we added some console output, and **we throw exceptions both in the work() and close() methods of the class**. Running the program yields the following output:

```
>>> work()
  >>> close()
  MyException: Exception in work()
       at AutoClose.work(AutoClose.java:11)
       at AutoClose.main(AutoClose.java:16)
       Suppressed: java.lang.RuntimeException: Exception in close()
            at AutoClose.close(AutoClose.java:6)
            at AutoClose.main(AutoClose.java:17)
```

The output clearly proves that **close() was indeed called before entering the catch block that should handle the exception**. Yet, the Java developer on or after Java SE 7 might be surprised to see the exception stack trace line prefixed by "**Suppressed: (…)**".

Exception Masking

Let us **get rid of the try-with-resources statement for a moment**, and manually rewrite a correct resource management code.

First, let us extract the following static method to be invoked by the main method:

```
public static void runWithMasking() throws MyException {
  AutoClose autoClose = new AutoClose();
  try {
        autoClose.work();
  } finally {
        autoClose.close();
  }
}
```

Then, let us transform the main method accordingly:

```
public static void main(String[] args) {
  try {
        runWithMasking();
  } catch (Throwable t) {
        t.printStackTrace();
  }
}
```

Now, running the program gives the following output:

```
>>> work()
  >>> close()
  java.lang.RuntimeException: Exception in close()
      at AutoClose.close(AutoClose.java:6)
      at AutoClose.runWithMasking(AutoClose.java:19)
      at AutoClose.main(AutoClose.java:52)
```

This code **shows the problem of one exception being masked by another exception**. Indeed, **the client code to the runWithMasking() method is notified of an exception being thrown in the close() method, while in reality, a first exception had been thrown in the work() method**.

However, **only one exception can be thrown at a time, meaning that even correct code misses information while handling exceptions**. Developers lose significant time debugging when a main exception is masked by a further exception being thrown while closing a resource. The astute reader could question such claims, because exceptions can be nested, after all. However, nested exceptions should be used for causality between one exception and another, typically to wrap a low-level exception within one aimed at higher layers of an application architecture. A good example is a JDBC driver wrapping a socket exception into a JDBC connection. Here, there are really two exceptions: one in work() and one in close(), and there is absolutely no causality relationship between them.

Supporting "Suppressed" Exceptions

Java SE 7 extends exceptions so that **“suppressed” exceptions can be attached to primary exceptions and is called Exception Masking**.
Going back to the previous runWithMasking() method, let us rewrite it with support for suppressed exceptions in mind:

The extensions to **java.lang.Throwable** are as follows:
public final void addSuppressed(Throwable exception) appends a suppressed exception to another one, so as to avoid exception masking.
public final Throwable[] getSuppressed() gets the suppressed exceptions that were added to an exception.

```
public static void runWithoutMasking() throws MyException {
  AutoClose autoClose = new AutoClose();
  MyException myException = null;
  try {
        autoClose.work();
  } catch (MyException e) {
        myException = e;
        throw e;
  } finally {
        if (myException != null) {
              try {
                    autoClose.close();
              } catch (Throwable t) {
                    myException.addSuppressed(t);
              }
        } else {
              autoClose.close();
        }
  }
}
```

A local variable is used to capture the primary exception, that is, the one that the work() method may throw. If such an exception is thrown, it is captured and then thrown again immediately, so as to delegate the remaining work to the finally block.
Entering the finally block, the reference to the primary exception is checked. If an exception was thrown, the exception that the close() method may throw would be attached to it as a suppressed exception.

```
>>> work()
  >>> close()
  MyException: Exception in work()
      at AutoClose.work(AutoClose.java:11)
      at AutoClose.runWithoutMasking(AutoClose.java:27)
      at AutoClose.main(AutoClose.java:58)
      Suppressed: java.lang.RuntimeException: Exception in close()
           at AutoClose.close(AutoClose.java:6)
           at AutoClose.runWithoutMasking(AutoClose.java:34)
           ... 1 more
```

<u>Syntantic Sugar Demystified</u>

```
public static void runInARM() throws MyException {
   try (AutoClose autoClose = new AutoClose()) {
             autoClose.work();
   }
}
```

In reality, the **Java *compiler automatically expands (for ARM to do auto-close functionality) to code* that matches the code of runWithoutMasking() for the above method which uses a try-with-resources statement**:

This can be checked through decompilation using tools like JD-GUI. The below code is the decompiled code. Compiler automatically adds the below code.

```
public static void runInARM() throws MyException {
   AutoClose localAutoClose = new AutoClose();
   Object localObject1 = null;
   try {
             localAutoClose.work();
   } catch (Throwable localThrowable2) {
             localObject1 = localThrowable2;
             throw localThrowable2;
   } finally {
             if (localAutoClose != null) {
                       if (localObject1 != null) {
                                 try {
                                           localAutoClose.close();
                                 } catch (Throwable localThrowable3) {
                                           localObject1.addSuppressed(localThrowable3);
                                 }
                       } else {
                                 localAutoClose.close();
                       }
             }
   }
}
```

A legitimate question regarding the use of the try-with-resources statement is that of performance impact compared to manually crafting proper resource management code. **There is actually <u>no performance impact</u>, because the compiler infers the minimal correct code for properly handling all exceptions**, as we illustrated through decompilation in the previous examples.

At the end of the day, **try-with-resources statements are syntactic sugar just like the enhanced for loops introduced in Java SE 5 for expanding loops over iterators**.

References
https://www.oracle.com/technetwork/articles/java/trywithresources-401775.html

Javascript ES6

Thursday, May 30, 2019 4:49 PM

export and import of modules

file1.js:

```
const firstname = 'Robin';
const lastname = 'Wieruch';
export { firstname, lastname };
```

You can also spare additional lines and export the variables directly for named exports:

```
export const firstname = 'Robin';
export const lastname = 'Wieruch';
```

Then you can import them in another file with a relative path to the first file:

```
import { firstname, lastname } from './file1.js';
console.log(firstname);
```

You can also **import all exported variables from another file as one object**:

```
import * as person from './file1.js';
console.log(person.firstname);
```

imports can have an alias. It can happen that you import functionalities from multiple files that have the same named export. That's why you can use a n alias:

```
import { firstname as username } from './file1.js';
console.log(username);
```

file1.js:

```
export const firstname = 'Robin';
export const lastname = 'Wieruch';
const address = {
  street,
  pincode,
};
export default address;
```

And **import the default (**developer**) or the named exports (**firstname,lastname**)** from another file. Note **the default import can have a different name** and need not the same name as the default export. **The default export address from file1.js is imported as developer**

```
import developer, { firstname, lastname } from './file1.js';
console.log(developer);
console.log(firstname, lastname);
```

<u>Summary</u>

import all exports from another file as one object	***import * as person*** *from './file1.js';*
imports can have an alias	*import {* ***firstname as username*** *} from './file1.js';*
import both the default and the named exports	*import developer, { firstname, lastname } from './file1.js';*

Data Access

Wednesday, June 12, 2019 12:15 PM

How to get Auto-generated Key with JdbcTemplate

Example 1

```
CREATE TABLE IF NOT EXISTS sys_message (
   id bigint(20) NOT NULL AUTO_INCREMENT,
   name varchar(100) NOT NULL,
   PRIMARY KEY (id)
);
```

```
import org.springframework.jdbc.support.GeneratedKeyHolder;
import org.springframework.jdbc.support.KeyHolder;

final PreparedStatementCreator psc = new PreparedStatementCreator() {
        @Override
        public PreparedStatement createPreparedStatement(final Connection connection) throws SQLException {
                final PreparedStatement ps = connection.prepareStatement("INSERT INTO sys_message (name) VALUES (?)",
                                                                            Statement.RETURN_GENERATED_KEYS);
                ps.setString(1, name);
                return ps;
        }
};

// The newly generated key will be saved in this object
final KeyHolder holder = new GeneratedKeyHolder();
jdbcTemplate.update(psc, holder);
final long newNameId = holder.getKey().longValue();
return newNameId;
```

Example 2

```
CREATE TABLE SOURCE_TAB (
   TRADE_ID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY
                        MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1
                        START WITH 1 CACHE 20 NOORDER NOCYCLE NOT NULL ENABLE,
   SOURCE_SYSTEM VARCHAR2(20 CHAR)
)
```

```
String sql = "INSERT INTO SOURCE_TAB (SOURCE_SYSTEM ) VALUES (:sourceSystem)" ;
MapSqlParameterSource params = new MapSqlParameterSource();
params.addValue("sourceSystem", blotterRecord.getSourceSystem());
String[] keyColumnNames = {"TRADE_ID"};
KeyHolder keyHolder = new GeneratedKeyHolder();
namedJdbcTemplate.update(sql, params, keyHolder, keyColumnNames);
dbTradeId = BigInteger.valueOf(keyHolder.getKey().longValue());
```

References

http://www.javacreed.com/how-to-get-auto-generated-key-with-jdbctemplate/

NodeJS

Wednesday, June 19, 2019 9:45 AM

NPM Install

Install modules and save them to your package.json as a dependency
Regular: **npm install <module> --save**
Shorthand: **npm i -S <module>**

Install Modules and Save Them to Your package.json as a **developer dependency**
Regular: **npm install <module> --save-dev**
Shorthand: **npm i -D <module>**

Installing a package globally (if you want to use it as a command line tool)
Regular: **npm i --global <module>**
Shorthand: **npm i -g <module>**

Here, <module> is the name of the module you want to install

If no package.json exists, the below options are ignored. If it exists, they'll update them if they are already there.
--save: Package will appear in your depedencies.
--save-dev: Package will appear in your devDependencies.
--save-optional: Package will appear in your optionalDependencies.
--global: Package will be installed globally (if you want to use it as a command line tool for example)

package.json:

```
{
 "name": "metaverse",
 "version": "0.92.12",
 "description": "The Metaverse virtual reality. The final outcome of all virtual worlds, augmented reality, and the Internet.",
 "main": "index.js"
 "license": "MIT",
 "devDependencies": {
   "mocha": "~3.1",
       ....
 },
 "dependencies": {
   "fill-keys": "^1.0.2",
       ..
 }
}
```

NPM Build

D:\Projects\edist-repo-sr\blaze-web\ui>npm build
npm WARN build `npm build` called with no arguments. Did you mean to `npm run-script build`?

package.json:

```
....
....
 "scripts": {
   "start": "react-scripts start",
   "build": "react-scripts build",
   "test": "react-scripts test",
   "eject": "react-scripts eject",
   "postbuild": "RMDIR /Q/S ..\\src\\main\\webapp & MKDIR ..\\src\\main\\webapp & XCOPY /E/Y build ..\\src\\main\\webapp\\. & XCOPY /E/I/Y WEB-INF ..
\\src\\main\\webapp\\WEB-INF"
 },
....
....
```

Unfortunately, npm build is already an internal command.
Because that command (npm build) already exists, it always shadows over your "build": "react-scripts build"

The fully-qualified way to run your own script is with run-script or its alias run:
$ **npm run build**

npm start and others are the short-hand way, but is only an option when an existing npm command doesn't shadow it, like npm build does.

Reference: https://stackoverflow.com/questions/29939697/npm-build-doesnt-run-the-script-named-build-in-package-json

VS Code

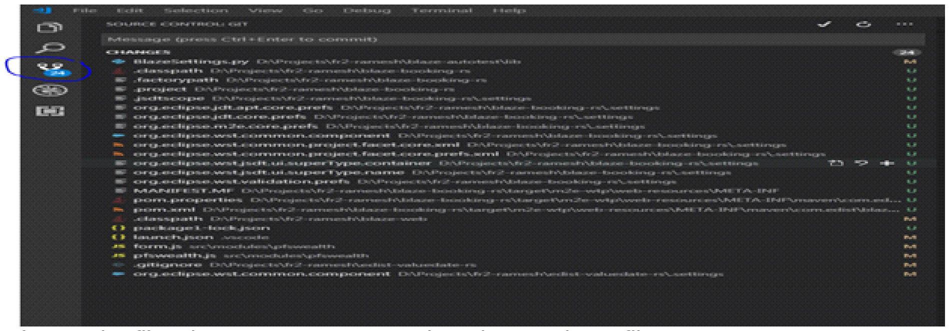
Thursday, June 20, 2019 10:59 AM

Commit and push a single file to GIT from VisualStudio Code

Choose the files that you want to commit and select Stage.

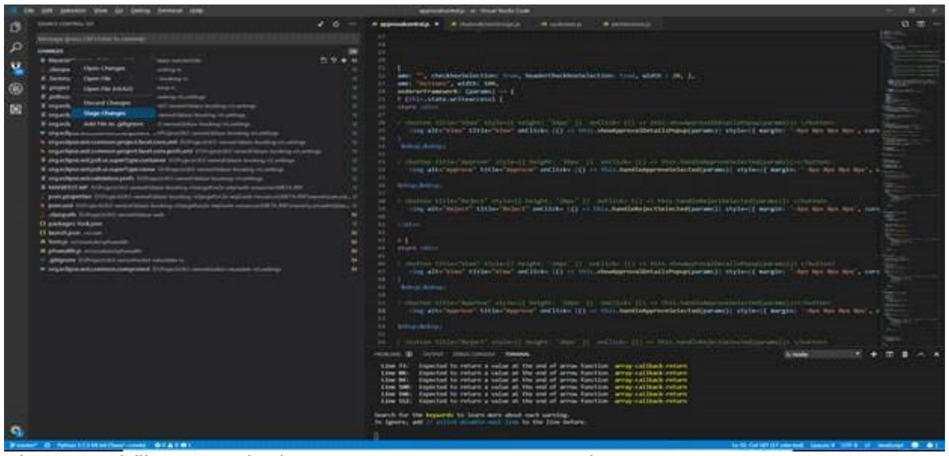
Now you'll have a new section above Changes called **Staged Changes** that only contain the file(s) you want. Type in your commit message and then select **Commit Staged**. Everything else will still be untouched under Changes.

Open the Source Control: GIT view by clicking the highlighted icon on the left.

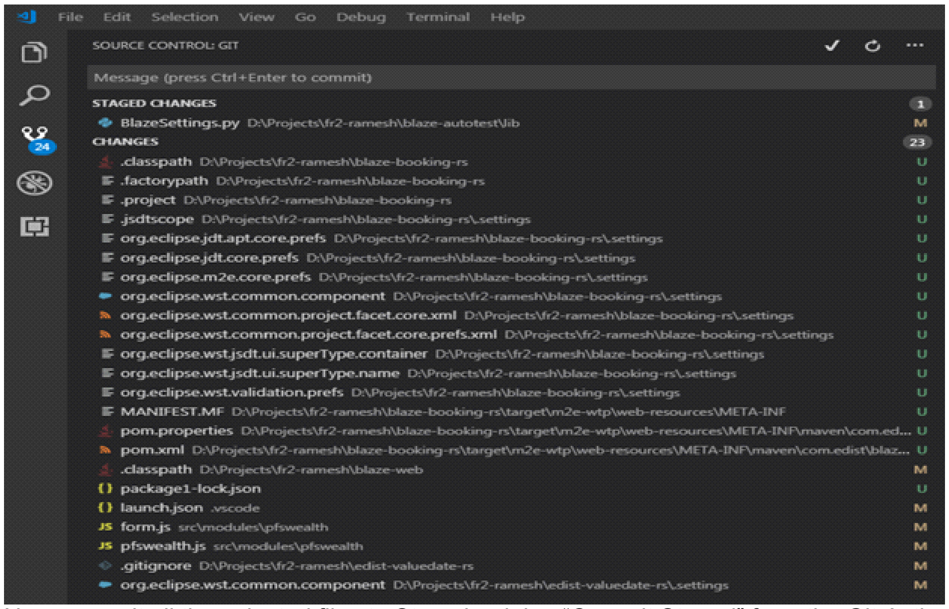

File Edit Selection View Go Debug Terminal Help
SOURCE CONTROL: GIT
Message (press Ctrl+Enter to commit)
CHANGES
24

```
BlazeSettings.py D:\Projects\fr2-ramesh\blaze-autotest\lib  M
.classpath D:\Projects\fr2-ramesh\blaze-booking-rs  U
.factorypath D:\Projects\fr2-ramesh\blaze-booking-rs  U
.project D:\Projects\fr2-ramesh\blaze-booking-rs  U
.jsdtscope D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.jdt.apt.core.prefs D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.jdt.core.prefs D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.m2e.core.prefs D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.wst.common.component D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.wst.common.project.facet.core.xml D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.wst.common.project.facet.core.prefs.xml D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.wst.jsdt.ui.superType.container D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.wst.jsdt.ui.superType.name D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.wst.validation.prefs D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
MANIFEST.MF D:\Projects\fr2-ramesh\blaze-booking-rs\target\m2e-wtp\web-resources\META-INF  U
pom.properties D:\Projects\fr2-ramesh\blaze-booking-rs\target\m2e-wtp\web-resources\META-INF\maven\com.ed...  U
pom.xml D:\Projects\fr2-ramesh\blaze-booking-rs\target\m2e-wtp\web-resources\META-INF\maven\com.edist\blaz...  U
.classpath D:\Projects\fr2-ramesh\blaze-web  M
package1-lock.json  U
launch.json .vscode  M
form.js src\modules\pfswealth  M
pfswealth.js src\modules\pfswealth  M
.gitignore D:\Projects\fr2-ramesh\edist-valuedate-rs  M
org.eclipse.wst.common.component D:\Projects\fr2-ramesh\edist-valuedate-rs\.settings  M
```

Choose the files that you want to commit and stage those files

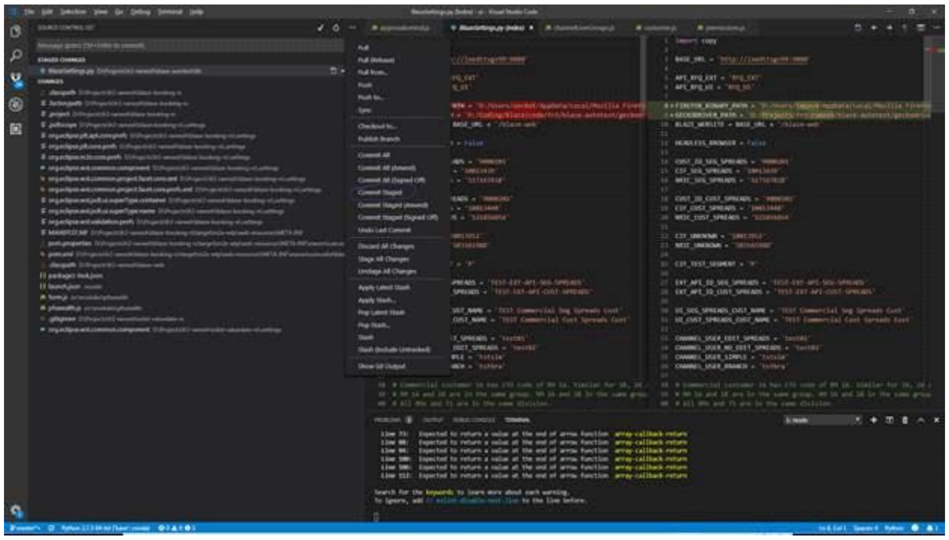


The staged files come in the STAGED CHANGES section



```
Message (press Ctrl+Enter to commit)
STAGED CHANGES  1
BlazeSettings.py D:\Projects\fr2-ramesh\blaze-autotest\lib  M
CHANGES  23
.classpath D:\Projects\fr2-ramesh\blaze-booking-rs  U
.factorypath D:\Projects\fr2-ramesh\blaze-booking-rs  U
.project D:\Projects\fr2-ramesh\blaze-booking-rs  U
.jsdtscope D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.jdt.apt.core.prefs D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.jdt.core.prefs D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.m2e.core.prefs D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.wst.common.component D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.wst.common.project.facet.core.xml D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.wst.common.project.facet.core.prefs.xml D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.wst.jsdt.ui.superType.container D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.wst.jsdt.ui.superType.name D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
org.eclipse.wst.validation.prefs D:\Projects\fr2-ramesh\blaze-booking-rs\.settings  U
MANIFEST.MF D:\Projects\fr2-ramesh\blaze-booking-rs\target\m2e-wtp\web-resources\META-INF  U
pom.properties D:\Projects\fr2-ramesh\blaze-booking-rs\target\m2e-wtp\web-resources\META-INF\maven\com.ed...  U
pom.xml D:\Projects\fr2-ramesh\blaze-booking-rs\target\m2e-wtp\web-resources\META-INF\maven\com.edist\blaz...  U
.classpath D:\Projects\fr2-ramesh\blaze-web  M
package1-lock.json  U
launch.json .vscode  M
form.js src\modules\pfswealth  M
pfswealth.js src\modules\pfswealth  M
.gitignore D:\Projects\fr2-ramesh\edist-valuedate-rs  M
org.eclipse.wst.common.component D:\Projects\fr2-ramesh\edist-valuedate-rs\.settings  M
```

Now, commit all the selected files to Stage by doing “Commit Staged” from the Git-Actions menu (…)



```
import copy

BASE_URL = '[illegible]'

API_RFQ_EXT = 'RFQ_EXT'
API_RFQ_US = 'RFQ_US'

FIREFOX_BINARY_PATH = '[illegible]'
GECKODRIVER_PATH = '[illegible]'
BLAZE_WEBSITE = BASE_URL + '/blaze-web'

HEADLESS_BROWSER = False

CUST_ID_SEG_SPREADS = '[illegible]'
CIF_SEG_SPREADS = '[illegible]'
WBIC_SEG_SPREADS = '[illegible]'

CUST_ID_CUST_SPREADS = '[illegible]'
CIF_CUST_SPREADS = '[illegible]'
WBIC_CUST_SPREADS = '[illegible]'

CIF_UNKNOWN = '[illegible]'
WBIC_UNKNOWN = '[illegible]'

CIF_TEST_SEGMENT = 'F'

EXT_API_ID_SEG_SPREADS = 'TEST-EXT-API-SEG-SPREADS'
EXT_API_ID_CUST_SPREADS = 'TEST-EXT-API-CUST-SPREADS'

UI_SEG_SPREADS_CUST_NAME = 'TEST Commercial Seg Spreads Cust'
UI_CUST_SPREADS_CUST_NAME = 'TEST Commercial Cust Spreads Cust'

CHANNEL_USER_EDIT_SPREADS = '[illegible]'
CHANNEL_USER_NO_EDIT_SPREADS = '[illegible]'
CHANNEL_USER_SIMPLE = '[illegible]'
CHANNEL_USER_BRANCH = '[illegible]'

# Commercial customer [illegible]
# [illegible]
# All RMs and TS are in the same division.
```

```
  Line 73:   Expected to return a value at the end of arrow function  array-callback-return
  Line 88:   Expected to return a value at the end of arrow function  array-callback-return
  Line 94:   Expected to return a value at the end of arrow function  array-callback-return
  Line 100:  Expected to return a value at the end of arrow function  array-callback-return
  Line 106:  Expected to return a value at the end of arrow function  array-callback-return
  Line 112:  Expected to return a value at the end of arrow function  array-callback-return

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

TDD

Monday, June 24, 2019 11:40 AM

<u>What Is TDD and Why It Is Not Unit Testing: Executive Briefing</u>

Many people in technology view "speed" and "quality" as a dichotomy.
ie., You can have quality and that it would be take longer. You can have speed. But, that will have defects (less quality)
Can we have both?. If we need to increase the speed. Then, we need to focus on Quality.

Dichotomy-vs-Oxymoron
Oxymoron: Historically, an (term) was "a (paradox) with a point". Oxymoron is a figure of speech in which two words with oppo sing meanings are used together intentionally for effect.
Dichotomy: A separation or division into two.

Automated Tests and why we need it?
Automated Tests are code that are used to test other code.
Automated tests are repeatable, triggered via just a click or even by a code commmit.
Automated tests run in a consistent and in a quick way when compared to manual testing.
With automated tests, we can confidently ship software and release new features even quicker.

The difference between manual and automated tests.
I test (verb) and I have the tests (noun)

TDD Cycle: RED-GREEN-REFACTOR cycle

TDD is not the same as Unit Testing. TDD is a process/approach to write tests and can be used for many different types of tes ts like Intergration tests and acceptance tests. TDD is generally a Test-First approach where tests are written before hand.
Code Coverage: Its a metric/measurement which tells the %age of production code that is executed by the automated tests.

When it comes to a debate, there are both opponents and proponents.
TDD has many advantages. But, it also comes with a cost.
TDD increase the quality of both the code and the tests.
TDD impacts productivity. Productivity decreases in short-term. But, increases in long-term

<u>Internal Quality Problems</u>
Code Mess - Difficult to understand and maintain
Key people and dependency issues - Representing an element of risk when they leave
Onboarding woes - Onboarding developers take long time
Growing Complexity - Creating features take longer and longer each time the code is touched.

<u>External Quality Problems</u>
Large backlog of Bugs
Frequent reports of issues
No time for feature work as most of time is spent on fixing bugs

Long Term Productivity
Developer Engagement - TDD puts a level of responsibility on the developer to develop a great product and happens to be their intrinsic motivation.

When TDD is not useful
The long investment on TDD for rapid prototyping is not useful if the prototype is thrown out once it serves the purpose(name ly validating the idea)
Speed to Market.

Security

Sunday, June 30, 2019 10:49 PM

XML XXE Vulnerability

XXE (XML External Entity) Incubates in DOCTYPES. You can turn off these unwanted features by telling the XML Parser to ignore the DOCTYPE Declaration.
Then, we effectively mitigate a host of vulnerabilities.Disallowing the doctype declaration means that the payload won't be parsed if it has a DOCTYPE declaration.
Rejecting the payload when it has a DTD is easier than trying to neutralize, sanitize or whitelist the payload.

Disabling DOCTYPES

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
final String DISALLOWEXTERNALDTD_FEATURE = "http://apache.org/xml/features/disallow-doctype-decl";
documentBuilderFactory.setFeature(DISALLOWEXTERNALDTD_FEATURE, true);
```

The same option of setFeature to turn off, will work for SaxParser, XMLReader and others.

```
saxReader.setFeature(DISALLOWEXTERNALDTD_FEATURE, true);
xmlReader.setFeature(DISALLOWEXTERNALDTD_FEATURE, true);
```

Setting the above feature will result in error when parsing an XML that has a DTD. This puts a solid defence against XXE.

Neutralize DOCTYPES

Sometimes, we don't have an option of parsing an XML without DTD. **We need to parse the XML with DTD and still eliminate XXE**.
In that case, we need to **neutralize the DTD processing with a No-Op EntityResolver**.
EntityResolver is the class which the XML parsers normally use to parse External Entities. Neutralize the EntityResolver with a noop EntityResolver.

```
// EntityResolver that does nothing. But, return an empty string (Return empty string and not null).
// But, returning null means to allow the default parser behaviour
private static final EntityResolver noop = (publicId, systemId) -> new InputSource(new StringReader(""));
DocumentBuilder builder = documentbuilderFactory.newDocumentBuilder();
builder.setEntityResolver(noop);
```

Instead of disabling DOCTYPES entirely or setting an noop Entity Resolver, the parsers have features(to neutralize) that can turn on specific features instead of throwing error while parsing the XML when DTD is present.

References

https://www.acunetix.com/blog/articles/xml-external-entity-xxe-vulnerabilities/
https://www.ivankrizsan.se/2013/08/02/disabling-loading-of-external-dtds-in-xml-documents-read-by-jaxp/
Pluralsight Course: https://app.pluralsight.com/library/courses/securing-java-web-application-data/table-of-contents

Bolt

Monday, July 08, 2019 11:47 AM

Wednesday, May 29, 2019 1:02 PM

Functional and class components in React

```
import React from 'react';
function App() {
  const greeting = 'Hello Function Component!';
  return <Welcome value={greeting} />;
}
export default App;
```

Class Component: ES6 class syntax to write components.	Function Components

class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.value}</h1>;
  }
}
	It's just a function which accepts props and returns a React element.
	function Welcome(props) {
	  return <h1>Hello, {props.value}</h1>;
	}

When should I use a function and when a class?
A functional component is just a plain JavaScript function which accepts props as an argument and returns a React element.
A class component requires you to extend from React.Component and create a render function which returns a React element.

Restrictions in Functional Components (Prior to React 16.8):

1. **You cannot use setState() in your functional component**. That's the reason why they **also get called functional stateless components**. So every time you see a functional component you can be sure that this particular component doesn't have its own state. If you need a state in your component you will either need to create a class component or you lift the state up to the parent component and pass it down the functional component via props.
2. Another feature which you cannot use in functional components are lifecycle hooks. The reason is the same like for state, **all lifecycle hooks are coming from the React.Component which you extend from in class components**.

The above two restrictions of Functional component doesn't exist from React 16.8 onwards with the Hooks Feature.
You can now use the **useEffect** hook to use lifecycle events in your functional components and **useState** hook that lets you add React state to function components

React Hooks

React Hooks was introduced to do state management, lifecycle management and side-effects in function components which was traditionally possible only in Class Component React function components without the need to refactor a React function component to a React class component for using lifecycle methods, in order to use have side-effects, or local state. React Hooks enable us to write React applications with only function components.

Basics
Define the Component as a Function rather than Class Component.
No longer need to import Component, instead we'll need the useState and useEffect hooks.
Constructor and initial state: Instead of defining initial state in a constructor, we will call the useState hook in the function body.
Replace all other references to this.state and this.setState with the methods returned from useState hooks.

References
https://itnext.io/add-state-and-lifecycle-methods-to-function-components-with-react-hooks-8e2bdc44d43d
https://www.robinwieruch.de/react-hooks/
https://reactjs.org/docs/hooks-rules.html
https://reactjs.org/docs/hooks-effect.html

React - setState

Tuesday, July 02, 2019 11:33 AM

React - setState

The first thing to be aware of is the fact that **the setState function in React works in an asynchronous fashion**. Therefore, the state values are not immediately available after the update.

You can use **this.setState to change single, multiple, or even all properties of state at a time**. When setting state for multiple properties, **only the properties passed in will be updated**.

```
/* Given this state */
this.state = {
  key1: value1,
  key2: value2,
  key3: value3
}
```

For example, the following shows how the properties key1 and key3 of a state could be updated.

```
/* We could use code like the following to update specific properties */
this.setState({ key1: newValue1, key3: newValue3 });
```

You can change multiple properties of a state like this.

```
this.setState({ openedCard: null, offeredChips: 0, activePlayer: nextPlayer });
```

(OR)

```
this.setState({openedCard:null});
this.setState({offeredChips:0});
this.setState({activePlayer:nextPlayer });
```

AG Grid

Tuesday, July 09, 2019 10:02 AM

Export to CSV

Exporting to Excel is an enterprise feature. However, in absence of an enterprise license, you should be able to call the gri dOptions API to export the data to a .csv, which can be opened in Excel.

Example 1:

```
const params = {
        columnGroups: true,
        allColumns: true,
        fileName: '<filename_of_your_choice>',
        columnSeparator: ","
};
this.gridApi.exportDataAsCsv(params);
```

Example 2 (Export only specific columns):

```
const params = {
       columnGroups: true,
       columnKeys: ['permission','description'],
       allColumns: false,
       fileName: 'permissions.csv',
       columnSeparator: ","
};
this.api2.exportDataAsCsv(params);
```

Reference:
https://stackoverflow.com/questions/49885722/export-json-to-excel-csv-using-ag-grid

Http requests

Tuesday, July 09, 2019 12:37 PM

How do we access the network in a React app

There are many HTTP libraries we can use to fetch data from a endpoint. The most popular ones are:
fetch
axios
Superagent
XMLHttpRequest

It is a best practice that the Ajax calls in React are made in the componentDidMount lifecycle method. This is so because at the instantiation of a component its JSX markup is not yet in the DOM. React first creates an instance of a component, calls componentWillMount and mount its JSX markup in the DOM by calling its render method. After this is successful, it will call componentDidMountto tell you that the mounting of the component in the DOM is successful, so you can add a code to execute at this point.
We know that the data gotten from an API endpoint is used to update our component in the DOM, so naturally, it is the best bet to do it when the component is already on the DOM. componentDidMount provides us that opportunity to render our data from a network.

setState Problem and componentWillUnMount
We fetch the data over the network in the componentDidMount lifecycle method and update the state in the callbacks.
What if the component is unmounted even before the data resolves?. Any reference to methods and properties to the case would be undefined. In our case, this.setState would not be available. This is because Promises is delegated run in an event loop. Once the JS engine hit a Promise it moves the Promise code to an Event Loop and continues with the code. So the component may be removed and its DOm dismounted but its Promise code is still running in the Event loop when the Promise code resolves the JS engine will found out that the callback in the Promise has lost reference because the class is no longer in memory and has lost reference. **The Promise(s) will hit a reference error**.

To abort silently and make our code not to throw reference errors, we need to cancel the HTTP request. The HTTP request returns a Promise and in the resolution, we update the state, so if we cancel the HTTP request the Promise won't resolve thereby not calling any member methods or properties
Where and when do we need to cancel HTTP requests? How do we know when our component will be unmounting?
We need to cancel the HTTP request when the component dismounts. **React provides with a lifecycle that lets us run some code before a component is to be dismounted**. **It is called componentWillUnMount**. That is, when a component is designated for removal from the DOM, a hook(componentWillUnMount) is called to signal the user that the component is to be dismounted from the DOM.

Axios
npm i axios -S

```
import axios from 'axios'
class UsersComponent extends React.Component {
  constructor() {
    this.state = {
      users: [],
      done: false
    }
  }
  componentDidMount() {
    axios.get(`/api/users`)
      .then(response => response.json())
      .then(json => this.setState({ users: json.data, done: true }))
  }
  render() {
    if(!this.state.done) {
      return (
        <div>
          Users Loading
        </div>
      )
    } else {
      return (
        <div>
          Users: {this.state.users}
        </div>
      )
    }
  }
}
```

get('/api/users') call performs a HTTPS GET requests to api/users the above json data is returned. **The returned data is encapsulated in a Promise**.
Now, if we load the component we will briefly see only Users: before seeing the fetched data rendered. **That brief moment before seeing the fetched data, the user will think the app is stuck.** So in order stave off that, we will add logic to show some text to tell the user that something is loading.

By default, axios serializes JavaScript objects to JSON

fetch
Fetch is a **native browser API for making HTTP requests**, we can use it in our React component to fetch data over a network. In the fetch(...) call we also set the done to true we the data is successfully fetched.

```
componentDidMount() {
    fetch(`/api/users`)
        .then(res => res.json())
        .then(result => this.setState({ users: result.users, done: true }));
}
```

XMLHttpRequest

We can use the good ol' XMLHttpRequest to access network:

```
constructor() {
    this.state = {
        users: [],
        done: false
    }
    this.xhr = null
}
componentDidMount() {
    this.xhr = new XMLHttpRequest()
    this.xhr.open('GET', '/api/users', true);
    this.xhr.onload = () => {
        if (request.status >= 200 && request.status < 400) {
            this.setState({users: xhr.responseText, done: true})
        }
    };
    this.xhr.send();
}
componentWillUnmount() {
    // Cancel the xhr request, so the callback is never called
    if (this.xhr && this.xhr.readyState != 4) {
        this.xhr.abort();
    }
}
```

Here, we made xhr globally so we can access in the componentWillUnMount method. There **we call this.xhr.abort() on the XMLHttpRequest to abort or stop the HTTP requests when the component is unmounted.**

Reference:
https://blog.bitsrc.io/making-https-request-in-react-3a2777700c5d

Cancelling Http Requests

How to use ComponentWillUnMount to setState and cancel pending promises

Reference:
https://redux-resource.js.org/recipes/canceling-requests
https://github.com/axios/axios#cancellation

SC Loan Repayment

Wednesday, June 19, 2019 10:10 AM

~~19-Jun-19~~	~~45~~
~~21-Jun-19~~	~~48~~
~~23-Jun-19~~	~~51~~
~~25-Jun-19~~	~~54~~
~~27-Jun-19~~	~~57~~
~~29-Jun-19~~	~~60~~
~~01-Jul-19~~	~~63~~
~~03-Jul-19~~	~~66~~
~~05-Jul-19~~	~~69~~
~~07-Jul-19~~	~~72~~
~~09-Jul-19~~	~~75~~
~~11-Jul-19~~	~~78~~
~~13-Jul-19~~	~~81~~
15-Jul-19	84
17-Jul-19	87
19-Jul-19	90
21-Jul-19	93
23-Jul-19	96
25-Jul-19	99
27-Jul-19	102
29-Jul-19	105
31-Jul-19	108
02-Aug-19	111
04-Aug-19	114
06-Aug-19	117
08-Aug-19	120
10-Aug-19	123
12-Aug-19	126
14-Aug-19	129
16-Aug-19	132
18-Aug-19	135
20-Aug-19	138
22-Aug-19	141
24-Aug-19	144
26-Aug-19	147
28-Aug-19	150

30-Aug-19	153
01-Sep-19	156
03-Sep-19	159
05-Sep-19	162
07-Sep-19	165
09-Sep-19	168
11-Sep-19	171
13-Sep-19	174
15-Sep-19	177
17-Sep-19	180
19-Sep-19	183
21-Sep-19	186
23-Sep-19	189
25-Sep-19	192
27-Sep-19	195
29-Sep-19	198
01-Oct-19	201
03-Oct-19	204
05-Oct-19	207
07-Oct-19	210
09-Oct-19	213
11-Oct-19	216
13-Oct-19	219
15-Oct-19	222
17-Oct-19	225
19-Oct-19	228
21-Oct-19	231
23-Oct-19	234
25-Oct-19	237
27-Oct-19	240
29-Oct-19	243
31-Oct-19	246
02-Nov-19	171
	10065

option

Tuesday, July 02, 2019 5:49 PM

S No	Date - Withdraw	Amount	Request Number	Date - Credited	Credit Amt - SGD
1	25-Jun-19	100	16347725088	27-Jun-19	139.81
2	28-Jun-19	100	16426833966	29-Jun-19	139.96
3	02-Jul-19	500	16531841396	04-Jul-19	700.7
4	04-Jul-19	600	16571204665	05-Jul-19	840.9
5	10-Jul-19	700	16711843910	12-Jul-19	984.61