# PreExperiment Data Collector

## Introduction

PreExperiment Data Collector is dedicated to gather all information from the user that can be provided prior to the experiment. Such information can be, for instance, a sample name or a set of parameters for the technique which will be used during the experiment.

The information is specified by the experiment operator. Meta data which describes the information is stored in json format.

The Data Collector is implemented as a standalone web application which also supports Tango interface [tango]. Therefore the application can be easily integrated into any experiment‘s environment operated by Tango.

## Implementation

The implementation of the application is based on the modern web technologies. The overall architecture is presented in the fig.1.

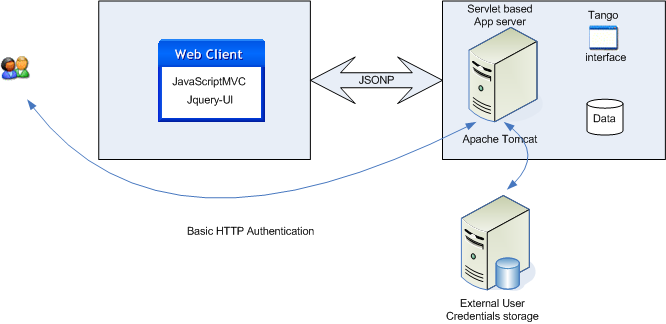


Figure 1: Overall architecture overview. Users interact with the server through browser. In order to gain access to the application users must be authenticated. Server consults remote Authentication server to authenticate users. All data on the server is also accessible through Tango interface.

Client implementation is based on JavaScriptMVC framework [jsMVC]. First of all this framework has very good implementation of the Model-View-Controller pattern [mvc]. It provides very well defined guidance for developing Rich Internet Applications [ria] as well an integration with software automation build tools, ant for example [ant]. Finally it provides facilities for testing and application deployment.

Complex user interface structures are implemented with jquery-ui and organized into engines. Engine is a small JavaScriptMVC application which can be easily integrated into the main application. Engine greatly increases code reusage factor. The client architecture is presented in the fig. 2.

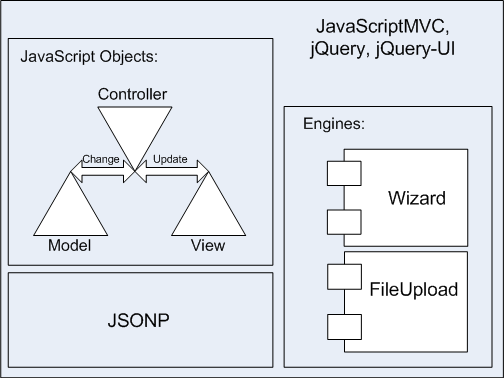


Figure 2: The client overview. Client is a JavaScriptMVC application which is executed in browser. jQuery and jQuery-UI are used in the Client’s implementation. Models are wrappers of the JSONP requests. Large parts of the common functionality are separated into the Engines which in turn are small JavaScriptMVC applications.

JSONP (JSON with Padding) is used in communications between Client and Server [jsonp]. In JavaScriptMVC JSONP is implemented through dynamic adding a script tag to the page.

Servlet technology is used to implement server side logic. Servlets are implemented in a stateless way so they do not track user’s session. Implementation is based on the commons apache libraries. So java representation of the meta data described in json is dynamically generated during the initialization of application. Server side implementation is shown in the fig. 3.

Any servlet container can be used for hosting this application. In this implementation Apache TomCat 7 is used [tomcat].

Users are authenticated by custom realm configured to use remote Kerberos server. It can also be configured to use any other remote authentication service. Basic HTTP authentication is used between client and server. This is considered to be secure enough due to the fact that the application will be accessed only within local intranet of the facility.

## Benefits of this implementation

Client side is done in a way of a very user-friendly wizard. Steps of the wizard are defined in json format and can be easily adjusted for different needs, for instance addition of a new field into one of the steps does not require any developer efforts at all.

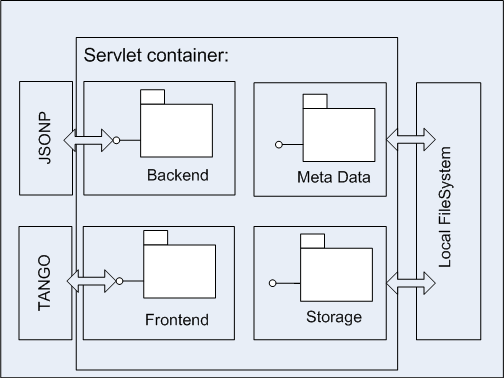


Figure 3: Server overview. Servlet container is a Java process where servlets (special Java classes) are being executed. Servlets are organized in different packages. Each package responsible for its own function, i.e. handle JSONP request. Data on the server is stored in the local file system and can be obtained through Tango interface.

Usage of JavaScriptMVC to implement client makes it possible to split development of the client and server sides. It also provides us code generation tools and testing facilities. JavaDoc like documentation [java-doc] makes it much easier to share knowledge across development team. JavaScriptMVC official site contains tons of useful information and examples so it is easy and quick to learn the framework.

Since JSONP is used for communication between client and server there is no need to manually handle authentication headers in every request. Browser automatically adds necessary headers to the request.

Stateless realization (no session tracking is needed) of the server greatly decreases developer’s efforts on implementing and supporting the application’s server side. Being stateless server side can be easily scaled horizontally across a cluster since no replication of the session is needed any more.

Well-designed server side allows to easily replace or adjust any part of the system, i.e. storage or meta data factory.

Since Apache TomCat is implemented in java it can be run anywhere where java virtual machine can be executed (on Windows, Linux, MacOS etc). The client is just a web browser. Therefore this implementation is absolutely cross platfroming.

All technologies used in this implementation are open source and therefor freely available.

## References

[tango] <http://www.tango-controls.org/>

[jsMVC] <http://wiki.javascriptmvc.com/wiki/index.php?title=Main_Page>

[mvc] <http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

[ria] <http://en.wikipedia.org/wiki/Rich_Internet_application>

[jsonp] <http://en.wikipedia.org/wiki/JSONP>

[tomcat] <http://tomcat.apache.org/>

[java-doc] <http://en.wikipedia.org/wiki/Javadoc>