CallCenterSolution 説明資料 2017.09

本資料はコールセンター支援システムβ版のオペレータサイト（CallCenterSolution）の構成と開発環境の作成方法について説明するものです。

なお、資料内の手順等は現時点のものであり、修正／変更する場合がありますのでご了承ください。

目　次

[1 概要 2](#_Toc494212377)

[2 構成 2](#_Toc494212378)

[2.1 アプリケーション構成 2](#_Toc494212379)

[2.2 ディレクトリ構成（概要） 3](#_Toc494212380)

[3 開発環境構築 6](#_Toc494212381)

[3.1 Java 6](#_Toc494212382)

[3.2 Tomcat 6](#_Toc494212383)

[3.3 Eclipse 6](#_Toc494212384)

[3.4 MySQL 6](#_Toc494212385)

[3.5 Eclipse プロジェクト作成 6](#_Toc494212386)

[3.6 WebStreamTalkAnalyze を取り込み 6](#_Toc494212387)

[3.7 Eclipse 上でのサーバ作成～Tomcat 起動 7](#_Toc494212388)

# 概要と要望

* + - maven は利用していません。（JRE と Tomcat に含まれるものを除いて、全ての jar は war/WEB-INF/lib に直接配置します。）
    - 動作確認は Eclipse を利用して実施しています。
    - リリース時には ant を使用して war ファイルを作成し、リリース先の Tomcat に配置しています。

# 構成

## アプリケーション構成

* + - Springframework（DIコンテナ）
    - Hibernate／MyBatis（O-R Mapper）
    - RESTEasy／Jackson2（REST フレームワーク／JSON パーサ）
    - 1 画面ごとに 1 html の構成（表示データは Ajax 通信でサーバから取得）
    - サーバの I/F は全て REST API

フレームワーク構成

（アプリケーション）

RESTEasy

Hibernate

MyBatis

Spring

処理フロー

クライアント

サーバ

PC

html

jQuery

http

*RestEasy*

*MyBatis*

table

table

spring bean

*Tx 境界*

*Tx 境界*

*Hibernate*

*Tx 境界*

*JDBC*

DB

～.hbm

.xml

～.hbm

.xml

Hibernate による

O/R Mapping

1 テーブルに

* 1 Entity クラス
* 1 Dao クラス

## ディレクトリ構成（概要）

CallCenterSolution

├─DDL

│ schema-create.sql テーブル生成用クエリ

│ schema-drop.sql

├─src

│ │ application.properties

│ │ jdbc.properties DB 接続情報を記述

│ │ log4j.xml

│ ├─cloud

│ │ └─optim

│ │ └─callcentersolution

│ │ ├─api

│ │ │ ├─app

│ │ │ │ ├─calllog 通話ログ TBL 操作用の機能一式

│ │ │ │ │ CallLog-selectItem.context.xml

│ │ │ │ │ CallLog.sqlmap.xml

│ │ │ │ │ CallLogMapper.java

│ │ │ │ │ CallLogRequest.java

│ │ │ │ │ CallLogResponse.java

│ │ │ │ │ CallLogRestService.java REST の入り口

│ │ │ │ │ CallLogRestValidator.java

│ │ │ │ │ CallLogSearchForm.java

│ │ │ │ │ CallLogSearchResult.java

│ │ │ │ │ CallLogService.java

│ │ │ │ │ SearchForm.java

│ │ │ │ │ SearchResult.java

│ │ │ │ │

│ │ │ │ ├─keyword キーワード API 機能一式（calllogと同様）

│ │ │ │ ├─knowledge ナレッジ TBL 操作用の機能一式（calllogと同様）

│ │ │ │ └─tag タグ TBL 操作用の機能一式（calllogと同様）

│ │ │ │

│ │ │ ├─entity O-R マッピング用のクラス／設定ファイル群

│ │ │ └─util

│ │ │ ExtractUtil.java

│ │ │ KuromojiAnalyzer.java KUROMOJI 実装

│ │ │ MorphologicalAnalyzer.java 形態素解析 I/F

│ │ └─core 各種共通処理／共通クラス

│ │ ├─common

│ │ │ │ DaoException.java

│ │ │ ├─fileupload

│ │ │ │ FileRegistHandler.java

│ │ │ ├─listener

│ │ │ │ HttpSessionListenerImpl.java

│ │ │ │ ServletContextListenerImpl.java

│ │ │ │ ShutdownListener.java

│ │ │ ├─log4j

│ │ │ │ DailySizeRollingFileAppender.java

│ │ │ └─utility

│ │ │ FileHelper.java

│ │ │ HankakuKanaConverter.java

│ │ │ IPAddressChecker.java

│ │ │ Network.java

│ │ │ QueryHelper.java

│ │ │ SystemUniqNo.java

│ │ │ ToStringHelper.java

│ │ │

CallCenterSolution

├─src

│ ├─cloud

│ │ └─optim

│ │ └─callcentersolution

│ │ └─core

│ │ └─modules

│ │ ├─filter ServletFilter 群

│ │ │ MappedDiagnosticContextLogFilter.java

│ │ │ NestedDiagnosticContextLogFilter.java

│ │ │ UserInfoFilter.java

│ │ ├─image

│ │ │ ImageResizeUtility.java

│ │ │ ResizeOption.java

│ │ ├─loginutil

│ │ │ CustomUser.java

│ │ │ CustomUserHolder.java

│ │ │ LoginUtility.java

│ │ │ PasswordUtility.java

│ │ ├─rest REST 用の共通処理／共通クラス

│ │ │ │ CustomExceptionMapper.java

│ │ │ │ CustomJsonProvider.java

│ │ │ │ ExceptionUtil.java

│ │ │ │ MessageUtility.java

│ │ │ │ ObjectMapperFactory.java

│ │ │ │ RequestStatusHolder.java

│ │ │ │ ResponseCode.java

│ │ │ │ RestException.java

│ │ │ │ RestLog.java

│ │ │ │ RestResult.java

│ │ │ │ RestValidatorUtils.java

│ │ │ │ SortForm.java

│ │ │ └─app

│ │ │ ├─fileupload

│ │ │ │ FileUploadResponse.java

│ │ │ │ FileUploadRestService.java

│ │ │ │ FileUploadRestValidator.java

│ │ │ │ FileUploadUtility.java

│ │ │ │ ThumbnailUtility.java

│ │ │ └─image

│ │ │ BinaryImageRequest.java

│ │ │ BinaryImageRestService.java

│ │ │ BinaryImageRestValidator.java

│ │ └─validator

│ │ ValidatorUtility.java

│ │ ValidatorUtils.java

│ └─resource

│ message.properties

CallCenterSolution

└─war 公開ディレクトリ（概ね、この下のファイルを war としてリリースします）

├─ui

│ │ admin.html 管理画面

│ │ edit.html 登録画面

│ │ error.html エラー画面

│ │ operation.html オペレーション画面

│ ├─app

│ │ │ admin.js 管理画面処理

│ │ │ ccs.css

│ │ │ edit.js 登録画面処理

│ │ │ layout.js 全画面のヘッダ処理

│ │ │ operation.js オペレーション画面処理

│ │ ├─common 共通処理

│ │ │ api.js

│ │ │ auth.js

│ │ │ ccs.js

│ │ │ prop.js

│ │ │ user.js

│ │ │ util.js

│ │ │ view.js

│ │ └─lib ライブラリ

│ │ └─optim

│ │ analyze\_communicator.js

│ │ communicator\_base.js

│ │ digest\_communicator.js

│ │ mic.js

│ │ wav.js

│ ├─css （デザイン CSS）

│ ├─fonts （デザインフォント）

│ ├─img （デザイン画像）

│ └─js

│ └─libs （デザインスクリプト）

└─WEB-INF

│ web.xml

├─classes

├─conf

│ application-base.context.xml Springframework の設定

│ sqlmap-config.xml MyBatis の設定

└─lib Java ライブラリ群

# 開発環境構築

## Java

Java 8 以上の SDK をインストールします。

* Eclipse 付属の JRE を利用するとコンパイルまたは実行時にエラー（「非推奨のメソッドを使用しています」）が発生する場合があるようです。その場合は、単独のディストリビューションを使用してください。

## Tomcat

Tomcat 8.5 （Servlet3.1）をインストールします。

## Eclipse

Eclipseをインストールします（Oxygen）。「動的 Web プロジェクト」を作成するため、JEE または All In Oneが必要です。

## MySQL

※データ登録を行わない場合、この手順は不要です。

5.7 をインストールします。以下の設定を行います。

* + - スキーマ（データベース）作成：callcentersolution  
      character set はutf8mb4 を指定（Collator は utf8mb4\_unicode\_ci）
    - ユーザ作成：ccs\_user（パスワード password）
    - character set は my.ini で全体的に utf8mb4 を指定（Collator は utf8mb4\_unicode\_ci）

## Eclipse プロジェクト作成

プロジェクトを Eclipse に取り込む場合は、「動的 Web プロジェクト」として作成してください。  
プロジェクト作成ウィザード中に注意が必要な項目は以下の通りです。

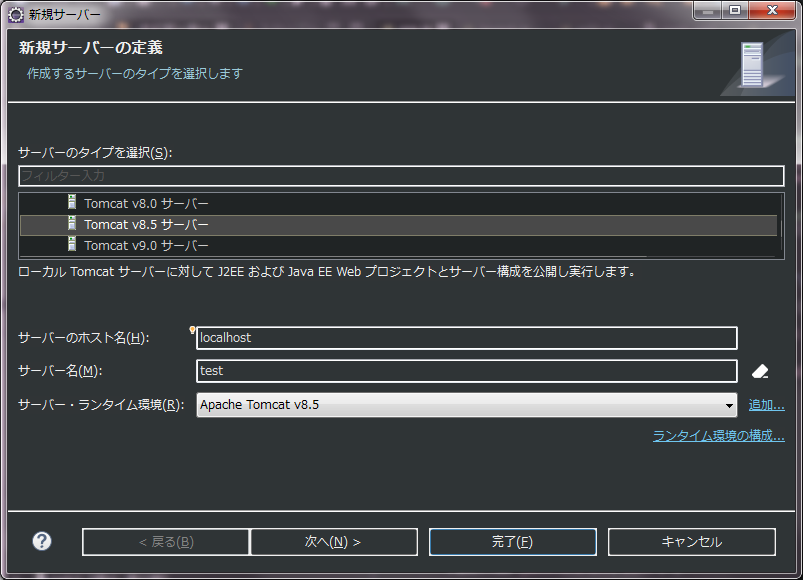
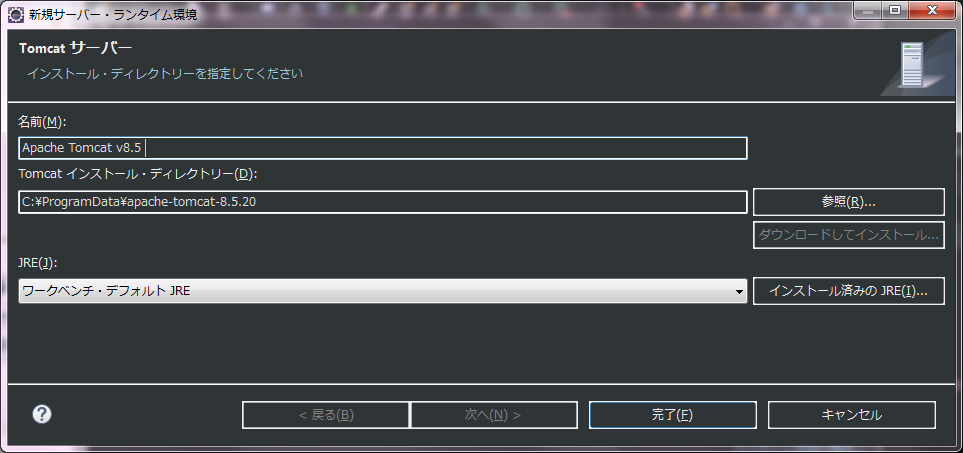
* ターゲット・ランタイム ： Apache Tomcat v8.5
* 動的 web モジュールバージョン ： 3.1
* デフォルト出力フォルダー ： war/WEB-INF/classes（デフォルトは「build\classes」なので変更する）
* コンテンツ・ディレクトリー ： war（デフォルトは「WebContent」なので変更する）

## WebStreamTalkAnalyze を取り込み

既存のプロジェクトとしてインポートします。

## Eclipse 上でのサーバ作成～Tomcat 起動

※図中の「WebAITextAnalyze」は、WebStreamTalkAnalyzeに読み替えて下さい。

* + 1. 「ウィンドウ」－「ビューの表示」から「サーバー」ビューを表示
    2. サーバービュー上で右クリックして「新規」－「サーバー」を選択し、新規サーバーウィンドウを表示
    3. 以下のように入力して「次へ」を押下  
       サーバーのタイプ：Tomcat v8.5 サーバー  
       サーバーのホスト名：localhost  
       サーバー名：（任意）  
       サーバー・ランタイム環境：Apache Tomcat v8.5
    4. 
       1. サーバー・ランタイム環境に Tomcat 8.5 が表示されない場合は「追加...」リンクを押下して新規サーバー・ランタイム環境ウィンドウを表示
       2. Tomcat インストール・ディレクトリーを設定して「完了」を押下
    5. 
    6. 「使用可能」リストに CallCenterSolution と WebStreamTalkAnalyzeが表示されるので、両方選択して「追加」を押下（CallCenterSolution と WebStreamTalkAnalyzeが「構成済み」リストに移動）
    7. 
    8. 
    9. 「完了」を押下すると、「サーバー」ビューに作成したサーバが表示されます。
       - 右クリック－「開始」で、アプリが起動します。
       - ダブルクリックすると、サーバの設定を変更できます。起動時にタイムアウトするようなら「タイムアウト」の項目を調節してください。  
         