# Composite Service

Our service has root in the CompositeService-class which consists of a number of static methods.

**REST** was easy to work with and our code is both compact and easy to read. REST is intuitive, transparent and easy to debug as the error messages are standardized (HTTP errors). We quickly got the code working.

**SOAP** gave us hours of work. It was difficult and the code is not nearly as clear and clean as our REST code. The main problem is with the debuggability: As SOAP is a more complex way to communicate with servers, it also has more strict requirements, and (unfortunately) less clear error messages.

In the end, we realized that modern IDEs for a great part can generate the most SOAP boilerplate code for you, making the task easy.

We prefer REST, as it also allows for looser coupling and is not tied to certain protocols and formats in the same way as SOAP.

\_\_\_\_\_\_web service stuff\_\_\_\_\_\_

## Testing

We use JUnit as our testing framework, to test the general functionality. As the functionality of our web service is quite simple, our primary test makes sure we can successfully get XML correctly. When this has been assured, the other tests either create or delete elements, verifying their effect with the data-getting method.

\_\_\_\_\_testing with marshalling\_\_\_\_\_