Java Immersion Projects Overview

There are two projects that are part of Java Immersion Training (JIT). The first project deals with a Customer Ordering System, the second project deals with a World Geography Application.

Java is evolving and the technologies are changing. The first project, the Customer Ordering System will focus on the "classic" Java technologies, including using JDBC to access databases, and writing web pages with Servlets and JavaServer Pages (JSP). Classic Java web applications are very common, and the concepts used in them form the foundation beneath the next generation of Java web applications.

The second project, the World Geography Application will focus on the next generation of Java Enterprise technologies, including using Java Persistence API (JPA) to access a database, and providing a REST Web Service interface to support mobile applications.

Both projects will include Best Practices including:

- Separation of application logic from persistence routines
- Separation of web front end code from application logic
- Creating Interfaces for all external I/O (Action) classes
- Documentation using JavaDocs
- Unit Testing using JUnit
- Mocking of dependencies during testing using Mockito
- Database unit testing using DBUnit
- Web page testing using Selenium or REST Web Service testing using SoapUI
- Build management using Maven
- Automated Code Review
- Accessibility compliance for all web pages
- Logging using Log4J
- IRS Java Coding Standards compliance

Customer Ordering System

Application background:
The Customer Ordering System offers a web interface to managing the Customer database. This database tracks Customer, Products, Suppliers and Orders.

The web interface for the database allows the User to search for a specific Customer, Order, Product or Supplier. The web interface will then display the detailed information about the requested record. The web interface will allow authorized users to add/update/delete database records.

Application requirements:

- All users must be authenticated
- Users authenticated as a Manager are authorized to add/change/delete records
- Customer records are searchable by Name
- Orders records are searchable by Customer or Order Date
- Products records are searchable by Name or Supplier
- Suppliers records are searchable by Company Name

Business Rules:

- All records follow USA format for addresses, phone numbers, currency and dates
- Price values must be greater than zero
- All quantities must be greater then zero
- All required fields must not be empty
- All string field inputs must be sanitized to avoid security vulnerabilities
- All string fields outputs must be sanitized to avoid security vulnerabilities
- All Order Dates must be current date or earlier
- All errors must be logged in addition to being displayed for the User
- All Database constraints must be followed

Database Structures:

```
CREATE TABLE CUSTOMER (
    ID INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
        (START WITH 1, INCREMENT BY 1),
    FIRST_NAME VARCHAR(40) NOT NULL,
    LAST_NAME VARCHAR(40) NOT NULL,
    CITY VARCHAR(40),
    COUNTRY VARCHAR(40),
    PHONE VARCHAR(30),
    CONSTRAINT PK_CUSTOMER PRIMARY KEY (ID)
);

CREATE TABLE ORDERS (
    ID INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
        (START WITH 1, INCREMENT BY 1),
    ORDER_DATE DATE NOT NULL,
    CUSTOMER_ID INTEGER,
    TOTAL_AMOUNT DECIMAL(12,2),
    ORDER_NUMBER VARCHAR(10),
    CONSTRAINT PK_ORDER PRIMARY KEY (ID)
);

CREATE TABLE ORDER_ITEM (
    ID INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
        (START WITH 1, INCREMENT BY 1),
    ORDER_ID INTEGER NOT NULL,
    PRODUCT_ID INTEGER NOT NULL,
    UNIT_PRICE DECIMAL(12,2) NOT NULL,
    QUANTITY INTEGER NOT NULL,
    CONSTRAINT PK_ORDERITEM PRIMARY KEY (ID)
);

CREATE TABLE PRODUCT (
    ID INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
        (START WITH 1, INCREMENT BY 1),
    PRODUCT_NAME VARCHAR(50) NOT NULL,
    SUPPLIER_ID INTEGER NOT NULL,
    UNIT_PRICE DECIMAL(12,2),
    PACKAGE VARCHAR(30),
    IS_DISCONTINUED BOOLEAN,
    CONSTRAINT PK_PRODUCT PRIMARY KEY (ID)
);
```

```
CREATE TABLE SUPPLIER (
    ID INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
        (START WITH 1, INCREMENT BY 1),
    COMPANY_NAME VARCHAR(40) NOT NULL,
    CONTACT_NAME VARCHAR(50),
    CITY VARCHAR(40),
    COUNTRY VARCHAR(40),
    PHONE VARCHAR(30),
    FAX VARCHAR(30),
    CONSTRAINT PK_SUPPLIER PRIMARY KEY (ID)
);

ALTER TABLE ORDERS
    ADD CONSTRAINT FK_ORDER_CUSTOMER
        FOREIGN KEY (CUSTOMER_ID)
    REFERENCES CUSTOMER (ID);

ALTER TABLE ORDER_ITEM
    ADD CONSTRAINT FK_ORDER_ITEM_ORDER
        FOREIGN KEY (ORDER_ID)
    REFERENCES ORDERS (ID);

ALTER TABLE ORDER_ITEM
    ADD CONSTRAINT FK_ORDER_ITEM_PRODUCT
        FOREIGN KEY (PRODUCT_ID)
    REFERENCES PRODUCT (ID);

ALTER TABLE PRODUCT
    ADD CONSTRAINT FK_PRODUCT_SUPPLIER
        FOREIGN KEY (SUPPLIER_ID)
    REFERENCES SUPPLIER (ID);
```

```
CREATE VIEW COUNT_PRODUCT_ORDERS
    AS
        SELECT OI.QUANTITY AS QUANTITY, P.PRODUCT_NAME
        FROM (SELECT SUM(QUANTITY) AS QUANTITY, PRODUCT_ID
            FROM ORDER_ITEM GROUP BY PRODUCT_ID) AS OI
        INNER JOIN PRODUCT P
        ON OI.PRODUCT_ID = P.ID

    UNION

        SELECT 0 AS QUANTITY, P.PRODUCT_NAME
        FROM PRODUCT P
        WHERE P.ID NOT IN (SELECT DISTINCT PRODUCT_ID
            FROM ORDER_ITEM)
        ORDER BY QUANTITY;

CREATE PROCEDURE SUSPEND_SUPPLIER(IN SUPPLIER_ID_IN INTEGER,
            OUT ROWS_EFFECTED INTEGER)
    INTEGER ROW_COUNT =
        UPDATE PRODUCT SET IS_DISCONTINUED = TRUE
            WHERE SUPPLIER_ID = SUPPLIER_ID_IN

    RETURN ROW_COUNT
```

World Geography Application

Application background:
The World Geography Application offers a web interface to navigating a collection of Countries, Cities, Continents and Organizations. In addition to these basic collections this application offers a collection of Languages by Country, and Countries within an Organization.

The REST web service interface for the database allows the User to search for a specific City, Country, Continent or Organization. The web service will then provide the detailed information about the requested record. The web service also allows searching for Country by Language or Organization.

Application requirements:

- All users may be autonomous or authenticated
- Autonomous are restricted to query operations only
- Users authenticated as a Manager are authorized to add/change/delete records
- City records are searchable by Name or Country
- Country records are searchable by Name, Organization or Language
- Continent records are searchable by Name
- Organization records are searchable by Name or Year founded

Business Rules:

- All elevation and surface area values are in meters
- All year values are 4 digits
- All numeric values must be greater then zero
- All required fields must not be empty
- All string field inputs must be sanitized to avoid security vulnerabilities
- All string fields outputs must be sanitized to avoid security vulnerabilities
- The IS_OFFICIAL value is ‘T’ or ‘F’
- All code values must in in uppercase
- All Database constraints must be followed

Database Structures:

```
CREATE TABLE COUNTRY (
    CODE CHAR(3) NOT NULL,
    NAME CHAR(52) NOT NULL,
    CONTINENT VARCHAR(15),
    REGION CHAR(26) NOT NULL,
    SURFACE_AREA DECIMAL(10,2) NOT NULL,
    INDEP_YEAR SMALLINT,
    POPULATION INTEGER NOT NULL,
    LIFE_EXPECTANCY DECIMAL(3,1),
    GNP DECIMAL(10,2),
    LOCALNAME CHAR(45),
    GOVERNMENT_FORM CHAR(45),
    CAPITAL INTEGER,
    CODE2 CHAR(2),
    CONSTRAINT PK_COUNTRY PRIMARY KEY (CODE)
);

CREATE TABLE CITY (
    ID INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
        (START WITH 1, INCREMENT BY 1),
    NAME CHAR(35) NOT NULL,
    COUNTRY_CODE CHAR(3) NOT NULL,
    DISTRICT CHAR(20) NOT NULL,
    POPULATION INTEGER NOT NULL,
    CONSTRAINT PK_CITY PRIMARY KEY (ID)
);

CREATE TABLE CONTINENT (
    CODE CHAR(2) NOT NULL,
    NAME VARCHAR(15) NOT NULL,
    HIGHEST_POINT VARCHAR(20) NOT NULL,
    HIGHEST_ELEVATION INTEGER NOT NULL,
    LOWEST_POINT VARCHAR(20) NOT NULL,
    LOWEST_ELEVATION INTEGER NOT NULL,
    SURFACE_AREA INTEGER NOT NULL,
    CONSTRAINT CSTR_CONTINENT_NAME UNIQUE (NAME),
    CONSTRAINT PK_CONTINENT PRIMARY KEY (CODE)
);
```

```
CREATE TABLE COUNTRY_LANGUAGE (
    COUNTRY_CODE CHAR(3) NOT NULL,
    LANGUAGE_NAME CHAR(30) NOT NULL,
    IS_OFFICIAL CHAR(1) NOT NULL,
    PERCENTAGE DECIMAL(4,1) NOT NULL,
    CONSTRAINT PK_COUNTRYLANGUAGE
    PRIMARY KEY (COUNTRY_CODE, LANGUAGE_NAME)
);

CREATE TABLE ORGANIZATION (
    CODE VARCHAR(7) NOT NULL,
    NAME VARCHAR(50) NOT NULL,
    YEAR_FOUNDED INTEGER,
    CONSTRAINT PK_ORGANIZATION PRIMARY KEY (CODE)
);

CREATE TABLE COUNTRY_ORGANIZATION (
    COUNTRY_CODE CHAR(3) NOT NULL,
    ORGANIZATION_CODE VARCHAR(7) NOT NULL,
    CONSTRAINT PK_COUNTRY_ORGANIZATION
    PRIMARY KEY (COUNTRY_CODE, ORGANIZATION_CODE)
);

ALTER TABLE COUNTRY_ORGANIZATION
    ADD CONSTRAINT FK_COUNTRY_ORGANIZATION_COUNTRY
        FOREIGN KEY (COUNTRY_CODE)
    REFERENCES COUNTRY(CODE);

ALTER TABLE COUNTRY_ORGANIZATION
    ADD CONSTRAINT FK_COUNTRY_ORGANIZATION_ORGANIZATION
        FOREIGN KEY (ORGANIZATION_CODE)
    REFERENCES ORGANIZATION(CODE);

ALTER TABLE COUNTRY
    ADD CONSTRAINT FK_COUNTRY_CONTINENT
        FOREIGN KEY (CONTINENT)
    REFERENCES CONTINENT(NAME);

ALTER TABLE COUNTRY
    ADD CONSTRAINT FK_COUNTRY_CITY
        FOREIGN KEY (CAPITAL)
    REFERENCES CITY(ID);
```

```
ALTER TABLE CITY
    ADD CONSTRAINT FK_CITY_COUNTRY
        FOREIGN KEY (COUNTRY_CODE)
    REFERENCES COUNTRY(CODE);

ALTER TABLE COUNTRY_LANGUAGE
    ADD CONSTRAINT FK_COUNTRY_LANGUAGE_COUNTY
        FOREIGN KEY (COUNTRY_CODE)
    REFERENCES COUNTRY(CODE);

CREATE VIEW V_ORGANIZATION_SUMMARY
        (COUNTRY_NAME, ORGANIZATION_NAME, ORGANIZATION_CODE)
    AS
        SELECT CTRY.NAME, ORG.NAME, ORG.CODE
            FROM ORGANIZATION ORG
        JOIN COUNTRY_ORGANIZATION CO
            ON CO.ORGANIZATION_CODE = ORG.CODE
        JOIN COUNTRY CTRY
            ON CO.COUNTRY_CODE = CTRY.CODE
        ORDER BY CTRY.NAME;
```