**Macchinettaオンライン版**

**機能検証実施手順書**

(第1.1.1版)

2015/08/05

日本電信電話株式会社

**変更履歴**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 項番 | 版数 | 制定日/改訂日 | 変更者 | 変更箇所 | 変更内容 |
|  | 1.0 | 2014/02/28 | - | - | 初版作成 |
|  | 1.0 | 2014/03/28 |  | 2.1.4日本語環境が存在しない場合 | localeにjaがあれば必要ないという注意書きを削除  (なくても動作するが、VNCで接続した場合に文字化けするため) |
|  | 1.0 | 2014/03/28 |  | 3.2Firefox | 自動更新停止の方法を削除  (自動更新機能はtar.gzを解凍した場合にはないため) |
|  | 1.0 | 2014/03/28 |  | 3.2.2インストール方法 | chownを削除 |
|  | 1.0 | 2014/03/28 |  | 5.1対象プロジェクト一覧 | 「spring-functionaltest-test-tomcat-+postgres」となっている箇所を「spring-functionaltest-test-tomcat+postgres」に修正 |
|  | 1.0 | 2014/04/02 |  | 3.3PostgreSQL | 不要なパーミッション変更を削除 |
|  | 1.0 | 2014/04/02 |  | 3.4Tomcat | 所有者、パーミッション変更の記述追記 |
|  | 1.0 | 2014/04/02 |  | 3.4.3.5使用メモリ/ログディレクトリの設定 | testATHR0101003でエラーになる場合の対処方法を追記 |
|  | 1.0 | 2014/04/02 |  | 4.1.2配置方法 | ファイル名をspring-functionaltest.zipから  spring-functionaltest-linux.zipへ変更。 |
|  | 1.0 | 2014/04/02 |  | 5.3.5spring-functionaltest-initdb-postgresql  5.3.6spring-functionaltest-release-tomcat+postgres | JenkinsJOBの設定値を修正  ビルドのパラメータのVERSIONは不要なため削除 |
|  | 1.0 | 2014/04/02 |  | 5.3.7spring-functionaltest-selenium-tomcat+postgres  5.3.8spring-functionaltest-test-tomcat+postgres | 不要なパラメータであるTARGET\_CLASSを削除 |
|  | 1.0 | 2014/04/02 |  | 5.3.7spring-functionaltest-selenium-tomcat+postgres  5.3.9spring-functionaltest-selenium-tomcat+postgres-target | 実行速度向上のための  selenium.enablePageSource=false  をプロパティに追記 |
|  | 1.0 | 2014/04/03 |  | 1.2.4推奨構成 | 参照ファイル名誤りを修正 |
|  | 1.0 | 2014/04/03 |  | 1.2.5使用ソフトウェア | tomcatのリンク切れURLを修正 |
|  | 1.0 | 2014/04/03 |  | 3.3.2インストール方法 | 実行ディレクトリを任意ではなく、rpmファイルのあるディレクトリに変更 |
|  | 1.0 | 2014/04/03 |  | 3.3.3.3パスワード変更 | 不要なパスワード入力を削除 |
|  | 1.0 | 2014/04/03 |  | 3.3.3.5認証方法変更 | IPアドレスが実値であるのを変更箇所の記述に変更  PostgreSQLの再実行要であることを追記 |
|  | 1.0 | 2014/04/03 |  | 3.4.3.4管理者画面開設設定用 | 単に追加となってため、「tomcat-usersタグ内に」追加を明記 |
|  | 1.0 | 2014/04/03 |  | 3.4.3.6(補足)試験に必要なオプションの設定 | 3.4.3.5.使用メモリ及び試験に必要なオプションの設定を必須の使用メモリの設定と試験に必要なオプションに分割 |
|  | 1.0 | 2014/04/03 |  | 3.7.3.1起動停止シェル作成 | シェルのstop部分のgrep -vのハイフンが文字化けしていたのを修正 |
|  | 1.1.0 | 2015/03/13 |  | 第1章はじめに | 本ドキュメントが更新していくため、内容理解に影響がない場合、キャプチャは最新ではない場合があることを追記 |
|  | 1.1.0 | 2015/03/13 |  | 1.1.3本ドキュメントでのサーバ | IPアドレス変更 |
|  | 1.1.0 | 2015/03/13 |  | 1.2.3検証構成 | 検証をVMWare PlayerからVirtualBoxへ変更、CentOSを6.4から6.5に変更、HDDを20Gから10Gに変更 |
|  | 1.1.0 | 2015/03/13 |  | 1.2.4推奨構成 | エラー時参照のファイル名を変更 |
|  | 1.1.0 | 2015/03/13 |  | 1.2.5使用ソフトウェア | 以下の使用ソフトウェアをバージョンアップ  Java：1.7.0\_72  Tomcat：7.0.56  PostgreSQL：9.3  Jenkins：1.599  Firefox:31.3.0  Maven:3.2.3  バージョンの記述してある箇所の変更 |
|  | 1.1.0 | 2015/03/13 |  | 2.1.4日本語環境が存在しない場合 | 他のyumコマンド同様に「-y」オプション追加 |
|  | 1.1.0 | 2015/03/13 |  | 2.4.1権限付与 | sudoのコマンドを追加 |
|  | 1.1.0 | 2015/03/13 |  | 3.3.2インストール方法  3.6.2インストール方法 | 他のソフトウェアのインストール方法と同様  ダウンロードしたファイルの削除手順を追加  ・rpmパッケージ削除 |
|  | 1.1.0 | 2015/03/13 |  | 3.4.2インストール方法 | Tomcatのホームディレクトリにシンボリックリンクを追加 |
|  | 1.1.0 | 2015/03/13 |  | 3.4.3.5使用メモリ/ログディレクトリの設定 | ログ出力ディレクトリの環境変数と作成を追加 |
|  | 1.1.0 | 2015/03/13 |  | 3.4.3.8JDBCドライバの配置 | 使用ソフトウェア（PostgreSQL）のバージョンアップにより  「postgresql-9.2-1004.jdbc4.jar」から「postgresql-9.3-1102.jdbc41.jar」へ修正 |
|  | 1.1.0 | 2015/03/13 |  | 3.6.2インストール方法 | 不要であった"Desktop"のインストールを削除 |
|  | 1.1.0 | 2015/03/13 |  | 3.7.2インストール方法 | 誤記「jankins.war」を「jenkins.war」へ修正 |
|  | 1.1.0 | 2015/03/13 |  | 3.7.3.4perl-URのインストール | deploy時の使用で追加したため、インストールを追加 |
|  | 1.1.0 | 2015/03/13 |  | 3.7.3.10.1Xvncプラグインの設定 | 画面のサイズを800x600から1024x768に変更 |
|  | 1.1.0 | 2015/03/13 |  | 5.3.2spring-functionaltest-env | 「ビルド」－「シェル実行」の設定内容に「cp -r /home/spring/spring-functionaltest/spring-functionaltest-env/3rdparty ${WORKSPACE}/.」を追記 |
|  | 1.1.0 | 2015/03/13 |  | 5.3.6spring-functionaltest-release-tomcat+postgres | シェルの実行で不要なアンデプロイを削除、デプロイ方法を変更 |
|  | 1.1.0 | 2015/03/13 |  | 表 5.3‑1 指定可能テストケース名一覧 | CommonLibraryValidationTest,DataAccessMyBatis3AddTest,SpringSecurityTestの追加 |
|  | 1.1.0 | 2015/03/13 |  | 5.4.1実行順序 | spring-functionaltest-parentから呼び出されるJOBの不足を追加。(spring-functionaltest-env) |
|  | 1.1.0 | 2015/03/13 |  | 第5章Jenkinsの設定 | ビルドのパラメータのVERSIONを、1.0.0-SNAPSHOTから1.0.0.RELEASEに変更  Jenkinsのバージョンを変更したことによるキャプチャの張替 |
|  | 1.1.0 | 2015/03/13 |  | 5.5.2エラーについての結果確認方法 | 追加　以前もエラーの確認については記述していたが、正常と異常の場合が入り組んでいてわかりづらかったため、改善。 |
|  | 1.1.0 | 2015/03/13 |  | 全体 | 不要なバスのetc/rc.d/init.dのrc.dを削除 |
|  | 1.1.1 | 2015/08/05 |  | 3.3.3.4データベース作成 | 試験内容の変更に伴い、作成するデータベースを変更。  削除したデータベース作成  ・spring-functionaltest-admin  ・spring-functionaltest-staff  ・spring-functionaltest-user  追加したデータベース作成  ・spring-functionaltest-open  ・spring-functionaltest-close |
|  | 1.1.1 | 2015/08/05 |  | 5.3.5spring-functionaltest-initdb-postgresql | データベースの変更により初期化しているmavenの呼び出しも変更。  削除したmavenの呼び出し（ゴールのみ記述）  ・sql:execute -P local-postgres-admin  ・sql:execute -P local-postgres-staff  ・sql:execute -P local-postgres-user  追加したmavenの呼び出し（ゴールのみ記述）  ・sql:execute -P local-postgres-open  ・sql:execute -P local-postgres-close |

[第1章. はじめに 1](#_Toc413850535)

[1.1. 本ドキュメントの位置づけ 2](#_Toc413850536)

[1.1.1. 本ドキュメントの目的 2](#_Toc413850537)

[1.1.2. 本ドキュメントの記述方針 2](#_Toc413850538)

[1.1.3. 本ドキュメントでのサーバ 2](#_Toc413850539)

[1.2. 前提 3](#_Toc413850540)

[1.2.1. 試験環境構成 3](#_Toc413850541)

[1.2.2. ソフトウェアの使用ポート番号 4](#_Toc413850542)

[1.2.3. 検証構成 4](#_Toc413850543)

[1.2.4. 推奨構成 4](#_Toc413850544)

[1.2.5. 使用ソフトウェア 5](#_Toc413850545)

[第2章. OSの設定 6](#_Toc413850546)

[2.1. 日本語環境の設定 7](#_Toc413850547)

[2.1.1. ロケール/言語/文字コード 7](#_Toc413850548)

[2.1.2. 時刻設定 7](#_Toc413850549)

[2.1.3. キーボードの設定 8](#_Toc413850550)

[2.1.4. 日本語環境が存在しない場合 8](#_Toc413850551)

[2.1.5. 確認 8](#_Toc413850552)

[2.2. OSグループの作成 9](#_Toc413850553)

[2.2.1. グループの作成 9](#_Toc413850554)

[2.3. OSユーザの作成 10](#_Toc413850555)

[2.3.1. ユーザの作成 10](#_Toc413850556)

[2.4. グループでのsudo実行権限の付与 11](#_Toc413850557)

[2.4.1. 権限付与 11](#_Toc413850558)

[2.5. ポート接続設定 12](#_Toc413850559)

[2.5.1. Firewallの無効化 12](#_Toc413850560)

[2.6. Proxy接続設定 13](#_Toc413850561)

[第3章. ソフトウェア類のカスタマイズ 14](#_Toc413850562)

[3.1. Java 15](#_Toc413850563)

[3.1.1. インストール先 15](#_Toc413850564)

[3.1.2. インストール方法 15](#_Toc413850565)

[3.1.3. 設定 15](#_Toc413850566)

[3.1.3.1. 優先度向上 15](#_Toc413850567)

[3.1.3.2. 環境変数追加 17](#_Toc413850568)

[3.2. Firefox 18](#_Toc413850569)

[3.2.1. インストール先 18](#_Toc413850570)

[3.2.2. インストール方法 18](#_Toc413850571)

[3.2.3. 設定 18](#_Toc413850572)

[3.2.3.1. 環境変数追加 18](#_Toc413850573)

[3.3. PostgreSQL 19](#_Toc413850574)

[3.3.1. インストール先 19](#_Toc413850575)

[3.3.2. インストール方法 19](#_Toc413850576)

[3.3.3. 設定 20](#_Toc413850577)

[3.3.3.1. 環境変数追加 20](#_Toc413850578)

[3.3.3.2. 自動起動設定 20](#_Toc413850579)

[3.3.3.3. パスワード変更 21](#_Toc413850580)

[3.3.3.4. データベース作成 21](#_Toc413850581)

[3.3.3.5. 認証方法変更 22](#_Toc413850582)

[3.4. Tomcat 23](#_Toc413850583)

[3.4.1. インストール先 23](#_Toc413850584)

[3.4.2. インストール方法 23](#_Toc413850585)

[3.4.3. 設定 24](#_Toc413850586)

[3.4.3.1. 環境変数追加 24](#_Toc413850587)

[3.4.3.2. 起動停止シェル作成 24](#_Toc413850588)

[3.4.3.3. URIEncoding変更 27](#_Toc413850589)

[3.4.3.4. 管理者画面開設設定用 27](#_Toc413850590)

[3.4.3.5. 使用メモリ/ログディレクトリの設定 28](#_Toc413850591)

[3.4.3.6. (補足)試験に必要なオプションの設定 29](#_Toc413850592)

[3.4.3.7. 自動起動設定 30](#_Toc413850593)

[3.4.3.8. JDBCドライバの配置 30](#_Toc413850594)

[3.4.3.9. Tomcat起動 31](#_Toc413850595)

[3.5. Maven 32](#_Toc413850596)

[3.5.1. インストール先 32](#_Toc413850597)

[3.5.2. インストール方法 32](#_Toc413850598)

[3.5.3. 設定 32](#_Toc413850599)

[3.5.3.1. 環境変数追加 32](#_Toc413850600)

[3.5.3.2. バージョン確認 33](#_Toc413850601)

[3.5.3.3. Proxyを使用する場合 33](#_Toc413850602)

[3.6. vncserver 34](#_Toc413850603)

[3.6.1. インストール先 34](#_Toc413850604)

[3.6.2. インストール方法 34](#_Toc413850605)

[3.6.3. 設定 34](#_Toc413850606)

[3.7. Jenkins 36](#_Toc413850607)

[3.7.1. インストール先 36](#_Toc413850608)

[3.7.2. インストール方法 36](#_Toc413850609)

[3.7.3. 設定 36](#_Toc413850610)

[3.7.3.1. 環境変数追加 36](#_Toc413850611)

[3.7.3.1. 起動停止シェル作成 38](#_Toc413850612)

[3.7.3.2. 自動起動設定 39](#_Toc413850613)

[3.7.3.3. Jenkins起動 40](#_Toc413850614)

[3.7.3.4. perl-URのインストール 40](#_Toc413850615)

[3.7.3.5. プラグイン導入 40](#_Toc413850616)

[3.7.3.6. ユーザ管理 41](#_Toc413850617)

[3.7.3.6.1. ユーザ管理方法の設定 41](#_Toc413850618)

[3.7.3.6.2. ユーザ作成 42](#_Toc413850619)

[3.7.3.6.3. 権限の付与 42](#_Toc413850620)

[3.7.3.7. JAVA\_HOMEの設定 44](#_Toc413850621)

[3.7.3.8. Mavenの設定 45](#_Toc413850622)

[3.7.3.9. Jenkinsの位置の設定 46](#_Toc413850623)

[3.7.3.10. プラグインの設定 47](#_Toc413850624)

[3.7.3.10.1. Xvncプラグインの設定 47](#_Toc413850625)

[第4章. テスト対象資材の配置 48](#_Toc413850626)

[4.1. 配置場所・配置方法 49](#_Toc413850627)

[4.1.1. 配置場所 49](#_Toc413850628)

[4.1.2. 配置方法 49](#_Toc413850629)

[第5章. Jenkinsの設定 50](#_Toc413850630)

[5.1. 対象プロジェクト一覧 51](#_Toc413850631)

[5.2. JOBの作成 52](#_Toc413850632)

[5.3. プロジェクト毎の設定 54](#_Toc413850633)

[5.3.1. spring-functionaltest-parent 54](#_Toc413850634)

[5.3.2. spring-functionaltest-env 57](#_Toc413850635)

[5.3.3. spring-functionaltest-domain 58](#_Toc413850636)

[5.3.4. spring-functionaltest-web 59](#_Toc413850637)

[5.3.5. spring-functionaltest-initdb-postgresql 60](#_Toc413850638)

[5.3.6. spring-functionaltest-release-tomcat+postgres 61](#_Toc413850639)

[5.3.7. spring-functionaltest-selenium-tomcat+postgres 65](#_Toc413850640)

[5.3.8. spring-functionaltest-test-tomcat+postgres 67](#_Toc413850641)

[5.3.9. spring-functionaltest-selenium-tomcat+postgres-target 69](#_Toc413850642)

[5.3.10. spring-functionaltest-test-tomcat+postgres-target 70](#_Toc413850643)

[5.4. Jenkinsの実行 72](#_Toc413850644)

[5.4.1. 実行順序 72](#_Toc413850645)

[5.4.2. 実行方法 73](#_Toc413850646)

[5.5. Jenkinsの結果確認 74](#_Toc413850647)

[5.5.1. 結果確認方法 74](#_Toc413850648)

[5.5.1.1. spring-functionaltest-parent 74](#_Toc413850649)

[5.5.1.2. spring-functionaltest-test-tomcat+postgres 78](#_Toc413850650)

[5.5.2. エラーについての結果確認方法 81](#_Toc413850651)

[5.5.2.1. テスト結果の確認 81](#_Toc413850652)

[5.5.2.2. エビデンスの確認 84](#_Toc413850653)

# はじめに

本書の位置づけと本書の構成について記載する。

本ドキュメント内で使用しているキャプチャ画面に表示されているバージョンは本ドキュメントを更新していく過程で内容理解に影響が無い場合は、最新になっていない場合があります。

## 本ドキュメントの位置づけ

### 本ドキュメントの目的

本ドキュメントは、AP動作環境検証プログラムの試験環境の構築方法を明確にする。

### 本ドキュメントの記述方針

本ドキュメントの記述内容は、ソフトウェア類などのデフォルト設定以外について、記載する。

テキストボックス内はSSHを使用し、実行ユーザ・ディレクトリは各手順に記載している。

本ドキュメントでは、簡易にサーバ構築することを目的としているため、パスワードやパーミッションについては、わかりやすく簡易にしている。本番での構築をする場合はセキュリティを考慮すること。

### 本ドキュメントでのサーバ

本ドキュメントで構築しているサーバのIPアドレスは「10.0.2.2」である。VMをポートフォワードしているため、記述上、localhostを指定している場合がある。

## 前提

### 試験環境構成

検証手順の確認は、クライアントとサーバを1台のマシンで構成した環境で行った。

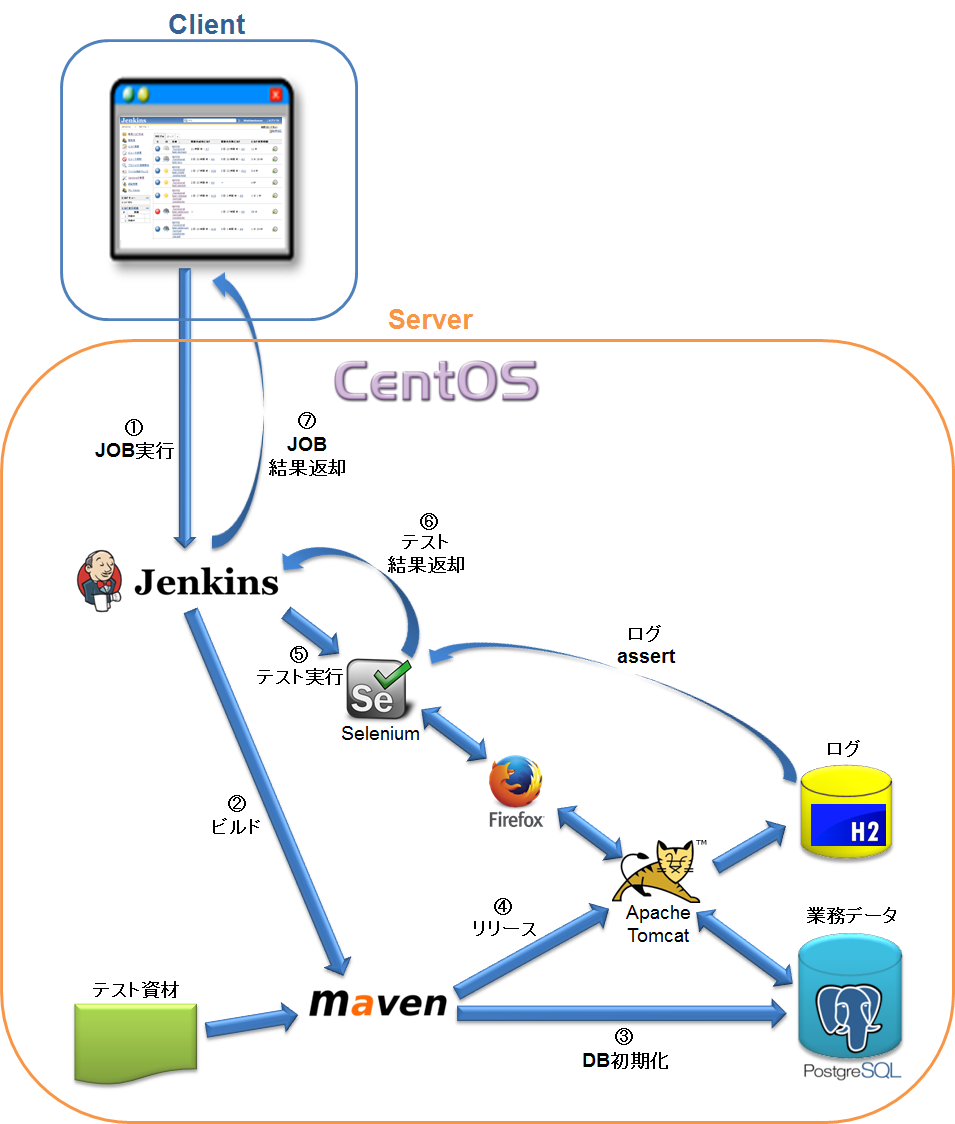


図 1.2‑1 試験環境構成

### ソフトウェアの使用ポート番号

|  |  |
| --- | --- |
| 種類 | ポート番号 |
| Tomcat | 8080 |
| Jenkins | 8090 |
| PostgreSQL | 5432 |

### 検証構成

検証手順はPC(windows 7 Enterprise 64bit)上に構築した次に示す仮想マシン上で確認した。

表 1.2‑1 仮想マシン

|  |  |
| --- | --- |
| 種類 | 詳細 |
| 仮想ソフト | VirtualBox 4.3.12 |
| CPU | Intel Core i3-2100 3.10GHz |
| メモリ | 2.0G |
| HDD | 10G |
| OS | CentOS 6.4 64bit |

### 推奨構成

表 1.2‑2 推奨マシン

|  |  |
| --- | --- |
| 種類 | 詳細 |
| CPU | 2コア、2.5GHz以上 |
| メモリ | 2.0G以上 |
| HDD | 10G以上 |
| OS | CentOS 6.4 64bit |

* 上記、スペックを満たさない場合、またはVMが動作するのに十分な物理マシンでは無い場合、タイミングの問題が発生する場合がある。  
  その場合は、「別紙\_機能検証エラー発生時の対処方法について.pdf/pptx」を参照し、追加パラメータを変更すること。

### 使用ソフトウェア

検証手順の確認に使用したソフトウェアのバージョンは以下の通りである。

表 1.2‑3 検証手順の使用ソフトウェアバージョン

|  |  |
| --- | --- |
| ソフトウェア | バージョン |
| Java | 1.7.0\_72 |
| Tomcat | 7.0.56 |
| PostgreSQL | 9.3 |
| Jenkins | 1.599 |
| Firefox | 31.3.0 |
| Maven | 3.2.3 |
| vncserver | tigervnc-server-1.1.0-5.el6\_4.1.x86\_64 |

〔参考〕

* Java

<http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html#jdk-7u72-oth-JPR>

* バージョンを指定してWindowsのブラウザでダウンロードするとデフォルトのファイル名が「jdk-7u72-linux-x64.gz」となるが、中身はtar.gzである。
* Tomcat

[http://archive.apache.org/dist/tomcat/tomcat-7/v7.0.56/bin/apache-tomcat-7.0.56.tar.gz](http://archive.apache.org/dist/tomcat/tomcat-7/v7.0.50/bin/apache-tomcat-7.0.50.tar.gz)

* Jenkins

<http://mirrors.jenkins-ci.org/war/latest/jenkins.war>

* Firefox

<http://ftp.mozilla.org/pub/mozilla.org/firefox/releases/31.3.0esr/linux-x86_64/ja/firefox-31.3.0esr.tar.bz2>

* Maven

<http://repo1.maven.org/maven2/org/apache/maven/apache-maven/3.2.3/apache-maven-3.2.3-bin.tar.gz>

* バージョンを変えて実行する場合は、バージョン部分を読み変えて検証環境構築すること。
* また、エラーが起きた際は、検証済のバージョンでの実施確認をして問題の切り分けをすることを推奨する。

# OSの設定

OSユーザの作成例を記載する。動作すれば、グループ、ユーザの作成は指定の通りでなくても良い。

## 日本語環境の設定

### ロケール/言語/文字コード

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/sysconfig/i18n |

〔結果ファイル〕

|  |
| --- |
| LANG=jp\_JP.utf8  LC\_CTYPE="ja\_JP.utf8"  LC\_NUMERIC="ja\_JP.utf8"  LC\_TIME="ja\_JP.utf8"  LC\_COLLATE="ja\_JP.utf8"  LC\_MONETARY="ja\_JP.utf8"  LC\_MESSAGES="ja\_JP.utf8"  LC\_PAPER="ja\_JP.utf8"  LC\_NAME="ja\_JP.utf8"  LC\_ADDRESS="ja\_JP.utf8"  LC\_TELEPHONE="ja\_JP.utf8"  LC\_MEASUREMENT="ja\_JP.utf8"  LC\_IDENTIFICATION="ja\_JP.utf8"  LC\_ALL= |

### 時刻設定

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| cp -f /usr/share/zoneinfo/Japan /etc/localtime |

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/sysconfig/clock |

〔結果ファイル〕

|  |
| --- |
| ZONE="Japan"  UTC=false |

### キーボードの設定

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/sysconfig/keyboard |

〔結果ファイル〕

|  |
| --- |
| KEYTABLE="jp106"  MODEL="pc105+inet"  LAYOUT="jp"  KEYBOARDTYPE="pc" |

### 日本語環境が存在しない場合

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| yum -y groupinstall "Japanese Support" |

### 確認

再起動して、以下の出力例の形式であればOK.

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| env | grep LANG |

〔出力例〕

|  |
| --- |
| LANG=jp\_JP.utf8 |

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| date |

〔出力例〕

|  |
| --- |
| 2014年 2月 7日 金曜日 10:35:29 JST |

## OSグループの作成

OSグループの作成例を記載する。動作すれば、グループの作成は指定の通りでなくても良い。

### グループの作成

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| groupadd -g 1000 spring |

## OSユーザの作成

OSユーザの作成例を記載する。動作すれば、ユーザの作成は指定の通りでなくても良い。

### ユーザの作成

実行ユーザ ：root

実行ディレクトリ ：任意

ユーザの作成

|  |
| --- |
| useradd -u 1001 -g spring -d /home/spring -m -s /bin/bash spring  useradd -u 1002 -g spring -d /home/jenkins -m -s /bin/bash jenkins  useradd -u 1003 -g spring -d /home/postgres -m -s /bin/bash postgres  useradd -u 1004 -g spring -d /home/tomcat -m -s /bin/bash tomcat |

実行ユーザ ：root

実行ディレクトリ ：任意

ユーザ毎のパスワードの設定

|  |
| --- |
| passwd spring |

パスワードはspringを設定

|  |
| --- |
| passwd jenkins |

パスワードはjenkinsを設定

|  |
| --- |
| passwd postgres |

パスワードはpostgresを設定

|  |
| --- |
| passwd tomcat |

パスワードはtomcatを設定

## グループでのsudo実行権限の付与

jenkinsからtomcatの再起動が可能になるようにグループによるsudoでのパスワード不要設定を行う。

### 権限付与

* + - グループユーザが各サービスの実行可能とするようにsudoのパスワード無し権限を付与

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| visudo |

以下の行をコメントアウト

|  |
| --- |
| **#**Defaults requiretty |

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/sudoers.d/tomcat-init |

以下の行を追加

|  |
| --- |
| # ホスト名は構築したサーバのホスト名に変更すること  jenkins ***ホスト名***=(tomcat) NOPASSWD: /opt/tomcat/tomcat/bin/, /etc/init.d/tomcat, /sbin/service, /bin/rm, /bin/cp, /bin/mkdir  tomcat ***ホスト名***=(tomcat) NOPASSWD: /opt/tomcat/tomcat/bin/, /etc/init.d/tomcat, /sbin/service, /bin/rm, /bin/cp, /bin/mkdir |

## ポート接続設定

JenkinsやTomcatへアクセスするポートを他のマシンからアクセス可能とするため、開放する。

### Firewallの無効化

* + - 各ソフトウェアにアクセスできるように内部環境でのみの試験を想定して、Firewallを無効化する。

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| /etc/init.d/iptables stop |

* + - Firewallの自動起動無効設定

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| chkconfig iptables off |

* + - Jenkins自動起動設定確認

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| chkconfig --list iptables |

〔出力例〕

以下のように出力されればOK

|  |
| --- |
| iptables 0:off 1:off 2:off 3:off 4:off 5:off 6:off |

## Proxy接続設定

Proxyが存在する場合は自ホストIPを除外リストにすること。

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/profile.d/proxy.sh |

自ホストIPを追加

|  |
| --- |
| export NO\_PROXY="localhost,127.0.0.1***,自ホストIP***"  export no\_proxy="localhost,127.0.0.1***,自ホストIP*** " |

# ソフトウェア類のカスタマイズ

ソフトウェア類のカスタマイズ部分を記載する。

## Java

### インストール先

/opt/java/jdk1.7.0\_72

### インストール方法

* 配置先ディレクトリ作成

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| mkdir -p /opt/java |

* ファイルの配置

ダウンロードした「jdk-7u72-linux-x64.gz」を/opt/javaに配置していること。

実行ユーザ ：root

実行ディレクトリ ：/opt/java

|  |
| --- |
| tar zxvf jdk-7u72-linux-x64.gz |

実行ユーザ ：root

実行ディレクトリ ：/opt/java

|  |
| --- |
| rm -f jdk-7u72-linux-x64.gz |

### 設定

優先度向上

* 事前確認

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| update-alternatives --config java |

ダウンロードしたjdk1.7.0\_72は対象に出てこない。

〔出力例〕

|  |
| --- |
| 2 プログラムがあり 'java' を提供します。  選択 コマンド  -----------------------------------------------  \*+ 1 /usr/lib/jvm/jre-1.7.0-openjdk.x86\_64/bin/java  2 /usr/lib/jvm/jre-1.6.0-openjdk.x86\_64/bin/java |

* 個別にダウンロードしたjavaを優先的に使用するように変更

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| update-alternatives --install /usr/bin/java java /opt/java/jdk1.7.0\_72/bin/java 17001 |

* 個別にインストールしたjavaを優先的に使用するように変更

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| update-alternatives --config java |

〔出力例〕

|  |
| --- |
| 3 プログラムがあり 'java' を提供します。  選択 コマンド  -----------------------------------------------  \*+ 1 /usr/lib/jvm/jre-1.7.0-openjdk.x86\_64/bin/java  2 /usr/lib/jvm/jre-1.6.0-openjdk.x86\_64/bin/java  3 /opt/java/jdk1.7.0\_72/bin/java  Enter を押して現在の選択 [+] を保持するか、選択番号を入力します:**3** |

3を選択して、インストールしたJavaが使用されるように変更

* 事後確認

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| update-alternatives --config java |

〔出力例〕

|  |
| --- |
| 3 プログラムがあり 'java' を提供します。  選択 コマンド  -----------------------------------------------  \* 1 /usr/lib/jvm/jre-1.7.0-openjdk.x86\_64/bin/java  2 /usr/lib/jvm/jre-1.6.0-openjdk.x86\_64/bin/java  + 3 /opt/java/jdk1.7.0\_72/bin/java |

インストールしたJavaに「+」が付いていること。

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| java -version |

〔出力例〕

|  |
| --- |
| java version "1.7.0\_72"  Java(TM) SE Runtime Environment (build 1.7.0\_72-b14)  Java HotSpot(TM) 64-Bit Server VM (build 24.72-b04, mixed mode) |

環境変数追加

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/bashrc |

最後に以下の行を追加

|  |
| --- |
| export JAVA\_HOME=/opt/java/jdk1.7.0\_72 |

## Firefox

### インストール先

/opt/firefox

### インストール方法

* ファイルの配置

ダウンロードした「firefox-31.3.0esr.tar.bz2」を/optに配置していること。

実行ユーザ ：root

実行ディレクトリ ：/opt

|  |
| --- |
| tar jxvf firefox-31.3.0esr.tar.bz2 |

実行ユーザ ：root

実行ディレクトリ ：/opt

|  |
| --- |
| rm -f firefox-31.3.0esr.tar.bz2 |

### 設定

環境変数追加

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/bashrc |

最後に以下の行を追加

|  |
| --- |
| export FIREFOX\_HOME=/opt/firefox  export PATH=${FIREFOX\_HOME}:${PATH} |

## PostgreSQL

### インストール先

/usr/pgsql-9.3/

### インストール方法

* rpmパッケージ取得

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| wget http://yum.postgresql.org/9.3/redhat/rhel-6-x86\_64/pgdg-centos93-9.3-1.noarch.rpm |

* 検証した環境は9.3.5であるが、ダウンロードするタイミングによってはメンテナンスバージョンはアップデートされる場合がある。
* rpmインストール

実行ユーザ ：root

実行ディレクトリ ：上記で取得したrpmファイルの存在するディレクトリ

|  |
| --- |
| rpm -ivh pgdg-centos93-9.3-1.noarch.rpm |

* yumで本体をインストール

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| yum install -y postgresql93-server postgresql93-devel postgresql93-contrib |

* rpmパッケージ削除

実行ユーザ ：root

実行ディレクトリ ：上記で取得したrpmファイルの存在するディレクトリ

|  |
| --- |
| rm -rf pgdg-centos93-9.3-1.noarch.rpm |

* DB初期化

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| /etc/init.d/postgresql-9.3 initdb |

〔出力例〕

以下のように出力されればOK

|  |
| --- |
| データベースを初期化中: [ OK ] |

* postgresql-9.3起動

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| /etc/init.d/postgresql-9.3 start |

〔出力例〕

以下のように出力されればOK

|  |
| --- |
| postgresql-9.3 サービスを開始中: [ OK ] |

### 設定

環境変数追加

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/bashrc |

最後に以下の行を追加

|  |
| --- |
| export POSTGRES\_HOME=/usr/pgsql-9.3/  export PATH=${POSTGRES\_HOME}/bin:${PATH} |

自動起動設定

* + - postgresql-9.3自動起動設定

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| chkconfig postgresql-9.3 on |

* + - postgresql-9.3自動起動設定確認

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| chkconfig --list postgresql-9.3 |

〔出力例〕

以下のように出力されればOK

|  |
| --- |
| postgresql-9.3 0:off 1:off 2:on 3:on 4:on 5:on 6:off |

パスワード変更

実行ユーザ ：root

実行ディレクトリ ：任意

postgresユーザにスイッチする。

|  |
| --- |
| su - postgres |

psqlツールをpostgresユーザで起動する。

|  |
| --- |
| psql -U postgres |

psqlツール内でパスワード変更を実施する。

|  |
| --- |
| ALTER Role postgres PASSWORD 'P0stgres'; |

データベース作成

パスワード変更に引き続きpsqlを使用

3つのデータベースを作成する。

|  |
| --- |
| CREATE DATABASE "spring-functionaltest" WITH OWNER = postgres ENCODING = 'utf8' TABLESPACE = pg\_default LC\_COLLATE = 'ja\_JP.utf8' LC\_CTYPE = 'ja\_JP.utf8' CONNECTION LIMIT = -1;  CREATE DATABASE "spring-functionaltest-open" WITH OWNER = postgres ENCODING = 'utf8' TABLESPACE = pg\_default LC\_COLLATE = 'ja\_JP.utf8' LC\_CTYPE = 'ja\_JP.utf8' CONNECTION LIMIT = -1;  CREATE DATABASE "spring-functionaltest-close" WITH OWNER = postgres ENCODING = 'utf8' TABLESPACE = pg\_default LC\_COLLATE = 'ja\_JP.utf8' LC\_CTYPE = 'ja\_JP.utf8' CONNECTION LIMIT = -1; |

psqlツールを終了する。

|  |
| --- |
| \q |

認証方法変更

PostgresへのTCP/IPでの認証を無条件で通すために設定を変更する。

実行ユーザ ：postgres

実行ディレクトリ ：/var/lib/pgsql/9.3/data

|  |
| --- |
| vi pg\_hba.conf |

〔変更前〕

|  |
| --- |
| # "local" is for Unix domain socket connections only  local all all peer  # IPv4 local connections:  host all all 127.0.0.1/32 ident  # IPv6 local connections:  host all all ::1/128 ident |

〔変更後〕

|  |
| --- |
| # "local" is for Unix domain socket connections only  local all all **trust**  # IPv4 local connections:  **#**host all all 127.0.0.1/32 ident  # IPv6 local connections:  **#**host all all ::1/128 ident  **host all all 127.0.0.1/32 trust**  **host all all *自IPアドレス帯域* trust**  **host all all 0.0.0.0 0.0.0.0 trust** |

変更を反映する。

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| /etc/init.d/postgresql-9.3 restart |

## Tomcat

### インストール先

/opt/tomcat/apache-tomcat-7.0.56

### インストール方法

* 配置先ディレクトリ作成

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| mkdir -p /opt/tomcat |

* ファイルの配置

ダウンロードした「apache-tomcat-7.0.56.tar.gz」を/opt/tomcatに配置していること。

実行ユーザ ：root

実行ディレクトリ ：/opt/tomcat

|  |
| --- |
| tar zxvf apache-tomcat-7.0.56.tar.gz |

実行ユーザ ：root

実行ディレクトリ ：/opt/tomcat

|  |
| --- |
| rm -f apache-tomcat-7.0.56.tar.gz |

実行ユーザ ：root

実行ディレクトリ ：/opt/tomcat

|  |
| --- |
| ln -s apache-tomcat-7.0.56 tomcat |

* 所有者の変更

実行ユーザ ：root

実行ディレクトリ ：/opt

|  |
| --- |
| chown -R tomcat:spring tomcat |

* パーミッションの追加（jenkinsユーザでも書き込めるようにグループに書込権限を追加）

実行ユーザ ：root

実行ディレクトリ ：/opt

|  |
| --- |
| chmod -R g+w tomcat |

### 設定

環境変数追加

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/bashrc |

最後に以下の行を追加

|  |
| --- |
| export TOMCAT\_HOME=/opt/tomcat/tomcat  export PATH=${TOMCAT\_HOME}/bin:${PATH} |

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/profile.d/tomcat.sh |

以下の行を追加

|  |
| --- |
| export TOMCAT\_HOME=/opt/tomcat/tomcat  export CATALINA\_HOME=/opt/tomcat/tomcat  export CLASSPATH=${CLASSPATH}:${CATALINA\_HOME}/common/lib |

起動停止シェル作成

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/init.d/tomcat |

以下の行を追加

|  |
| --- |
| #!/bin/sh  #  # Startup script for the Tomcat  #  # chkconfig: 345 80 15  # description: Tomcat is a Servlet+JSP Engine.  # Source function library.  . /etc/init.d/functions  # set environment  source /etc/profile.d/tomcat.sh  case "${1}" in  'start')  if [ -f ${TOMCAT\_HOME}/bin/startup.sh ]; then  echo "Starting Tomcat service"  sudo -u tomcat "${TOMCAT\_HOME}/bin/startup.sh"  else  echo "file not found. ${TOMCAT\_HOME}/bin/startup.sh"  fi  ;;  'stop')  if [ -f ${TOMCAT\_HOME}/bin/shutdown.sh ]; then  echo "Stopping Tomcat service."  sudo -u tomcat "${TOMCAT\_HOME}/bin/shutdown.sh"  else  echo "file not found. ${TOMCAT\_HOME}/bin/shutdown.sh"  fi  ;;  'restart')  ${0} stop  ${0} start  ;;  'status')  NUMPROC=`ps -ef | grep ${TOMCAT\_HOME} |grep -v grep | wc -l`  if [ ${NUMPROC} -gt 0 ]; then  echo "Tomcat is running..."  else  echo "Tomcat is stopped..."  fi  ;;  \*)  echo "Usage: ${0} {start|stop|restart|status}"  ;;  esac  exit 0 |

* + - Tomcat起動スクリプトに実行権限付加

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| chmod a+x /etc/init.d/tomcat |

URIEncoding変更

実行ユーザ ：tomcat

実行ディレクトリ ：任意

|  |
| --- |
| vi /opt/tomcat/tomcat/conf/server.xml |

コメントアウトされていない

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

に対してURIEncodingを追加

|  |
| --- |
| <Connector port="8080" protocol="HTTP/1.1"  connectionTimeout="20000"  redirectPort="8443" URIEncoding="UTF-8"/> |

管理者画面開設設定用

実行ユーザ ：tomcat

実行ディレクトリ ：任意

|  |
| --- |
| vi /opt/tomcat/tomcat/conf/tomcat-users.xml |

以下の行をtomcat-usersタグ内に追加

|  |
| --- |
| <role rolename="manager-gui"/>  <user username="tomcat" password="tomcat" roles="manager-gui,manager-script"/> |

使用メモリ/ログディレクトリの設定

実行ユーザ ：tomcat

実行ディレクトリ ：任意

|  |
| --- |
| vi /opt/tomcat/tomcat/bin/setenv.sh |

以下の行を追加

|  |
| --- |
| #!/bin/sh  export CATALINA\_OPTS="-Xmx1024m -XX:MaxPermSize=512M -Dapp.log.dir=/var/log/gfw/app/tomcat" |

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| mkdir -p /var/log/gfw/app/tomcat  chown -R tomcat:spring /var/log/gfw |

(補足)試験に必要なオプションの設定

testATHR0101003のテストケースでエラーとなる場合、以下のJAVA\_OPTSの設定が必要となる。

実行ユーザ ：tomcat

実行ディレクトリ ：任意

|  |
| --- |
| vi /opt/tomcat/tomcat/bin/setenv.sh |

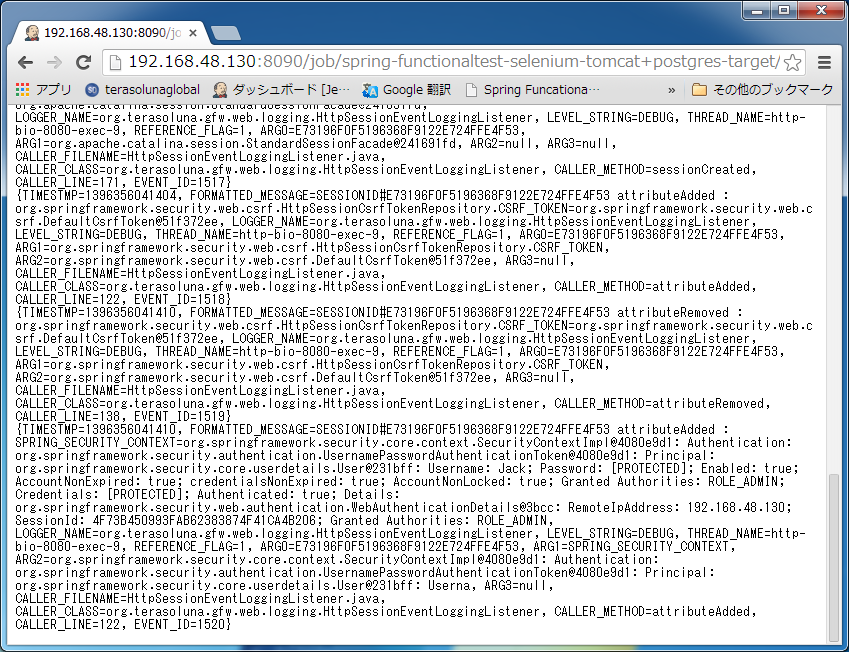
以下の行を最後に追加

|  |
| --- |
| export JAVA\_OPTS="-DRemoteIpAddress=***ホストのIPアドレス***" |

testATHR0101003はリクエストのあったIPアドレスを許可し、それ以外からのアクセスを拒否する試験である。

testATHR0101003のテストケースでエラーになる場合は、下記を参照し、IPアドレスを特定する。

Jenkins上で「テストプロジェクト」 -> 「workspace」-> 「evidence」-> 「athr/ATHR0101003」-> 「dblog\_001-failed-logging\_event.log」を開いた際の下記のRemoteIpAddressに出力されているIPアドレスが対象になる。



自動起動設定

* + - Tomcat起動スクリプトをchkconfigへ追加

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| chkconfig --add tomcat |

* + - Tomcat自動起動設定

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| chkconfig tomcat on |

* + - Tomcat自動起動設定確認

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| chkconfig --list tomcat |

〔出力例〕

以下のように出力されればOK

|  |
| --- |
| tomcat 0:off 1:off 2:on 3:on 4:on 5:on 6:off |

JDBCドライバの配置

${TOMCAT\_HOME}/libにjdbcのjarの配置が必要。

必要なJDBCドライバ

postgresql-9.3-1102.jdbc41.jar

〈参考〉

<http://jdbc.postgresql.org/download/postgresql-9.3-1102.jdbc41.jar>

Tomcat起動

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| /etc/init.d/tomcat start |

〔出力例〕

|  |
| --- |
| Starting Tomcat service  Using CATALINA\_BASE: /opt/tomcat/tomcat  Using CATALINA\_HOME: /opt/tomcat/tomcat  Using CATALINA\_TMPDIR: /opt/tomcat/tomcat/temp  Using JRE\_HOME: /usr  Using CLASSPATH: /opt/tomcat/tomcat/bin/bootstrap.jar:/opt/tomcat/tomcat/bin/tomcat-juli.jar  Tomcat started. |

起動後に作成されるディレクトリに権限を与えるため、再度、パーミッションの追加を実施する。

* パーミッションの追加（jenkinsユーザでも書き込めるようにグループに書込権限を追加）

実行ユーザ ：root

実行ディレクトリ ：/opt

|  |
| --- |
| chmod -R g+w tomcat |

## Maven

### インストール先

/opt/maven/apache-maven-3.2.3

### インストール方法

* 配置先ディレクトリ作成

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| mkdir -p /opt/maven |

* ファイルの配置

ダウンロードした「apache-maven-3.2.3-bin.tar.gz」を/opt/mavenに配置していること。

実行ユーザ ：root

実行ディレクトリ ：/opt/maven

|  |
| --- |
| tar zxvf apache-maven-3.2.3-bin.tar.gz |

実行ユーザ ：root

実行ディレクトリ ：/opt/maven

|  |
| --- |
| rm -f apache-maven-3.2.3-bin.tar.gz |

### 設定

環境変数追加

|  |
| --- |
| vi /etc/bashrc |

最後に以下の行を追加

|  |
| --- |
| export MAVEN\_HOME=/opt/maven/apache-maven-3.2.3  export PATH=${MAVEN\_HOME}/bin:${PATH} |

バージョン確認

環境変数読込

実行ユーザ ：root

実行ディレクトリ ：/opt/maven

|  |
| --- |
| . /etc/bashrc |

実行ユーザ ：root

実行ディレクトリ ：/opt/maven

|  |
| --- |
| mvn --version |

〔出力例〕

|  |
| --- |
| Apache Maven 3.2.3 (33f8c3e1027c3ddde99d3cdebad2656a31e8fdf4; 2014-08-12T05:58:10+09:00)  Maven home: /opt/maven/apache-maven-3.2.3  Java version: 1.7.0\_72, vendor: Oracle Corporation  Java home: /opt/java/jdk1.7.0\_72/jre  Default locale: ja\_JP, platform encoding: UTF-8  OS name: "linux", version: "2.6.32-431.el6.x86\_64", arch: "amd64", family: "unix" |

Proxyを使用する場合

本試験では、jenkinsユーザで実行するため、jenkinsのホームディレクトリに「.m2」ディレクトリが必要となり、settings.xmlに設定を行う必要がある。

## vncserver

### インストール先

/usr/bin/vncserver

### インストール方法

* X-window-systemが動作していれば、この部分は不要。

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| yum -y groupinstall "X Window System" |

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| yum -y install tigervnc-server xorg-x11-fonts-Type1 |

### 設定

vncのパスワードが設定されていないと実行時にエラーとなるため、実行するユーザでvncpasswdを実行し、パスワードファイルを作成する。

実行ユーザ ：jenkins

実行ディレクトリ ：任意

|  |
| --- |
| vncpasswd |

〔対話コンソール〕

|  |
| --- |
| Password:jenkins  Verify:jenkins |

ここではパスワードの例としてjenkinsを指定しているが、任意で問題ありません。

〈確認〉

実行ユーザ ：jenkins

実行ディレクトリ ：任意

|  |
| --- |
| ls /home/jenkins/.vnc |

〔出力例〕

|  |
| --- |
| passwd |

passwdファイルが作成されていればOK

〈参考〉

vncserver初回起動後

実行ユーザ ：jenkins

実行ディレクトリ ：任意

|  |
| --- |
| ls /home/jenkins/.vnc |

〔出力例〕

|  |
| --- |
| ***ホスト名***:***画面番号***.log passwd xstartup |

## Jenkins

### インストール先

/opt/jenkins

### インストール方法

* 配置先ディレクトリ作成

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| mkdir -p /opt/jenkins |

* 資材の配置

実行ユーザ ：root

実行ディレクトリ ：任意

ダウンロードした「jenkins.war」を/opt/jenkinsに配置する。

* 所有者の変更

実行ユーザ ：root

実行ディレクトリ ：/opt

|  |
| --- |
| chown -R jenkins:spring jenkins |

* パーミッションの追加（jenkinsユーザでも書き込めるようにグループに書込権限を追加）

実行ユーザ ：root

実行ディレクトリ ：/opt

|  |
| --- |
| chmod -R g+w jenkins |

### 設定

環境変数追加

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/bashrc |

最後に以下の行を追加

|  |
| --- |
| export JENKINS\_HOME=/opt/jenkins  export PATH=${JENKINS\_HOME}:${PATH} |

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/profile.d/jenkins.sh |

以下の行を追加

|  |
| --- |
| export JAVA\_ARGS=-Dhudson.util.ProcessTree.disable=true  export MEMORY\_ARGS="-Xmx256m -XX:MaxPermSize=256M"  export JENKINS\_HOME=/opt/jenkins  export JENKINS\_WAR=${JENKINS\_HOME}/jenkins.war  export JENKINS\_PORT=--httpPort=8090 |

#### 起動停止シェル作成

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| vi /etc/init.d/jenkins |

以下の行を追加

|  |
| --- |
| #!/bin/sh  # chkconfig: 345 81 16  # description: jenkins  # set environment  source /etc/profile.d/jenkins.sh  case "${1}" in  'start')  if [ -f ${JENKINS\_WAR} ]; then  echo "Starting jenkins"  su - jenkins -c "nohup java ${JAVA\_ARGS} ${MEMORY\_ARGS} -jar ${JENKINS\_WAR} ${JENKINS\_PORT} > /dev/null 2>&1 &"  else  echo "file not found. ${JENKINS\_WAR}"  fi  ;;  'stop')  if [ -f ${JENKINS\_WAR} ]; then  echo "Stopping jenkins."  kill -9 `ps -ef | grep [j]enkins.war | grep -v grep | awk '{ print $2 }'`  else  echo "file not found. ${JENKINS\_WAR}"  fi  ;;  'restart')  ${0} stop  ${0} start  ;;  'status')  NUMPROC=`ps -ef | grep [j]enkins.war | grep -v grep | wc -l`  if [ ${NUMPROC} -gt 0 ]; then  echo "Jenkins is running..."  else  echo "Jenkins is stopped..."  fi  ;;  \*)  echo "Usage: ${0} {start|stop|restart|status}"  ;;  esac  exit 0 |

* + - Jenkins起動スクリプトに実行権限付加

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| chmod a+x /etc/init.d/jenkins |

自動起動設定

* + - Jenkins起動スクリプトをchkconfigへ追加

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| chkconfig --add jenkins |

* + - Jenkins自動起動設定

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| chkconfig jenkins on |

* + - Jenkins自動起動設定確認

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| chkconfig --list jenkins |

〔出力例〕

以下のように出力されればOK

|  |
| --- |
| jenkins 0:off 1:off 2:on 3:on 4:on 5:on 6:off |

Jenkins起動

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| /etc/init.d/jenkins start |

perl-URのインストール

warの配置しているディレクトリをURI形式に変換するため

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| yum -y install perl-URI |

プラグイン導入

* + - 関連するjobをtriggerで実行する際にパラメータを引き継げるようにparameterized-trigger.hpiを導入する。

〈参考〉

<http://updates.jenkins-ci.org/latest/parameterized-trigger.hpi>

* + - vnc接続ができるようにxvnc.hpiを導入する。

〈参考〉

<http://updates.jenkins-ci.org/latest/xvnc.hpi>

* + - ${JENKINS\_HOME}/plugins/ にhpiを配置する。
  + 配置して再起動するとプラグインが有効になる。

実行ユーザ ：root

実行ディレクトリ ：任意

|  |
| --- |
| /etc/init.d/jenkins restart |

ユーザ管理

##### ユーザ管理方法の設定

Jenkinsにクライアントのブラウザから「http://***サーバのURL***:8090」へアクセスする。

Jenkins のサイドメニューの「Jenkinsの管理」 -> 「グローバルセキュリティの設定」



「セキュリティを有効化」にチェックを入れ、ユーザ管理方法を指定する。



保存すると先頭画面に戻される。

##### ユーザ作成

サインアップし、ユーザを作成する。

ユーザ名とパスワードを設定してサインアップする。

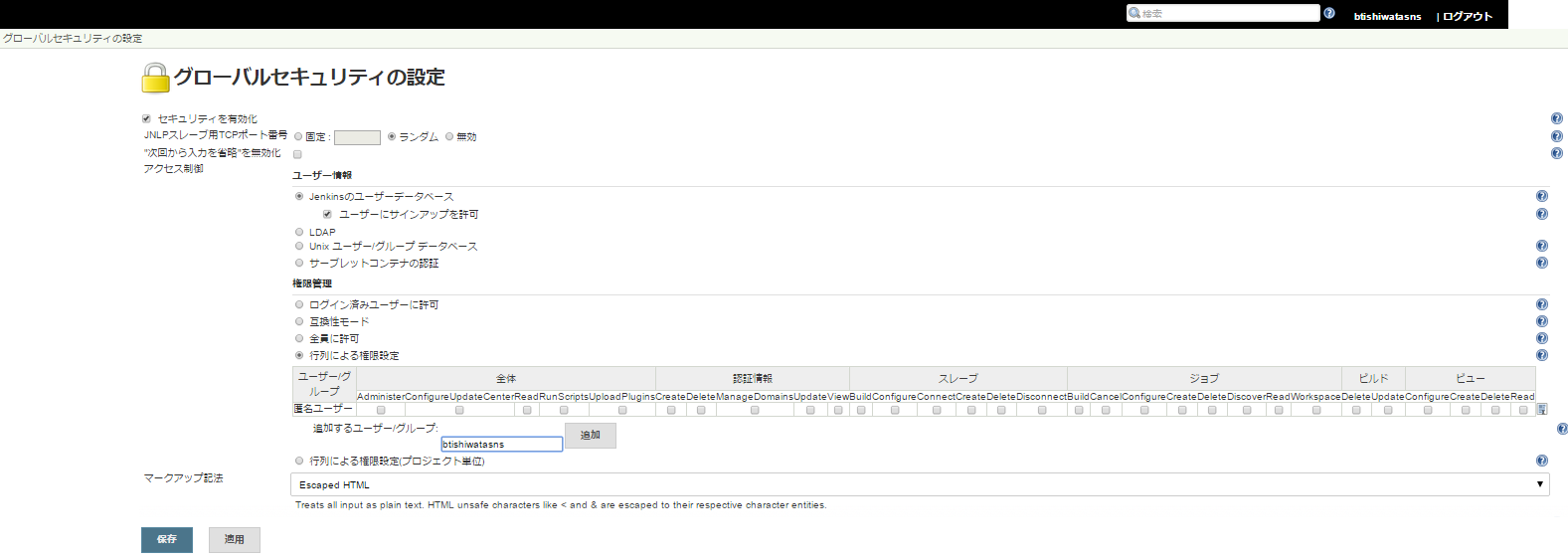


サインアップ後はログイン状態となる。

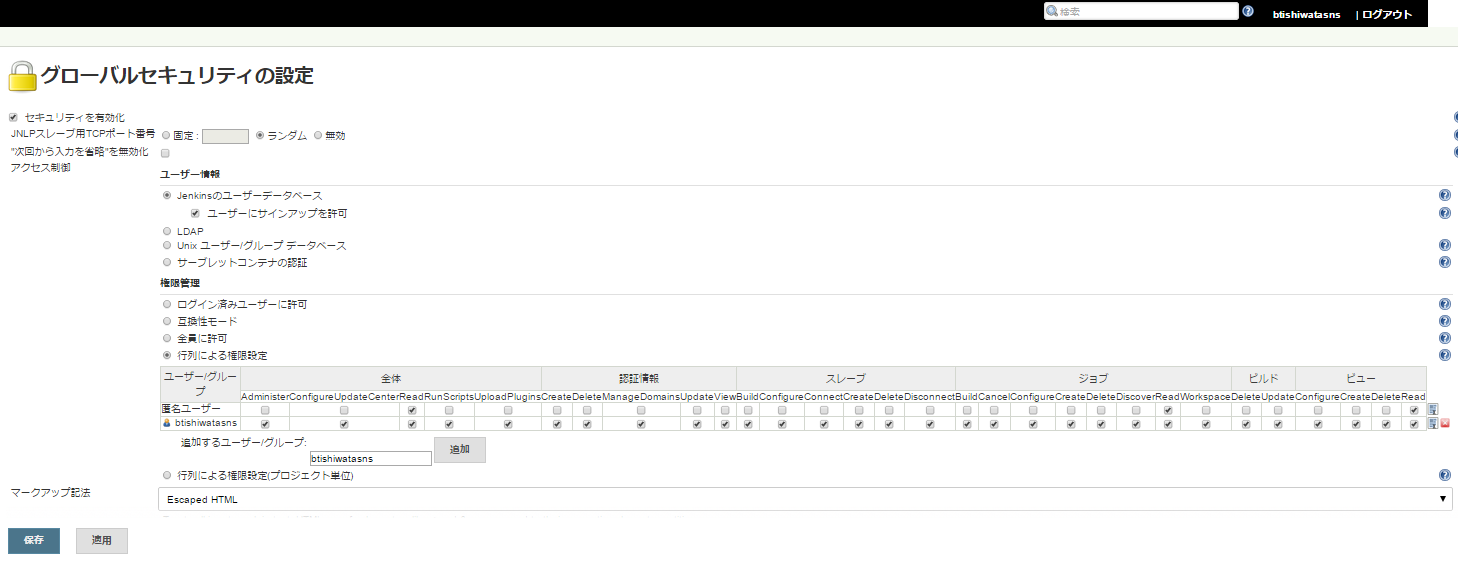
##### 権限の付与

Jenkinsのサイドメニューの「Jenkinsの管理」 -> 「グローバルセキュリティの設定」 ->「権限管理」 -> 「行列による権限設定」

サインアップしたユーザを指定して追加する。



管理者は全て権限をつけて、匿名ユーザはReadのみに設定する。

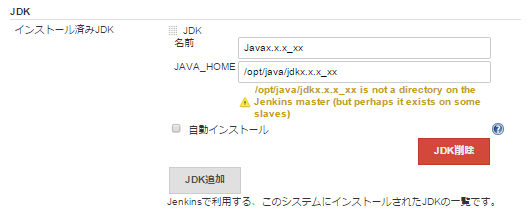


JAVA\_HOMEの設定

Jenkinsのサイドメニューの「Jenkinsの管理」 -> 「システムの設定」



JDK「JDK追加」ボタンを押して開く



* JDK
  + 名前

Java1.7.0\_72

* + JAVA\_HOME

/opt/java/jdk1.7.0\_72

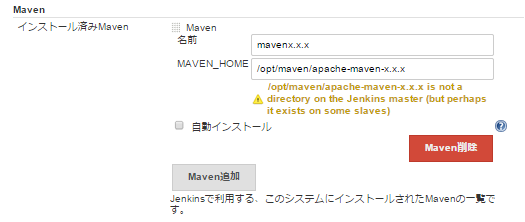
* + キャプチャのxは置き換えて下さい。

Mavenの設定

Jenkinsのサイドメニューの「Jenkinsの管理」 -> 「システムの設定」



Maven「Maven追加」ボタンを押して開く



* Maven
  + 名前

maven3.2.3

* + MAVEN\_HOME

/opt/maven/apache-maven-3.2.3

* + キャプチャのxは置き換えて下さい。

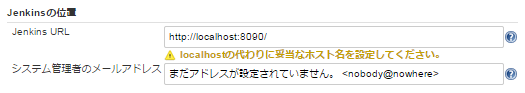
Jenkinsの位置の設定

この設定はデフォルトのままなので、localhostで実行出来る場合は変更する必要はない。

localhost以外の場合は、IPアドレスを指定すること。

Jenkinsのサイドメニューの「Jenkinsの管理」 -> 「システムの設定」





* Jenkinsの位置
  + Jenkins

<http://localhost:8090/>

* + システム管理者のメールアドレス

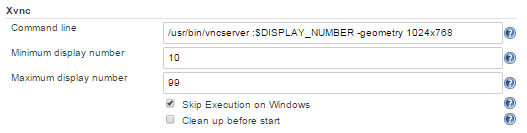
指定無し

プラグインの設定

##### Xvncプラグインの設定

Jenkinsのサイドメニューの「Jenkinsの管理」 -> 「システムの設定」





* Xvnc
  + Command line

/usr/bin/vncserver :$DISPLAY\_NUMBER -geometry 1024x768

* + Minimum display number

10

* + Maximum display number

99

* + skip Execution on Windows

checked

# テスト対象資材の配置

テスト対象資材はサーバ上のローカルディスクに配置するものとする。

## 配置場所・配置方法

### 配置場所

/home/spring

### 配置方法

* パーミッションの追加（jenkinsユーザでも書き込めるようにグループに書込権限を追加）

実行ユーザ ：root

実行ディレクトリ ：/home

|  |
| --- |
| chmod -R g+rwx spring |

* + - 資材

テスト資材である「spring-functionaltest-linux.zip」を/home/springに配置していること。

* + - 解凍

実行ユーザ ：spring

実行ディレクトリ ：/home/spring

|  |
| --- |
| unzip spring-functionaltest-linux.zip |

* + - 確認

実行ユーザ ：spring

実行ディレクトリ ：/home/spring/spring-functionaltest

|  |
| --- |
| ls -1 |

〔出力例〕

|  |
| --- |
| spring-functionaltest-domain  spring-functionaltest-env  spring-functionaltest-initdb  spring-functionaltest-parent  spring-functionaltest-selenium  spring-functionaltest-web |

# Jenkinsの設定

テストを実行するJenkinsの設定を記載する。

## 対象プロジェクト一覧

表 5.1‑1 対象プロジェクト一覧

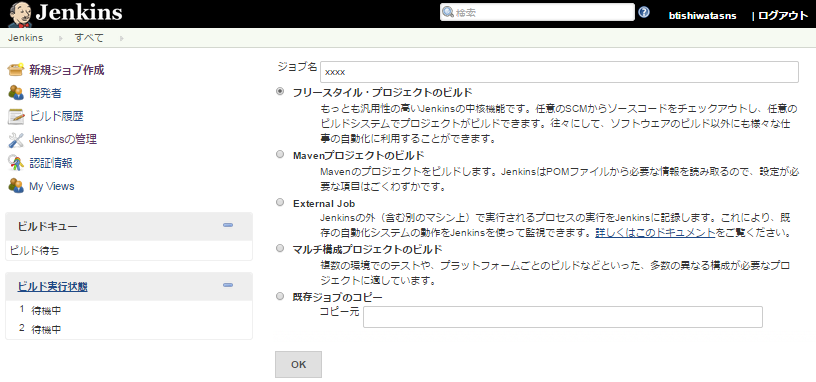
|  |  |
| --- | --- |
| プロジェクト名 | 説明 |
|  | プロジェクトで必要となるOSSのパッケージをローカルリポジトリにインストールする。 これによりローカルにある他のプロジェクトから参照することができるようになる。  依存関係があり、parentを実行することにより、domain,webは実行される。 |
| spring-functionaltest-env | 環境依存性設定ファイルを集めたプロジェクトをビルドします。  特にprofileは指定せず、デフォルトのままです。 |
| spring-functionaltest-env | ドメイン層のプロジェクトをビルドし、jarを作成する。 |
|  | Web層のプロジェクトをビルドして、warを作成する。 |
|  | spring-functionaltestデータベース(PostgreSQL)の初期化 |
|  | Tomcatへのリリース |
|  | seleniumを利用して、tomcatへリリースした資材を試験する。DBはpostgresを使用。 |
|  | initdb,release,seleniumを順番に実行する。  テストを実施する際にはこのJOBのみ実行すれば良い。 |
|  | TestClass単位で実行するためのJOB |
|  | TestClass単位で実行するためのJOB |

## JOBの作成

1. 新規ジョブ作成



1. フリースタイル・プロジェクトのビルドを選んで作成する。



1. この画面でプロジェクトを作成する。



## プロジェクト毎の設定

JOBの作成は依存関係があり、最初は選択、入力できない箇所が存在するが、全てのJOBを作成すれば、互いに参照が可能となる。

JOB作成時にデフォルトから変更する設定値を青色で記述している。また、デフォルトのままであるが迷いそうな項目について、灰色で記述している。

### spring-functionaltest-parent

* 番号は例として次ページの画面キャプチャと対応させている。
* プロジェクト名

|  |
| --- |
| spring-functionaltest-parent |

* 説明

|  |
| --- |
| プロジェクトで必要となるOSSのパッケージをローカルリポジトリにインストールする。  これによりローカルにある他のプロジェクトから参照することができるようになる。  依存関係があり、parentを実行することにより、env,domain,webは実行される。 |

* ビルドのパラメータ化
  + 文字列
    - 名前

|  |
| --- |
| VERSION |

* + - デフォルト値

|  |
| --- |
| 1.1.0.RELEASE |

* ソースコード管理

|  |
| --- |
| なし |

* ビルド
  + シェルの実行

|  |
| --- |
| cp -r /home/spring/spring-functionaltest/spring-functionaltest-parent/\* ${WORKSPACE}/. |

* + mavenの呼び出し
    - 使用するMaven

|  |
| --- |
| maven3.2.3 |

* + - ゴール

|  |
| --- |
| install |

* + - 高度な設定
      * POM

|  |
| --- |
| pom.xml |

* ビルド後の処理
  + Trigger parameterized build on other project
    - Build Triggers
      * Projects to build

|  |
| --- |
| spring-functionaltest-env |

* + - * Trigger when build is

|  |
| --- |
| Stable |

* + - * Add Parameters

|  |
| --- |
| Current build paramaters |

〔例として対応位置を示す〕

**①**

* １



**⑩**

* １

**⑨**

* １

**⑧**

* １

**⑦**

* １

**⑥**

* １

**④**

* １

**⑤**

* １

**③**

* １

**②**

* １

### spring-functionaltest-env

* プロジェクト名

|  |
| --- |
| spring-functionaltest-env |

* 説明

|  |
| --- |
| 環境依存性設定ファイルを集めたプロジェクトをビルドします。  特にprofileは指定せず、デフォルトのままです。 |

* ソースコード管理

|  |
| --- |
| なし |

* ビルド
  + シェルの実行

|  |
| --- |
| cp /home/spring/spring-functionaltest/spring-functionaltest-env/pom.xml ${WORKSPACE}/.  cp -r /home/spring/spring-functionaltest/spring-functionaltest-env/src ${WORKSPACE}/.  cp -r /home/spring/spring-functionaltest/spring-functionaltest-env/configs ${WORKSPACE}/.  cp -r /home/spring/spring-functionaltest/spring-functionaltest-env/3rdparty ${WORKSPACE}/. |

* + mavenの呼び出し
    - 使用するMaven

|  |
| --- |
| maven3.2.3 |

* + - ゴール

|  |
| --- |
| clean install |

* + - 高度な設定
      * POM

|  |
| --- |
| pom.xml |

* ビルド後の処理
  + Trigger parameterized build on other project
    - Build Triggers
      * Projects to build

|  |
| --- |
| spring-functionaltest-domain |

* + - * Trigger when build is

|  |
| --- |
| Stable |

* + - * Add Parameters

|  |
| --- |
| Current build paramaters |

### spring-functionaltest-domain

* プロジェクト名

|  |
| --- |
| spring-functionaltest-domain |

* 説明

|  |
| --- |
| ドメイン層のプロジェクトをビルドし、jarを作成する。 |

* ソースコード管理

|  |
| --- |
| なし |

* ビルド
  + シェルの実行

|  |
| --- |
| cp -r /home/spring/spring-functionaltest/spring-functionaltest-domain/\* ${WORKSPACE}/. |

* + mavenの呼び出し
    - 使用するMaven

|  |
| --- |
| maven3.2.3 |

* + - ゴール

|  |
| --- |
| clean install |

* + - 高度な設定
      * POM

|  |
| --- |
| pom.xml |

* ビルド後の処理
  + Trigger parameterized build on other project
    - Build Triggers
      * Projects to build

|  |
| --- |
| spring-functionaltest-web |

* + - * Trigger when build is

|  |
| --- |
| Stable |

* + - * Add Parameters

|  |
| --- |
| Current build paramaters |

### spring-functionaltest-web

* プロジェクト名

|  |
| --- |
| spring-functionaltest-web |

* 説明

|  |
| --- |
| Web層のプロジェクトをビルドして、warを作成する。 |

* ソースコード管理

|  |
| --- |
| なし |

* ビルド
  + シェルの実行

|  |
| --- |
| cp -r /home/spring/spring-functionaltest/spring-functionaltest-web/\* ${WORKSPACE}/. |

* + mavenの呼び出し
    - 使用するMaven

|  |
| --- |
| maven3.2.3 |

* + - ゴール

|  |
| --- |
| clean install -P warpack |

* + - 高度な設定
      * POM

|  |
| --- |
| pom.xml |

### spring-functionaltest-initdb-postgresql

* プロジェクト名

|  |
| --- |
| spring-functionaltest-initdb-postgresql |

* 説明

|  |
| --- |
| spring-functionaltestデータベース(PostgreSQL)の初期化 |

* ソースコード管理

|  |
| --- |
| なし |

* ビルド
  + シェルの実行

|  |
| --- |
| cp -r /home/spring/spring-functionaltest/spring-functionaltest-initdb/\* ${WORKSPACE}/. |

* + mavenの呼び出し
    - 使用するMaven

|  |
| --- |
| maven3.2.3 |

* + - ゴール

|  |
| --- |
| sql:execute |

* + mavenの呼び出し
    - 使用するMaven

|  |
| --- |
| maven3.2.3 |

* + - ゴール

|  |
| --- |
| sql:execute -P local-postgres-open |

* + mavenの呼び出し
    - 使用するMaven

|  |
| --- |
| maven3.2.3 |

* + - ゴール

|  |
| --- |
| sql:execute -P local-postgres-close |

### spring-functionaltest-release-tomcat+postgres

* プロジェクト名

|  |
| --- |
| spring-functionaltest-release-tomcat+postgres |

* 説明

|  |
| --- |
| Tomcatへのリリース |

* ソースコード管理

|  |
| --- |
| なし |

* ビルド
  + シェルの実行

|  |
| --- |
| cp -r /home/spring/spring-functionaltest/spring-functionaltest-env/\* ${WORKSPACE}/. |

* + mavenの呼び出し
    - 使用するMaven

|  |
| --- |
| maven3.2.3 |

* + - ゴール

|  |
| --- |
| clean package -P tomcat-postgresql |

* + - 高度な設定
      * プロパティ

|  |
| --- |
| source.skip=true |

* + mavenの呼び出し
    - 使用するMaven

|  |
| --- |
| maven3.2.3 |

* + - ゴール

|  |
| --- |
| org.apache.maven.plugins:maven-dependency-plugin:2.5:get |

* + - 高度な設定
      * プロパティ

|  |
| --- |
| groupId=jp.co.ntt.fw.spring.functionaltest  artifactId=spring-functionaltest-web  version=${VERSION}  packaging=war  dest=${WORKSPACE}/target/spring-functionaltest-web.war |

* + シェルの実行

|  |
| --- |
| #!/bin/sh  APP\_NAME=spring-functionaltest  ENV\_ID=tomcat-postgresql  WEBAPP\_NAME=${APP\_NAME}-web  WEBAPP\_ENV\_JARS\_DIR=/opt/jenkins/webapps-env-jars/${APP\_NAME}-env-${ENV\_ID}  WAR\_FILE\_PATH=`echo "${WORKSPACE}/target/${WEBAPP\_NAME}.war" | perl -MURI::Escape -lne 'print uri\_escape($\_)'`  HTTP\_USER=tomcat  HTTP\_PASSWD=tomcat  **# Tomcatが実行されているマシンのIPアドレスに変更すること**  TARGET\_SRV="***IPアドレス:ポート番号***"  TARGET\_PROJECT="spring-functionaltest-web"  STOP\_URL="http://${TARGET\_SRV}/manager/text/stop?path=/${WEBAPP\_NAME}"  START\_URL="http://${TARGET\_SRV}/manager/text/start?path=/${WEBAPP\_NAME}"  UNDEPLOY\_URL="http://${TARGET\_SRV}/manager/text/undeploy?path=/${WEBAPP\_NAME}"  DEPLOY\_URL="http://${TARGET\_SRV}/manager/text/deploy?path=/${WEBAPP\_NAME}&war=${WAR\_FILE\_PATH}"  # 再起動時の不要ファイルの削除  sudo -u tomcat rm -fr ${WEBAPP\_ENV\_JARS\_DIR}/\*.jar  sudo -u tomcat rm -fr ${TOMCAT\_HOME}/conf/Catalina/localhost/${WEBAPP\_NAME}.xml  sudo -u tomcat rm -fr ${TOMCAT\_HOME}/webapps/${WEBAPP\_NAME}  sudo -u tomcat rm -fr ${TOMCAT\_HOME}/webapps/${WEBAPP\_NAME}.war  # Tomcat再起動  sudo -u tomcat /etc/init.d/tomcat restart  # Tomcatの起動確認  WAIT\_TIME=10  MAX\_COUNT=10  NOW\_COUNT=0  RET=1  while [ "${RET}" -ne "0" -a "${NOW\_COUNT}" -lt "${MAX\_COUNT}" ]  do  sleep ${WAIT\_TIME}  wget -O - ${TARGET\_SRV}  RET=$?  if [ "${RET}" -ne "0" ]; then  echo "Tomcatがまだ正常起動していません。"  echo "リトライ回数：${NOW\_COUNT}回目、sleep秒数=${WAIT\_TIME}"  else  echo "Tomcatが正常起動しました。"  RET=0  fi  NOW\_COUNT=`expr ${NOW\_COUNT} + 1`  done  if [ ${RET} -ne "0" -o "${NOW\_COUNT}" -eq "${MAX\_COUNT}" ]; then  echo "Tomcatが正常起動していません。"  echo "戻り値=${RET}"  echo "リトライ回数=${NOW\_COUNT}回目、最大リトライ回数=${MAX\_COUNT}"  exit 1  fi  # config(env)のjarファイルの設置  sudo -u tomcat mkdir -p ${WEBAPP\_ENV\_JARS\_DIR}  sudo -u tomcat cp ${WORKSPACE}/target/${APP\_NAME}\*.jar ${WEBAPP\_ENV\_JARS\_DIR}/.  # コンテキスト設定xmlファイルを設置  sudo -u tomcat mkdir -p ${TOMCAT\_HOME}/conf/Catalina/localhost  sudo -u tomcat cp ${WORKSPACE}/configs/${ENV\_ID}/ContainerConfigXML/context.xml ${TOMCAT\_HOME}/conf/Catalina/localhost/${WEBAPP\_NAME}.xml  # funcationtestのデプロイ  wget -O - --http-user=${HTTP\_USER} --http-password=${HTTP\_PASSWD} ${DEPLOY\_URL}  RET=$?  if [ "${RET}" -ne "0" ]; then  return ${RET}  fi  # funcationtestをstart  wget -O - --http-user=${HTTP\_USER} --http-password=${HTTP\_PASSWD} ${START\_URL}  RET=$?  if [ "${RET}" -ne "0" ]; then  return ${RET}  fi  # spring-fanctinaltest-webの起動確認  WAIT\_TIME=10  MAX\_COUNT=20  NOW\_COUNT=0  RET=1  while [ "${RET}" -ne "0" -a "${NOW\_COUNT}" -lt "${MAX\_COUNT}" ]  do  sleep ${WAIT\_TIME}  wget -O - ${TARGET\_SRV}/${TARGET\_PROJECT}  RET=$?  if [ "${RET}" -ne "0" ]; then  echo "spring-fanctinaltest-webがまだ正常起動していません。"  echo "リトライ回数：${NOW\_COUNT}回目、sleep秒数=${WAIT\_TIME}"  else  echo "spring-fanctinaltest-webが正常起動しました。"  RET=0  fi  NOW\_COUNT=`expr ${NOW\_COUNT} + 1`  done  if [ ${RET} -ne "0" -o "${NOW\_COUNT}" -eq "${MAX\_COUNT}" ]; then  echo "spring-fanctinaltest-webが正常起動していません。"  echo "戻り値=${RET}"  echo "リトライ回数=${NOW\_COUNT}回目、最大リトライ回数=${MAX\_COUNT}"  exit 1  fi |

### spring-functionaltest-selenium-tomcat+postgres

* プロジェクト名

|  |
| --- |
| spring-functionaltest-selenium-tomcat+postgres |

* 説明

|  |
| --- |
| seleniumを利用して、tomcatへリリースした資材を試験する。DBはpostgresを使用。 |

* ソースコード管理

|  |
| --- |
| なし |

* ビルド環境
  + Run Xvnc during build

|  |
| --- |
| Checked |

* ビルド
  + シェルの実行

|  |
| --- |
| cp -r /home/spring/spring-functionaltest/spring-functionaltest-selenium/\* ${WORKSPACE}/. |

* + mavenの呼び出し
    - 使用するMaven

|  |
| --- |
| maven3.2.3 |

* + - ゴール

|  |
| --- |
| clean test |

* + - 高度な設定
      * POM

|  |
| --- |
| pom.xml |

* + - * プロパティ

|  |
| --- |
| selenium.serverUrl=http://***IPアドレス:ポート番号***  selenium.contextName=spring-functionaltest-web  selenium.applicationContextUrl=http://***IPアドレス:ポート番号***/spring-functionaltest-web  spring.profiles.active=firefox  selenium.enableCapture=false  selenium.enablePageSource=false  selenium.evidenceBaseDirectory=${WORKSPACE}/evidence |

* ビルド後の処理
  + JUnitテスト結果の集計
    - テスト結果XML

|  |
| --- |
| target/surefire-reports/\*.xml |

推奨スペックに充たしていない環境でテストする場合は、

「実行順序性の調整パラメータ」を指定(変更)が必要になるケースがあるため、

1.2.4推奨構成の※を参照。

### spring-functionaltest-test-tomcat+postgres

* プロジェクト名

|  |
| --- |
| spring-functionaltest-test-tomcat+postgres |

* 説明

|  |
| --- |
| initdb,release,seleniumを順番に実行する。  テストを実施する際にはこのJOBのみ実行すれば良い。 |

* ビルドのパラメータ化
  + 文字列
    - 名前

|  |
| --- |
| VERSION |

* + - デフォルト値

|  |
| --- |
| 1.1.0.RELEASE |

* ソースコード管理

|  |
| --- |
| なし |

* ビルド
  + Trigger / call builds on other project
    - Projects to build

|  |
| --- |
| spring-functionaltest-initdb-postgresql |

* + - Block until the triggered projects finish their builds

|  |
| --- |
| checked |

* + - Fail this build step if the triggered build is worse or equal to

|  |
| --- |
| FAILURE |

* + - Fail this build step if the triggered build is worse or equal to

|  |
| --- |
| FAILURE |

* + - Fail this build step if the triggered build is worse or equal to

|  |
| --- |
| UNSTABLE |

* + - Add Parameters

|  |
| --- |
| Current build paramaters |

* + Trigger / call builds on other project
    - Projects to build

|  |
| --- |
| spring-functionaltest-release-tomcat+postgres |

* + - Block until the triggered projects finish their builds

|  |
| --- |
| checked |

* + - Fail this build step if the triggered build is worse or equal to

|  |
| --- |
| FAILURE |

* + - Fail this build step if the triggered build is worse or equal to

|  |
| --- |
| FAILURE |

* + - Fail this build step if the triggered build is worse or equal to

|  |
| --- |
| UNSTABLE |

* + - Add Parameters

|  |
| --- |
| Current build paramaters |

* + Trigger / call builds on other project
    - Projects to build

|  |
| --- |
| spring-functionaltest-selenium-tomcat+postgres |

* + - Block until the triggered projects finish their builds

|  |
| --- |
| checked |

* + - Fail this build step if the triggered build is worse or equal to

|  |
| --- |
| FAILURE |

* + - Fail this build step if the triggered build is worse or equal to

|  |
| --- |
| FAILURE |

* + - Fail this build step if the triggered build is worse or equal to

|  |
| --- |
| UNSTABLE |

* + - Add Parameters

|  |
| --- |
| Current build paramaters |

### spring-functionaltest-selenium-tomcat+postgres-target

からJOBをコピーして作成すること。

* プロジェクト名

|  |
| --- |
| spring-functionaltest-selenium-tomcat+postgres-target |

* 説明

|  |
| --- |
| クラス単位に試験を実行する。  対応のための  のクラス名指定バージョン |

* ビルド
  + mavenの呼び出し
    - 使用するMaven

|  |
| --- |
| maven3.2.3 |

* + - ゴール

|  |
| --- |
| clean test |

* + - 高度な設定
      * POM

|  |
| --- |
| pom.xml |

* + - * プロパティ

|  |
| --- |
| selenium.serverUrl=http://***IPアドレス:ポート番号***  selenium.contextName=spring-functionaltest-web  selenium.applicationContextUrl=http://***IPアドレス:ポート番号***/spring-functionaltest-web  spring.profiles.active=firefox  selenium.enableCapture=false  selenium.enablePageSource=false  selenium.evidenceBaseDirectory=${WORKSPACE}/evidence  **test=${TARGET\_CLASS}** |

* **test=${TARGET\_CLASS}**を追加するのみが変更点である。

### spring-functionaltest-test-tomcat+postgres-target

からJOBをコピーして作成すること。

* プロジェクト名

|  |
| --- |
| spring-functionaltest-test-tomcat+postgres-target |

* 説明

|  |
| --- |
| クラス単位に試験を実行するためのJOB  のクラス名指定バージョン |

* ビルドのパラメータ化
  + 文字列
    - 名前

|  |
| --- |
| **TARGET\_CLASS** |

* + - デフォルト値

|  |
| --- |
| **TilesLayoutTest** |

* **TARGET\_CLASS**を追加する。デフォルト値は実行で最も軽いテストを選択している。実行するクラスを「表 5.3‑1 指定可能テストケース名一覧」から選択し、jenkins実行時のパラメータ付きビルドで「TARGET\_CLASS」の値を変更すること。
* ビルド
  + Trigger / call builds on other project
    - Projects to build

|  |
| --- |
| **spring-functionaltest-selenium-tomcat+postgres-target** |

* + - Block until the triggered projects finish their builds

|  |
| --- |
| checked |

* + - Fail this build step if the triggered build is worse or equal to

|  |
| --- |
| FAILURE |

* + - Fail this build step if the triggered build is worse or equal to

|  |
| --- |
| FAILURE |

* + - Fail this build step if the triggered build is worse or equal to

|  |
| --- |
| UNSTABLE |

* + - Add Parameters

|  |
| --- |
| Current build paramaters |

* spring-functionaltest-selenium-tomcat+postgresから**spring-functionaltest-selenium-tomcat+postgres-targetに変更**する。

指定に意味はないが、一番軽い処理をデフォルトに設定している。

パラメータ付きビルドで対象のクラスを変更して実行すると対象クラスのみのテスト実行となる。

表 5.3‑1 指定可能テストケース名一覧

|  |
| --- |
| テストケースクラス名 |
| AjaxTest |
| ApplicationLayerTest |
| AuthenticationTest |
| AuthorizationTest |
| BeanMappingTest |
| CommonLibraryValidationTest |
| CorrelationValidationTest |
| CSRFProtectionTest |
| CustomBeanValidationTest |
| DataAccessCommonTest |
| DataAccessMyBatis3AddTest |
| DataAccessMyBatis3Test |
| DateOperationTest |
| DefineValidationMessagesTest |
| DomainLayerTest |
| ExceptionHandlingTest |
| ExclusiveControlTest |
| FileDownloadTest |
| FileUploadTest |
| InternationalizationForDeLocaleBrowserTest |
| InternationalizationForEnLocaleBrowserTest |
| InternationalizationForFrLocaleBrowserTest |
| InternationalizationForNonLocaleBrowserTest |
| PaginationTest |
| PasswordHashTest |
| PropertyManagementTest |
| SessionManagementTest |
| SimpleValidationTest |
| SpringSecurityTest |
| TilesLayoutTest |

## Jenkinsの実行

### 実行順序

spring-functionaltest-parent

（trigger起動

→spring-functionaltest-env

→spring-functionaltest-domain

→spring-functionaltest-web

)

↓

spring-functionaltest-test-tomcat+postgres

（trigger起動

→spring-functionaltest-initdb-postgresql

→spring-functionaltest-release-tomcat+postgres

→spring-functionaltest-selenium-tomcat+postgres

)

の順に実行する。

* 2つのJOBを順番に実行すると他のJOBはtrigger起動に記述した順で実行される。

### 実行方法

ビルドボタンをクリック



## Jenkinsの結果確認

### 結果確認方法

spring-functionaltest-parent

Jenkinsで「spring-functionaltest-parent」の実行確認手順を説明する。

Jenkinsのページで「spring-functionaltest-parent」の「ビルドボタン」を押下する。

（ログインしていないとビルドボタンが押せないので、注意すること。）



ログインするとログインユーザの名称となる。

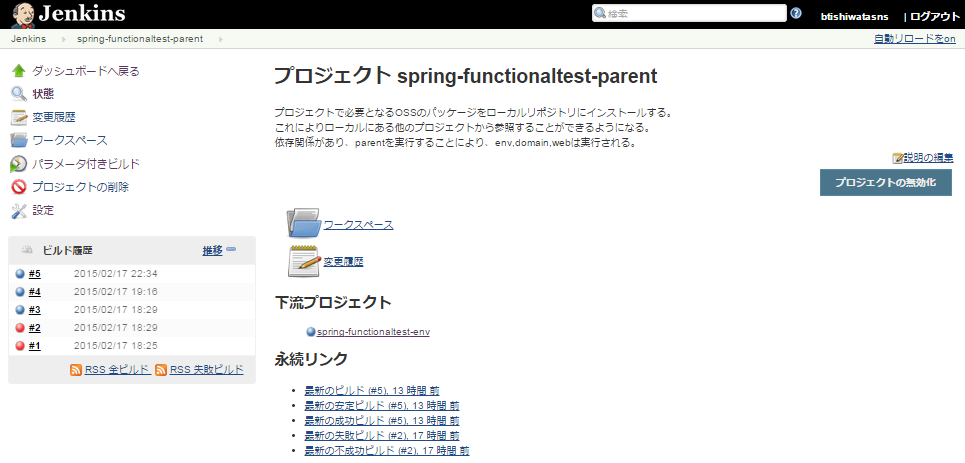
前回の実行が成功であると青になり、失敗であると赤になる。

ビルドの実行中、または完了後の実行確認を行う。

「spring-functionaltest-parent」のJOBの中に入っていく。



ビルドした時間の「ビルド履歴」をクリック

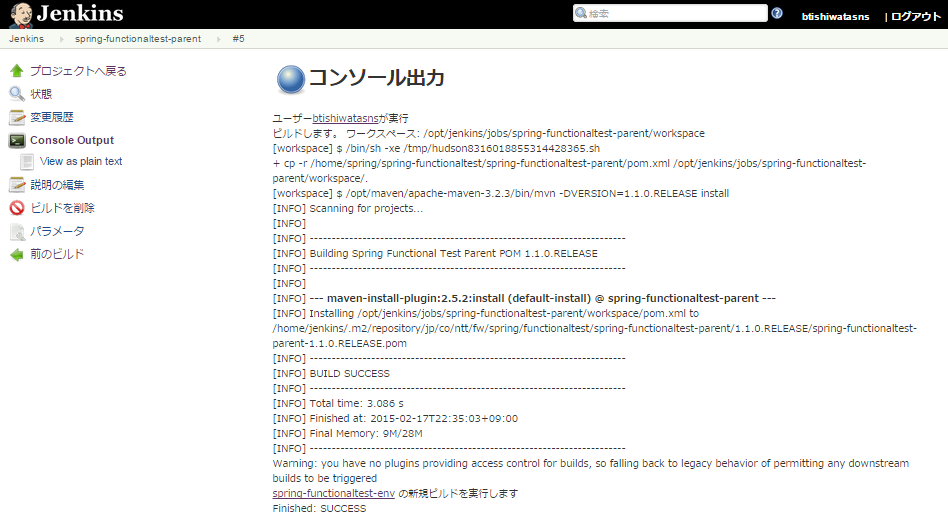


「Console Output」をクリック



実行したユーザが表示されます。

コンソール出力の内容を確認する。



SUCCESSが出力されていれば、OK.

エラーがある場合は、「5.5.2エラーについての結果確認方法」があるので、エラー部分について、解析する。

下位プロジェクトがある場合は、下位プロジェクトへ遷移し、コンソール出力の確認を行う。



spring-functionaltest-test-tomcat+postgres

テストの実行確認

実行中、または実行後に確認。

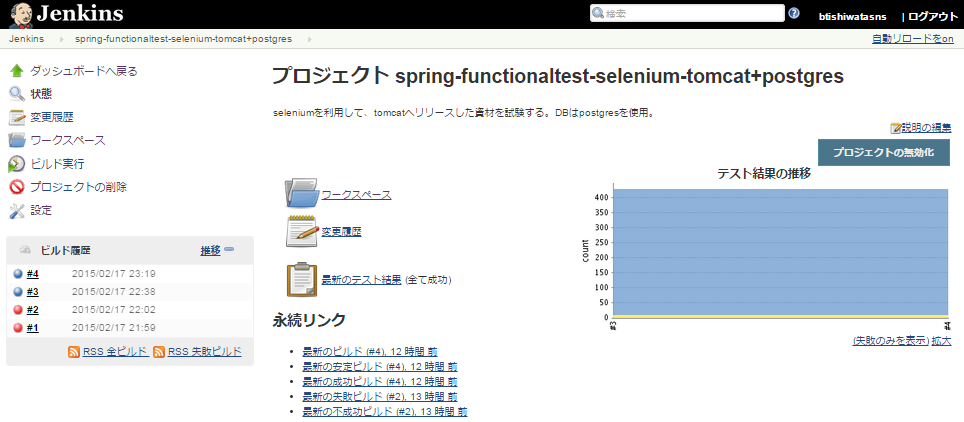
「spring-functionaltest-test-tomcat+postgres」をクリック。



テストの結果自体はsubプロジェクトの「spring-functionaltest-selenium-tomcat+postgres」にあるため、クリックして移動する。



実行した時間の「ビルド履歴」をクリック



実行したビルド履歴がSUCCESS、「最新のテスト結果」が（全て成功に）なっていればOK

テストしたケースを確認したい場合は「最新のテスト結果」をクリック。

テスト結果の一覧が出力される。



### エラーについての結果確認方法

ここでは故意に発生させたエラーで見方を説明するため、「spring-functionaltest-selenium-tomcat+postgres-test」で説明する。

「spring-functionaltest-selenium-tomcat+postgres」での見方も同様である。

#### テスト結果の確認

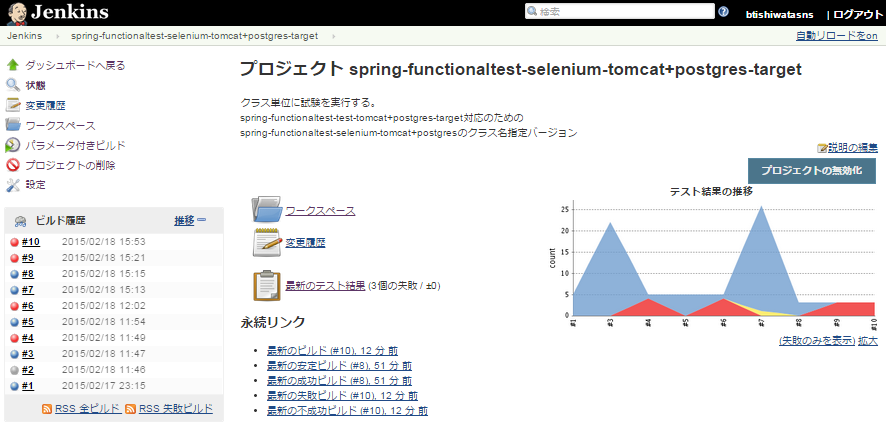
「spring-functionaltest-selenium-tomcat+postgres-test」をクリック



「spring-functionaltest-selenium-tomcat+postgres」をクリック



ビルドした時刻の「ビルド履歴」をクリック



「テスト結果」が何分のいくつ失敗したかが表示される。複数有る場合は「失敗したすべてのテストを表示」が表示されるのでクリックする。



エラーとなったテストケースをクリック。



エラーメッセージやスタックトレースが出力されているので、エラー内容を確認する。



推奨スペックに充たしていない環境でテストする場合は、

「実行順序性の調整パラメータ」を指定(変更)が必要になるケースがあるため、

1.2.4推奨構成の※を参照。

#### エビデンスの確認

エビデンスはエラーが発生したときのみ、workspaceにevidenceディレクトリが作成される。

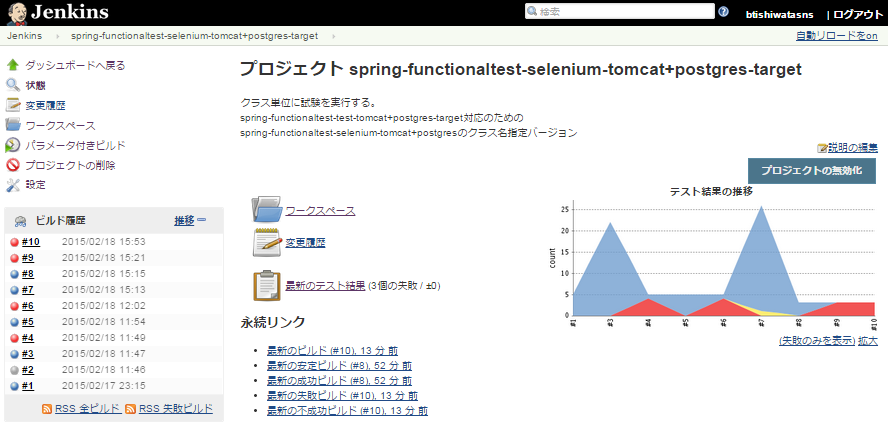
「spring-functionaltest-selenium-tomcat+postgres-test」をクリック



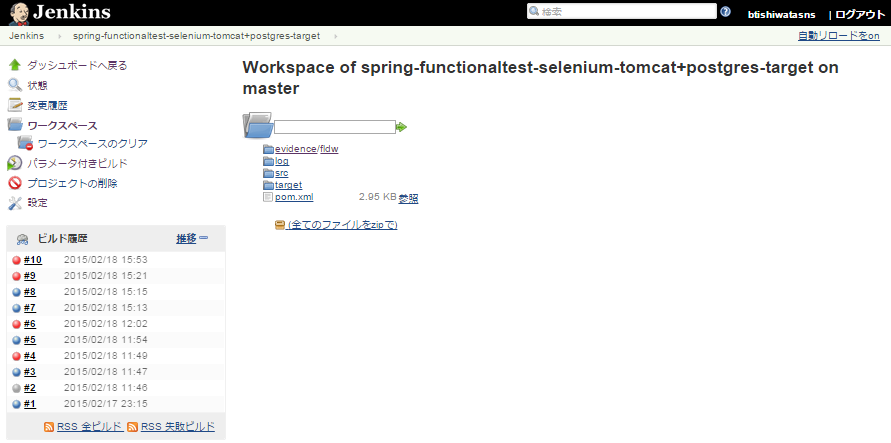
「spring-functionaltest-selenium-tomcat+postgres」をクリック



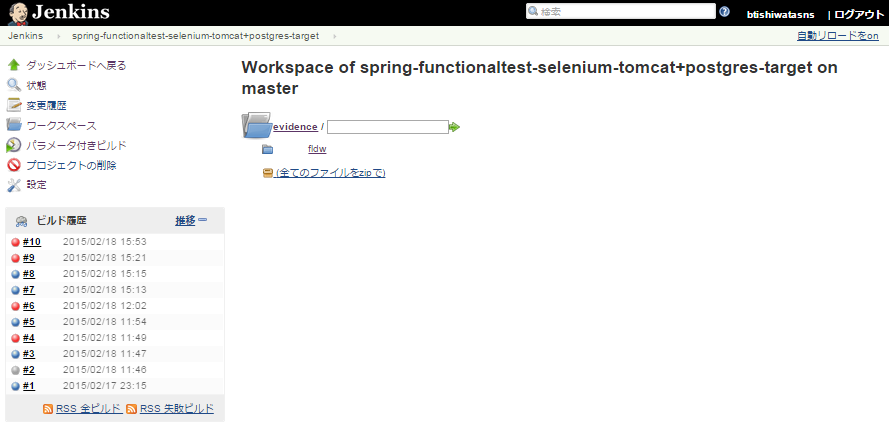
「ワークスペース」をクリック



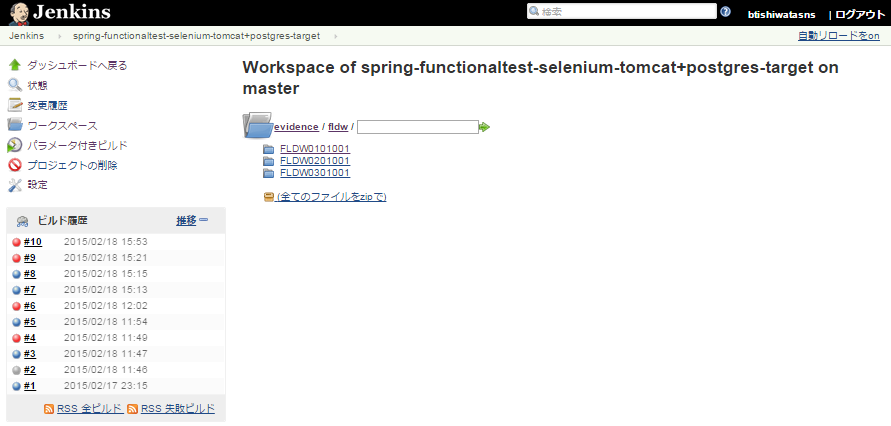
「evidence」をクリック



「機能ID」をクリック



「テストケース」をクリック



成功しているとエビデンスで見るものはほぼ無い。

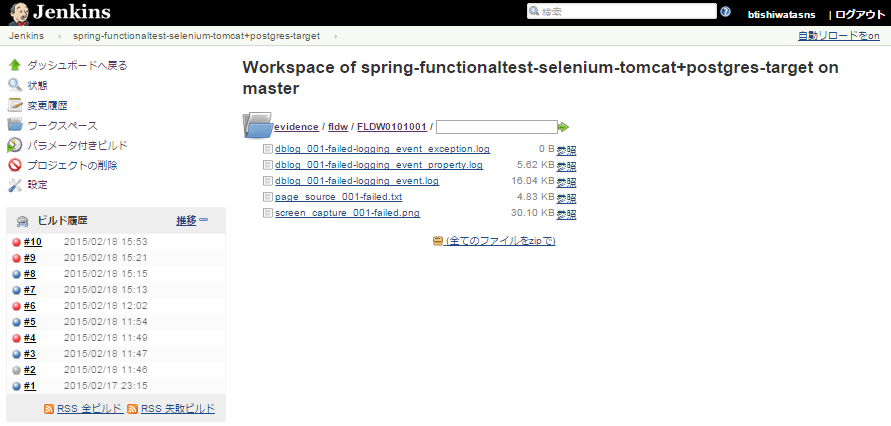


表 5.5‑1 対象エビデンス一覧

|  |  |  |
| --- | --- | --- |
| ファイル名 | 内容 | 説明 |
| dblog\_001-failed-logging\_event\_exception.log | 例外発生時のスタックトレースが出力される。 |  |
| dblog\_001-failed-logging\_event\_property.log | MDCの値を出力 | 画面エビデンスを全て取った場合、一致させることができる。 |
| dblog\_001-failed-logging\_event.log | アプリケーションでの出力ログ |  |
| page\_source\_001-failed.txt | キャプチャしたときのHTMLレスポンス（Body部のみ） |  |
| screen\_capture\_001-failed.png | テストケースでの最終画面のキャプチャ |  |