SOAP-based Service

How to Use the Service-Based Layout in Your Project

One of the goals of KWebS is to provide the KIELER layout to a broader range of users by hiding its java nature behind a web service. As the server uses a SOAP web service, it describes the user interface in a platform independent way using the Web Service Description Language (WSDL). You can use this contract to automatically generate the code necessary to access the operations provided by KWebS. For example, if you have a java based project, you can use the tool wsimport:

```
wsimport -p my.kwebs.client -keep http://layout.rtsys.informatik.uni-kiel.de:9442/layout?wsdl
```

Normally, `wsimport` will derive the package where it puts the generated classes into from the XML name space declared in the contract, but this will in general not fit to your needs. You can declare a different package with the **-p** option, in the given example the classes will be generated to the package `my.kwebs.client`. The `-keep` option tells `wsimport` not to delete the generated java sources after it has compiled them to class files. As a last option, `wsimport` requires the contract from which it shall generate the code. This may be a file on your local computer, or as an alternative, `wsimport` can download it from the internet. The public demo service of KWebS is located at **http://layout.rtsys.informatik.uni-kiel.de:9442/layout**, and you can access its contract by adding `?wsdl` to this location.

After code generation, you have a bunch of java source and class files. They contain the code necessary for invoking the service-based layout. The most important classes are `LayoutService` and `LayoutServicePort`. You need them to create a *proxy* to the service:

```
package my.kwebs.client;

import java.net.URL;
import javax.xml.namespace.QName;

public class KWebSClient {

    // The endpoint address the service is bound to
    private static final String SERVICE_URL
        = "http://layout.rtsys.informatik.uni-kiel.de:9442/layout";

    // The qualified name of the service interface
    private static final String SERVICE_QNAME
        = "http://layout.rtsys.informatik.uni-kiel.de/layout";

    // The name of the service
    private static final String SERVICE_NAME
        = "LayoutService";

    /**
     * Entry point of the client application
     */
    public static void main(final String args[]) {

        try {

            // Connect to the service
            LayoutService layoutService = new LayoutService(
                new URL(SERVICE_URL + "?wsdl"),
                new QName(SERVICE_QNAME, SERVICE_NAME)
            );

            // Create a proxy to access its operations
            LayoutServicePort layoutServicePort = layoutService.getLayoutServicePort();

            // Create the serial notation of the source graph
            String graph = ...

            // Call the "graphLayout" operation of the service. As result you get the
            // serial notation of the layouted graph in the same serial notation as
            // you used for invoking the layout.
            String layout = layoutServicePort.graphLayout(
                graph,                     // the source model
                "de.cau.cs.kieler.kgraph", // we use the KGraph format
                null,                      // we want the result in KGraph format
                null                       // we don't declare any options
            );

        // Handle exceptions here
        } catch (final Exception e) {
            e.printStackTrace();
        }

    }

}
```