American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-4, Issue-10, pp-18-22
www.ajer.org
Research Paper Open Access

A Generic Adaptive Method for Corruption Mitigation in Trial Monitoring System with Restful Authorization

Suryanaraayan.B[1], Subramanian.M[2], Saikrishna.K.K[3], Sujitha.G[4]
[1,2,3,4]*(Department of Computer Science and Engineering, Rajalakshmi Engineering College/Anna University, India)*

Abstract: *The purpose of a Trial Monitoring System is to provide a comprehensive suite where cases are created. Trial proceedings are monitored progressively to make informed decisions that include assignment of investigating entities and requesting advisors for opinions to take over the prosecution of the case. It provides a platform for applying counter petitions against an allegation's disposal and integrate proceedings in different levels of scrutiny and across tribunal entities. The outcome is the aggregation of such a data, to classify cases for statistical information and relaying the information in presentable format.*

Keywords – *Caching, Object Relational Mapping, RESTful Web Service*

I. Introduction

A Tribunal Entity, admit about thousand cases in average every year. The accumulated data is very difficult to maintain. The existing system maintains records which either doesn't confirm to relational normalization which leads to a poor database implementation and give raise to various inconsistencies like a case of not able to track disposals, or the evolving case proceedings.

The system's scope is limited to entering data. Because of this disadvantage, the organizational entity cannot get any meaningful information or use the system productively for truly monitoring. Most of the today systems are loosely integrated, so functionalities away from the domain of operation of the system can utilize features, as a service from a remote external entity, rather than itself implementing it. Such a design allows for broader purpose of use and a better scalability as the information can be pulled instantly only when a request is processed.

Since the system naively uses such architecture, the possibility of expansion or integration is close to null. Mishaps in exercising of duties of investigating entities are unmanageable and thus may raise corruption within a tribunal entity. The workflow, is only utilized based on the traditional customary and relies on managing based on physical copies of documents, either completely duplicated or partially persisted. In such a situation, the possibility of data getting lost amidst the others is highly likely, leading to irregularity whose identification and rectification is a very time consuming and costly operation. Also the classification of records based on different criterion is time consuming as the query needs to eliminate duplicates and validate referential integrity.

Though, the existing system has some of few advantages which are like less hardware and software required, cheap in comparison of computerized system. Migration from the existing system to a new better system, calls for utilizing a rather robust requirement for the new automated system.

II. Proposed model

The Objective of designing a new system is to improve the monitoring capabilities over that of the conventional system. The proposed system suggests an adaptive method in Trial Monitoring System using RESTful Web Services and Authorization. The system provides a comprehensive suite where cases are created and the trial proceedings are monitored progressively to make informed decisions that includes assignment of investigating entities, requesting advisors for opinions to take over the prosecution of the case and applying counter petitions against an allegation's disposal by providing a platform to integrate proceedings in different levels of scrutiny and across tribunal entities, and aggregation of such a data, to classify cases for statistical information and relaying the information in presentable format.

2.1 System as RESTful Web Service

Considering REST as a design pattern is primarily to utilize the system architectural styles consisting of guidelines and best practices for creating scalable web services. It is by far optimal and supports readability of data with respect to conventional dynamic web services over HTTP. The logic of the controller is interfaced through a REST Handler, which maps the sanitized URI to one of the corresponding REST API Methods, which handle the service request.

2.1.1 REST Framework

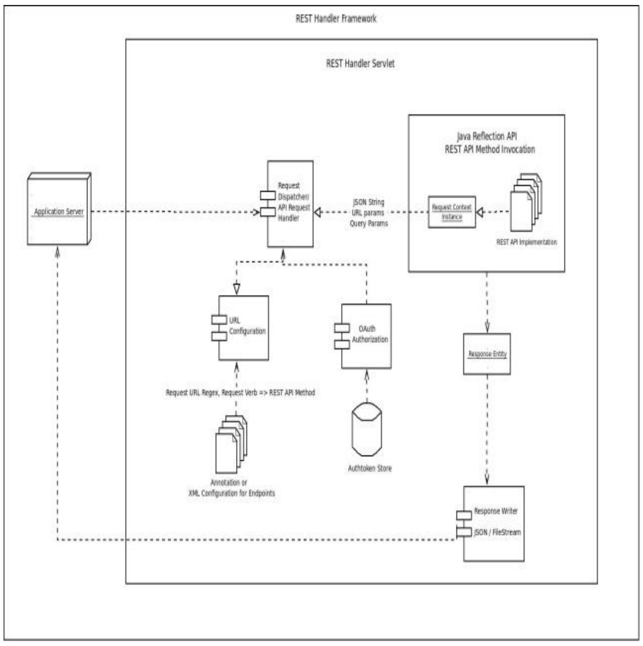

Figure 1: REST Framework

The Trial Monitoring System is proposed with a customized REST Handling Framework optimized to provide coherence with Role Based Authorization. It has been modelled to be development friendly and at the same time provide faster lookup.

2.1.2 Initiating REST Service

The framework is written with a start point of a Servlet which accepts all requests within the API context, and invokes (Java Reflection) a corresponding API method passing the context parameters, query parameters in case of GET request and additional parameter of JSON String that comes as the request payload in case of POST, PUT and DELETE requests.

The returned object from the method is an instance of either an entity in which case, it is converted to JSON or a File Stream which is written as raw bytes to the response body with content disposition of attachment. Exceptions in API, are specially handled by throwing custom exception instance and corresponding error message is written to the response.

2.1.3 Registering an Endpoint

The framework provides an elegant way to register an endpoint (map URL pattern with corresponding API method). Endpoints are registered by adding a custom annotation to each of the API methods providing the Request Method, URLRegex, and a name (that uniquely identifies an endpoint) or providing the mapping in XML files corresponding to each Request Method.

2.1.4 Optimization:

A URL Configuration singleton is instantiated, which reads the annotated methods and configuration XMLs for the endpoints. An in-memory mapping is constructed eliminating duplications. The URL Regex is compiled for once and saved in memory so that subsequent matching time is reduced. Each of the endpoints are stored in the database as an Activity and indexed using an integer key. The optimization with respect to Role Based Authorization is achieved by using this key of each endpoint for comparison, eliminating a costly operation of comparison of strings, thus making it faster.

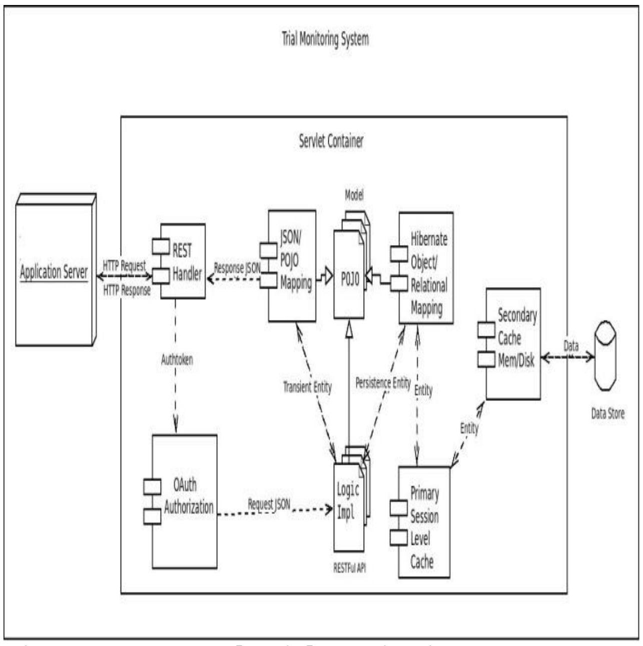

Figure 2: Proposed Trial Monitoring System

The proposed REST Handler Framework consists of a Servlet, which accepts all the requests from the context of the API. Statelessness is achieved by authorization from accepting the "Authorization Token" which is passed as with Request Header. This token is passed to the Authorization Service, which authorizes the given request to access the REST APIs, on validation against a list of services the user possessing the authtoken is eligible to exercise. The token is primarily obtained from a trusted entity, which is a RESTful service of Identity Access Management, by requesting for a token, providing the required identity information. A user may obtain more than one token and thus use the token as a grant towards executing the listed services.

Upon Authorization, the request entity is State Transferred from conventional JSON to a Java instance which is passed to the corresponding REST API corresponding the Request URI and the Request Verb (GET, POST, PUT, DELETE) used. The Response from the REST API is either one of (1) Response Entity corresponding to the given request. (2) A file or stream of content, whose content disposition is of an attachment. (3) An Error explaining any or all of exceptions with the request Entity.

Each response is associated with a status code explaining the status of the response. Conventionally, the status code corresponds to the HTTP status codes. The framework is customized in a way that allows for a clearer API implementation. It uses the context parameters reference as a single point message passing between the REST API and the REST Handler, which takes care of the Representational State Transfer. This is unlike the existing REST Frameworks that uses annotations for mapping requests and constructing separate Response object for each request, and that forces the developer to declare method arguments for each input from the request which leads to more redundancy.

2.2 Role Based Authorization

An important component of the proposed system is to exercise Role Base Authorization. Every user is associated with a predefined Role, which allows the minimum level of services accessibility of the system for that role. The system allows for customization of role permissions and overriding the access constraints to any particular user. This allows reusability and complete customization of actions and corresponding events.

2.3 Productivity and Collaboration

The system opens scope for enhanced productivity and seamless integration between other systems through reverse API, also known as WebHooks. Basically, the system incorporates a event driven approach, which allows logging of event actions remotely through an access point that connects the system's Internal API to the external API. This is done by registering a HTTP URL endpoint for selected actions within the system. The system triggers a HTTP Request to these selected endpoints in the event of selected action, passing the event data JSON as URL Parameter.

Examples of integration include integration with remote Email Server, or SMS Server, which can be used for notification of the event and ease of monitoring. More sophisticated integrations include, integration of the system with another system that runs independently and sinks services with a different level of scrutiny, which requires the data in this system, for preliminary processing.

The system offers reporting as a service, which helps classify the cases based on different criterion and relay the information in a presentable manner. The system offers collaboration of different entities by allowing every participating entity to comment and discuss on a case for making informed decision.

2.4 Platform Independence

The system operates externally using JSON standard representation, which provides a consistent data format for interoperability. Further the system suggests usage of Java as a platform, as the executable modules are platform independent and thus can run across multitude of operating systems and usage of Object Relational Mapper as an interface of accessing the database, eliminating the dependency on schema implementation across different database management systems.

2.5 Performance Evaluation

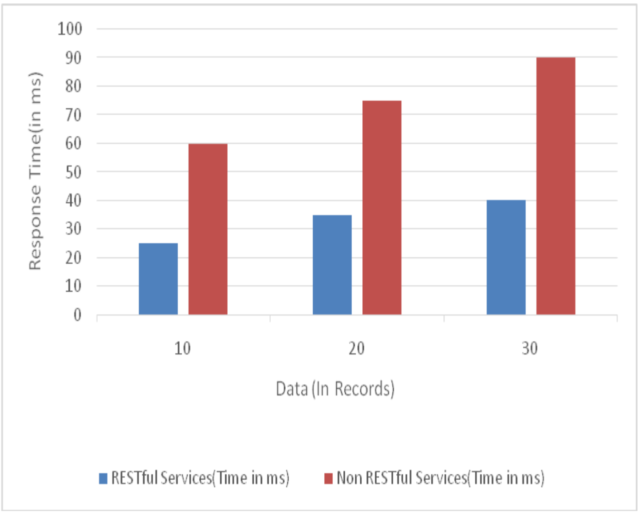

Figure 3: Performance Measure: Service Architecture

RESTful Service implementation using JSON provides better semantics and data representation, hence the added payload of other HTML, CSS or other resources for constructing the web page are loaded once. Raw JSON is used to populate the data in the webpage, thus reducing the response time for each request.

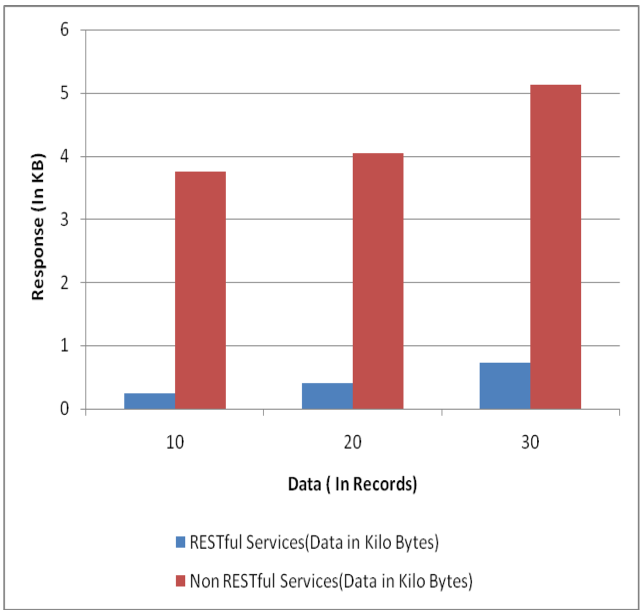

Figure 4: Performance Measure: Data Size

Conventional dynamic web apps mix HTML code with raw data, thereby adding overhead each time to the response message. On the contrary, in the RESTful implementation, Raw JSON is used to populate the data in the webpage, thus reducing the data size for each response.

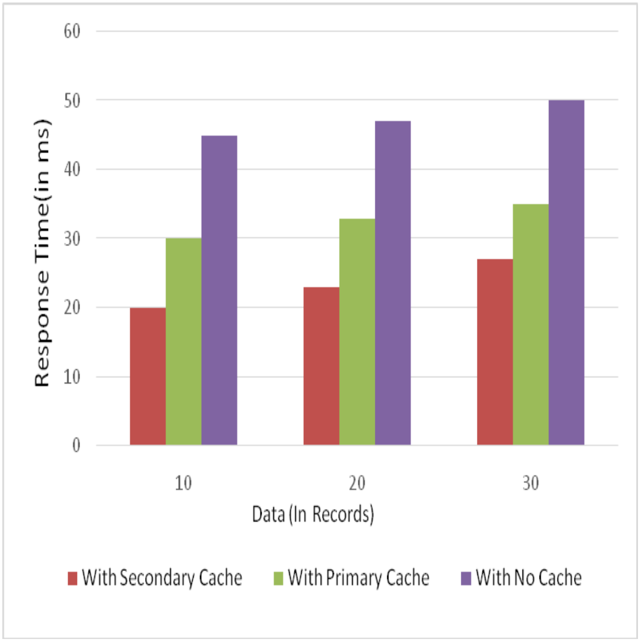

Figure 5: Performance Measure: Caching

Since the system has been modeled as entities, the query overhead and database lookup overheads can be optimized by introducing caching of entities. It has since been observed that the performance of system can still be optimized by introducing a secondary level of caching, with extended time to live, thus eliminating the buffering in the primary cache thereby lowering the response time.

III. Conclusions

The proposed Trial Monitoring System provides an adaptive method for monitoring case entities. Assuming that all the requirements are intact, the system is scalable to better levels and secure. RESTful implementation in Trial Monitoring System is the first of its kind in serving large data based on the user roles. Data access is authorized at varied levels based on the role of the user. Future work in this system, include integration of the system to higher levels of the tribunal entity and provide varied notification such as SMS and Email as a service from an external provider.

REFERENCES

[1] Leonard Richardson, Sam Ruby, David Heinemeier Hansson, *RESTful Web Services* (Sebastopol, O'Reilly Media, 2007)
[2] Bill Burke, *RESTful Java with JAX-RS* (Sebastopol, O'Reilly Media, 2009)
[3] Subbu Allamaraju, *RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity* (Sebastopol, O'Reilly Media / Yahoo Press, 2010)
[4] Gavin King, Christian Bauer, *Java Persistence with Hibernate* (Greenwich, Manning Publications, 2006)
[5] David Heffelfinger, *JasperReports 3.5 for Java Developers* (Birmingham, Packt Publishing Ltd, 2009)
[6] Charbonneau. N, Newman. B, Pecelli. D, Security Incident Origin Discovery (SIOD) IP Transaction Tracking for Centralized Cyber Defense, Proc. *Military Communications Conference (MILCOM) IEEE, Baltimore*, MD, 2014, 30-39.
[7] Rouached. M, Sallay. H, RESTful Web Services for High Speed Intrusion Detection Systems, *Proc. Web Services (ICWS), IEEE 20th International Conference, Santa Clara*, CA, 2013, 62-622.