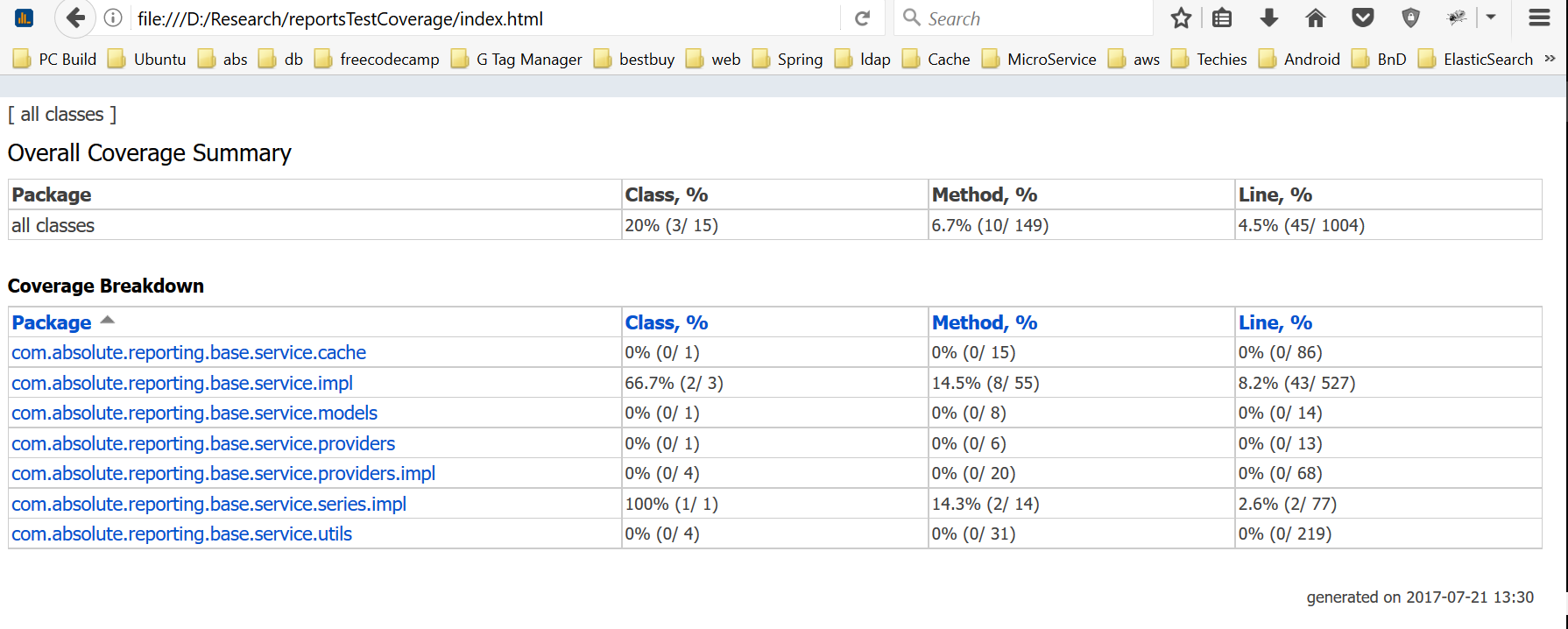
# JUnit

Code Coverage



Using Intellij

Right-click on file/folder Run … with Coverage.

# Java

## Clone

https://stackoverflow.com/questions/2156120/java-recommended-solution-for-deep-cloning-copying-an-instance

For deep cloning (clones the entire object hierarchy):

commons-lang SerializationUtils - using serialization - if all classes are in your control and you can force implementing Serializable.

Java Deep Cloning Library - using reflection - in cases when the classes or the objects you want to clone are out of your control (a 3rd party library) and you can't make them implement Serializable, or in cases you don't want to implement Serializable.

For shallow cloning (clones only the first level properties):

commons-beanutils BeanUtils - in most cases.

Spring BeanUtils - if you are already using spring and hence have this utility on the classpath.

I deliberately omitted the "do-it-yourself" option - the API's above provide a good control over what to and what not to clone (for example using transient, or String[] ignoreProperties), so reinventing the wheel isn't preferred.

https://github.com/kostaskougios/cloning/

Here is an example of its usage:

Cloner cloner=new Cloner();

MyClass clone=cloner.deepClone(o);

// clone is a deep-clone of o

#For first level implements Clonable and override clone()

@Override

protected Person clone() {

Person clone = null;

try {

clone = (Person) super.clone();

} catch (CloneNotSupportedException e) {

e.printStackTrace();

}

return clone;

}