BUSINESS COMPONENT DEVELOPMENT - II (JIAT/BCD II)

STUDENT NAME	**: M.R.P.N.THARUKSHA RAJAPAKSHA**
BCU NO.	**: 22178965**
NIC NO.	**: 200019401866**
BRANCH	: COLOMBO

Introduction

I have developed a project management part for the NDCamera web application. I have used best practices and standards to develop this application and this application is user-friendly, efficient, and secure. This application has included Time services to manage task scheduling, Interceptor classes to log user activity, Different types of transactions to ensure data consistency and reliability, Robust security architecture to protect user data and ensure privacy, Exception handling to optimize application performance, EJB components in a split directory structure, and Testing tools to test the application works.

This presentation summarizes the key features and benefits of the NDCamera project.

Key Features

- Java Class Library
- Java Enterprise Application
- EJB Module
- Java Web Application
- Time Services
- Interceptor Classes
- Transactions
- Robust Security Architecture
- Exception Handling
- EJB Components

Java Class Library

A collection of pre-written Java code, commonly referred to as the Java Standard Library, the Java Class Library offers a broad range of functionality for creating Java applications. It is a crucial component of the Java Development Kit (JDK) and includes tens of thousands of classes and interfaces, grouped into packages that address I/O, networking, GUI (Graphical User Interface), collections, concurrency, database connectivity, security, and other topics.

In the NDCamera Project, I have used this to create NDCameraDBLibrary. ProductSessionBeanRemote and TimerSessionBeanRemote session bean remotes, Product entity class, and AuthException exception class are within it. And, NDCameraDBLibrary is added as a library to the NDCameraEnterpriseApplication-ejb EJB Module and NDCameraEnterpriseApplication-war web application.

Benefits of Java Class Library

- Reusability: There is a sizable selection of pre-built classes and interfaces available in the Java Class Library that can be utilized in many Java applications. This promotes code reusability and minimizes duplication by allowing developers to use existing code rather than beginning from scratch, saving time and effort during development.
- Productivity: The Java Class Library offers a comprehensive collection of APIs (Application Programming Interfaces), which encapsulate complicated functionality into user-friendly procedures. As a result, developers may write code more quickly and effectively because there is no longer a need to manually apply complex logic.
- Platform independence: Because the Java Class Library is intended to be platform-independent, Java programs created with its help can run without modification on any system that supports the Java Virtual Machine (JVM). This supports Java's "Write Once, Run Everywhere" philosophy, which makes it incredibly portable and adaptable.
- Robustness: Java applications are robust and reliable because the Java Class Library contains classes and interfaces that adhere to strict coding rules and guidelines. It offers techniques for handling errors, exceptions, and input and output, which aids in producing more reliable and stable programs.

Benefits of Java Class Library

- Scalability: The Java Class Library offers scalability capabilities that enable developers to create high-performance, scalable programs. These features include support for multithreading, networking, and collections. Java programs can effectively handle big data volumes, concurrent users, and heavy traffic loads thanks to their features.
- Security: The Java Class Library has security features that aid in the development of secure Java applications, including encryption, authentication, and access control. Java applications are more secure because the Java Class Library adheres to stringent security standards to guard against vulnerabilities like buffer overflow, SQL injection, and cross-site scripting (XSS).
- Community-driven: A sizable and vibrant Java community supports the Java Class Library and continuously contributes to its growth and upkeep. This means that new features, bug fixes, and performance enhancements are frequently added to the Java Class Library, giving Java developers access to a dependable and current collection of tools.

Java Enterprise Application

A platform and set of requirements for creating and deploying massive, enterprise-level Java applications are together referred to as Java Enterprise Application (Java EE) or Jakarta EE. It offers an environment that is stable, scalable, and secure for developing applications that can manage heavy loads and intricate business logic.

In the NDCamera Project, I have used this to create NDCameraEnterpriseApplication. NDCameraEnterpriseApplication-ejb EJB Module and NDCameraEnterpriseApplication-war web application are within it.

Benefits of Java Enterprise Application

- Portability: Java EE applications are platform-independent and can be installed on any application server that supports Java EE. This offers simple migration and scalability, as well as flexibility in selecting multiple application servers or cloud platforms.
- Scalability: Java EE offers distributed computing, enabling the deployment of applications over a number of servers to handle heavy loads and offer horizontal scalability. Building high-performance applications may be done with it because it also enables clustering, load balancing, and failover.
- Security: To shield applications from security threats, Java EE offers a full range of security capabilities, including authentication, authorisation, and encryption. Additionally, it supports the Java Authorization Contract for Containers (JACC) and Java Authentication and Authorization Service (JAAS), which aid in the development of secure applications.
- Productivity: With support for messaging, transactions, persistence, and web services, Java EE offers a comprehensive set of APIs and tools for developing sophisticated enterprise applications. Additionally, it has features like JavaServer Faces (JSF), Enterprise JavaBeans (EJB), and Java Persistence API (JPA) that facilitate quick development and cut down on boilerplate code.

Benefits of Java Enterprise Application

- Extensibility: Because Java EE is so versatile, programmers can create unique plug-ins and components to meet unique requirements. This gives developers the freedom to create applications that interface with current systems and cater to particular business needs.
- Community & Ecosystem: The Java EE platform includes a sizable and vibrant community of vendors, consumers, and developers that offers a multitude of tools, libraries, and frameworks. Enterprise applications can be efficiently built using a variety of application servers, tools, and frameworks that are part of the Java EE ecosystem.

EJB Module

A Java EE (Enterprise Edition) component known as an EJB (Enterprise JavaBeans) module enables programmers to build distributed, reusable, server-side business logic components. EJBs make it simpler to construct reliable and scalable corporate applications since they are made to be utilized in a distributed, scalable, and transactional environment. They also offer a variety of services, including database access, transaction management, security, and messaging.

In the NDCamera Project, I have used this to create NDCameraEnterpriseApplication-ejb. ProductSessionBean and TimerSessionBean Session Beans, and LoggingInterceptor Interceptor classes are within it.

Benefits of EJB Module

- Component-based design: EJBs offer a component-based architecture, allowing for the modularization of business logic into reusable components, simplifying application maintenance and updating.
- Distributed computing: When components are placed on many servers and communicate with one another through remote method invocations (RMI), distributed computing is made possible by EJBs, enabling scalability and load balancing.
- Transaction management: EJBs come with integrated support for transaction management, enabling multi-step, complicated business procedures to be completed in a single, atomic transaction.
- Security: To ensure that corporate applications are secure and meet security standards, EJBs include a built-in security framework that enables the security of business logic components such authentication, authorization, and role-based access control (RBAC).
- Scalability and performance: EJBs can be deployed on a cluster of servers and are made to be scalable, enabling high availability and load balancing. Enterprise applications are effective thanks to EJBs' additional provision of techniques for caching and speed optimization.

Benefits of EJB Module

- Interoperability: EJBs can be deployed on any Java EE compliant application server and are built on the platform-independent Java technology, which makes them interoperable and enables their interaction with other Java EE technologies and third-party libraries.
- Productivity of developers: EJBs come with a number of services and capabilities out of the box, including transaction management, security, and messaging, which can shorten the development process and free up developers to concentrate on business logic rather than intricate technical details.

Java Web Application

A software program created with Java technologies and intended to operate on a web server is known as a Java web application. It is commonly accessed by users using a web browser and is composed of Java code, HTML, CSS, and JavaScript.

In the NDCamera Project, I have used this to create NDCameraEnterpriseApplication-war. 403.jsp, error.jsp, index.jsp and login.jsp jsp files and DeleteProduct, InsertProduct, Logout, SearchProduct, and UpdateProduct servlets are within it.

Benefits of Java Web Application

- Portability: Because Java is platform-independent, Java web applications can operate without modification on a variety of operating systems, including Windows, Linux, and macOS. Java web applications are hence very flexible to various settings and portable.
- Scalability: Java web applications are renowned for their capacity to support numerous concurrent users and large-scale applications. Java is a good choice for creating enterprise-level applications because it offers strong libraries and frameworks for managing complicated business logic, controlling database activities, and managing user sessions.
- Security: Java comes with built-in security mechanisms that assist stop unauthorized access, data breaches, and other security concerns. These features include a powerful type-checking system and a strong security manager. Java online applications can benefit from a variety of security frameworks and libraries for encryption, authentication, and authorisation.
- Rich ecosystem: Java has a large and established ecosystem of libraries, frameworks, and tools that speed up and improve the efficiency of web application development. Prominent Java web development frameworks like Spring, JavaServer Faces (JSF), and Struts offer tools for creating dependable and scalable web applications.

Benefits of Java Web Application

- Developer output: Because Java is a popular programming language with a huge developer community, there are a lot of learning tools and documentation available for debugging. Java also encourages the use of modular programming, which boosts the maintainability, extensibility, and reuse of code, hence boosting developer productivity.
- Integration capabilities: Due to its support for a number of integration protocols and standards, Java web applications can readily interface with other systems and technologies, including databases, messaging systems, and web services. As a result, integrating Java web applications with existing systems and utilizing their features is made simpler.
- Support for multithreading: Java provides native multithreading capability, enabling Java web applications to effectively handle several concurrent requests, enhancing performance and responsiveness.

Time Services

The Time Services API in Enterprise JavaBeans (EJB) offers scheduling features for carrying out operations at certain times or at regular intervals. Developers can schedule actions like method calls, message sending, and other actions to be carried out at certain times or on a regular basis using EJB Time Services.

In the NDCamera Project, I have used this to create scheduler(), createTimer(), and timeoutHandler() methods in the TimerSessionBean.

Benefits of Time Services

- Reliability: Even in the event of failures or system restarts, EJB Time Services' reliable scheduling features make sure that tasks are carried out at the appointed time. By ensuring that crucial operations are carried out exactly as planned, this increases the system's overall reliability.
- Flexibility: EJB Time Services allow for task scheduling flexibility. The execution of tasks can be scheduled by developers for specific days and times, as well as at regular intervals such every minute, hour, or day. This enables precise control over task scheduling depending on the unique requirements of the application.
- Scalability: EJB Time Services are made to function well in complex, distributed systems. Task distribution in a distributed environment is facilitated by the ability to plan execution of tasks to run on particular EJB instances or across a cluster of EJB instances.
- Simplified Development: EJB Time Services offer a straightforward and standardized API for scheduling tasks, making it simple for developers to integrate scheduling features into their applications without needing to create intricate code to handle timers or scheduling logic. This may lead to quicker development and less maintenance work.

Benefits of Time Services

- Integration with EJB Container: Because EJB Time Services are integrated with the EJB container, other EJB services like transactions, security, and messaging can be used with them without any problems. It is now possible to schedule tasks consistently and cohesively within the EJB framework, which makes it simpler to manage and maintain scheduling capabilities as a component of the broader EJB application.

Interceptor Classes

An interceptor is an unique kind of class that may intercept method calls on EJB components in the context of Enterprise JavaBeans (EJB). This enables the application of cross-cutting issues like logging, security, and transaction management in a modular and reusable manner.

In EJB, there are two different kinds of interceptor classes:

- Method interceptors: Each individual method of an EJB component is subject to a method interceptor. At the class or method level, they can be specified using the @Interceptors annotation. Method interceptors can be used to add actions like logging, auditing, or security checks. They can intercept business methods as well as the lifecycle methods of an EJB component.
- Lifecycle interceptors: They are used during the whole lifespan of an EJB component. The @AroundConstruct, @PostConstruct, @PreDestroy, and @AroundInvoke annotations can be used to define them. Lifecycle interceptors can be used to add actions like resource management, caching, or exception handling by catching the creation, initialization, and destruction of an EJB component as well as the execution of business methods.

In the NDCamera Project, I have used this to create the LoggingInterceptor class, and the LoggingInterceptor class is annotated on the methods in the ProductSessionBean.

Benefits of Interceptor Classes

- Reusability: Cross-cutting concerns can be implemented in a reusable manner using interceptor classes, which can be developed once and used to numerous EJB components. This discourages duplication and encourages code reuse.
- Modularity: Because interceptor classes may be used to encapsulate particular functions like logging, security, or transaction management, they enable the separation of concerns. A modular and maintainable architecture is encouraged by this.
- Separation of concerns: By separating business logic from cross-cutting issues using interceptor classes, the software becomes cleaner and simpler to comprehend.
- Flexibility: As interceptor classes may easily be added, removed, or modified without altering the underlying logic of EJB components, this flexibility allows for changing or extending the functionality of EJB components without changing how they are implemented.
- Consolidated management: By allowing for consistent application of cross-cutting concerns across numerous EJB components, interceptor classes can be centrally managed, improving maintainability and lowering the chance of introducing inconsistencies.

Transactions

A server-side component technology called Enterprise JavaBeans (EJB) is used to create distributed, scalable, and reliable Java applications. Several forms of transactions that are used to control the consistency and integrity of data in distributed applications are supported by EJB.

The various EJB transaction types:

- Container-Managed Transactions (CMT)
- Bean-Managed Transactions (BMT)
- No Transaction

In the NDCamera Project, I have used this to create the ProductSessionBean Session Bean.

Container-Managed Transactions (CMT)

The EJB container handles transaction management in CMT. Based on the transaction attributes supplied in the EJB deployment descriptor or annotations, the container automatically manages the transactions for EJB methods.

Benefits of CMT:

- Simplified development: As the container manages transaction management automatically, developers don't need to write explicit transaction management code in their EJB methods.
- Consistency: The container ensures that transactions are consistently managed across all EJB methods, providing a consistent approach to transaction management.
- Scalability: In multi-user scenarios, CMT's support for concurrent access to EJB methods makes it possible to scale.
- Portability: CMT enables the deployment of EJBs in various application servers that meet the Java EE specification, offering portability across several platforms.

Bean-Managed Transactions (BMT)

While utilizing BMT, the Java Transaction API is used by the developer to explicitly manage the transaction management in the EJB methods (JTA).

Benefits of BMT:

- Fine-grained control: Since developers have full control over transaction management, they can precisely regulate the transaction boundaries.
- Customization: Based on particular business needs, developers can alter the transaction management logic.
- Flexibility: Depending on the needs of the application, BMT supports a variety of transaction techniques, including distributed transactions, two-phase commit, and others.

No Transaction

The EJB methods are not subject to transaction management in this kind of transaction. There is no transactional coordination across various EJB methods; rather, each method invocation is handled as a separate transaction.

Benefits of No Transaction:

- Simplicity: The simplest method of transaction management requires neither writing nor configuring any transaction management code.
- Performance: As each method call is handled as a separate transaction, there is no transaction overhead.
- Read-only operations appropriate: No Transaction is appropriate for read-only actions where efficiency is a top concern but data consistency is not essential.

Robust Security Architecture

In an environment where Java EE (Enterprise Edition) application server is used, the robust Security Architecture in EJB (Enterprise JavaBeans) is a set of security features and procedures created to provide robust and safe access control to EJB components.

In the NDCamera Project, I have used this to create the login and authentication part. To create this part I have created three user ids in the sever-config>security on the server side. The configurations are set in the web.xml file in NDCameraEnterpriseApplication-war web application.

Benefits of Robust Security Architecture

- Authentication: EJB offers means for identifying clients accessing EJB components so that the application server may confirm their identities. This assists in making sure that the protected EJB components can only be accessed by authorized customers.
- Authorization: The EJB platform enables fine-grained authorization, allowing programmers to specify access control policies at the method or even the individual parameter level of the EJB components. By guaranteeing that clients may only access the methods or parameters to which they are authorized, this offers a granular level of protection.
- Declarative Security: EJB enables programmers to specify security restrictions declaratively in the deployment descriptor or by making use of source code annotations. It is easier to maintain and manage security settings when security concerns are separated from the business logic of the EJB components.
- Role-based Access Control (RBAC): The EJB framework supports RBAC, enabling developers to establish roles and assign them to users or groups. Due to the easier management and control of access to EJB components based on user roles, this aids in imposing access control based on roles.

Benefits of Robust Security Architecture

- Secure Communication: To protect communication between clients and EJB components, EJB supports a number of secure communication protocols, including SSL/TLS. This guarantees that the information transmitted between clients and EJB components is secure and shielded from spying on or alteration.
- Fine-grained Exception Handling: The exception handling capabilities offered by EJB enable developers to specify fine-grained exception handling techniques, such as the mapping of exceptions to certain error codes or error pages. In a controlled and safe manner, this aids in managing security-related exceptions.
- Monitoring and Auditing: EJB offers tools for auditing and monitoring that let administrators keep track of and keep an eye on security-related events like unsuccessful login attempts and illegal access attempts, among other things. This facilitates quick detection and action in the face of security risks.

Exception Handling

Enterprise Java Beans (EJB) use an exception handling technique to deal with runtime faults and exceptions that may arise while EJB components, like session beans or message-driven beans, are being executed in a Java EE (Enterprise Edition) application server.

In the NDCamera Project, I have used this everywhere in the project to handle the exceptions as try-catches and throws. Specially I have handled the EJBAccessException in the DeleteProduct, InsertProduct, SearchProduct, and UpdateProduct servlets in the NDCameraEnterpriseApplication-war web application. And, I have manually created the AuthException class to handle the RuntimeException in the NDCameraDBLibrary. The AuthException is used to throw the exceptions in the methods in ProductSessionBean.

Benefits of Exception Handling

- Robustness: By enabling developers to build code that can manage unforeseen faults and runtime exceptions, EJB components become more durable and reliable.
- Fault tolerance: EJB capabilities like transaction management and container-managed persistence allow for the possibility of exceptions being issued in the event of failures such problems with database connectivity. Developers may respond appropriately to such problems and restart the operation or roll back the transaction thanks to exception handling.
- Scalability: In a distributed and scalable environment, exception handling aids in controlling faults and exceptions. Exception handling enables proper handling of exceptions across numerous nodes, ensuring that the system is reliable and available. EJB applications frequently run in a clustered environment.
- Maintainability: Handling exceptions makes building maintainable code easier. Developers can guarantee that error messages are logged or reported in a meaningful fashion, making it simpler to identify and fix problems in the production environment, by handling exceptions properly.

Benefits of Exception Handling

- Error reporting and logging: Proper error reporting and logging are made possible by exception handling, which aids in identifying problems and resolving them during the development, testing, and production phases of an application's lifecycle. Exceptions that are correctly logged and reported offer useful information for troubleshooting and debugging.

EJB Components

A Java-based framework called EJB (Enterprise JavaBeans) offers a selection of server-side components for creating distributed and scalable applications. EJB components offer a number of advantages and can be deployed on Java EE (Enterprise Edition) application servers.

The various EJB Components:

- Session Beans
- Entity Beans
- Message-Driven Beans

In the NDCamera Project, I have used this everywhere the project to develop.

Session Beans

Business logic is contained in stateful or stateless components called session beans, which offer services to customers. Concurrency management, security, and transaction processing can all be implemented using them.

Benefits of Session Beans:

- Scalability: Session Beans are appropriate for high-performance applications because they can be readily expanded horizontally by adding new instances to manage growing demands.
- Transaction Management: The handling of database operations and other resource management is made simpler by Session Beans' ability to take part in container-managed transactions.
- Encapsulation of Business Logic: Session Beans encapsulate business logic, improving concern separation and code maintainability.
- Remote Access: Clients can connect to Session Beans from a distance, enabling distributed architectures and allowing remote access to business logic.

Entity Beans

Entity Beans represent persistent data in a relational database and offer an easy way to use object-oriented programming to communicate with the database.

Benefits of Entity Beans:

- Object-Relational Mapping (ORM): Entity Beans offer ORM capabilities, making it simpler to interact with databases in an object-oriented way by allowing developers to map Java objects to database tables.
- Transaction Management: The consistency and integrity of the data in the database are ensured by the participation of entity beans in container-managed transactions.
- Scalability: Entity Beans can be cached and optimized for speed, enabling quick data manipulation and retrieval operations.
- Data Integrity: By enforcing data validation and business rules, entity beans can guarantee data integrity in the database.

Message-Driven Beans

Java EE applications use message-driven beans for asynchronous message processing. They process messages asynchronously after receiving them from a message broker.

Message-Driven Beans:

- Asynchronous Processing: Message-Driven Beans enable asynchronous message processing, which can enhance the responsiveness and scalability of programs.
- Loose Coupling: Message-Driven Beans make it possible for components to be loosely coupled, which gives distributed systems' architectures more flexibility.
- Scalability: Message-Driven Beans are suitable for high-throughput messaging systems because they can be readily extended horizontally by adding more instances to manage growing message loads.
- Reliability: With functions like message acknowledgment and error handling, Message-Driven Beans offer dependable message processing.

Summary

This presentation summarizes the key features and benefits of the NDCamera project.

Thank You !