Design Specifications for Cisco Log Parser Project

Log Files

The logs in the log files have the structure of:
TIMESTAMP | TYPE | THREAD | | CLASS | MESSAGE |
in two log files named "spf-device-manager" and "spf-service-manager"

Database

The data base in this project has one table called 'info' with the following structure:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
time	datetime	YES		NULL	
ms	int(11)	YES		NULL	
type	varchar(30)	YES		NULL	
threadname	varchar(100)	YES		NULL	
classname	varchar(100)	YES		NULL	
message	longtext	YES		NULL	
exception	text	YES		NULL	
servicename	varchar(50)	YES		NULL	

Implementation specifications

- The project was developed in the Eclipse IDE environment. using a maven project structure.
- Hibernate ORM was used to do the connection and operations with the database.
- Also the project implemented Spring framework for the fetching of the service beans that run the Hibernate ORM.
- Structure of the project contained DAO for the logs which for simplicity was just left as the POJO of the LOG class.
- RESTful API was implemented using the Jersey framework to achieve the backend aspect. the API uses http GET requests and returns JSONs containing required log objects.
- The API allows for multiple searches among the recorded logs, including: search by ID, by Class, Thread, Type, filtering logs for those with exception, and filtering logs between certian dates.
- UI was implemented mainly using the JQuery widget "DataTables" and some filtering feilds made manually using JavaScript.
- Unit testing was implemented using the JUnit testing framework with Mockito for testing the logic of various services in the project.
- Continuous integration was also implemented using Jenkins and Git.

UI

using HTML, CSS, and JS we created a simple website to view and interact with the logs