P02: REST calls

1
We offer a server infrastructure with three APIs to handle notes. You will get the base URL during your lab. The API can be called by <BaseURL>/api/<API Name>, i.e. http://123.212.121.11/api/addNote:

1. **addNote** : adds a new note to the server. Possible URL parameters include *subject, message* and *creator*. The call returns an *Id* of the newly created note element in JSON format:
   {"id":"f658c7ae-2f9d-cd2a-7030-0b642c2cf3d9"}
2. **notes**: returns a list of all notes on the server in the form {"notes":[<<note>>,...]} and each note containing the properties *id, subject, message,* and *dateTime* (in UTC).
3. **deleteNote**: deletes a note from the server. A required parameter is the Id of the note to delete.

Use your web browser to test each API call. Develop an HTML/JavaScript web page that contains a list of all notes (stored on the server) and a form where a new note can be added. The list of notes should offer a delete link for each note.

2
The API server from (1) is called REST. Think about why the current implementation is not REST conform. However, the current implementation has one advantage over REST. What is this advantage?

3
A lot of web sites are already using REST JSON Apis. One example is the Tagesanzeiger (www.tagesanzeiger.ch).

1. Open the homepage and one article of Tagesanzeiger. Use your browser development tools to monitor the network traffic between client and server. Look for JSON requests. Find the URL for the JSON API to retrieve all comments of an article.
2. Develop a small HTML page that uses a JQuery call on the comments JSON API and print out the comments for one article in a list with the fields first name, last name and message. Example:
   Walter Brun: Der Bundesrat sucht verzweifelt...
   Katharina Saluz: Allen Drohungen zum Trotz setzen...