IMu Documentation

KE IMu API Technical Overview

Document Version 1.1

IMu Version 1.0.03

EMu
Museum
Management
System

KE Software

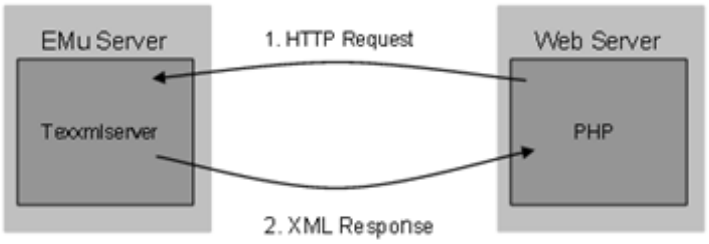
Contents

SECTION 1

Introduction

KE Software has offered a web API in one form or another since the mid 1990s. As technology has evolved, KE's web technology has been updated and rewritten. In 2003 PHP was adopted as the client-side technology for KE's web interface. This web interface was designed to be configurable by technically minded museum staff and so produced much of the required HTML for the pages. In 2006 the release of PHP version 5 provided an opportunity to rewrite the lowest levels of this interface to provide a clearer API for use by KE staff and other developers.

Both these previous versions, collectively known as EMuWeb, follow a very similar architecture: they communicate over HTTP to a database service which returns structured XML:



The client-side PHP library is responsible for forming any required TexQL, any security mechanisms and any user session handling.

This approach has a number of issues:

- Record security is managed by the PHP library on the web server.
- The connection between the PHP library and the database is stateless, so a true database cursor cannot be maintained.
- The API is available in PHP only.
- Integration with Third party systems requires development on the client (web server) side.

IMu has been developed to address these issues. With IMu:

- Security is handled by the server.
- A stateful connection, and hence a true database cursor, can be maintained with EMu.
- The API library is available in PHP, Perl, ASP.NET and Java.
- Integration with Third party systems can be handled on the server by exposing custom API hooks through IMu handlers.

SECTION 2

IMu architecture

IMu Server

Central to IMu is the IMu Server, which handles communication and management of the different handlers available to the client API. Data is transferred over the network using JSON. The IMu Server may be on the same machine, or a separate machine, to the EMu database.

IMu Handlers

Handlers provide the application functionality of the system in IMu. There may be any number of handlers in an IMu system, some distributed with EMu by default, and others written specifically to suit a customer's needs.

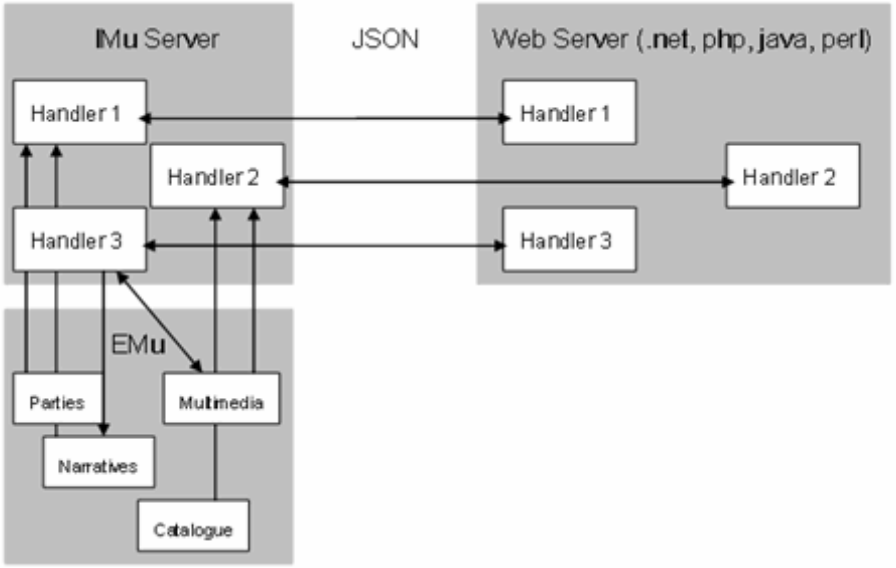
A server-side handler is matched by a client-side handler, and the pair communicate transparently through the IMu Server, which maintains state as necessary.

Two IMu Handlers, `IMuCursor` and `IMuModule`, are distributed with the IMu client API and their usage is outlined in the relevant API documentation. They provide a low-level query mechanism to EMu and require a level of system knowledge to use.

The handler model allows specific pieces of functionality to be produced for individual customers, and custom hooks to be exposed through the API.

API Hooks can simplify complex development by exposing custom functionality to the client-side and returning data structured to suit different needs as specified.

Schematic



Arrows show direction of data flow; data and EMu modules used will vary between customer and application.

SECTION 3

Advantages and benefits

Amongst the advantages and benefits offered by IMu are the following:

Security

- Movement of security logic to the server portion of IMu results in an increase in security. With the previous EMuWeb API, validation of user input to prevent SQL injection had to be performed at the client (web server) side. With IMu, additional safeguards can be put in place server-side to check against this.
- Because state can be maintained between web client and IMu handler, enhanced log-in and client authentication mechanisms are available to developers with very little additional programming overhead.
- Database updates and insertions can be moved server-side where code can be supported by KE Software.

The IMu model is inherently more secure than the previous API.

State can be maintained

- Logic within the IMu Server means that state between JSON transactions may be maintained between web client and server requests. Thus a record-set may be uniquely associated with a user's browser session, but the data of the set held entirely server-side. This offers a significant advantage over standard web methods available, particularly when dealing with museum collections of many thousands of records.
- Very large sorts, record updates, batch updates, batch deletions and record-set merging are all available with a database cursor.

Maintaining a stateful connection into the system is possible and is handled simply by IMu.

Less network traffic

- JSON incurs far less network overhead than XML.
- Maintenance of state means less data needs to be sent between database server and client application: data is only sent when it is needed.

Integration and use by Third parties

With any database integration the most common obstacle is knowledge of the database schema. EMu's object orientated model has a schema that has been designed to fit SPECTRUM standards for the needs of all collections management processes. However this is a flexible model and is configured and used in different ways depending on an organization's needs. As such KE Software is often asked to provide integration work to ensure the data model and relationships are maintained. The exception to this is if there are in-house developers who understand the schema, have the technical skills and can support their own integration work.

By developing customized IMu handlers KE can expose necessary parts of the system to Third party developers. Thus a Third party is able to specify precisely which data it would like, and what API calls it would like to make to get it. Additionally, database updates and insertions can be validated by a server-side IMu handler before being applied to EMu.

The availability of the API in most commonly used web languages (PHP, Perl, ASP.NET and Java) also makes IMu easy to use by Third parties.

The IMu API is much more accessible to Third parties because of its availability in many languages and its ease of use.

Ongoing system support

By moving much of the business logic into an IMu handler produced by KE Software the customer makes most use of the support agreement which is often in place with KE.

Previously, any database updates or insertions would have to validated at the web developer / web server side. This would often prove a barrier to many customers who would be hesitant in placing this level of trust in a party with which they had a transient relationship. The IMu model makes it possible for KE to handle data updates (maintaining any such logic under the support agreement) and therefore makes the capture of user contributed information much more accessible to our customers.

By producing server-side API hooks KE can cover the business logic and database interactions under the pre-existing support agreement.

Index