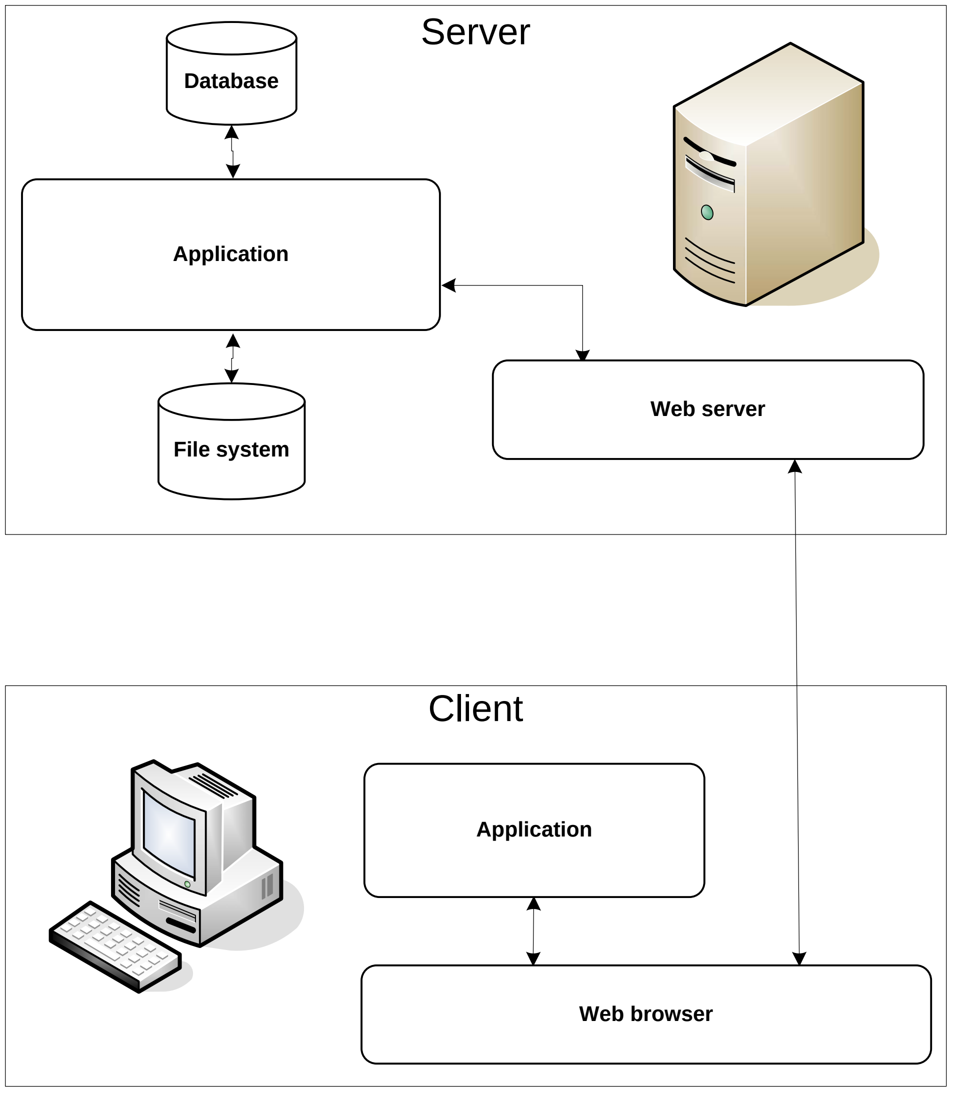
2.2 Overall Architecture Description



The general architecture applied is a two tier (semi) fat client web service system, meaning that the server basically does nothing more then receive requests, processes them and sends back the requested information. Semi-fat means that there’s an event driven application on the client side that does some of the html generation and injection. The actual pages being viewed aren’t fully complete (at least in some cases), the missing information is fetched and processed real time when the page is loading and/or as a response to user interaction.

On the server side there exists a web server which handles the low level http protocol requests and reroutes them to the web service application which decides how to respond. The application may also request information from the database or from a separate file.

On the client side a fairly modern web browser handles the requests to the server and presents the results to the end user. In aid of presenting the information and interacting with the user, the browser uses various applications.

2.3 Detailed Architecture

In this section a more detailed version of the architecture is explained.

2.3.1 Server Architecture

The server as a whole is made out of several major components as depicted in the figure in section 2.3.1.2. Each of the individual components will be described in the next section followed by a control and data flow description in section 2.3.1.2.

2.3.1.1 Server Architecture Modules

Tomcat

Apache Tomcat[1] is a web server derived from Apache Hypertext Transfer Protocol (HTTP) Server[2] which adds support for Hypertext Transfer Protocol over Secure Socket Layer (HTTPS) protocol as well as being the standard implementation for JEE Web Application Technologies.

Java Enterprise Edition (JEE) Web Application Technologies

The JEE Web Application Technologies[3] is a collection of technologies for Java based web applications. Two of these are Java Servlets and Java Server Pages (JSP) which we will be using extensively. It is basically a layer or interface linking the web server with the custom written code Java code.

When an incoming http request (which is mapped to some servlet) is received by Tomcat it is forwarded to the JEE Web Application technologies which will handle the details concerning the http request and then call the correct custom Servlet. The JEE Web services are run inside/on top of the Java Virtual Machine (JVM).

[1] http://tomcat.apache.org/
[2] http://httpd.apache.org/
[3] http://java.sun.com/javaee/technologies/webapps/

Service Application

The service application is a collection of custom written Servlets and JSP pages that will receive different requests made by the user in the form of HTTP requests, process them and if required, return the appropriate information. It is here that all of the server side logic of our software system resides. It will use the database (through Hibernate) to store all of the data concerning the users and the system, and the file system to store the user drawings.

Hibernate

Hibernate[4] is a sort of middleware which will simplify the integration of a Structured Query Language (SQL) database with the rest of the application by acting as a link between the programming language and the database, doing on the fly (two way) translation between Java objects and SQL. In simplified terms this means that it will automatically transform Java objects to SQL queries and database table rows. It will also do the opposite, that is, transform the result of a query back to Java objects. This will simplify the development of a data intense application such as this as the code need not to be cluttered with SQL code and will also greatly simplify the evolution of the data model as the code more or less needs not to be aware of the database schema.

JVM

The JVM will run all the java related code. It needs to comply to the Java 1.5 standard as Generics will be used.

The database

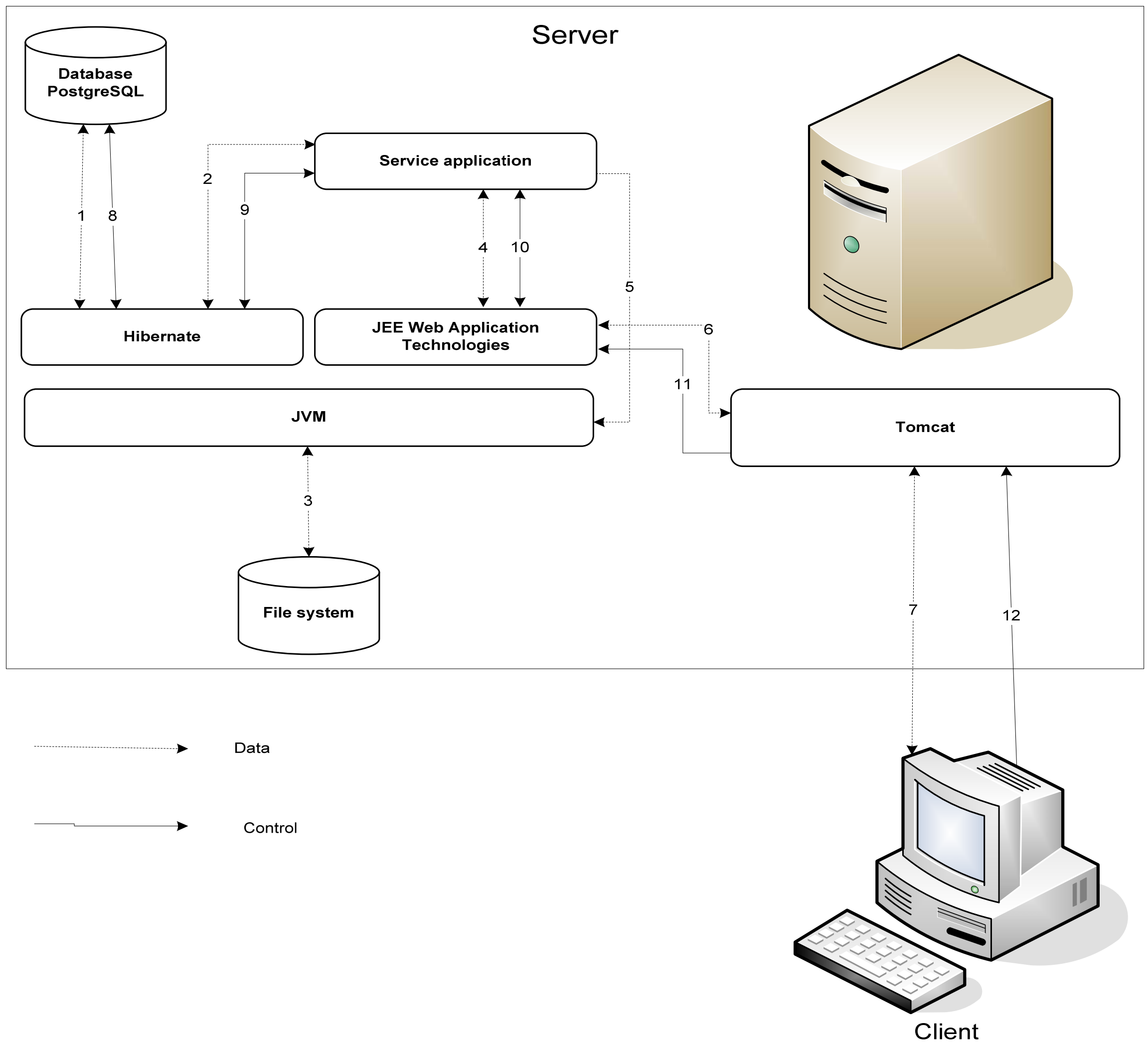
The database will be a standard PostgreSQL[5] setup which need not be on the same physical machine as the Apache Tomcat server.

The file system

The underlying file system will be used to store the user created pictures. As the server side application is entirely written in Java, it is independent of the platform. So the only requirement on the file system is that it supports long file names (up to 256 characters) and can be a network mapped drive.

[4] http://www.hibernate.org/
[5] http://www.postgresql.org/

2.3.1.2 Server architecture control and data flow



This section will provide a general and simplified overview of the control and data flow of the architecture model. The text will often be followed by a number inside a parenthesis. This number is a reference to the numbered arrows in the figure above. The different steps in this process are described below.

1. The client will make HTTP/HTTPS requests to the server (12). In some cases (such as login) this request will be made synchronously, meaning that the control will be given over to the server and nothing more will occur until a response (7) has been received. But in most case it be an asynchronous request and the client can go on to do whatever it wishes.

2. Tomcat will handle this request and forward (11) it to JEE Web Application layer along with any additional data (6) in the HTTP request.

3. The JEE Application layer will in turn forward (10) this to the correct custom written Servlet or JSP along with the parsed extra data (4).

4. The custom servlet will process the request and gather any additional data required through Hibernate (9) which will return the data (2).

5. Hibernate will translate the request to SQL and send (8) a query to the database. It will receive the response (1) and transform it back to Java objects which will be returned (2) to the custom Servlet.

6. Having the required data, a correct response can now be constructed by the custom Servlet. It might require a specific file which will be fetched (5 and 3).

7. The response is now constructed by the Servlet and returned (4) to JEE Web Application layer which returns it to Tomcat (6), which in turn, forwards (7) it to the client.

8. The request is now complete.

2.3.2 Client Architecture

The client is also made out of several major components. Again, first a description of the components then a description of the control and data flow is described.

2.3.2.1 Client Architecture Modules

Web browser

The web browser serves as the HTTP client that makes and handles HTTP requests and responses to and from the server. It views the html and host the JavaScript interpreter, and plugins such as Flash Player.

Flash Player

The Flash Player is an ActionScript engine that executes ActionScript source code.

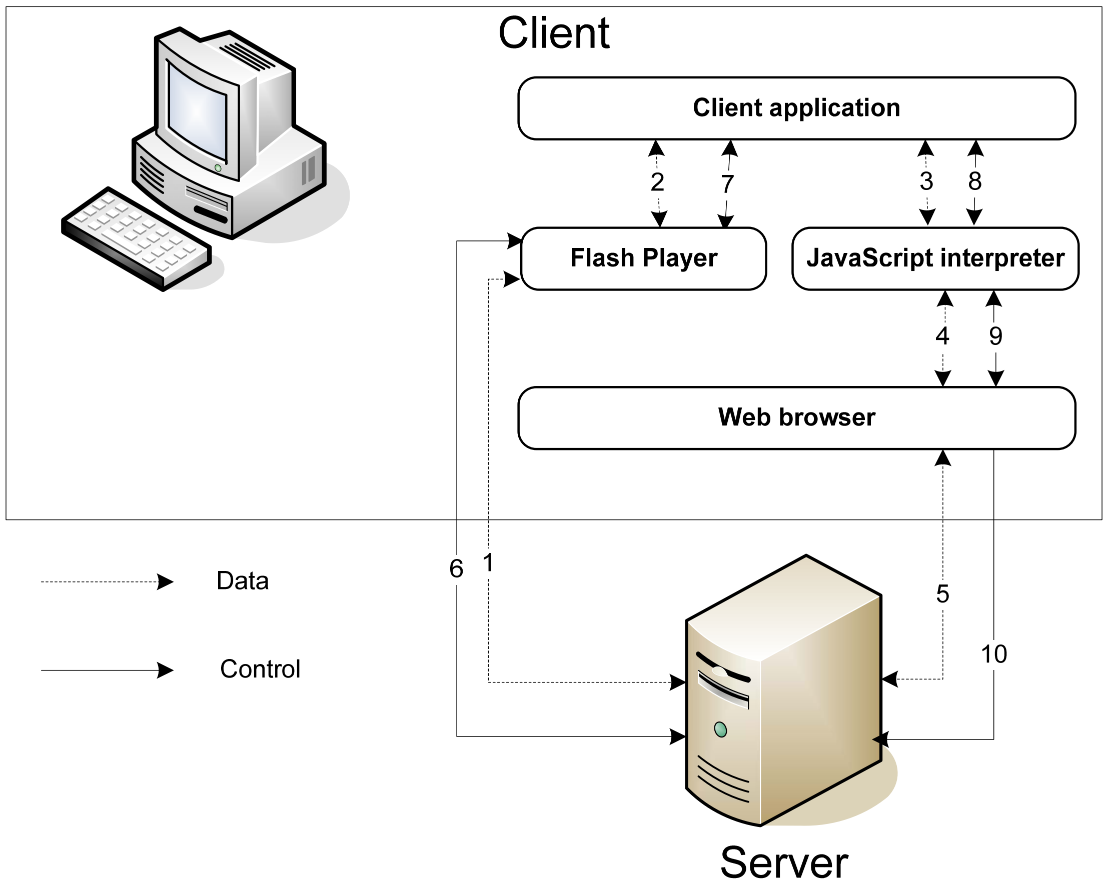
JavaScript interpreter

The JavaScript interpreter will run the actual client application. Which handles different events and performs the corresponding action.

Client application

The client application is a collection of custom ECMAScripts written mainly in JavaScripts with the exception of the one painting component that is written in ActionScript. It is here that all the client side logic of the system resides. Takes care of the user interaction with the system and generates HTML for different actions.

2.3.2.2 Server Architecture Control and Data flow



This section corresponds to section 2.3.1.2 and will provide a general and simplified overview of the Control and Data flow of the architecture model for the client. Again, a number wrapped in parentheses implies the respective arrow in the figure above. The different steps in the process are described below in the (mostly) general case.

1. The user interacts with the html page which triggers an event that is sent to (4) the JavaScript interpreter.

2. The JavaScript interpreter calls (3) the JavaScript function which is mapped to the corresponding event.

3. The function might need to fetch or register some data on the server. So an asynchronous HTTP request (8, 9, 10) is made.

4. The server will respond (5, 4, 3) which will trigger the corresponding JavaScript event handler (10, 9, 8)

5. The event handler will do the required action, i.e. generate the required HTML and inject it into the html page.