Slightly

Programming exercise to do at home. The candidate should have at least a weekend to do it.

This page has all the instructions. (So we can send a PDF extract of only this page to the candidate.)

Please use the attached package Slightly.zip as starting point.

Context: Adobe's new Sightly

Adobe Experience Manager 6.0 introduces a new server-side html templating language called "Sightly", to replace JSP+JSTL.

The main idea is that the server-side template is already valid html. The usual operations of conditional and loop are written in HTML5's data-attributes.

This exercise: Slightly

For this exercise, you will implement something similar to Sightly, called "Slightly".

It is a small server-side html templating language with some server-side scripting in javascript.

You need to know Java, servlets, HTML5, and javascript. You don't need to know any AEM.

The aim is to have a template file like this in the web server:

```
<!DOCTYPE html>
<html>
 <script type="server/javascript">
  importClass(Packages.biz.netcentric.Person)
  var id=request.getParameter("id")
  var person=Person.lookup(id)
 </script>
 <head>
  <title>${person.name}</title>
 </head>
 <body>
  <h1 title="${person.name}">${person.name}</h1>
  <h2 data-if="person.married" title="${person.spouse}">Spouse: ${person.spouse}</h2>
  <div data-for-child="person.children">Child: ${child}</div>
 </body>
</html>
```

which, provided we have a Java class like:

```
package biz.netcentric;
public class Person{

 public static Person lookup(String id){ return ...; }

 public String getName(){...}
 public boolean isMarried(){...}
 public String getSpouse(){...}

 public List<String> getChildren(){...}
}
```

renders an html page like:

```
<!DOCTYPE html>
<html>
 <head>
  <title>Maria</title>
 </head>
 <body>
  <h1 title="Maria">Maria</h1>
  <h2 title="Pere">Spouse: Pere</h2>
  <div>Child: Anna</div>
  <div>Child: Berta</div>
  <div>Child: Clara</div>
 </body>
</html>
```

You will write a servlet that responds to requests for html files. The servlet reads the requested file, parses the html, and processes the server-side javascript and server-side data-attributes, and finally sends the resulting html to the browser.

Specification

<script> element

A server-side script element is marked by the attribute type="server/javascript". If the servlet finds such a script element, it should execute its body as javascript. The element itself produces no output for the browser.

For javascript-execution you may use Java's built-in Javascript engine. Java 7 includes a version of Mozilla's Rhino engine. Java 8 has a similar Javascript engine called "Nashorn". You may also include Mozilla's Rhino engine.

The script element may put new javascript variables in the state, like "person" in the example. Initially, the state should have at least one useful variable: "request", being the current HttpServletRequest object. Using that, the template can depend on input from URL parameters.

From the javascript in the element it is possible to call Java classes. The details of exactly that works depend on which Javascript engine you use. In general, there will be facilities to import Java packages and classes, instantiate Java classes, and call methods on them, with a close correspondence between the Javascript syntax and Java syntax. All three mentioned Javascript engines know the getter-setter pattern. So the javascript *person.name* will be mapped to the Java methods getName() or setName(String) in the usual way.

$-expressions

There may be server-side $-expressions in template. They may appear inside the element bodies and inside the values of attributes. That means, they may not appear in the element tags or in the keys of attributes.

When the servlet finds an expression, it should evaluate the javascript expression in the current javascript state, and render the result in the html output.

If the result is rendered inside the body of an html element, you must escape any characters that are not allowed inside an html element.

If the result is rendered inside the value of an attribute, you must escape any characters that are not allowed inside an html attribute value.

data-if

An html element may have an attribute with key "data-if".

When the servlet finds such an attribute, it should evaluate the attribute value as a javascript expression, in the current javascript state.

If the result is *true*, the servlet should render the element in the normal way, but without the data-if attribute.

If the result if *false*, the servlet should not render the element.

data-for-x

An html element may have an attribute like "data-for-x", where "x" stands for a something that can be a variable name in javascript.

When the servlet finds such an attribute, it should evaluate the attribute value as a javascript expression, in the current javascript state.

If the result is an array (or a similar thing), then the servlet should render the element once for every item in the array.

During each rendering of the element, the variable *x* stands for the item in the array.

The result of the javascript evaluation may come from a java method call, therefore objects similar should also work:

- javascript array
- java array
- java collections.

If an element has both *data-if* and *data-for* attributes, what should happen? What do you think is most useful?

Implementation hints

You should be able to run the servlet with the provided package using mvn jetty:run.

For parsing html, you can use any built-in library, your own code, or any libraries you like, as long as they are open source and you add them to the project. Make your life easy: You may assume all html files are valid XML.

Having parsed html, the servlet should traverse the html elements in tree-order, and render them, in order to send them to the browser. The formatted html needn't be the same character-by-character, as long as it is equivalent as html.

Optional: local variables

How could we add local variables? Maybe I want variable "person" to stand for the value of "new Person("Maria")", but only during the rendering of one -element. After that element, the variable should go back to whatever value it had before, if it had any.

Optional: inclusion

How could one template include another? Can we use the new html5 template-element? Can there be a data-attribute that includes another template?

In general

- You may use libraries, if they are open-source and you include them.
- You may use maven.
- Clean code, please.
- Tests that demonstrate that the code is correct are good.
- Document your design a little. Argue for it. Mention any alternatives that you have discarded.
- You may use Google and any other sources of information, of course.
- Don't publicise the exercise to future candidates please. If you find a solution please tell us.