```java
import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import java.io.Reader;
import org.apache.jena.rdf.model.InfModel;
import org.apache.jena.reasoner.Reasoner;
import org.apache.jena.reasoner.rulesys.GenericRuleReasoner;
import org.apache.jena.reasoner.rulesys.Rule;


// Create an RDF model from the XHTML file (sourceURL – is an URL of the XHTML
// file with RDFa content)…

Model modelFromFile = null;
try {
            Reader in = Utils.getRequest(sourceURL); // see below
            modelFromFile = ModelFactory.createDefaultModel();
            modelFromFile.read(in,"","RDF/XML");
} catch (IOException e) {
            System.out.println("Impossible to find the specified file");
            System.out.println(e.getMessage());
}



// Create an inference RDF model based on the base RDF model and string-based
// rules.


InfModel finalModel = null;

String rulesStr = "@prefix foaf: <http://xmlns.com/foaf/0.1/>.\n@prefix dc:
<http://purl.org/dc/elements/1.1/>.\n[rule1: (?a foaf:interest ?b) (?b dc:title
?c) -> (?a foaf:nick ?c)]";


try {
    String filePath = Utils.setRule(rulesStr); // see below
    System.out.println("Path: "+filePath);
    Reasoner reasoner = new GenericRuleReasoner(Rule.rulesFromURL(filePath));

//    Reasoner reasoner = new GenericRuleReasoner(Rule.parseRules(rulesStr)); -
did not find the way to make this option to work…

    reasoner.setDerivationLogging(true);
    finalModel = ModelFactory.createInfModel(reasoner, modelFromFile);
} catch(Rule.ParserException e) {
    System.out.println(e.getMessage());
} catch (IOException e) {
    System.out.println(e.getMessage());
}
```

To proceed with further query – just create dataset instance based on corresponding model…

```java
dataset = DatasetFactory.create(finalModel);
```

or

```java
dataset = DatasetFactory.create(modelFromFile);
```

Utils class that contains functions to read XHTML file and writing string to the file (used in one of the ways to read the Rule from the string via file writer/reader)…

```java
import java.io.BufferedWriter;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Reader;
import java.io.StringReader;
import java.net.URISyntaxException;

import org.apache.any23.Any23;
import org.apache.any23.extractor.ExtractionException;
import org.apache.any23.http.HTTPClient;
import org.apache.any23.source.DocumentSource;
import org.apache.any23.source.HTTPDocumentSource;
import org.apache.any23.writer.RDFXMLWriter;
import org.apache.any23.writer.TripleHandler;
import org.apache.any23.writer.TripleHandlerException;

public class Utils {
    public static String setRule(String rules) throws IOException {
        File file = File.createTempFile("rules_", ".txt");
        file.deleteOnExit();
        BufferedWriter bw = new BufferedWriter(new FileWriter(file));
        bw.write(rules);
        bw.close();
        return file.getAbsolutePath();
    }
    public static Reader getRequest(String url) throws IOException{
        Any23 runner = new Any23();
        runner.setHTTPUserAgent("test-user-agent");
        HTTPClient httpClient = runner.getHTTPClient();
        DocumentSource source = null;
        try {
            source = new HTTPDocumentSource(httpClient, url);
        } catch (URISyntaxException e) {
            e.printStackTrace();
        }
        ByteArrayOutputStream output = new ByteArrayOutputStream();
        TripleHandler handler = new RDFXMLWriter(output);
        try {
            runner.extract(source, handler);
        } catch (ExtractionException e) {
            e.printStackTrace();
        } finally {
            try {
                handler.close();
            } catch (TripleHandlerException e) {
                e.printStackTrace();
            }
        }
        return new StringReader(output.toString());
    }
}
```