Chapter 1

Managment framework

The core runtime constitutes the main part of the project, but in and of itself only allows running challenges and games given the necessary actors and game libraries. To make it accessible as a server side application, a standalone program is needed to store and manage actors and games, and also provide a web frontend for human interaction.

1.1 Persistence

Data persistence is by default handled by an in–process HSQLDB instance running in embedded file mode. The connection parameters and database location is configurable, and if the database is not found the program attempts to create it at startup. Database access is maintained over JDBC using standard SQL.

In lieu of an ORM system, a simple submodule is used for object persisting and querying. I have choosen this solution because I have felt the project does not require the complex solution of most ready–to–use libraries and a custom set of abstractions and helpers will do. These resources are well integrated into the Kreator dependency injection framework and match my preferred style of data handling. Repositories can be defined for the desired entites, that need only to define two–way object–row mapping to enable various data access functionality (result mapping, single– or multi selection, single– or multi entity persisting, deletion), atop which more behavior can be built. Transaction management is also handled through utilities which build on JDBC checkpoints and multiple connections to provide nested and guarded transactional behavior that is well–integrated to kotlin's capabilities like with–receiver lambda expressions and extension methods.

1.2 Game management

Game management consists of queuing future games, running them using the runtime library, persisting the results, and ranking bots.

1.3 Web interface

The web backend is a kotlin standalone application which provides a RESTful API via JAX-RS with Jersey as its implementation. The program launches an embedded Apache Jetty web server to handle requests. It provides

The frontend web application is written in javascript using Vue.js. It is a single page application that uses Vue Router.