# Online Job Portal

**Introduction**

The Online Job Portal project is developed with the aim of revolutionizing the traditional job search process by providing a comprehensive online platform for job seekers and employers to connect and interact. In today's digital era, the need for efficient and user-friendly job portals has become increasingly important to bridge the gap between job seekers and employers.

The project is built using Java, JSP (JavaServer Pages), Servlets, JDBC (Java Database Connectivity), MySQL, Maven, MVC (Model-View-Controller) architecture, and is deployed on the Tomcat server. These technologies have been chosen for their robustness, scalability, and compatibility with web development.

The Online Job Portal offers a wide range of features and functionalities to both job seekers and employers, making the entire job search and recruitment process streamlined and efficient. Job seekers can create their profiles, upload resumes, search for job vacancies, and apply directly through the platform. On the other hand, employers can post job vacancies, search for suitable candidates, and manage the entire recruitment process seamlessly.

The project focuses on enhancing the user experience by providing a simple and intuitive interface. It aims to minimize the time and effort required for job seekers to find suitable opportunities and for employers to identify and hire qualified candidates. By leveraging the power of technology and connectivity, the Online Job Portal project aims to bridge the gap between job seekers and employers, facilitating efficient and effective recruitment processes.

Additionally, the project emphasizes the importance of security and privacy. It incorporates a robust authentication and authorization mechanism to ensure that only registered and authorized users can access the system. Furthermore, user data and confidential information are stored securely using industry-standard encryption techniques.

The project follows the MVC architectural pattern, which promotes code modularity, maintainability, and scalability. The separation of concerns ensures that the application's logic, data storage, and presentation layers are distinct and can be modified independently, making it easier to maintain and extend the project in the future.

Overall, the Online Job Portal project aims to revolutionize the way job seekers find employment opportunities and employers discover talented individuals. It provides a seamless and efficient platform that simplifies the job search and recruitment process, saving time and effort for all parties involved.

##### Overview of the Project

The Online Job Portal project is an advanced web application that serves as a platform for job seekers and employers to connect and streamline the recruitment process. It aims to simplify the job search and hiring processes by leveraging technology and providing a user-friendly interface.

The project offers a range of features and functionalities to cater to the needs of both job seekers and employers. Job seekers can create personalized profiles, upload their resumes, and search for job vacancies based on various criteria such as location, industry, experience level, and job type. They can also save job listings, track their applications, and receive notifications about new job opportunities.

Employers, on the other hand, can post job vacancies, specifying the required qualifications, responsibilities, and other details. They can search for suitable candidates based on specific criteria and manage applications received from interested job seekers. Employers also have access to applicant profiles, enabling them to review qualifications, contact candidates, and schedule interviews.

The project incorporates a secure user authentication system to ensure that only registered and authorized users can access the platform. User data and confidential information are stored securely using encryption techniques, ensuring the privacy and confidentiality of both job seekers and employers.

The Online Job Portal project follows the MVC (Model-View-Controller) architecture, which provides a structured approach to development. The model layer handles data storage and retrieval, including interactions with the MySQL database using JDBC. The view layer, implemented using JSP (JavaServer Pages), focuses on rendering dynamic web pages and displaying information to users. The controller layer, implemented using Servlets, manages the flow of the application, handles user requests, and invokes appropriate actions.

Maven is used as a build automation tool, managing project dependencies and facilitating project management. The project is deployed on the Tomcat server, making it accessible through a web browser from any device with an internet connection.

The primary goals of the Online Job Portal project are to enhance the efficiency and effectiveness of the job search and recruitment processes. By providing a user-friendly interface, advanced search options, and seamless communication channels between job seekers and employers, the project aims to bridge the gap and facilitate successful job placements.

Overall, the Online Job Portal project aims to revolutionize the traditional job search and recruitment methods, making it easier for job seekers to find suitable employment opportunities and for employers to identify and hire qualified candidates. It empowers individuals in their job search and assists organizations in building their workforce efficiently.

**Abstraction**

Abstraction is a fundamental concept in programming and software development that involves simplifying complex systems by focusing on essential details while hiding unnecessary or irrelevant information. It allows us to create models, interfaces, and classes that capture the essential characteristics and behavior of a system without getting into the implementation details.

In the context of the Online Job Portal project, abstraction can be applied at various levels:

**User Interface Abstraction:** The user interface of the Online Job Portal can be designed with abstraction in mind. It should provide a clean and intuitive user experience, hiding unnecessary complexities and presenting only the relevant information and actions to the users. The interface should be designed with the user's perspective in mind, abstracting away the underlying technical complexities.

**Modular Code Abstraction:** The project's codebase can be organized into modular components, each responsible for a specific set of functionalities. Each module can encapsulate its internal logic and data structures, providing a simplified and abstracted interface for interaction with other modules. This way, the complexities of individual modules are hidden, promoting code reuse, maintainability, and ease of understanding.

**Data Abstraction:** The data model of the Online Job Portal can be designed with abstraction in mind. It should focus on capturing the essential entities, attributes, and relationships required for the system's functionality while hiding unnecessary details. By abstracting the data model, it becomes easier to manipulate and interact with the data without being overwhelmed by low-level implementation details.

By applying abstraction in these various aspects, the Online Job Portal project can achieve a higher level of simplicity, maintainability, and scalability. Abstraction helps in managing complexity, promoting code reusability, and facilitating the understanding and maintenance of the system. It enables developers and users to focus on the essential aspects of the project while abstracting away unnecessary complexities, resulting in a more efficient and manageable software solution.

**Technologies Used**

The online Job Portal project utilizes several tools and technologies to build a robust and efficient e-commerce platform. The key tools and technologies employed in the project include:

Java: Java is a widely used programming language known for its versatility, platform independence, and object-oriented approach. It forms the foundation of the project, providing the core logic and functionality.

**JSP (JavaServer Pages):** JSP is a technology that allows the creation of dynamic web pages using Java. It enables the seamless integration of Java code within HTML pages, facilitating the generation of dynamic content and interaction with the server.

**Servlets:** Servlets are Java classes that extend the capabilities of a server, enabling the processing of requests and generating dynamic responses. They play a crucial role in handling user interactions, managing sessions, and processing form data in the online nursery shop project.

**HTML/CSS:** HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are the fundamental building blocks of web pages. HTML provides the structure and content of the web pages, while CSS handles the visual presentation and layout.

**JavaScript:** JavaScript is a client-side scripting language used for enhancing interactivity and user experience on web pages. It enables dynamic content updates, form validation, and event handling.

**MySQL:** MySQL is a popular open-source relational database management system (RDBMS). It is used to store and manage the project's data, including product information, user profiles, orders, and reviews.

**JDBC (Java Database Connectivity):** JDBC is a Java API that allows Java applications to interact with relational databases. It provides a standardized interface for connecting to databases, executing queries, and managing data.

**IDE (Integrated Development Environment):** An IDE such as Eclipse or IntelliJ IDEA can be used to write, debug, and test Java code efficiently. These IDEs provide a range of tools and features that enhance productivity during the development process.

**Apache Tomcat:** Apache Tomcat is a widely used web server and servlet container. It is responsible for hosting and executing the Java web application, providing the runtime environment for JSP and Servlets.

These tools and technologies form the backbone of the online nursery shop project, enabling the development of a scalable, user-friendly, and secure e-commerce platform. They provide the necessary frameworks and functionalities to implement the project's key features and ensure a seamless shopping experience for customers.

**Technology/Domain:** Java

**Front-End:** JSP, Html, CSS, JS, Bootstrap.

**Server-side:** Servlet.

**Back-end:** MYSQL.

**Server:** Tomcat 8.5

**System Architecture(MVC)**

The online Job Portal project follows the Model-View-Controller (MVC) architectural pattern to structure and organize the codebase effectively. MVC separates the application into three interconnected components: the Model, View, and Controller. Let's explore how each component is utilized in this project:

**Model:**

The Model represents the data and business logic of the application. In the online nursery shop project, the Model consists of classes that handle data management and processing. This includes classes for managing user profiles, product information, order details, and reviews. The Model component is responsible for interacting with the database, retrieving and updating data, and performing necessary calculations or validations.

**View:**

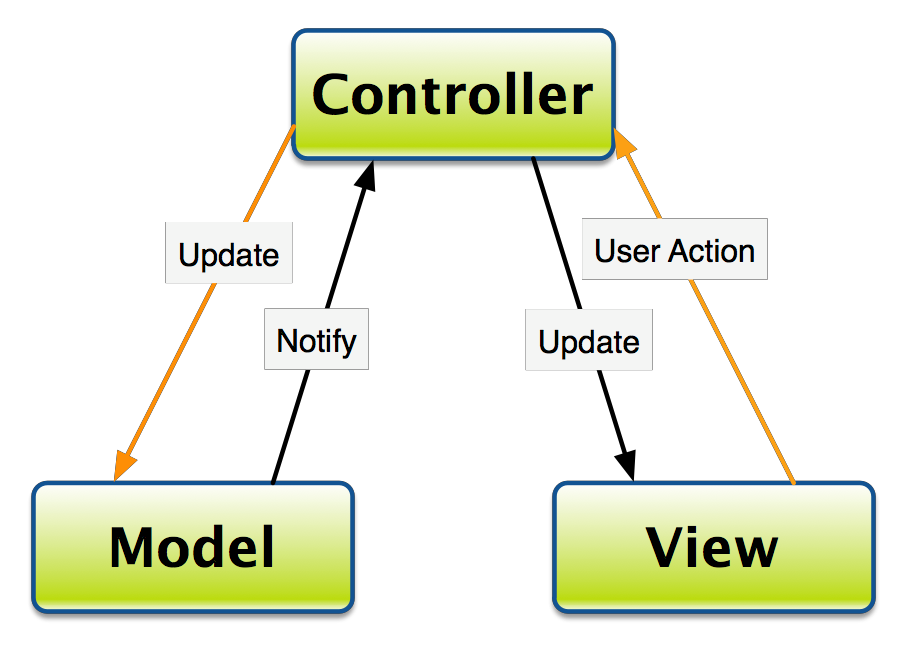
The View is responsible for presenting the user interface to the customers. In this project, the View component comprises JSP (JavaServer Pages) files and HTML templates. These files define the layout, structure, and visual representation of the web pages. The View component interacts with the Model to fetch and display the relevant data, ensuring a dynamic and interactive user experience. It also handles user input and form submissions, forwarding data to the Controller for processing.

**Controller:**

The Controller acts as an intermediary between the Model and the View, handling user requests, coordinating data flow, and controlling the application's behavior. In the online nursery shop project, the Controller component consists of Servlets that receive and process user requests from the View. Servlets interact with the Model to fetch or update data, perform necessary operations, and direct the appropriate response to the View. They also manage session handling, authentication, and request routing.

The MVC architecture in this project promotes separation of concerns, making the codebase modular and maintainable. It allows for independent development and testing of each component, enhancing code reusability and flexibility. The Model encapsulates the data and business logic, ensuring data integrity and consistency. The View focuses on presenting the information to the users in an aesthetically pleasing and user-friendly manner. The Controller orchestrates the flow of data and controls the application's behavior based on user input.

By adopting the MVC architecture, the online nursery shop project ensures a clear separation of responsibilities, facilitating collaborative development and future enhancements. It enhances code readability, scalability, and maintainability, making it easier to modify or extend specific components without impacting the entire system. The MVC pattern also enables efficient debugging and troubleshooting by isolating potential issues within specific components.



**Development Process**

**Requirements gathering and analysis:**

Requirements gathering and analysis is a crucial phase in the development process of any software project, including the online nursery shop project. It involves understanding and documenting the needs, expectations, and constraints of stakeholders to define the project's scope and deliverables. Here is an overview of the requirements gathering and analysis process for the online nursery shop project:

**Identify Stakeholders:**

The first step is to identify the key stakeholders who will be impacted by the project. This may include customers, administrators, managers, and other relevant individuals. Understanding their perspectives and requirements is essential for ensuring the project's success.

**Conduct Interviews and Workshops:**

Interviews and workshops with stakeholders are conducted to gather insights about their expectations and goals. These sessions provide an opportunity to ask specific questions, clarify doubts, and obtain detailed information about the desired features, functionalities, and user experience.

**Document Functional Requirements:**

Functional requirements define the specific actions and behaviors the system should exhibit. These requirements outline the core features and functionalities expected from the online nursery shop, such as user registration, product catalog browsing, shopping cart management, order placement, and payment integration. Each requirement should be documented in a clear and unambiguous manner.

**Identify Non-Functional Requirements:**

Non-functional requirements specify the attributes and qualities of the system, including performance, security, usability, scalability, and reliability. For example, the project may require fast page load times, secure user authentication, responsive design for mobile devices, and the ability to handle a large number of concurrent users. These requirements help shape the overall system architecture and guide the selection of appropriate technologies.

**Consider Legal and Regulatory Requirements:**

Depending on the nature of the project, it is important to consider any legal and regulatory requirements that need to be addressed. This may include data protection and privacy regulations, secure payment processing, or compliance with industry standards.

**Analyze and Prioritize Requirements:**

Once all the requirements are gathered, they should be analyzed to identify any conflicts or inconsistencies. The requirements can then be prioritized based on their importance and impact on the project's success. This helps in allocating resources effectively and managing the project timeline.

**Create Requirement Specifications:**

The gathered requirements should be documented in a comprehensive manner, creating requirement specifications. These documents serve as a reference for the development team and other stakeholders throughout the project's lifecycle. They provide a clear understanding of what needs to be implemented and serve as a basis for future verification and validation activities.

Requirements gathering and analysis is an iterative process, involving continuous communication and collaboration with stakeholders. It is important to ensure that the requirements are well-documented, understood, and agreed upon by all parties involved. Any changes or updates to the requirements should be carefully managed through a formal change control process to maintain project alignment and minimize scope creep.

By conducting thorough requirements gathering and analysis, the online nursery shop project can establish a solid foundation for successful development, ensuring that the end product meets the needs and expectations of its stakeholders.

**Database design**

Database design is a critical aspect of the online nursery shop project, as it determines the structure and organization of data to support the system's functionality and ensure efficient data management. Here are key considerations for the database design:

**Identify Entities and Relationships:**

Start by identifying the main entities within the online nursery shop system. These entities could include users, products, orders, reviews, and categories. Determine the relationships between these entities, such as one-to-many, many-to-many, or one-to-one relationships. For example, a user can have multiple orders, and an order can contain multiple products.

**Define Tables and Fields:**

Create database tables to represent each entity. Each table will have columns that correspond to specific fields or attributes of the entity. For instance, the "Users" table may have columns like user ID, name, email, password, and address. Similarly, the "Products" table may have columns like product ID, name, description, price, and category ID.

**Establish Primary and Foreign Keys:**

Identify the primary key for each table, which uniquely identifies each record in that table. For example, the user ID in the "Users" table or the product ID in the "Products" table can serve as the primary key. Establish appropriate foreign keys to establish relationships between tables. For instance, the order table could have a foreign key referencing the user who placed the order.

**Normalize the Database:**

Apply normalization techniques to ensure data integrity and eliminate redundancy. Normalize the tables by breaking them down into smaller, more manageable tables and reducing data duplication. The normalization process helps in minimizing data anomalies and maintaining consistency within the database.

**Consider Data Constraints:**

Define appropriate data constraints to ensure data accuracy and integrity. This includes setting data types for each field, specifying field lengths, enforcing unique constraints, and defining default values or validation rules. For example, the email field in the "Users" table can have a unique constraint, and the price field in the "Products" table can have a constraint to only accept positive values.

**Plan Indexing and Optimization:**

Identify the fields that will be frequently used for searching or filtering data and consider creating indexes on those fields. Indexing can significantly improve query performance. Additionally, plan for database optimization techniques such as query optimization, caching, and database tuning to enhance system performance.

**Consider Database Security:**

Implement appropriate security measures to protect the database and its data. This may involve setting up user roles and permissions, encrypting sensitive information, and implementing measures to prevent unauthorized access or data breaches. Ensure compliance with relevant data protection regulations.

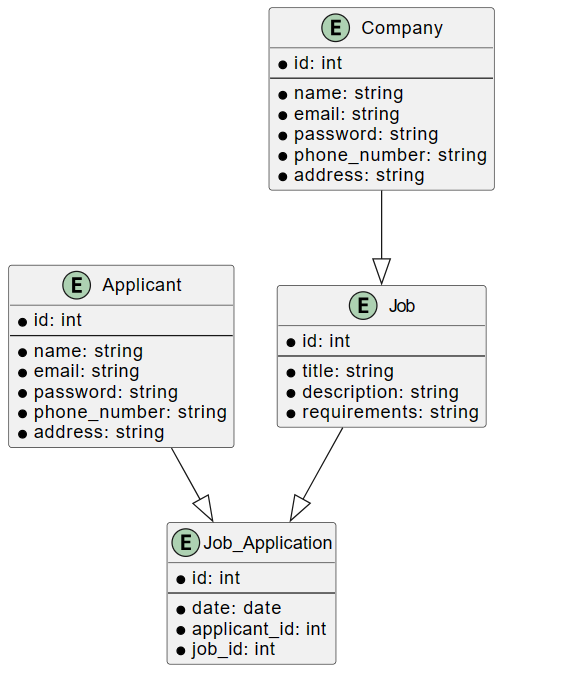
**Test and Refine the Database Design:**

Perform thorough testing of the database design, including inserting sample data, running queries, and verifying the results. Identify any design flaws or performance bottlenecks and make necessary refinements to optimize the database structure and ensure its efficiency.

It is essential to consider the scalability and future growth of the online nursery shop project during the database design phase. Plan for potential expansion, such as accommodating a larger product inventory or an increasing customer base. Regular maintenance and monitoring of the database will be necessary to ensure its optimal performance and data integrity as the project evolves.

**Entity Relationship Diagram**

An Entity Relationship Diagram (ERD) is a visual representation of the entities, attributes, and relationships within a database. It provides a clear and concise overview of the database structure and helps in understanding the relationships between different entities. Here is an example of an ERD for the online nursery shop project:



**System Design**

System design in the context of the online nursery shop project involves defining the architecture and components that make up the system, determining their interactions and dependencies, and ensuring that the system meets the desired requirements. Here are key considerations for the system design:

High-Level Architecture: Determine the overall architecture of the system. This could involve a multi-tier architecture, where the system is divided into presentation layer, business logic layer, and data layer. Alternatively, a microservices architecture could be considered, where different functionalities are split into separate services.

Component Design: Identify the major components of the system and define their functionalities. For example, components could include user management, product catalog, shopping cart, order management, and payment processing. Assign responsibilities to each component and define the interfaces through which they communicate.

Data Flow and Integration: Define how data flows between different components and modules of the system. Identify the data exchange mechanisms, such as API calls, message queues, or direct database interactions. Determine how the different components integrate with each other to ensure smooth data flow and consistency.

Scalability and Performance: Consider the scalability requirements of the system. Design the system in a way that allows it to handle increased load and accommodate future growth. This could involve techniques such as load balancing, caching, and database optimization to ensure optimal performance.

Security: Address the security aspects of the system design. Define mechanisms for user authentication and authorization. Implement measures to protect sensitive data, prevent SQL injections, cross-site scripting (XSS) attacks, and other common security vulnerabilities. Consider encryption, secure communication protocols, and best practices for data protection.

Error Handling and Logging: Incorporate error handling mechanisms to gracefully handle exceptions and errors. Implement logging functionality to record system activities, errors, and debugging information. This will assist in diagnosing issues and tracking system behavior.

User Interface Design: Design an intuitive and user-friendly interface for the online nursery shop. Consider usability principles, responsive design for different devices, and accessibility requirements. Create wireframes or prototypes to visualize the user interface and gather user feedback.

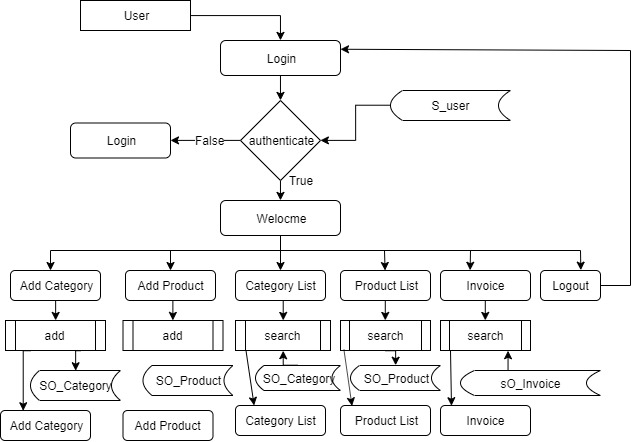
**Data Flow Diagram**

A Data Flow Diagram (DFD) provides a graphical representation of how data flows within a system, illustrating the processes, inputs, outputs, and data stores. Here is an example of a high-level DFD for the online nursery shop project:

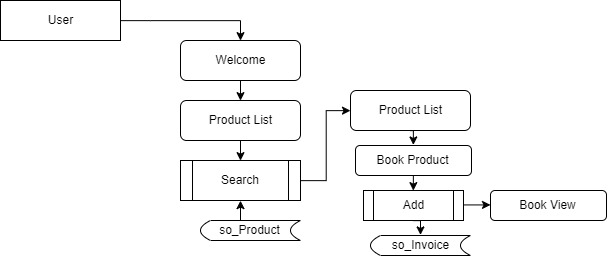
**Level 0 DFD**



##### Level 1 DFD (Administrative Panel) :



##### Level 1 DFD (User Panel) :



**Development**

Development in the context of the online nursery shop project involves the actual implementation of the system based on the requirements, design, and chosen technologies. Here are key aspects of the development process:

**Technology Selection:** Based on the project requirements and the desired system architecture, select the appropriate technologies for development. This may include Java as the programming language, JSP and Servlets for server-side development, HTML/CSS/JavaScript for client-side development, and a relational database management system (such as MySQL or PostgreSQL) for data storage.

**Environment Setup:** Set up the development environment with the necessary tools and frameworks. Install the required software, such as an Integrated Development Environment (IDE) like Eclipse or IntelliJ, a web server like Apache Tomcat, and any additional libraries or frameworks that will be used during development.

**Modular Development:** Adopt a modular approach to development, breaking down the system into smaller, manageable modules or components. Each module should have well-defined responsibilities and interfaces. This promotes code reusability, maintainability, and easier collaboration among developers.

**Backend Development:** Begin by developing the backend components of the system, such as the servlets and Java classes responsible for handling requests, processing data, and interacting with the database. Implement the business logic, validation rules, and data manipulation functionalities.

**Frontend Development:** Concurrently, work on the frontend components of the system, including the user interface design and implementation. Develop the JSP files, HTML templates, CSS stylesheets, and JavaScript code to create a visually appealing and interactive user interface. Ensure responsiveness for different devices and browsers.

Database Implementation: Implement the database design by creating the necessary tables, columns, and relationships based on the defined database schema. Write SQL queries or use Object-Relational Mapping (ORM) frameworks like Hibernate to interact with the database from the backend code.

Testing and Debugging: Perform thorough testing at various stages of development. Write unit tests to validate the functionality of individual components and integration tests to ensure the smooth interaction between different modules. Use debugging tools to identify and fix any issues or errors that arise during development.

**Iterative Development:** Adopt an iterative and incremental development approach. Develop and test the system in iterations, delivering incremental functionality and gathering feedback from stakeholders. This allows for early validation of requirements, quicker identification of issues, and flexibility in accommodating changes or enhancements.

**Version Control and Collaboration:** Use a version control system like Git to manage the source code, enabling collaboration among developers, tracking changes, and ensuring code stability. Establish proper branching and merging strategies to manage concurrent development and enable easy rollbacks if needed.

**Documentation:** Document the development process, including system architecture, code structure, and any specific implementation details. Maintain up-to-date documentation for future reference, maintenance, and onboarding of new team members.

**Deployment and Release:** Prepare the system for deployment to a production environment. Set up the necessary infrastructure, configure servers, and deploy the application. Perform final testing in the production environment before making the system available to users.

**Continuous Improvement:** Development is an ongoing process, even after the initial release. Continuously gather user feedback, monitor system performance, and address any bugs or issues that arise. Plan for future enhancements and feature additions based on user requirements and evolving business needs.

By following a systematic and disciplined approach to development, the online nursery shop project can ensure the successful implementation of the system, meeting the defined requirements, and delivering a reliable and user-friendly online shopping experience

**Testing**

Testing is a crucial phase in the development lifecycle of the online nursery shop project. It involves the verification and validation of the system to ensure that it functions as intended, meets the specified requirements, and is free from defects. Here are key aspects of the testing process:

Test Planning: Define a comprehensive test plan that outlines the objectives, scope, and approach for testing. Identify the different types of tests to be conducted, such as functional testing, usability testing, performance testing, security testing, and compatibility testing. Determine the test environment, test data, and resources required for testing.

Unit Testing: Start with unit testing, which focuses on testing individual components or modules in isolation. Develop unit test cases for each component to verify its functionality and behavior. Use testing frameworks like JUnit for Java to automate the execution of unit tests and ensure consistent results.

Integration Testing: Conduct integration testing to validate the interactions between different components or modules of the system. Test the integration points, data flow, and dependencies between modules. This ensures that the system functions correctly as a whole and that there are no integration issues or data inconsistencies.

Functional Testing: Perform functional testing to validate the system's compliance with the specified requirements. Develop test cases based on functional requirements and user stories. Test various scenarios, inputs, and user interactions to verify that the system behaves as expected and produces the correct outputs.

Usability Testing: Evaluate the user-friendliness and ease of use of the online nursery shop through usability testing. Gather feedback from actual users or representative users to assess the intuitiveness of the user interface, the clarity of instructions, and the overall user experience. Identify areas for improvement and make necessary refinements to enhance usability.

Performance Testing: Test the performance of the system under expected and peak load conditions. Measure response times, throughput, and resource utilization to ensure that the system performs efficiently. Conduct stress testing and load testing to assess the system's scalability and ability to handle a large number of concurrent users or transactions.

Security Testing: Conduct security testing to identify and mitigate potential vulnerabilities and protect the system from security threats. Perform penetration testing, vulnerability scanning, and code reviews to detect any security weaknesses. Verify that sensitive information is properly encrypted, access controls are in place, and potential attack vectors are mitigated.

Compatibility Testing: Test the compatibility of the online nursery shop with different web browsers, operating systems, and devices. Verify that the system functions correctly across various platforms and configurations, ensuring a consistent user experience for all users.

Regression Testing: Perform regression testing to validate that new changes or bug fixes have not introduced any unintended side effects or regressions in the system. Re-run previously executed tests to ensure that the existing functionality has not been adversely affected.

Error and Exception Handling: Test the system's error handling capabilities by intentionally triggering errors, exceptions, or boundary conditions. Verify that appropriate error messages are displayed, and the system gracefully handles unexpected situations without crashing or compromising data integrity.

Test Documentation: Maintain comprehensive documentation of the test cases, test results, and any issues or defects encountered during testing. This documentation serves as a reference for future testing cycles and helps in identifying patterns, trends, and areas that require further attention.

Continuous Testing: Implement continuous integration and continuous testing practices to automate and streamline the testing process. Use tools like Jenkins or GitLab CI/CD to automate the execution of tests whenever there are code changes, ensuring early detection of defects and promoting faster feedback loops.

By conducting thorough and systematic testing, the online nursery shop project can ensure that the system meets the desired quality standards, functions as intended, and provides a reliable and satisfactory user experience. Regular testing throughout the development lifecycle helps in identifying and resolving issues early, reducing the risk of critical failures and enhancing

**Deployment**

Deploying the online nursery shop project on Tomcat localhost involves configuring the Tomcat web server to host the application locally. Here are the steps to deploy the project on Tomcat:

**Install Tomcat:** Download and install the Apache Tomcat server on your local machine. Ensure that you have the appropriate version of Tomcat that is compatible with your project.

**Build the Project:** Build the online nursery shop project using your preferred build tool, such as Apache Maven or Gradle. This step compiles the source code, resolves dependencies, and generates the necessary artifacts for deployment.

**Generate the WAR File:** Package the project into a Web Application Archive (WAR) file. A WAR file is a compressed file format that contains all the necessary files and resources for the web application. This can be done using the build tool or manually by creating a WAR file from the project's build output.

**Stop Tomcat:** If Tomcat is already running, stop the server to ensure a clean deployment. This can be done by running the shutdown script or by stopping the Tomcat service if running as a service.

**Deploy the WAR File:** Copy the generated WAR file to the webapps directory of your Tomcat installation. By default, this directory is located at <Tomcat installation directory>/webapps. Paste the WAR file in this directory.

**Start Tomcat:** Start the Tomcat server by running the startup script or by starting the Tomcat service. This will initiate the deployment process and Tomcat will automatically extract the contents of the WAR file and deploy the web application.

**Verify Deployment:** Once Tomcat has started, you can access the online nursery shop by opening a web browser and navigating to http://localhost:8080/<war-file-name>. Replace <war-file-name> with the actual name of the WAR file deployed in the webapps directory.

**Test the Application:** Interact with the online nursery shop application in the web browser to ensure that it is functioning correctly. Test different features, functionalities, and user flows to validate the deployment.

During the deployment process, Tomcat will create a separate context for the web application based on the name of the WAR file. This context can be accessed through the URL mentioned in Step 7.

Note: Ensure that the required database configurations, such as connection URL, username, and password, are correctly specified in the project's configuration files to establish a successful database connection during deployment.

By following these steps, you can deploy the online nursery shop project on Tomcat localhost and access it locally through the Tomcat server.

**Screenshots**

Take sceenshots from latest project and add them here.

**Source code**

package in.co.job.portal.bean;

import java.sql.Timestamp;

import java.util.Date;

public class JobBean extends BaseBean {

private String companyName;

private String language;

private String Description;

private Date postDate;

private String postByRecId;

private String address;

private long recruiterId;

private long jobId;

private long userId;

private Timestamp applayDate;

private String userName;

private String resumeFile;

public String getResumeFile() {

return resumeFile;

}

public void setResumeFile(String resumeFile) {

this.resumeFile = resumeFile;

}

public String getUserName() {

return userName;

}

public void setUserName(String userName) {

this.userName = userName;

}

public long getRecruiterId() {

return recruiterId;

}

public void setRecruiterId(long recruiterId) {

this.recruiterId = recruiterId;

}

public long getJobId() {

return jobId;

}

public void setJobId(long jobId) {

this.jobId = jobId;

}

public long getUserId() {

return userId;

}

public void setUserId(long userId) {

this.userId = userId;

}

public String getCompanyName() {

return companyName;

}

public Timestamp getApplayDate() {

return applayDate;

}

public void setApplayDate(Timestamp applayDate) {

this.applayDate = applayDate;

}

public void setCompanyName(String companyName) {

this.companyName = companyName;

}

public String getLanguage() {

return language;

}

public void setLanguage(String language) {

this.language = language;

}

public String getDescription() {

return Description;

}

public void setDescription(String description) {

Description = description;

}

public Date getPostDate() {

return postDate;

}

public void setPostDate(Date postDate) {

this.postDate = postDate;

}

public String getPostByRecId() {

return postByRecId;

}

public void setPostByRecId(String postByRecId) {

this.postByRecId = postByRecId;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

@Override

public String getKey() {

// TODO Auto-generated method stub

return null;

}

@Override

public String getValue() {

// TODO Auto-generated method stub

return null;

}

}

**package** in.co.job.portal.bean;

**import** java.util.Date;

**public** **class** MakeResumeBean **extends** BaseBean {

**private** String name;

**private** String email;

**private** String mobile;

**private** String gCourceName;

**private** String gInsName;

**private** String gPercentage;

**private** String hCourceName;

**private** String hInsName;

**private** String hPercentage;

**private** String sCourceName;

**private** String sInsName;

**private** String sPercentage;

**private** String skill;

**private** String hobbies;

**private** String pDetail;

**private** String declaration;

**private** Date date;

**private** String place;

**private** String objective;

**public** String getObjective() {

**return** objective;

}

**public** **void** setObjective(String objective) {

**this**.objective = objective;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getMobile() {

**return** mobile;

}

**public** **void** setMobile(String mobile) {

**this**.mobile = mobile;

}

**public** String getgCourceName() {

**return** gCourceName;

}

**public** **void** setgCourceName(String gCourceName) {

**this**.gCourceName = gCourceName;

}

**public** String getgInsName() {

**return** gInsName;

}

**public** **void** setgInsName(String gInsName) {

**this**.gInsName = gInsName;

}

**public** String getgPercentage() {

**return** gPercentage;

}

**public** **void** setgPercentage(String gPercentage) {

**this**.gPercentage = gPercentage;

}

**public** String gethCourceName() {

**return** hCourceName;

}

**public** **void** sethCourceName(String hCourceName) {

**this**.hCourceName = hCourceName;

}

**public** String gethInsName() {

**return** hInsName;

}

**public** **void** sethInsName(String hInsName) {

**this**.hInsName = hInsName;

}

**public** String gethPercentage() {

**return** hPercentage;

}

**public** **void** sethPercentage(String hPercentage) {

**this**.hPercentage = hPercentage;

}

**public** String getsCourceName() {

**return** sCourceName;

}

**public** **void** setsCourceName(String sCourceName) {

**this**.sCourceName = sCourceName;

}

**public** String getsInsName() {

**return** sInsName;

}

**public** **void** setsInsName(String sInsName) {

**this**.sInsName = sInsName;

}

**public** String getsPercentage() {

**return** sPercentage;

}

**public** **void** setsPercentage(String sPercentage) {

**this**.sPercentage = sPercentage;

}

**public** String getSkill() {

**return** skill;

}

**public** **void** setSkill(String skill) {

**this**.skill = skill;

}

**public** String getHobbies() {

**return** hobbies;

}

**public** **void** setHobbies(String hobbies) {

**this**.hobbies = hobbies;

}

**public** String getpDetail() {

**return** pDetail;

}

**public** **void** setpDetail(String pDetail) {

**this**.pDetail = pDetail;

}

**public** String getDeclaration() {

**return** declaration;

}

**public** **void** setDeclaration(String declaration) {

**this**.declaration = declaration;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

**public** String getPlace() {

**return** place;

}

**public** **void** setPlace(String place) {

**this**.place = place;

}

@Override

**public** String getKey() {

// **TODO** Auto-generated method stub

**return** **null**;

}

@Override

**public** String getValue() {

// **TODO** Auto-generated method stub

**return** **null**;

}

}

package in.co.job.portal.bean;

import java.sql.Timestamp;

import java.util.Date;

/\*\*

\* User JavaBean encapsulates TimeTable attributes

\*

\* @author Navigable Set

\* @version 1.0

\* @Copyright (c) Navigable Set

\*

\*/

public class UserBean extends BaseBean {

private String firstName;

/\*\*

\* Last Name of User

\*/

private String lastName;

/\*\*

\* Login of User

\*/

private String login;

/\*\*

\* Password of User

\*/

private String password;

/\*\*

\* Confirm Password of User

\*/

private String confirmPassword;

/\*\*

\* Date of Birth of User

\*/

private Date dob;

/\*\*

\* MobielNo of User

\*/

private String mobileNo;

/\*\*

\* Role of User

\*/

private long roleId;

/\*\*

\* Number of unsuccessful login attempt

\*/

private String gender;

/\*\*

\* Last login timestamp

\*/

/\*\*

\* accessor

\*/

/\*\*

\* @return FirstName Of User

\*/

public String getFirstName() {

return firstName;

}

/\*\*

\* @param FirstName

\* To set User FirstName

\*/

public void setFirstName(String firstName) {

this.firstName = firstName;

}

/\*\*

\* @return LastName Of User

\*/

public String getLastName() {

return lastName;

}

/\*\*

\* @param LastName

\* To set User LastName

\*/

public void setLastName(String lastName) {

this.lastName = lastName;

}

/\*\*

\* @return Login id Of User

\*/

public String getLogin() {

return login;

}

/\*\*

\* @param Login

\* Id To set User Login ID

\*/

public void setLogin(String login) {

this.login = login;

}

/\*\*

\* @return Password Of User

\*/

public String getPassword() {

return password;

}

/\*\*

\* @param Password

\* To set User Password

\*/

public void setPassword(String password) {

this.password = password;

}

/\*\*

\* @return Confirm Password Of User

\*/

public String getConfirmPassword() {

return confirmPassword;

}

/\*\*

\* @param Confirm

\* PAssword To set User Confirm Password

\*/

public void setConfirmPassword(String confirmPassword) {

this.confirmPassword = confirmPassword;

}

/\*\*

\* @return Date Of Birth Of User

\*/

public Date getDob() {

return dob;

}

/\*\*

\* @param Date

\* Of Birth To set User Date Of Birth

\*/

public void setDob(Date dob) {

this.dob = dob;

}

/\*\*

\* @return Mobile No Of User

\*/

public String getMobileNo() {

return mobileNo;

}

/\*\*

\* @param Mobile

\* No To set User Mobile No

\*/

public void setMobileNo(String mobileNo) {

this.mobileNo = mobileNo;

}

/\*\*

\* @return ROle Id Of User

\*/

public long getRoleId() {

return roleId;

}

/\*\*

\* @param Role

\* Id To set User ROle Id

\*/

public void setRoleId(long roleId) {

this.roleId = roleId;

}

/\*\*

\* @return unSuccessfulLogin Of User

/\*\*

\* @return unSuccessfulLogin Of User

\*/

public String getGender() {

return gender;

}

/\*\*

\* @param Gender

\* To set User Gender

\*/

public void setGender(String gender) {

this.gender = gender;

}

public String getKey() {

return id + "";

}

public String getValue() {

return firstName + " " + lastName;

}

}

package in.co.job.portal.controller;

import java.io.IOException;

import java.nio.file.Paths;

import javax.servlet.ServletException;

import javax.servlet.annotation.MultipartConfig;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import javax.servlet.http.Part;

import org.apache.log4j.Logger;

import in.co.job.portal.bean.BaseBean;

import in.co.job.portal.bean.JobBean;

import in.co.job.portal.bean.UserBean;

import in.co.job.portal.exception.ApplicationException;

import in.co.job.portal.exception.DuplicateRecordException;

import in.co.job.portal.model.JobModel;

import in.co.job.portal.model.UserModel;

import in.co.job.portal.util.DataUtility;

import in.co.job.portal.util.DataValidator;

import in.co.job.portal.util.PropertyReader;

import in.co.job.portal.util.ServletUtility;

/\*\*

\*

\* Servlet implementation class ApplayJobCtl

\*/

@WebServlet(name="ApplayJobCtl",urlPatterns={"/ctl/ApplayJobCtl"})

@MultipartConfig(maxFileSize = 16177215)

public class ApplayJobCtl extends BaseCtl {

private static final long serialVersionUID = 1L;

private static Logger log = Logger.getLogger(ApplayJobCtl.class);

/\*\*

\* Loads list and other data required to display at HTML form

\*

\* @param request

\*/

/\*\*

\* Validate input Data Entered By User

\*

\* @param request

\* @return

\*/

@Override

protected boolean validate(HttpServletRequest request) {

log.debug("JobCtl Method validate Started");

boolean pass = true;

Part part = null;

try {

part = request.getPart("file");

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (ServletException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

String fileName = Paths.get(part.getSubmittedFileName()).getFileName().toString();

if (DataValidator.isNull(fileName)) {

request.setAttribute("file", PropertyReader.getValue("error.require", "Resume"));

pass = false;

}

log.debug("JobCtl Method validate Ended");

return pass;

}

@Override

protected BaseBean populateBean(HttpServletRequest request) {

log.debug("JobCtl Method populatebean Started");

JobBean bean = new JobBean();

bean.setId(DataUtility.getLong(request.getParameter("id")));

bean.setCompanyName(DataUtility.getString(request .getParameter("cName")));

bean.setLanguage(DataUtility.getString(request.getParameter("language")));

bean.setAddress(DataUtility.getString(request.getParameter("address")));

bean.setDescription(DataUtility.getString(request.getParameter("description")));

bean.setPostDate(DataUtility.getDate(request.getParameter("pDate")));

bean.setPostByRecId(DataUtility.getString(request.getParameter("postById")));

HttpSession session=request.getSession();

UserBean uBean=(UserBean)session.getAttribute("user");

bean.setUserId(uBean.getId());

populateDTO(bean, request);

log.debug("JobCtl Method populatebean Ended");

return bean;

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

log.debug("JobCtl Method doGet Started");

String op = DataUtility.getString(request.getParameter("operation"));

// get model

JobModel model = new JobModel();

long id = DataUtility.getLong(request.getParameter("id"));

if (id > 0 || op != null) {

JobBean bean;

try {

bean = model.findByPK(id);

ServletUtility.setBean(bean, request);

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

}

}

ServletUtility.forward(getView(), request, response);

log.debug("JobCtl Method doGet Ended");

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

log.debug("JobCtl Method doPost Started");

String op = DataUtility.getString(request.getParameter("operation"));

// get model

JobModel model = new JobModel();

long id = DataUtility.getLong(request.getParameter("id"));

if (OP\_PROCESS.equalsIgnoreCase(op)) {

JobBean bean = (JobBean) populateBean(request);

try {

UserModel uModel=new UserModel();

UserBean ubean= uModel.findByLogin(bean.getPostByRecId());

bean.setRecruiterId(ubean.getId());

bean.setJobId(bean.getId());

bean.setResumeFile(ServletUtility.subFile(request, response,ubean.getFirstName()));

long pk = model.jobadd(bean);

ServletUtility.setSuccessMessage("Job is successfully Applay",request);

ServletUtility.setBean(bean, request);

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

} catch (DuplicateRecordException e) {

ServletUtility.setBean(bean, request);

ServletUtility.setErrorMessage("Job Is Already Applyed", request);

}

} else if (OP\_DELETE.equalsIgnoreCase(op)) {

JobBean bean = (JobBean) populateBean(request);

try {

model.delete(bean);

ServletUtility.redirect(JPView.JOB\_LIST\_CTL, request,

response);

return;

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

}

} else if (OP\_CANCEL.equalsIgnoreCase(op)) {

ServletUtility.redirect(JPView.JOB\_LIST\_CTL, request, response);

}else if (OP\_RESET.equalsIgnoreCase(op)) {

ServletUtility.redirect(JPView.JOB\_CTL, request, response);

return;

}

ServletUtility.forward(getView(), request, response);

log.debug("JobCtl Method doPostEnded");

}

@Override

protected String getView() {

// TODO Auto-generated method stub

return JPView.Apply\_VIEW;

}

}

package in.co.job.portal.controller;

import java.io.IOException;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.apache.log4j.Logger;

import in.co.job.portal.bean.BaseBean;

import in.co.job.portal.bean.JobBean;

import in.co.job.portal.bean.UserBean;

import in.co.job.portal.exception.ApplicationException;

import in.co.job.portal.model.JobModel;

import in.co.job.portal.model.UserModel;

import in.co.job.portal.util.DataUtility;

import in.co.job.portal.util.PropertyReader;

import in.co.job.portal.util.ServletUtility;

/\*\*

\* Servlet implementation class ApplayJobListCtl

\*/

@WebServlet(name = "ApplayJobListCtl", urlPatterns = { "/ctl/ApplayJobListCtl" })

public class ApplayJobListCtl extends BaseCtl {

private static Logger log = Logger.getLogger(ApplayJobListCtl.class);

/\*\*

\* Populates bean object from request parameters

\*

\* @param request

\* @return

\*/

@Override

protected BaseBean populateBean(HttpServletRequest request) {

log.debug("JobListCtl populateBean method start");

JobBean bean = new JobBean();

bean.setUserName(DataUtility.getString(request.getParameter("name")));

log.debug("JobListCtl populateBean method end");

return bean;

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

log.debug("JobListCtl doGet Start");

List list = null;

int pageNo = 1;

int pageSize = DataUtility.getInt(PropertyReader.getValue("page.size"));

JobBean bean = (JobBean) populateBean(request);

String op = DataUtility.getString(request.getParameter("operation"));

// get the selected checkbox ids array for delete list

String ids = request.getParameter("ids");

JobModel model = new JobModel();

try {

HttpSession session=request.getSession();

UserBean uBean=(UserBean)session.getAttribute("user");

if(uBean.getRoleId()==2) {

bean.setRecruiterId(uBean.getId());

}else if(uBean.getRoleId()==3) {

bean.setUserId(uBean.getId());

}

list = model.jobSearch(bean, pageNo, pageSize);

if (list == null || list.size() == 0) {

ServletUtility.setErrorMessage("No record found ", request);

}

ServletUtility.setList(list, request);

ServletUtility.setPageNo(pageNo, request);

ServletUtility.setPageSize(pageSize, request);

ServletUtility.forward(getView(), request, response);

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

}

log.debug("JobListCtl doPOst End");

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

log.debug("JobListCtl doPost Start");

List list = null;

int pageNo = DataUtility.getInt(request.getParameter("pageNo"));

int pageSize = DataUtility.getInt(request.getParameter("pageSize"));

pageNo = (pageNo == 0) ? 1 : pageNo;

pageSize = (pageSize == 0) ? DataUtility.getInt(PropertyReader.getValue("page.size")) : pageSize;

JobBean bean = (JobBean) populateBean(request);

String op = DataUtility.getString(request.getParameter("operation"));

// get the selected checkbox ids array for delete list

String[] ids = request.getParameterValues("ids");

JobModel model = new JobModel();

try {

if (OP\_SEARCH.equalsIgnoreCase(op) || "Next".equalsIgnoreCase(op) || "Previous".equalsIgnoreCase(op)) {

if (OP\_SEARCH.equalsIgnoreCase(op)) {

pageNo = 1;

} else if (OP\_NEXT.equalsIgnoreCase(op)) {

pageNo++;

} else if (OP\_PREVIOUS.equalsIgnoreCase(op) && pageNo > 1) {

pageNo--;

}

} else if (OP\_NEW.equalsIgnoreCase(op)) {

ServletUtility.redirect(JPView.JOB\_CTL, request, response);

return;

} else if (OP\_DELETE.equalsIgnoreCase(op)) {

pageNo = 1;

if (ids != null && ids.length > 0) {

JobBean deletebean = new JobBean();

for (String id : ids) {

deletebean.setId(DataUtility.getInt(id));

model.delete(deletebean);

}

ServletUtility.setSuccessMessage("Data Deleted Successfully", request);

} else {

ServletUtility.setErrorMessage("Select at least one record", request);

}

} else if (OP\_RESET.equalsIgnoreCase(op)) {

ServletUtility.redirect(JPView.APPLY\_LIST\_CTL, request, response);

return;

}

HttpSession session=request.getSession();

UserBean uBean=(UserBean)session.getAttribute("user");

bean.setRecruiterId(uBean.getId());

list = model.jobSearch(bean, pageNo, pageSize);

if (list == null || list.size() == 0) {

ServletUtility.setErrorMessage("No record found ", request);

}

ServletUtility.setList(list, request);

ServletUtility.setPageNo(pageNo, request);

ServletUtility.setPageSize(pageSize, request);

ServletUtility.forward(getView(), request, response);

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

}

log.debug("JobListCtl doGet End");

}

@Override

protected String getView() {

// TODO Auto-generated method stub

return JPView.Apply\_LIST\_VIEW;

}

}

package in.co.job.portal.controller;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.apache.log4j.Logger;

import in.co.job.portal.bean.BaseBean;

import in.co.job.portal.bean.JobBean;

import in.co.job.portal.bean.UserBean;

import in.co.job.portal.exception.ApplicationException;

import in.co.job.portal.exception.DuplicateRecordException;

import in.co.job.portal.model.JobModel;

import in.co.job.portal.model.UserModel;

import in.co.job.portal.util.DataUtility;

import in.co.job.portal.util.DataValidator;

import in.co.job.portal.util.PropertyReader;

import in.co.job.portal.util.ServletUtility;

/\*\*

\* Servlet implementation class JobCtl

\*/

@WebServlet(name="JobCtl",urlPatterns={"/ctl/JobCtl"})

public class JobCtl extends BaseCtl {

private static final long serialVersionUID = 1L;

private static Logger log = Logger.getLogger(JobCtl.class);

/\*\*

\* Loads list and other data required to display at HTML form

\*

\* @param request

\*/

/\*\*

\* Validate input Data Entered By User

\*

\* @param request

\* @return

\*/

@Override

protected boolean validate(HttpServletRequest request) {

log.debug("JobCtl Method validate Started");

boolean pass = true;

if (DataValidator.isNull(request.getParameter("cName"))) {

request.setAttribute("cName",

PropertyReader.getValue("error.require", "Company Name"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("language"))) {

request.setAttribute("language",

PropertyReader.getValue("error.require", "Language"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("pDate"))) {

request.setAttribute("pDate",

PropertyReader.getValue("error.require", "Date"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("address"))) {

request.setAttribute("address",

PropertyReader.getValue("error.require", "Address"));

pass = false;

}

if (DataValidator.isNull(request.getParameter("description"))) {

request.setAttribute("description",

PropertyReader.getValue("error.require", "Description"));

pass = false;

}

log.debug("JobCtl Method validate Ended");

return pass;

}

@Override

protected BaseBean populateBean(HttpServletRequest request) {

log.debug("JobCtl Method populatebean Started");

JobBean bean = new JobBean();

bean.setId(DataUtility.getLong(request.getParameter("id")));

bean.setCompanyName(DataUtility.getString(request .getParameter("cName")));

bean.setLanguage(DataUtility.getString(request.getParameter("language")));

bean.setAddress(DataUtility.getString(request.getParameter("address")));

bean.setDescription(DataUtility.getString(request.getParameter("description")));

bean.setPostDate(DataUtility.getDate1(request.getParameter("pDate")));

HttpSession session=request.getSession();

UserBean uBean=(UserBean)session.getAttribute("user");

bean.setPostByRecId(uBean.getLogin());

populateDTO(bean, request);

log.debug("JobCtl Method populatebean Ended");

return bean;

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

log.debug("JobCtl Method doGet Started");

String op = DataUtility.getString(request.getParameter("operation"));

// get model

JobModel model = new JobModel();

long id = DataUtility.getLong(request.getParameter("id"));

if (id > 0 || op != null) {

JobBean bean;

try {

bean = model.findByPK(id);

ServletUtility.setBean(bean, request);

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

}

}

ServletUtility.forward(getView(), request, response);

log.debug("JobCtl Method doGet Ended");

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

log.debug("JobCtl Method doPost Started");

String op = DataUtility.getString(request.getParameter("operation"));

// get model

JobModel model = new JobModel();

long id = DataUtility.getLong(request.getParameter("id"));

if (OP\_SAVE.equalsIgnoreCase(op)) {

JobBean bean = (JobBean) populateBean(request);

try {

if (id > 0){

model.update(bean);

ServletUtility.setSuccessMessage("Data is successfully Updated", request);

} else {

long pk = model.add(bean);

// bean.setId(pk);

ServletUtility.setSuccessMessage("Data is successfully saved",request);

}

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

} catch (DuplicateRecordException e) {

ServletUtility.setBean(bean, request);

ServletUtility.setErrorMessage("Job id already exists", request);

}

ServletUtility.forward(getView(), request, response);

} else if (OP\_DELETE.equalsIgnoreCase(op)) {

JobBean bean = (JobBean) populateBean(request);

try {

model.delete(bean);

ServletUtility.redirect(JPView.JOB\_LIST\_CTL, request,

response);

return;

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

}

} else if (OP\_CANCEL.equalsIgnoreCase(op)) {

ServletUtility.redirect(JPView.JOB\_LIST\_CTL, request, response);

}else if (OP\_RESET.equalsIgnoreCase(op)) {

ServletUtility.redirect(JPView.JOB\_CTL, request, response);

return;

}

ServletUtility.forward(getView(), request, response);

log.debug("JobCtl Method doPostEnded");

}

@Override

protected String getView() {

// TODO Auto-generated method stub

return JPView.JOB\_VIEW;

}

}

package in.co.job.portal.controller;

import java.io.IOException;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.apache.log4j.Logger;

import in.co.job.portal.bean.BaseBean;

import in.co.job.portal.bean.JobBean;

import in.co.job.portal.bean.UserBean;

import in.co.job.portal.exception.ApplicationException;

import in.co.job.portal.model.JobModel;

import in.co.job.portal.model.UserModel;

import in.co.job.portal.util.DataUtility;

import in.co.job.portal.util.PropertyReader;

import in.co.job.portal.util.ServletUtility;

/\*\*

\* Servlet implementation class JobListCtl

\*/

@WebServlet(name = "JobListCtl", urlPatterns = { "/ctl/JobListCtl" })

public class JobListCtl extends BaseCtl {

private static final long serialVersionUID = 1L;

private static Logger log = Logger.getLogger(JobListCtl.class);

/\*\*

\* Populates bean object from request parameters

\*

\* @param request

\* @return

\*/

@Override

protected BaseBean populateBean(HttpServletRequest request) {

log.debug("JobListCtl populateBean method start");

JobBean bean = new JobBean();

bean.setCompanyName(DataUtility.getString(request.getParameter("cName")));

bean.setLanguage(DataUtility.getString(request.getParameter("language")));

log.debug("JobListCtl populateBean method end");

return bean;

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

log.debug("JobListCtl doGet Start");

List list = null;

int pageNo = 1;

int pageSize = DataUtility.getInt(PropertyReader.getValue("page.size"));

JobBean bean = (JobBean) populateBean(request);

String op = DataUtility.getString(request.getParameter("operation"));

// get the selected checkbox ids array for delete list

String ids = request.getParameter("ids");

JobModel model = new JobModel();

try {

if (ids != null ) {

JobBean deletebean = new JobBean();

deletebean.setId(DataUtility.getInt(ids));

model.delete(deletebean);

ServletUtility.setSuccessMessage("Data Deleted Successfully", request);

}

HttpSession session=request.getSession();

UserBean uBean=(UserBean)session.getAttribute("user");

if(uBean.getRoleId()==2) {

bean.setPostByRecId(uBean.getLogin());

}

list = model.search(bean, pageNo, pageSize);

if (list == null || list.size() == 0) {

ServletUtility.setErrorMessage("No record found ", request);

}

ServletUtility.setList(list, request);

ServletUtility.setPageNo(pageNo, request);

ServletUtility.setPageSize(pageSize, request);

ServletUtility.forward(getView(), request, response);

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

}

log.debug("JobListCtl doPOst End");

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

log.debug("JobListCtl doPost Start");

List list = null;

int pageNo = DataUtility.getInt(request.getParameter("pageNo"));

int pageSize = DataUtility.getInt(request.getParameter("pageSize"));

pageNo = (pageNo == 0) ? 1 : pageNo;

pageSize = (pageSize == 0) ? DataUtility.getInt(PropertyReader.getValue("page.size")) : pageSize;

JobBean bean = (JobBean) populateBean(request);

String op = DataUtility.getString(request.getParameter("operation"));

// get the selected checkbox ids array for delete list

String[] ids = request.getParameterValues("ids");

JobModel model = new JobModel();

try {

if (OP\_SEARCH.equalsIgnoreCase(op) || "Next".equalsIgnoreCase(op) || "Previous".equalsIgnoreCase(op)) {

if (OP\_SEARCH.equalsIgnoreCase(op)) {

pageNo = 1;

} else if (OP\_NEXT.equalsIgnoreCase(op)) {

pageNo++;

} else if (OP\_PREVIOUS.equalsIgnoreCase(op) && pageNo > 1) {

pageNo--;

}

} else if (OP\_NEW.equalsIgnoreCase(op)) {

ServletUtility.redirect(JPView.JOB\_CTL, request, response);

return;

} else if (OP\_DELETE.equalsIgnoreCase(op)) {

pageNo = 1;

if (ids != null && ids.length > 0) {

JobBean deletebean = new JobBean();

for (String id : ids) {

deletebean.setId(DataUtility.getInt(id));

model.delete(deletebean);

}

ServletUtility.setSuccessMessage("Data Deleted Successfully", request);

} else {

ServletUtility.setErrorMessage("Select at least one record", request);

}

} else if (OP\_RESET.equalsIgnoreCase(op)) {

ServletUtility.redirect(JPView.JOB\_LIST\_CTL, request, response);

return;

}

HttpSession session=request.getSession();

UserBean uBean=(UserBean)session.getAttribute("user");

if(uBean.getRoleId()==2) {

bean.setPostByRecId(uBean.getLogin());

}

list = model.search(bean, pageNo, pageSize);

if (list == null || list.size() == 0) {

ServletUtility.setErrorMessage("No record found ", request);

}

ServletUtility.setList(list, request);

ServletUtility.setPageNo(pageNo, request);

ServletUtility.setPageSize(pageSize, request);

ServletUtility.forward(getView(), request, response);

} catch (ApplicationException e) {

log.error(e);

ServletUtility.handleException(e, request, response);

return;

}

log.debug("JobListCtl doGet End");

}

@Override

protected String getView() {

// TODO Auto-generated method stub

return JPView.JOB\_LIST\_VIEW;

}

}

**package** in.co.job.portal.exception;

/\*\*

\* ApplicationException is propogated from Service classes when an business

\* logic exception occurered.

\*

\* **@author** Navigable Set

\* **@version** 1.0

\* **@Copyright** (c) Navigable Set

\*

\*/

**public** **class** ApplicationException **extends** Exception

{

/\*\*

\* **@param** msg

\* : Error message

\*/

**public** ApplicationException(String msg) {

**super**(msg);

}

}

**package** in.co.job.portal.exception;

/\*\*

\* DatabaseException is propogated by DAO classes when an unhandled Database

\* exception occurred

\*

\* **@author** Navigable Set

\* **@version** 1.0

\* **@Copyright** (c) Navigable Set

\*

\*/

**public** **class** DatabaseException **extends** Exception

{

/\*\*

\* **@param** msg

\* : Error message

\*/

**public** DatabaseException(String msg) {

**super**(msg);

}

}

**package** in.co.job.portal.exception;

/\*\*

\* DuplicateRecordException thrown when a duplicate record occurred

\*

\* **@author** Navigable Set

\* **@version** 1.0

\* **@Copyright** (c) Navigable Set

\*

\*/

**public** **class** DuplicateRecordException **extends** Exception

{

/\*\*

\* **@param** msg

\* error message

\*/

**public** DuplicateRecordException(String msg) {

**super**(msg);

}

}

**package** in.co.job.portal.exception;

/\*\*

\* RecordNotFoundException thrown when a record not found occurred

\*

\* **@author** Navigable Set

\* **@version** 1.0

\* **@Copyright** (c) Navigable Set

\*

\*/

**public** **class** RecordNotFoundException **extends** Exception

{

/\*\*

\* **@param** msg

\* error message

\*/

**public** RecordNotFoundException(String msg) {

**super**(msg);

}

}

package in.co.job.portal.model;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import org.apache.log4j.Logger;

import in.co.job.portal.bean.JobBean;

import in.co.job.portal.bean.RoleBean;

import in.co.job.portal.bean.UserBean;

import in.co.job.portal.exception.ApplicationException;

import in.co.job.portal.exception.DatabaseException;

import in.co.job.portal.exception.DuplicateRecordException;

import in.co.job.portal.exception.RecordNotFoundException;

import in.co.job.portal.util.DataUtility;

import in.co.job.portal.util.JDBCDataSource;

public class JobModel {

private static Logger log = Logger.getLogger(JobModel.class);

/\*\*

\* Find next PK of Role

\*

\* @throws DatabaseException

\*/

public Integer nextPK() throws DatabaseException {

log.debug("Model nextPK Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM P\_Job");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new DatabaseException("Exception : Exception in getting PK");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

return pk + 1;

}

/\*\*

\* Add a Role

\*

\* @param bean

\* @throws DatabaseException

\* @throws ApplicationException

\*

\*

\*/

public long add(JobBean bean) throws ApplicationException, DuplicateRecordException {

log.debug("Model add Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

pk = nextPK();

// Get auto-generated next primary key

System.out.println(pk + " in ModelJDBC");

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("INSERT INTO P\_Job VALUES(?,?,?,?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

pstmt.setString(2, bean.getCompanyName());

pstmt.setString(3, bean.getLanguage());

pstmt.setString(4, bean.getDescription());

pstmt.setDate(5, new java.sql.Date(bean.getPostDate().getTime()));

pstmt.setString(6, bean.getPostByRecId());

pstmt.setString(7, bean.getAddress());

pstmt.setString(8, bean.getCreatedBy());

pstmt.setString(9, bean.getModifiedBy());

pstmt.setTimestamp(10, bean.getCreatedDatetime());

pstmt.setTimestamp(11, bean.getModifiedDatetime());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException("Exception : add rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in add Role");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model add End");

return pk;

}

/\*\*

\* Delete a Role

\*

\* @param bean

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public void delete(JobBean bean) throws ApplicationException {

log.debug("Model delete Started");

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("DELETE FROM P\_job WHERE ID=?");

pstmt.setLong(1, bean.getId());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

// log.error("Database Exception..", e);

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException("Exception : Delete rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in delete Role");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model delete Started");

}

/\*\*

\* Find User by Role

\*

\* @param name

\* : get parameter

\* @return bean

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public JobBean findByName(String name) throws ApplicationException {

log.debug("Model findBy EmailId Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM P\_Job WHERE companyName=?");

JobBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setString(1, name);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new JobBean();

bean.setId(rs.getLong(1));

bean.setCompanyName(rs.getString(2));

bean.setLanguage(rs.getString(3));

bean.setDescription(rs.getString(4));

bean.setPostDate(rs.getDate(5));

bean.setPostByRecId(rs.getString(6));

bean.setAddress(rs.getString(7));

bean.setCreatedBy(rs.getString(8));

bean.setModifiedBy(rs.getString(9));

bean.setCreatedDatetime(rs.getTimestamp(10));

bean.setModifiedDatetime(rs.getTimestamp(11));

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by emailId");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findBy EmailId End");

return bean;

}

/\*\*

\* Find Role by PK

\*

\* @param pk

\* : get parameter

\* @return bean

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public JobBean findByPK(long pk) throws ApplicationException {

log.debug("Model findByPK Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM P\_job WHERE ID=?");

JobBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, pk);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new JobBean();

bean.setId(rs.getLong(1));

bean.setCompanyName(rs.getString(2));

bean.setLanguage(rs.getString(3));

bean.setDescription(rs.getString(4));

bean.setPostDate(rs.getDate(5));

bean.setPostByRecId(rs.getString(6));

bean.setAddress(rs.getString(7));

bean.setCreatedBy(rs.getString(8));

bean.setModifiedBy(rs.getString(9));

bean.setCreatedDatetime(rs.getTimestamp(10));

bean.setModifiedDatetime(rs.getTimestamp(11));

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

/\*\*

\* Update a Role

\*

\* @param bean

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public void update(JobBean bean) throws ApplicationException, DuplicateRecordException {

log.debug("Model update Started");

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement(

"UPDATE P\_Job SET companyNAME=?,language=?,DESCRIPTION=?,postDate=?,postbyrecId=?,address=?,CREATEDBY=?,MODIFIEDBY=?,CREATEDDATETIME=?,MODIFIEDDATETIME=? WHERE ID=?");

pstmt.setString(1, bean.getCompanyName());

pstmt.setString(2, bean.getLanguage());

pstmt.setString(3, bean.getDescription());

pstmt.setDate(4, new java.sql.Date(bean.getPostDate().getTime()));

pstmt.setString(5, bean.getPostByRecId());

pstmt.setString(6, bean.getAddress());

pstmt.setString(7, bean.getCreatedBy());

pstmt.setString(8, bean.getModifiedBy());

pstmt.setTimestamp(9, bean.getCreatedDatetime());

pstmt.setTimestamp(10, bean.getModifiedDatetime());

pstmt.setLong(11, bean.getId());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

log.error("Database Exception..", e);

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException("Exception : Delete rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception in updating Role ");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model update End");

}

/\*\*

\* Search Role

\*

\* @param bean

\* : Search Parameters

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public List search(JobBean bean) throws ApplicationException {

return search(bean, 0, 0);

}

/\*\*

\* Search Role with pagination

\*

\* @return list : List of Roles

\* @param bean

\* : Search Parameters

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\*

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public List search(JobBean bean, int pageNo, int pageSize) throws ApplicationException {

log.debug("Model search Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM P\_Job WHERE 1=1");

if (bean != null) {

if (bean.getId() > 0) {

sql.append(" AND id = " + bean.getId());

}

if (bean.getCompanyName() != null && bean.getCompanyName().length() > 0) {

sql.append(" AND companyNAME LIKE '" + bean.getCompanyName() + "%'");

}

if (bean.getLanguage() != null && bean.getLanguage().length() > 0) {

sql.append(" AND language LIKE '" + bean.getLanguage() + "%'");

}

if (bean.getPostByRecId() != null && bean.getPostByRecId().length() > 0) {

sql.append(" AND postbyrecId LIKE '" + bean.getPostByRecId() + "%'");

}

if (bean.getDescription() != null && bean.getDescription().length() > 0) {

sql.append(" AND DESCRIPTION LIKE '" + bean.getDescription() + "%'");

}

}

// if page size is greater than zero then apply pagination

if (pageSize > 0) {

// Calculate start record index

pageNo = (pageNo - 1) \* pageSize;

sql.append(" Limit " + pageNo + ", " + pageSize);

// sql.append(" limit " + pageNo + "," + pageSize);

}

ArrayList list = new ArrayList();

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new JobBean();

bean.setId(rs.getLong(1));

bean.setCompanyName(rs.getString(2));

bean.setLanguage(rs.getString(3));

bean.setDescription(rs.getString(4));

bean.setPostDate(rs.getDate(5));

bean.setPostByRecId(rs.getString(6));

bean.setAddress(rs.getString(7));

bean.setCreatedBy(rs.getString(8));

bean.setModifiedBy(rs.getString(9));

bean.setCreatedDatetime(rs.getTimestamp(10));

bean.setModifiedDatetime(rs.getTimestamp(11));

list.add(bean);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in search Role");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model search End");

return list;

}

/\*\*

\* Get List of Role

\*

\* @return list : List of Role

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public List list() throws ApplicationException {

return list(0, 0);

}

/\*\*

\* Get List of Role with pagination

\*

\* @return list : List of Role

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public List list(int pageNo, int pageSize) throws ApplicationException {

log.debug("Model list Started");

ArrayList list = new ArrayList();

StringBuffer sql = new StringBuffer("select \* from P\_Job");

// if page size is greater than zero then apply pagination

if (pageSize > 0) {

// Calculate start record index

pageNo = (pageNo - 1) \* pageSize;

sql.append(" limit " + pageNo + "," + pageSize);

}

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

JobBean bean = new JobBean();

bean.setId(rs.getLong(1));

bean.setCompanyName(rs.getString(2));

bean.setLanguage(rs.getString(3));

bean.setDescription(rs.getString(4));

bean.setPostDate(rs.getDate(5));

bean.setPostByRecId(rs.getString(6));

bean.setAddress(rs.getString(7));

bean.setCreatedBy(rs.getString(8));

bean.setModifiedBy(rs.getString(9));

bean.setCreatedDatetime(rs.getTimestamp(10));

bean.setModifiedDatetime(rs.getTimestamp(11));

list.add(bean);

}

rs.close();

} catch (Exception e) {

// log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting list of Role");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model list End");

return list;

}

public Integer jnextPK() throws DatabaseException {

log.debug("Model nextPK Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(ID) FROM P\_Applay");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new DatabaseException("Exception : Exception in getting PK");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

return pk + 1;

}

public long jobadd(JobBean bean) throws ApplicationException, DuplicateRecordException {

log.debug("Model add Started");

Connection conn = null;

int pk = 0;

UserModel umodel=new UserModel();

UserBean uBean=umodel.findByPK(bean.getUserId());

bean.setUserName(uBean.getFirstName()+" "+uBean.getLastName());

JobBean existBean=jobfindByJobId(bean.getJobId(),bean.getUserId());

if (existBean != null) {

throw new DuplicateRecordException("Job Id already Applyed");

}

try {

conn = JDBCDataSource.getConnection();

pk = jnextPK();

// Get auto-generated next primary key

System.out.println(pk + " in ModelJDBC");

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement("INSERT INTO P\_applay VALUES(?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

pstmt.setLong(2, bean.getRecruiterId());

pstmt.setLong(3, bean.getJobId());

pstmt.setLong(4, bean.getUserId());

pstmt.setTimestamp(5, DataUtility.getCurrentTimestamp());

pstmt.setString(6,bean.getUserName());

pstmt.setString(7,bean.getResumeFile());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException("Exception : add rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in add Role");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model add End");

return pk;

}

public List jobSearch(JobBean bean) throws ApplicationException {

return search(bean, 0, 0);

}

/\*\*

\* Search Role with pagination

\*

\* @return list : List of Roles

\* @param bean

\* : Search Parameters

\* @param pageNo

\* : Current Page No.

\* @param pageSize

\* : Size of Page

\*

\* @throws DatabaseException

\* @throws ApplicationException

\*/

public List jobSearch(JobBean bean, int pageNo, int pageSize) throws ApplicationException {

log.debug("Model search Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM P\_applay WHERE 1=1");

if (bean != null) {

if (bean.getId() > 0) {

sql.append(" AND id = " + bean.getId());

}

if (bean.getUserId() > 0) {

sql.append(" AND userid = " + bean.getUserId());

}

if (bean.getUserName() != null && bean.getUserName().length() > 0) {

sql.append(" AND userName LIKE '" + bean.getUserName() + "%'");

}

if (bean.getRecruiterId() > 0) {

sql.append(" AND RecruiterId = " + bean.getRecruiterId());

}

}

sql.append(" Order by ID Desc");

// if page size is greater than zero then apply pagination

if (pageSize > 0) {

// Calculate start record index

pageNo = (pageNo - 1) \* pageSize;

sql.append(" Limit " + pageNo + ", " + pageSize);

// sql.append(" limit " + pageNo + "," + pageSize);

}

ArrayList list = new ArrayList();

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new JobBean();

bean.setId(rs.getLong(1));

bean.setRecruiterId(rs.getLong(2));

bean.setJobId(rs.getLong(3));

bean.setUserId(rs.getLong(4));

bean.setApplayDate(rs.getTimestamp(5));

bean.setUserName(rs.getString(6));

bean.setResumeFile(rs.getString(7));

list.add(bean);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in search Role");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model search End");

return list;

}

public JobBean jobfindByPK(long pk) throws ApplicationException {

log.debug("Model findByPK Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM P\_applay WHERE ID=?");

JobBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, pk);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new JobBean();

bean.setId(rs.getLong(1));

bean.setRecruiterId(rs.getLong(2));

bean.setJobId(rs.getLong(3));

bean.setUserId(rs.getLong(4));

bean.setApplayDate(rs.getTimestamp(5));

bean.setUserName(rs.getString(6));

bean.setResumeFile(rs.getString(7));

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

public JobBean jobfindByJobId(long jID,long uId) throws ApplicationException {

log.debug("Model findByPK Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM P\_applay WHERE jobId=? and userID=?");

JobBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, jID);

pstmt.setLong(2,uId);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new JobBean();

bean.setId(rs.getLong(1));

bean.setRecruiterId(rs.getLong(2));

bean.setJobId(rs.getLong(3));

bean.setUserId(rs.getLong(4));

bean.setApplayDate(rs.getTimestamp(5));

bean.setUserName(rs.getString(6));

bean.setResumeFile(rs.getString(7));

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException("Exception : Exception in getting User by pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

public void updateApplayJob(JobBean bean) throws ApplicationException, DuplicateRecordException {

log.debug("Model update Started");

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn.prepareStatement(

"UPDATE P\_applay SET RecruiterId=?,JobId=?,UserId=?,applayDate=?,UserName=?,ResumeFile=? WHERE ID=?");

pstmt.setLong(1, bean.getRecruiterId());

pstmt.setLong(2, bean.getJobId());

pstmt.setLong(3, bean.getUserId());

pstmt.setTimestamp(4, DataUtility.getCurrentTimestamp());

pstmt.setString(5,bean.getUserName());

pstmt.setString(6,bean.getResumeFile());

pstmt.setLong(7, bean.getId());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

log.error("Database Exception..", e);

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException("Exception : Delete rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception in updating Role ");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model update End");

}

}

package in.co.job.portal.model;

import java.sql.Connection;

import java.sql.Date;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import org.apache.log4j.Logger;

import in.co.job.portal.bean.MakeResumeBean;

import in.co.job.portal.bean.RoleBean;

import in.co.job.portal.exception.ApplicationException;

import in.co.job.portal.exception.DatabaseException;

import in.co.job.portal.exception.DuplicateRecordException;

import in.co.job.portal.util.JDBCDataSource;

/\*\*

\* JDBC Implementation of Role Model

\*

\* @author Navigable Set

\* @version 1.0

\* @Copyright (c) Navigable Set

\*/

public class MakeResumeModel

{

private static Logger log = Logger.getLogger(MakeResumeModel.class);

/\*\*

\* Find next PK of Role

\*

\* @throws DatabaseException

\*/

public Integer nextPK() throws DatabaseException {

log.debug("Model nextPK Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn

.prepareStatement("SELECT MAX(ID) FROM P\_RESUME");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

pk = rs.getInt(1);

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new DatabaseException("Exception : Exception in getting PK");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model nextPK End");

return pk + 1;

}

/\*\*

\* Add a Role

\*

\* @param bean

\* @throws DatabaseException

\* @throws ApplicationException

\*

\*

\*/

public long add(MakeResumeBean bean) throws ApplicationException,

DuplicateRecordException {

log.debug("Model add Started");

Connection conn = null;

int pk = 0;

try {

conn = JDBCDataSource.getConnection();

pk = nextPK();

// Get auto-generated next primary key

System.out.println(pk + " in ModelJDBC");

conn.setAutoCommit(false); // Begin transaction

PreparedStatement pstmt = conn

.prepareStatement("INSERT INTO P\_RESUME VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");

pstmt.setInt(1, pk);

pstmt.setString(2, bean.getName());

pstmt.setString(3, bean.getEmail());

pstmt.setString(4, bean.getMobile());

pstmt.setString(5, bean.getgCourceName());

pstmt.setString(6, bean.getgInsName());

pstmt.setString(7, bean.getgPercentage());

pstmt.setString(8, bean.gethCourceName());

pstmt.setString(9, bean.gethInsName());

pstmt.setString(10, bean.gethPercentage());

pstmt.setString(11, bean.getsCourceName());

pstmt.setString(12, bean.getsInsName());

pstmt.setString(13, bean.getsPercentage());

pstmt.setString(14, bean.getSkill());

pstmt.setString(15, bean.getHobbies());

pstmt.setString(16, bean.getpDetail());

pstmt.setString(17, bean.getDeclaration());

pstmt.setDate(18, new java.sql.Date(bean.getDate().getTime()));

pstmt.setString(19, bean.getPlace());

pstmt.setString(20,bean.getObjective());

pstmt.executeUpdate();

conn.commit(); // End transaction

pstmt.close();

} catch (Exception e) {

e.printStackTrace();

log.error("Database Exception..", e);

try {

conn.rollback();

} catch (Exception ex) {

throw new ApplicationException(

"Exception : add rollback exception " + ex.getMessage());

}

throw new ApplicationException("Exception : Exception in add MakeResume");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model add End");

return pk;

}

public MakeResumeBean findByPK(long pk) throws ApplicationException {

log.debug("Model findByPK Started");

StringBuffer sql = new StringBuffer("SELECT \* FROM P\_Resume WHERE ID=?");

MakeResumeBean bean = null;

Connection conn = null;

try {

conn = JDBCDataSource.getConnection();

PreparedStatement pstmt = conn.prepareStatement(sql.toString());

pstmt.setLong(1, pk);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

bean = new MakeResumeBean();

bean.setId(rs.getLong(1));

bean.setName(rs.getString(2));

bean.setEmail(rs.getString(3));

bean.setMobile(rs.getString(4));

bean.setgCourceName(rs.getString(5));

bean.setgInsName(rs.getString(6));

bean.setgPercentage(rs.getString(7));

bean.sethCourceName(rs.getString(8));

bean.sethInsName(rs.getString(9));

bean.sethPercentage(rs.getString(10));

bean.setsCourceName(rs.getString(11));

bean.setsInsName(rs.getString(12));

bean.setsPercentage(rs.getString(13));

bean.setSkill(rs.getString(14));

bean.setHobbies(rs.getString(15));

bean.setpDetail(rs.getString(16));

bean.setDeclaration(rs.getString(17));

bean.setDate(rs.getDate(18));

bean.setPlace(rs.getString(19));

bean.setDeclaration(rs.getString(20));

}

rs.close();

} catch (Exception e) {

log.error("Database Exception..", e);

throw new ApplicationException(

"Exception : Exception in getting User by pk");

} finally {

JDBCDataSource.closeConnection(conn);

}

log.debug("Model findByPK End");

return bean;

}

}

package in.co.job.portal.util;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.OutputStream;

import java.util.Date;

import com.itextpdf.text.BaseColor;

import com.itextpdf.text.Chunk;

import com.itextpdf.text.Document;

import com.itextpdf.text.DocumentException;

import com.itextpdf.text.Element;

import com.itextpdf.text.Image;

import com.itextpdf.text.List;

import com.itextpdf.text.ListItem;

import com.itextpdf.text.Paragraph;

import com.itextpdf.text.pdf.PdfPCell;

import com.itextpdf.text.pdf.PdfPTable;

import com.itextpdf.text.pdf.PdfWriter;

public class CreateResume {

public CreateResume() throws FileNotFoundException, DocumentException {

try {

OutputStream file = new FileOutputStream(new File("F:\\JobPortal\\JobPortal\\src\\main\\webapp\\file\\PDF\_Java4s8.pdf"));

Document document = new Document();

PdfWriter.getInstance(document, file);

//Inserting Image in PDF

//Inserting Table in PDF

PdfPTable table=new PdfPTable(3);

PdfPCell cell = new PdfPCell (new Paragraph ("Java4s.com"));

cell.setColspan (3);

cell.setHorizontalAlignment (Element.ALIGN\_CENTER);

cell.setPadding (10.0f);

cell.setBackgroundColor (new BaseColor (140, 221, 8));

table.addCell(cell);

table.addCell("Name");

table.addCell("Address");

table.addCell("Country");

table.addCell("Java4s");

table.addCell("NC");

table.addCell("United States");

table.setSpacingBefore(30.0f); // Space Before table starts, like margin-top in CSS

table.setSpacingAfter(30.0f); // Space After table starts, like margin-Bottom in CSS

//Inserting List in PDF

List list=new List(true,30);

list.add(new ListItem("Java4s"));

list.add(new ListItem("Php4s"));

list.add(new ListItem("Some Thing..."));

//Text formating in PDF

Chunk chunk=new Chunk("Welecome To Java4s Programming Blog...");

chunk.setUnderline(+1f,-2f);//1st co-ordinate is for line width,2nd is space between

Chunk chunk1=new Chunk("Php4s.com");

chunk1.setUnderline(+4f,-8f);

chunk1.setBackground(new BaseColor (17, 46, 193));

//Now Insert Every Thing Into PDF Document

document.open();//PDF document opened........

document.add(Chunk.NEWLINE); //Something like in HTML 🙂

document.add(new Paragraph("Dear Java4s.com"));

document.add(new Paragraph("Document Generated On - "+new Date().toString()));

document.add(table);

document.add(chunk);

document.add(chunk1);

document.add(Chunk.NEWLINE); //Something like in HTML 🙂

document.newPage(); //Opened new page

document.add(list); //In the new page we are going to add list

document.close();

file.close();

System.out.println("Pdf created successfully..");

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) throws FileNotFoundException, DocumentException {

new CreateResume();

}

}

package in.co.job.portal.util;

import java.sql.Timestamp;

import java.text.SimpleDateFormat;

import java.util.Date;

/\*\*

\* Data Utility class to format data from one format to another

\*

\* @author Navigable Set

\* @version 1.0

\* @Copyright (c) Navigable Set

\*/

public class DataUtility

{

/\*\*

\* Application Date Format

\*/

public static final String APP\_DATE\_FORMAT = "MM/dd/yyyy";

// public static final String APP\_TIME\_FORMAT = "MM/dd/yyyy HH:mm:ss";

/\*\*

\* Date formatter

\*/

private static final SimpleDateFormat formatter = new SimpleDateFormat(APP\_DATE\_FORMAT);

//private static final SimpleDateFormat timeFormatter = new SimpleDateFormat(APP\_TIME\_FORMAT);

/\*\*

\* Trims and trailing and leading spaces of a String

\*

\* @param val

\* @return

\*/

public static final String APP\_DATE\_FORMAT1 = "yyyy-MM-dd";

private static final SimpleDateFormat formatter1 = new SimpleDateFormat(APP\_DATE\_FORMAT1);

public static Date getDate1(String val) {

Date date = null;

try {

date = formatter1.parse(val);

} catch (Exception e) {

}

return date;

}

public static String getString(String val) {

if (DataValidator.isNotNull(val)) {

return val.trim();

} else {

return val;

}

}

/\*\*

\* Converts and Object to String

\*

\* @param val

\* @return

\*/

public static String getStringData(Object val) {

if (val != null) {

return val.toString();

} else {

return "";

}

}

/\*\*

\* Converts String into Integer

\*

\* @param val

\* @return

\*/

public static int getInt(String val) {

if (DataValidator.isInteger(val)) {

return Integer.parseInt(val);

} else {

return 0;

}

}

/\*\*

\* Converts String into Long

\*

\* @param val

\* @return

\*/

public static long getLong(String val) {

if (DataValidator.isLong(val)) {

return Long.parseLong(val);

} else {

return 0;

}

}

/\*\*

\* Converts String into Date

\*

\* @param val

\* @return

\*/

public static Date getDate(String val) {

Date date = null;

try {

date = formatter.parse(val);

} catch (Exception e) {

}

return date;

}

/\*\*

\* Converts Date into String

\*

\* @param date

\* @return

\*/

public static String getDateString(Date date) {

try {

if(date!=null) {

return formatter.format(date);

}

else{

return "";

}

} catch (Exception e) {

return "";

}

}

public static String getDateString1(Date date) {

try {

if(date!=null) {

return formatter1.format(date);

}

else{

return "";

}

} catch (Exception e) {

return "";

}

}

/\*\*

\* Gets date after n number of days

\*

\* @param date

\* @param day

\* @return

\*/

public static Date getDate(Date date, int day) {

return null;

}

/\*\*

\* Converts String into Time

\*

\* @param cdt

\* @return

\*/

public static Timestamp getTimestamp(long l) {

Timestamp timeStamp = null;

try {

timeStamp = new Timestamp(l);

} catch (Exception e) {

return null;

}

return timeStamp;

}

/\*\*

\* Converts String into Time

\*

\* @param cdt

\* @return

\*/

public static Timestamp getTimestamp(String cdt) {

Timestamp timeStamp = null;

try {

// timeStamp = new Timestamp((timeFormatter.parse(cdt)).getTime());

} catch (Exception e) {

return null;

}

return timeStamp;

}

/\*\*

\* Converts Time into Long

\*

\* @param tm

\* @return

\*/

public static long getTimestamp(Timestamp tm) {

try {

return tm.getTime();

} catch (Exception e) {

return 0;

}

}

/\*\*

\* Provide Current time

\*

\*

\* @return Time

\*/

public static Timestamp getCurrentTimestamp() {

Timestamp timeStamp = null;

try {

timeStamp = new Timestamp(new Date().getTime());

} catch (Exception e) {

}

return timeStamp;

}

public static void main(String[] args)

{

System.out.println(DataUtility.getDate1("2023-06-01"));;

}

}

package in.co.job.portal.util;

import java.io.File;

import java.io.IOException;

import java.util.List;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.Part;

import in.co.job.portal.bean.BaseBean;

import in.co.job.portal.controller.BaseCtl;

import in.co.job.portal.controller.JPView;

/\*\*

\* This class provides utility operation for Servlet container like forward,

\* redirect, handle generic exception=same time exception, manage success and

\* error message, manage default Bean and List, manage pagination parameters

\*

\* @author Navigable Set

\* @version 1.0

\* @Copyright (c) Navigable Set

\*/

public class ServletUtility {

public static void forward(String page, HttpServletRequest request, HttpServletResponse response)

throws IOException, ServletException {

RequestDispatcher rd = request.getRequestDispatcher(page);

System.out.println(page);

rd.forward(request, response);

}

public static void redirect(String page, HttpServletRequest request, HttpServletResponse response)

throws IOException, ServletException {

response.sendRedirect(page);

}

public static void handleException(Exception e, HttpServletRequest request, HttpServletResponse response)

throws IOException, ServletException {

request.setAttribute("exception", e);

ServletUtility.forward(JPView.ERROR\_CTL, request, response);

e.printStackTrace();

}

public static String getErrorMessage(String property, HttpServletRequest request) {

String val = (String) request.getAttribute(property);

if (val == null) {

return "";

} else {

return val;

}

}

public static String getMessage(String property, HttpServletRequest request) {

String val = (String) request.getAttribute(property);

if (val == null) {

return "";

} else {

return val;

}

}

public static void setErrorMessage(String msg, HttpServletRequest request) {

request.setAttribute(BaseCtl.MSG\_ERROR, msg);

}

public static String getErrorMessage(HttpServletRequest request) {

String val = (String) request.getAttribute(BaseCtl.MSG\_ERROR);

if (val == null) {

return "";

} else {

return val;

}

}

public static void setSuccessMessage(String msg, HttpServletRequest request) {

request.setAttribute(BaseCtl.MSG\_SUCCESS, msg);

}

public static String getSuccessMessage(HttpServletRequest request) {

String val = (String) request.getAttribute(BaseCtl.MSG\_SUCCESS);

if (val == null) {

return "";

} else {

return val;

}

}

public static void setBean(BaseBean bean, HttpServletRequest request) {

request.setAttribute("bean", bean);

}

public static BaseBean getBean(HttpServletRequest request) {

return (BaseBean) request.getAttribute("bean");

}

public static String getParameter(String property, HttpServletRequest request) {

String val = (String) request.getParameter(property);

if (val == null) {

return "";

} else {

return val;

}

}

public static void setList(List list, HttpServletRequest request) {

request.setAttribute("list", list);

}

public static List getList(HttpServletRequest request) {

return (List) request.getAttribute("list");

}

/\*\*

\* Sets Page Number for List pages

\*

\* @param pageNo

\* @param request

\*/

public static void setPageNo(int pageNo, HttpServletRequest request) {

request.setAttribute("pageNo", pageNo);

}

public static int getPageNo(HttpServletRequest request) {

return (Integer) request.getAttribute("pageNo");

}

public static void setPageSize(int pageSize, HttpServletRequest request) {

request.setAttribute("pageSize", pageSize);

}

public static int getPageSize(HttpServletRequest request) {

return (Integer) request.getAttribute("pageSize");

}

public static void setOpration(String msg, HttpServletRequest request) {

request.setAttribute("Opration", msg);

}

public static String getOpration(HttpServletRequest request) {

String val = (String) request.getAttribute("Opration");

if (val == null) {

return "";

} else {

return val;

}

}

public static String subFile(HttpServletRequest request, HttpServletResponse response,String name)

throws ServletException, IOException {

String savePath = DataUtility.getString(PropertyReader.getValue("filePath"));

File fileSaveDir = new File(savePath);

if (!fileSaveDir.exists()) {

fileSaveDir.mkdir();

}

Part part = request.getPart("file");

String fileName = extractFileName(part);

part.write(savePath + File.separator + name+fileName);

return name+fileName;

}

public static String extractFileName(Part part) {

String contentDisp = part.getHeader("content-disposition");

String[] items = contentDisp.split(";");

for (String s : items) {

if (s.trim().startsWith("filename")) {

return s.substring(s.indexOf("=") + 2, s.length() - 1);

}

}

return "";

}

}

##### Conclusions

In conclusion, the Online Job Portal project offers a comprehensive and user-friendly platform that revolutionizes the job search and recruitment process. By leveraging technologies such as Java, JSP, Servlets, JDBC, MySQL, Maven, MVC architecture, and Tomcat server, the project provides a robust and efficient solution for connecting job seekers and employers.

Through the Online Job Portal, job seekers can create personalized profiles, search for job vacancies, and apply for positions with ease. The advanced search options and filters enable them to find relevant opportunities quickly, while the ability to track applications and receive notifications streamlines the entire job search process. The project prioritizes user experience by providing a simple and intuitive interface, making it accessible to a wide range of users.

For employers, the Online Job Portal simplifies the recruitment process by allowing them to post job vacancies, search for qualified candidates, and manage applications seamlessly. The platform provides employers with the tools they need to identify the best candidates, review applicant profiles, and communicate efficiently. By streamlining the hiring process, the project saves time and resources for employers, enabling them to build a talented workforce effectively.

The project's adoption of the MVC architecture ensures a modular and maintainable codebase, promoting code organization and separation of concerns. It facilitates future enhancements and modifications, making it easier to scale the system and accommodate evolving requirements.

Moreover, the implementation of secure authentication and authorization mechanisms ensures the confidentiality and privacy of user data. The project prioritizes the protection of user information, employing encryption techniques and robust security measures to prevent unauthorized access.

Overall, the Online Job Portal project demonstrates the power of technology in revolutionizing the job search and recruitment domain. By providing a user-friendly interface, advanced search capabilities, secure authentication, and a robust architecture, it simplifies the job search process for job seekers and employers alike. The project's focus on abstraction, modularity, and usability contributes to its effectiveness in meeting the needs of users and fostering successful connections between job seekers and employers.

The Online Job Portal project serves as a testament to the potential of technology in transforming traditional processes, enabling individuals to find employment opportunities more efficiently and organizations to build a talented workforce effectively. With its user-centric design and robust functionality, the Online Job Portal project is poised to make a significant impact in the realm of job search and recruitment.