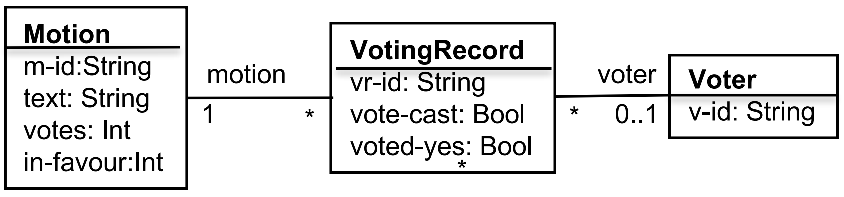
University of Leicester

SOA: Service-oriented Architectures

Reiko Heckel (rh122@le.ac.uk)

Worksheet 3: REST Service Testing and Modelling

You are asked to implement an API for electronic voting. The API is based on the following data model.



A motion is a proposal to vote on. It has a *text* attribute for the proposal motion text, and attributes *votes* and *in-favour* to record the total number of votes cast and the number of votes cast in favour of the motion.

For each motion a voter is allowed to vote on, a one voting record is required. The *vote-cast* attribute is *true* if a vote for this motion by this voter has been recorded, and in this case *voted-yes* is *true* if the vote was in favour. If no vote has been recorded yet, *vote-cast* is *false* and the value of *voted-yes* is irrelevant. Motions, records and voters are identified by id attributes.

The service should provide the following operations.

1. *createVoter(voter_id: String): Bool*
   Create voter *voter_id*
2. *createMotion(motion_id: String, text: String) : Bool*
   Create motion *motion_id* and proposal *text*
3. *createVotingRecord(motion_id, voter-id: record_id: String) : Bool*
   Create record *record_id*
4. *vote(voter_id: String, record_id: String, vote: Bool) : Bool*
   Voter *voter_id* records vote in voting record *record_id*
5. *clearDatabase() : Bool*
   To clear any voting record created
6. *getVoter(voter_id: String) : Voter*
   Return voter with *voter_id*
7. *getVotingRecordsForVoter(voter_id: String) : Set <VotingRecord>*
   For voter *voter_id* return their voting records for all motions
8. *getMotion(motion_id) : Motion*
   Returns motion *motion-id,* will be used to read out results of a vote from attributes *votes* and *in-favour* which should reflect the actual number of votes cast and the number of votes in favour on this motion up to this point in time

Operations of type Bool should return *true* if successful, otherwise return *false*. All results should be encoded as strings encoding JSON objects. All operations are further specified by the visual contracts provided with the assignment.

The WADL file describing the interface you should implement is provided with the assignment as

Make sure that your implementation realises correctly the behaviour specified by the visual contracts provided and use the class diagram as a guide to your internal data model as well as to encode your data objects in JSON for communication with your service.

Assignment

Your task is to implement the voting service specified by this WADL such that it can be invoked by the client application provided with the assignment on BlackBoard.

Submission

You should submit your executable WAR file, which we will run locally in our testing environment, as well the complete source code from which it was generated. Your code should be fully commented, explaining each step of your implementation.

Our lab instructions show how to develop a REST service in Java, but if you don't require technical support you can use any language you are comfortable with. However, in this case it is your responsibility to find a solution for hosting the service and submit to us the link where we can access and invoke your service.

Marking Scheme

Up to 100 marks can be obtained for implementing the operations as follows

- 5 marks for each of operation 1-4 implemented in a way so it can be invoked without runtime errors by the client application
- 10 marks for each of operation 1-4 that passes our tests
- 10 marks for each of the auxiliary operations 5-8 implemented correctly

The test client we will use for marking will be similar but not be identical to the one we have published, e.g. it may call operations in different order, create different voting records or vote differently, etc.

No marks will be awarded unless complete documented source code is submitted.

We will use a lab session after the submission deadline to follow up on any questions we may have on your implementation, so you have to be prepared to explain your code in an individual viva. This is individual work worth 20% of the module mark.