Mn.IT @ DOT – Business Applications Services:

Java Software Development Standards

Version 6.1
February 1, 2016

Authored by: Rick Meyer, Information and Technology Services, Systems Architect
e-mail: Rick.Meyer@dot.state.mn.us
1. Introduction

This document describes the software development environment to be used for Java / Java EE applications and services. It follows the Information and Technology Services strategic technology recommendations for Java applications.

2. Technical Overview

The application architecture will be Java servlet deployed to Apache Tomcat.

The client is typically a Web browser. The primary browser targets are IE 11 and Chrome, though the application should run in all the popular browsers (IE, Chrome, Firefox, Safari). The User Interface (UI) technology should consist of HTML5, CSS3, and JavaScript only. The UI should be responsive, ensuring it programmatically adapts to mobile devices (small, touch-enabled screens).

The database is typically Oracle or Microsoft SQL Server.

Reports should be developed for Business Objects (Crystal) Enterprise. Reports should be decoupled from the application and follow a service-oriented architecture.
3. Development Environment

3.1 Application Technology

The application development / maintenance technology for Java applications is as follows:

1. Application architecture: **Java Enterprise Edition**
2. Application development language: **Oracle Java 1.7**
3. Scripting Language:
   - **JavaScript**
   - **PHP**
4. Presentation:
   - **JSF, JSP, JSTL, HTML, CSS, JavaScript**
5. Reports solution:
   - **Business Objects (Crystal) Enterprise XI**
6. Database:
   - **Oracle 11g, MS SQL**
7. Object / relational mapping:
   - **Hibernate 3 (optional)**
8. Controller framework:
   - **JSF (Apache MyFaces), Spring (optional)**
9. Application server:
   - **Tomcat 7**
10. Code version control system / repository:
    - **Git**
11. Build / deploy / dependency tool:
    - **Ant & Maven 3**
12. Integrated Development Environment (IDE):
    - **Netbeans, Eclipse IDE**
13. Modeling / design tools:
    - **Netbeans, Eclipse IDE Visio**
14. Primary programming tool:
    - **Netbeans, Eclipse IDE**
15. Primary testing tools:
    - **Netbeans, Eclipse IDE, JUnit, HTTP Unit**
16. Git interface tools:
    - **Netbeans, Eclipse IDE, command line**
17. Defect / Issue tracking tool:
    - **JIRA, RallyDev**
18. Logging tool:
    - **Log4J**
19. Target Browsers:
    - **IE 11, Chrome**

Java Servlet Application Technology Model:

![Java Servlet Application Technology Model](image-url)
3.2 Development Environment Specifications and Processes:
Mn/DOT’s Information and Technology Services has specifications for a Java servlet that software developers will follow. These include:

1. Coding Standards:
   Developers will follow coding style guidelines. Located at [https://google.github.io/styleguide/javaguide.html](https://google.github.io/styleguide/javaguide.html)
   Developers will use PMD to check coding compliance to best practices and will deliver a clean PMD report at the end of each iteration. PMD is located here: [https://pmd.github.io/](https://pmd.github.io/).

   PMD must be used by Java developers on a regular basis to identify poor coding practices.

   PMD scans Java source code and looks for potential problems like:
   - Possible bugs - empty try/catch/finally/switch statements
   - Dead code - unused local variables, parameters and private methods
   - Suboptimal code - wasteful String/StringBuffer usage
   - Overcomplicated expressions - unnecessary if statements, for loops that could be while loops
   - Duplicate code - copied/pasted code means copied/pasted bugs

   Guidelines for use:
   - Developers must run PMD at the end of each project iteration for their own benefit.
   - Developers must examine and fix all level 1 & level 2 violations, or explain why they should not be fixed.
   - Contractors must deliver a PMD report in Excel format to the project manager at the end of each project iteration. This report must be void of level 1 & level 2 violations, or an explanation should be written as to why the violation should be ignored.
2. **Presentation (GUI) Standards:**

Developers will use HTML5, CSS3, and JavaScript with JavaServer Pages (JSP) for service-side processing. OR OPTIONALLY use [Apache MyFaces](http://myfaces.apache.org) JavaServer Faces framework, the [Apache Tomahawk](http://tomahawk.apache.org) library, [PrimeFaces](http://primefaces.org) user interface components, and [Facelets](http://facelets.apache.org).

Developers will follow Mn/DOT GUI standards for consistent “look and feel” with other Mn/DOT Web applications. See screen shots at the end of this document. Mn/DOT's Template Application includes pages that meet GUI standards. It's best to start with these pages. See “Template Application” for more detail.

Developers will follow these Screen Numbering Guidelines:

Developers should follow this example. Group the application's functionality into major categories and follow the example below. If there are more than 5 major categories, the developer can use smaller series’ (2000 – 2499, 2500 – 2999, etc).

- **General screens – 1000 series**
  - 1000 - Welcome or index.html
  - 1010 – Login
  - 1020 – Common / Site-level search & search results

- **Major Functional Category #1 – 2000 series**
  - 2000 – Major Function General Info
  - 2100 – View / Edit General Info
  - 2200 – View / Edit Specific Info

- **Major Functional Category #2 – 3000 series**
  - 3000 – Major Function General Info
  - 3100 – View / Edit General Info
  - 3200 – View / Edit Specific Info

- **Major Functional Category ...**

- **Report Screens - 7000 series**
  - 7000 - Report

- **Error Screens - 8000 series**
  - 8000 - Error

- **Help & About Screens – 9000 series**
  - 9000 – Help
  - 9500 – About
3. **Versioning Standards:**

Developers will follow these versioning guidelines:

A `<Major>`.<`Minor`>.<`Patch`> numbering system is used to track software revisions when a new version is released.

A Major Release is a full product upgrade of the software containing significant new functionality. A Major Release is necessary if there are changes to the model or compatibility with previous versions cannot be maintained. When the Major Release version number is incremented, the Minor and Patch version numbers are reset to zero.

A Minor Release is a planned update to the existing software incorporating standard maintenance, improvements to existing features, enhancements and bug fixes. When a Minor Release number is incremented, the Major Release version number remains unchanged and the Patch Release version number is reset to zero.

A Patch Release is distributed when necessary to correct critical or significant problems that impact a customer’s use of the system. When the Patch Release version number is incremented, the Major and Minor Release version numbers remain unchanged.

4. **Template Application:**

Developers will begin a project with a ‘template application’, which consists of the technology listed in Section 3.1 and base configuration. This provides developers with a jump-start shell of an application. It helps ensure the developer starts within a Model/View/Controller framework and uses built-in patterns and best practices. Contact the application development group for a copy of the template app.

5. **Software build, dependency, and version management:**

Developers will use Ant, Maven, and Git for build, dependency, and version management. Developers will provide an Ant or Maven build script that builds the project without any IDE dependencies.
6. **Source Code Maintenance:**
Developers will provide and maintain application source files, configuration files, database schemas and scripts, developer documentation, and other related artifacts in MnDOT’s Git repository.

Follow these rules for naming Git projects:
One directory per application (including all variants of that application).

The top level directory name should be short (8 characters or less), all lower case letters and/or digits with no "-" (dash), and "_" (underscore), "." (period), or spaces. The directory name should be the same named used for all project artifact prefixes.

The directory name is short to facilitate naming of resources and artifacts where long user-friendly names would be problematic. For example, database table names are often prepended with the application’s short name. Projects should determine the application's short name (often an acronym) early in the project so project artifacts and configuration files can use this name.

7. **Documentation Tags (Javadoc):**
Developers will use Javadoc tags for documentation. Please follow Oracle's Javadoc reference

8. **Object / Relational Mapping (Hibernate):**
Developers may use Hibernate for Object / Relational Mapping. The use of Hibernate is optional.

9. **Code Portability:**
Developers will follow practices that insure code portability. Peer reviews and documented justification is required for the use of proprietary application server extensions or libraries.

10. **Deployment:**
Developers will follow a standard deployment process.

**Process for development server and test server application deployment:**
Use Anonymous FTP to place the deployment file (.war) at the following directory structure “/pub/incoming/apps/<appName>”. The server will auto-deploy the .war within a few minutes of it being placed in its respective directory.

**Process for production server application deployment:**
1. Use Anonymous FTP to place a copy of the deployment files (.war) and any deployment instructions at the following directory structure “/pub/incoming/apps/<appName>”.
2. Submit a Service Desk Ticket to the server administrator to deploy the production application.
11. Auditing:
Models and code will be reviewed for compliance to the specifications listed in Section 3.

12. Controller Standards (JSF, Spring Framework)
Developers will use JavaServer Pages (JSP) OR OPTIONALLY use Apache MyFaces JavaServer Faces framework. The use of Spring Framework is optional.

13. Java Application Server (Tomcat)
Tomcat is used to host Java servlet applications.

14. Application Security:
Servlet security should follow Tomcat best-practices. Applications will use Microsoft Active Directory for user authentication and role-based (AD Groups) authorization.

15. Scripting Standards
Developers will use JavaScript for client-side scripting, and PHP for server-side scripting.

The use of Javascript for client-side scripting should be based on the following criteria.

1. The script is not used as the only means of security or input validation.
2. The script must follow coding best-practices:
   - Re-usable
   - Well commented
   - Well organized
16. Naming Standards for Web applications

User-friendly name
- The user-friendly name is the descriptive or popular name that identifies the application. The user-friendly name should be chosen by the project sponsor or application owner.
- Projects should determine the application's user-friendly name early in the project so it can be used in documentation.

URL
- The URL is the application's Web address.
- Projects must determine the URL for the application. The URL is typically based on the user-friendly name, and must follow these examples:
  - Production URL = appname.dot.state.mn.us
  - Test URL = appnamet.dot.state.mn.us
  - Development URL = appnamed.dot.state.mn.us

SSL for secure communications
- Projects must determine whether the application will use SSL (Secure Sockets) for secure communication early in the project because this affects the naming of server resources and configuration.

Short-name
- The application short-name is used to name application and data resources and artifacts where long user-friendly names would be problematic. For example, database schema names are often prepended with the application short-name.
- Projects must determine the application's short-name (typically an acronym) early in the project so project artifacts and configuration files can use this name.
17. Issue / Bug Tracking (JIRA)
Developers will use JIRA for issue / bug tracking.

JIRA is located at http://jira.dot.state.mn.us. You must be connected to a MnDOT network for access.

18. Developer Team Collaboration (Wiki)
Developers can use a Wiki available for collaboration and documentation, located at http://wiki.dot.state.mn.us. You must be connected to a Mn/DOT network for access.

19. Logging Standards
Developers must use Log4J. Developers must create application-specific log files. Do not write logs to the server's log file.

20. SSL for secure communications
SSL is provided by a reverse-proxy server for all external HTTP communications. There is no coding or configuration required by the developer. See “Model Java EE Web application in Mn/DOT’s current infrastructure” diagram.
21. Reports Standards

Reports are created using Crystal Reports and hosted on the Business Objects Enterprise in a separate Web application. Report links must never be hard coded in the application. Instead, developer must use a Mn/DOT custom Tag Library included in the Template Application (archetype).

Application Dynamic Link to Reporting Environment

The Java application report's link points to the report's start page by dynamically retrieving the URL from the database. This feature has been added to the Mn/DOT tag library (mndot-taglib:reportsLink). During application compilation, the URL is queried from the database based on the application name and database information and imbedded in the application reports link.

Following these steps enable this feature:

1 - import the maven 2 dependency, or else get the appropriate jar file onto the classpath:
   `<dependency>
   <groupId>us.mn.state.dot</groupId>
   <artifactId>mndot-taglib</artifactId>
   <version>2.1.0</version>
   </dependency>

2 - put this line of code at the top of your file:
   `<%@ taglib uri="http://dot.state.mn.us/customTags/DbStyle" prefix="mndot" %>`

3 - put this line of code where you'd like the reports link to appear:
   Internal:
   `<mndot:reportsLink dataSource="sampleDS" appName="sample">Reports</mndot:reportsLink>`
   
   External:
   `<mndot:reportsLink dataSource="sampleDS" appName="sample" external="true">Reports</mndot:reportsLink>`

If you are using the Mn/DOT standard applicationTemplate Version 1.3.0 or above, these steps are already included for you. However, you need to alter the datasource file name and application name specified at the step 3.

GIS applications have session reports. These reports allow the user to view the data in a session, but as soon as the session is closed the report goes away. These session reports are part of the application code and are stored in Git. These reports are different than the reports that are used in most business applications.

The Crystal reports (Business Objects) that are developed for business applications and are binary.rpt format. These reports are stored in the Business Objects repository.
Any reports done for GIS applications or Business applications that use the .rpt format are stored in the Business Objects repository.
3.3 Hosting Environment:

1. Application hosting services are typically provided by Mn.IT@DOT Infrastructure Section. Applications are hosted by Tomcat on a Red Hat Enterprise Linux server. Application hosting typically resides in a State data center. The customer provides a Web site to act as the customer-facing entry point for the application. This site could be an existing Web site located inside or outside a State data center.

2. Database hosting services are typically provided by Information and Technology Services Infrastructure Section. Databases reside on Oracle RAC or Microsoft SQL Server. Database hosting typically resides in a State data center.

3. Reports hosting services are typically provided by Information and Technology Services Infrastructure Section. The reporting solution is typically Business Objects (Crystal) Enterprise. Reports hosting typically resides in a State data center.

3.4 Development / Test Environment

Unit testing is done on development servers. Integration testing, system testing and user acceptance testing is done on test servers that reside in a test environment.

3.5 Production Environment

Production instances are deployed on production servers that reside in the production environment.
Model Java EE Web application in Mn/DOT's current infrastructure.
Standard Web Application Template (screen shots)

Login page:
Typical application screen layout:

Note: The yellow background in the upper-right corner will turn white when the app moves into production.
Application screen showing Reports link