
単体テストケース (クラステスト)

| | |
|--------|--|
| 管理番号 | |
| システムID | |
| システム名称 | |
| 改定日 | |
| 改訂者 | |

■単体テスト(クラステスト)の方針

(目的)

- ・特定のクラスやメソッドが仕様通りに機能するかを確認する。
- ・クラス内の各メソッドや処理が期待通りに動作するかを「個別に確認」する。
- ・補足: クラス間の連携は単体テストの範囲外とし、主にシステムテスト(統合テスト)で確認する。

(対象範囲)

- ・「クラスやメソッド単位」を範囲とし、依存する外部要素(認証情報のデータベースやセッションへの保存など)は除外する。
- ・テスト対象の処理だけを確認し、**外部の依存関係やフレームワークによる処理はテスト対象外とする。**

(前提)

- ・外部クラスへのアクセスや連携に関わる処理は単体テストの範囲外とし、システムテスト(統合テスト)で行う。
- ・クラス内で外部クラスのメソッドを呼び出す場合、その「処理結果のみを記録」することで確認とする。
- ・フレームワーク側の処理は、結果や戻り値のみを確認し、実際の内部処理はテスト対象外とする。

(テスト方法)

- ・対象行にブレークポイントを設置する。
- ・テスト対象の操作(例:「ログインボタンをクリック」など)を行う。
- ・Eclipseのデバッグモードでテストを実行し、処理が該当行で停止することを確認する。
- ・Eclipseにて、該当行のステップオーバーを実行し、ステップオーバーの前後で「変数」ウィンドウの値を記録する。
- ・テスト結果や観測した変数の値を、テスト詳細に記録する。

■今後の方針

単体テスト終了後は、プログラムテスト(統合テスト)を実施する。

(目的)

クラス同士やモジュール間の連携が正しく機能するかを確認。

(対象範囲)

関連クラスやメソッド間の連携処理。データベースやセッション管理などの外部依存も含めてテスト。

(内容)

一連のプロセス(例: 認証結果の判定、セッションへのユーザー情報保持など)が連携し期待通りに動作するかを確認。

例: LoginAction クラスが実際にセッションへ認証情報を設定し、ユーザー権限に応じて画面遷移が行われるかを確認。

■単体テスト一覧

| No. | クラス名 | テストケース | 入力条件 | 期待する結果 | 実際の出力 | 結果 | テスト詳細 | 不具合情報 | ステータス | 備考 | |
|------|-----------------------------------|--|------|--|--|---|---|-------------------|---|---|--|
| 1.1 | LoginAction | LoginAction を介した認証結果の取得(admin) | | auth_type = "admin" | UserDAO.authenticate() メソッドが呼ばれ、変数 authenticatedUser にユーザ情報が格納され、ユーザ情報が返ることを確認。 デバッグログに「認証結果: ユーザーID=30, ユーザー名=mesuser1, 権限タイプ=1」が表示されること。 | 成功 | 認証成功、ユーザーID=30が認証されたログに出力される。 auth_type が1の場合、ログに「管理権限」。 | 成功 | 単体テストNo1.1 | 完了 | |
| 1.2 | LoginAction | 認証結果の判定(admin) | | auth_type = "admin" | auth_type が1である場合に、「管理権限」と、「認証結果: 管理権限での処理を実行 (adminを送る) (adminを送る)」のログが出力され、 "admin" が返されることを確認する。 | 成功 | 「認証結果: 管理権限での処理を実行 (adminを送る)」が出力され、メソッドが "admin" を返す。 | 無し | 単体テストNo1.2 | 完了 | |
| 1.3 | LoginAction | 認証失敗時の処理(admin) | | auth_type = "admin" | ・return LOGIN; が呼ばれ、ログイン画面に戻ることを確認する ・addActionError("ユーザー名かパスワードが不正です。") が呼ばれた際、エラーメッセージが表示されることを確認する ・ログイン画面に「ユーザー名かパスワードが不正です。」のメッセージが表示されること ・デバッグログで「認証失敗」が出力されること。 UserDAO.authenticate() メソッドが呼ばれ、変数 authenticatedUser が適切なユーザ情報を返ること。 デバッグログに「認証結果: ユーザーID=30, ユーザー名=mesuser1, 権限タイプ=0」が表示されること。 | return LOGIN によりログイン画面に遷移し、ログイン画面に「ユーザー名かパスワードが不正です。」の表示を確認。 デバッグログに「認証失敗」が出力されていることを確認。 | 成功 | 単体テストNo1.3 | 完了 | | |
| 2.1 | LoginAction | LoginAction を介した認証結果の取得(user) | | auth_type = "user" | auth_type が0である場合に、「一般ユーザー権限」と、「認証結果: 一般ユーザー権限での処理を実行 (userを送る)」のログが出力され、 "user" が返されることを確認する。 | 成功 | 認証成功、ユーザーID=30が認証されたログに出力される。 auth_type が0の場合、ログに「一般ユーザー権限」。 | 成功 | 単体テストNo2.1 | 完了 | |
| 2.2 | LoginAction | 認証結果の判定(user) | | auth_type = "user" | auth_type が0である場合に、「一般ユーザー権限」と、「認証結果: 一般ユーザー権限での処理を実行 (userを送る)」のログが出力され、 "user" が返されることを確認する。 | 成功 | 「認証結果: 一般ユーザー権限での処理を実行 (userを送る)」が出力され、メソッドが "user" を返す。 | 無し | 単体テストNo2.2 | 完了 | |
| 2.3 | LoginAction | 認証失敗時の処理(user) | | auth_type = "user" | ・return LOGIN; が呼ばれ、ログイン画面に戻ることを確認する ・addActionError("ユーザー名かパスワードが不正です。") が呼ばれた際、エラーメッセージが表示されることを確認する ・ログイン画面に「ユーザー名かパスワードが不正です。」のメッセージが表示されること ・デバッグログで「認証失敗」が出力されること。 | return LOGIN によりログイン画面に遷移し、ログイン画面に「ユーザー名かパスワードが不正です。」の表示を確認。 デバッグログに「認証失敗」が出力されていることを確認。 | 成功 | 単体テストNo2.3 | 完了 | | |
| 3.1 | UserDAO | ユーザー認証処理 | | 有効なユーザー名とパスワード | ・LoginActionクラスから渡されたユーザー名とパスワードを使ってデータベースからユーザー情報を取得できること ・取得したユーザー情報(ユーザーID, ユーザー名, 権限タイプなど)が期待値であること。 ・正しいユーザー情報を LoginAction クラスに返すことができること。 | ・user_name は "admin999999" に設定されていた ・password は "12345678" に設定されていた | 成功 | 単体テストNo3.1 | 完了 | このクラスの課題 ★セキュリティの観点で改善すべき点: 1.パスワードのハッシュ化(例: bcrypt)して格納し、認証時にハッシュを照合する。 2.2人か1人か(セッション/SQLインジェクション)によるため、ユーザー名やパスワードに対するチェックを強化する。 3.エラーハンドリング: 内部エラーメッセージやログメッセージをログに出力しつつ、ユーザーには簡潔な誤りメッセージを送る。 4.ログ管理: 個人情報やパスワードがログに出力されないようにする。(本番リリース時にデバッグログのオフ) 5.セッション: 管理の強化、セッションIDの再生成とタイムアウト設定でセッション固定攻撃に対応する。 ●セキュリティ対策 1.パスワードはハッシュ化(例: bcrypt)して格納し、認証時にハッシュを照合する。 2.SQLにおける最小権限の使用 3.エラーハンドリング: 内部エラーメッセージをログに出力しつつ、ユーザーには簡潔な誤りメッセージを送る。 | |
| 3.2 | UserDAO | 認証失敗処理 | | | ・LoginActionクラスから渡されたユーザー名とパスワードを使ってデータベースからユーザー情報を取得できること ・取得したユーザー情報(ユーザーID, ユーザー名, 権限タイプなど)が期待値であること。 ・正しいユーザー情報を LoginAction クラスに返すことができること。 | ・user_name は "admin999999" に設定されていた ・password は "12345678" に設定されていた | 成功 | 単体テストNo3.2 | 完了 | ●セキュリティ対策 1.パスワードはハッシュ化(例: bcrypt)して格納し、認証時にハッシュを照合する。 2.SQLにおける最小権限の使用 3.エラーハンドリング: 内部エラーメッセージをログに出力しつつ、ユーザーには簡潔な誤りメッセージを送る。 | |
| 4 | LogoutAction | ログアウト処理 | | ユーザーがログインしており、セッションがアクティブ状態で「ログアウト」ボタンをクリック。 | ・現在のセッションが取得され、無効化されること。 ・session.isValid が false になること。 ・無効化後、ログイン画面へ遷移すること。 ・ログアウト時のデバッグログが出力されること。 | ・現在のセッションが取得され、session.isValid が false であることを確認。 ・session.isValid が false であることを確認。 ・ログイン画面への遷移が確認できること。 ・ログアウト時のデバッグログが出力されたことを確認。 | 成功 | 単体テストNo4 | 完了 | | |
| 5.1 | MoveBulletinboardManagementAction | 掲示板管理画面への遷移処理 | | ・セッション情報 ・ログイン済みのユーザーセッションが存在している。 ・セッションには正しいユーザー情報(user_id, role など)が含まれている。 ・データベース状態 ・bulletinboardテーブルにデータが存在する(例:13レコード)。 ・各レコードには以下のフィールドが含まれる。 ・bulletinboard_id ・bulletinboard_title ・bulletinboard_content ・実行条件 ・アクションメソッド MoveBulletinboardManagementAction.execute() を実行呼び出す。 | 1.正常遷移の確認 ・表示はSUCCESSであること。 ・bulletinboardのリストが最新のデータで更新されていること。 2.表示項目のみの確認 ・リストのサイズがデータベースのbulletinboardテーブルのレコード数と一致していること。 ・各オブジェクトのフィールド(bulletinboard_id, bulletinboard_title, bulletinboard_content)などが期待される値であること。 | 1.表示は: SUCCESS 2.bulletinboards リスト ・リストサイズ: 13 ・各要素: ・例: 1件目 ・bulletinboard_id: 34 ・bulletinboard_title: 2024年7月18日 ・bulletinboard_content: 2024年7月18日平日掲示板、追記 3.エラーや例外の発生: なし | 成功 | 単体テストNo5.1 | 完了 | ・デバッグログで (rs.next()) の戻り値を明示(単体テスト用) コードに一時対応のログを追加して、(rs.next()) の戻り値を確認確認した。 事前にフック変数に結果を格納する処理を実装。 | |
| 5.2 | MoveBulletinboardManagementAction | セッションと、データベースの異常系処理 | | 1.セッション未設定の場合 ・管理メニュー画面にログインせず掲示板管理画面へのリンクを直接アクセスする。 2.データベース接続エラー ・正常にログインした状態でMySQLのサービスを一時停止し、掲示板管理画面へのリンクをクリックする。 | 1.セッションが未設定の場合、適切なエラー処理が行われることを確認 <期待する結果> ・戻り値がERRORであること。 ・addActionErrorに指定されたメッセージが設定されていること。 ・loggerにエラーログが記録されていること。 2.データベース接続エラーが発生した場合、適切に例外がキャッチされ、ログが記録されることを確認 <期待する結果> ・戻り値がERRORであること。 ・loggerに例外メッセージが記録されていること。 3.より実行中にエラーが発生した場合、例外が正しいハンドリングされることを確認 <期待する結果> ・戻り値がERRORであること。 ・loggerにエラーログが記録されていること。 | 1.セッションが未設定の場合 ・戻り値: ERROR ・エラーメッセージ: セッションが有効ではありません ・ログ内容: [ERROR] セッションが未設定のためエラーが発生しました 2.データベース接続エラー ・戻り値: ERROR ・ログ内容: [ERROR] データベース接続中にエラーが発生しました (例外内容) | 期待通り | 単体テストNo5.2 | 完了 | ・以下改善 エラー画面への遷移処理と、画面の実装 | |
| 6.1 | MoveUserManagementAction | ユーザー管理画面への遷移処理 | | ・セッション情報 ・ログイン済みのユーザーセッションが存在している。 ・セッションには正しいユーザー情報(user_id, role など)が含まれている。 ・データベース状態 ・users テーブルにデータが存在する(例:12レコード)。 ・各レコードには以下のフィールドが含まれる。 ・user_id ・user_name ・password ・auth_type ・delete_flag ・delete_day ・実行条件 ・アクションメソッド MoveUserManagementAction.execute() を実行呼び出す。 | 1.正常遷移の確認 ・表示はSUCCESSであること。 ・usersリストが最新のデータで更新されていること。 2.ユーザーリストのみの確認 ・リストのサイズがデータベースのusersテーブルのレコード数と一致していること。 ・各オブジェクトのフィールド(user_id, user_name, password, auth_type, delete_flag, など)が期待される値であること。 | 1.表示は: SUCCESS 2.users リスト ・リストサイズ: 12 ・各要素: ・例: 1件目 ・user_id: 15 ・user_name: hidetaka44 ・password: r5hV4kb ・auth_type: 1 ・delete_flag: 1 ・delete_day: 2025-12-31 00:00:00.000000 3.エラーや例外の発生: なし | 成功 | 単体テストNo6.1 | 完了 | ・デバッグログで (rs.next()) の戻り値を明示(単体テスト用) コードに一時対応のログを追加して、(rs.next()) の戻り値を確認確認した。 事前にフック変数に結果を格納する処理を実装。 | |
| 6.2 | MoveUserManagementAction | セッションと、データベースの異常系処理 | | 1.セッション未設定の場合 ・管理メニュー画面にログインせずユーザー管理画面へのリンクを直接アクセスする。 2.データベース接続エラー ・正常にログインした状態でMySQLのサービスを一時停止し、ユーザー管理画面へのリンクをクリックする。 | 1.セッションが未設定の場合、適切なエラー処理が行われることを確認 <期待する結果> ・戻り値がERRORであること。 ・addActionErrorに指定されたメッセージが設定されていること。 ・loggerにエラーログが記録されていること。 2.データベース接続エラーが発生した場合、適切に例外がキャッチされ、ログが記録されることを確認 <期待する結果> ・戻り値がERRORであること。 ・loggerに例外メッセージが記録されていること。 3.より実行中にエラーが発生した場合、例外が正しいハンドリングされることを確認 <期待する結果> ・戻り値がERRORであること。 ・loggerにエラーログが記録されていること。 | 1.セッションが未設定の場合 ・戻り値: ERROR ・エラーメッセージ: セッションが有効ではありません ・ログ内容: [ERROR] セッションが未設定のためエラーが発生しました 2.データベース接続エラー ・戻り値: ERROR ・ログ内容: [ERROR] データベース接続中にエラーが発生しました (例外内容) | 成功 | 単体テストNo6.2 | 完了 | ・以下改善 エラー画面への遷移処理と、画面の実装 | |
| 7.1 | MovePostManagementAction | 投稿管理画面への遷移 | | | | | | | | | |
| 7.2 | MovePostManagementAction | セッションと、データベースの異常系処理 | | | | | | | | | |
| 8 | goToManagementMenuAction | 管理メニューへ戻る処理 | | | ・リンクをクリックした際に、ManagementMenu.jsp にリダイレクトされ、管理メニュー画面が表示される。 ・ブラウザのアドレスバーに /view/ManagementMenu.jsp が表示されること。 | | 正常に ManagementMenu.jsp が表示されることを確認。 | 成功 | 単体テストNo8 | 完了 | |
| 9.1 | MoveCreateBulletinboardAction | 掲示板作成画面への遷移 | | | ・ボタンをクリックした際に、CreateBulletinboardScreen.jspにリダイレクトされ、掲示板作成画面が表示される。 ・ブラウザのアドレスバーに http://localhost:8080/Bulletinboard/CreateBulletinboardScreen.action が表示されること。 | | 正常に CreateBulletinboardScreen.jsp が表示されることを確認。 | 成功 | 単体テストNo9.1 | 完了 | |
| 9.2 | MoveCreateBulletinboardAction | 掲示板作成画面遷移時の、異常系処理 | | 1.セッション未設定の場合 <期待する結果> ・戻り値がERRORであること。 ・addActionErrorに指定されたメッセージが設定されていること。 ・loggerにエラーログが記録されていること。 | 1.セッションが未設定の場合 ・戻り値: ERROR ・ログ内容: Error in MoveBulletinboardManagementAction: User session is missing. | 成功 | 単体テストNo9.2 | 完了 | | | |
| 10.1 | InsertBulletinboardAction | 「bulletinboard」テーブルへの登録処理 | | ・「作成」ボタンをクリックする操作。 ・前提としては、セッションにログイン済みのユーザー情報(user_id)が含まれている。入力データ(掲示板タイトル、内容、削除フラグ、削除日)は正しい形式で提供されていること。 | ・データベースに正しいレコードが挿入される(挿入後、該当データをクエリで確認)こと。 ・戻り値が"success"である。 | | 正常に掲示板データが、掲示板テーブルに登録されたことを確認。 | 成功 | 単体テストNo10.1 | 完了 | |
| 10.2 | InsertBulletinboardAction | InsertBulletinboardActionの異常系処理 | | 1.セッション未設定の場合 ・掲示板管理画面にログインせず項目を入力後、セッションを削除し「作成」ボタンを押下する。 2.データベース接続エラー ・「掲示板作成」画面に必要項目を入力後、データベースを停止し「作成」ボタンを押下する。 | 1.セッションが未設定の場合、適切なエラー処理が行われることを確認 <期待する結果> ・戻り値がERRORであること。 ・addActionErrorに指定されたメッセージが設定されていること。 ・loggerにエラーログが記録されていること。 2.データベース接続エラーが発生した場合、適切に例外がキャッチされ、ログが記録されることを確認 <期待する結果> ・戻り値がERRORであること。 ・loggerに例外メッセージが記録されていること。 3.より実行中にエラーが発生した場合、例外が正しいハンドリングされることを確認 <期待する結果> ・戻り値がERRORであること。 ・loggerにエラーログが記録されていること。 | 1.セッションが未設定の場合 ・戻り値: ERROR ・エラーメッセージ: セッションが有効ではありません ・ログ内容: [ERROR] セッションが未設定のためエラーが発生しました 2.データベース接続エラー ・戻り値: ERROR ・ログ内容: [ERROR] データベース接続中にエラーが発生しました (例外内容) | 成功 | 単体テストNo10.2 | 完了 | | |
| 11.1 | EditBulletinboardAction | 掲示板編集画面のデータ表示処理 | | ・掲示板の「編集」リンクをクリックする操作。 ・前提条件として、セッションが有効であり、掲示板編集画面に正常にアクセスできていること。 ・掲示板編集画面に、該当掲示板の既存データが表示されること。 | ・表示はSUCCESSであること。 ・セッションが有効で、ログインユーザーの情報が保持されていること。 ・掲示板編集画面の表示項目 ・フォーム欄に該当掲示板の既存データが正しく表示されていること。 ・各項目 (bulletinboard_id, bulletinboard_title, bulletinboard_content など) の値が期待値と一致していること。 | 1.表示は: SUCCESS 2.掲示板編集画面の内容 ・項目名 表示内容 ・ ・ bulletinboard_title 2025年1月1日掲示板 ・ bulletinboard_content 1月8日に作成 ・ bulletinboard_delete_flag 削除しない ・ bulletinboard_delete_day 2050-12-31 04:00:00 3.エラーや例外の発生: なし | 成功 | 単体テストNo11.1 | 該当テストケースで特記事項なし(セッション保持、フォーム表示、例外発生なし)。 | 完了 | 単体テスト項目に以下のリンクを一緒に追加 ・セッションがユーザー情報を保持する処理 ・接続が有効かどうかを確認する処理 ・データベース接続確認をログに出力する処理 ※通常運用時には無効化(コメントアウト)予定。 |
| 11.2 | EditBulletinboardAction | EditBulletinboardAction - セッション未設定時およびDB接続エラーのハンドリング検証 | | 1.セッション未設定の場合 ・「掲示板管理」画面でセッションを削除した状態で、該当掲示板の「編集」リンクをクリックする。 2.データベース接続エラー ・「掲示板作成」画面に必要項目を入力後、データベースを停止し「作成」ボタンを押下する。 ・該当掲示板の「編集」リンクをクリックする。 | 1.セッションが未設定の場合、適切なエラー処理が行われることを確認 <期待する結果> ・戻り値がERRORであること。 ・loggerにエラーログが記録されていること。 2.データベース接続エラーが発生した場合、適切に例外がキャッチされ、ログが記録されることを確認 <期待する結果> ・戻り値がERRORであること。 ・loggerに例外メッセージが記録されていること。 | 1.セッションが未設定の場合 ・戻り値: ERROR ・エラーメッセージ: セッションが有効ではありません ・ログ内容: [ERROR] セッションが未設定のためエラーが発生しました 2.データベース接続エラー ・戻り値: ERROR ・ログ内容: [ERROR] データベース接続中にエラーが発生しました (例外内容) | 成功 | 単体テストNo11.2 | 該当テストケースで特記事項なし。 | 完了 | <課題> ・MySQL停止操作の自動化案 ・手動操作が難しいため、自動化スクリプトを作成する。 ・Windows環境では PowerShell スクリプトを活用し、MySQL サービスの停止/起動を自動化する。 例: Stop-Service -Name "mysqld" Start-Sleep -Seconds 10 Start-Service -Name "mysqld" ・Linux環境では、システムスクリプトを作成し、systemctl または service コマンドで同様の操作を実現する。 ・エラー画面のHTML要素の検証が不足している。エラーメッセージやリンクの正確性をHTML要素レベルで検証することを検討する。 例: エラー画面のDOM構造における <div> 要素や <p> 要素の内容をチェック。 ・自動化テストの活用 ・Selenium などを使用し、エラー画面のDOM要素やリンクの動作をプログラムで検証。 ・定期的検証ツールを用いて、エラー画面のHTML構造の正確性を確認。 |
| 12 | UpdateBulletinboardAction | UpdateBulletinboardAction - 「bulletinboard」テーブルへの更新処理 | | ・掲示板編集画面の「更新」ボタンをクリックする操作。 ・前提条件として、セッションが有効であること。 ・掲示板編集画面に正常にアクセスできていること。 ・掲示板編集画面のフォームに既存データを編集後、更新処理を実行。 | ・データベースに正しいレコードが挿入される(挿入後、該当データをクエリで確認)こと。 ・戻り値が"success"である。 | 1.表示は: SUCCESS 2.掲示板編集画面の内容 ・更新後タイトル: 2025年1月15日掲示板 ・掲示板本文: 1月15日に更新 ・掲示板削除フラグ: 削除する ・掲示板削除日: 2050-12-31 04:00:00 3.エラーや例外の発生: なし | 成功 | 単体テストNo12.1 | 単体テスト No.11.1 に基づく処理、エラーや例外が発生しないことを確認済み。 | 完了 | ■追加内容 ・SQLのエラーを分類するための SQLStateコードを取得する処理を実装。 ・取得したコードに基づいて適切なエラー処理を追加。 ・今回のテストケースでは SQLState による接続エラー例外は発生せず、正常に動作。 |
| 12.2 | UpdateBulletinboardAction | UpdateBulletinboardAction - セッション未設定時およびDB接続エラーのハンドリング検証 | | 1.セッション未設定の場合 ・「掲示板管理」画面でセッションを削除した状態で、掲示板編集画面の「更新」ボタンをクリックする。 2.データベース接続エラー ・「掲示板作成」画面に必要項目を入力後、データベースを停止し「更新」ボタンをクリックする。 ・該当掲示板の「更新」ボタンをクリックする。 | 1.セッションが未設定の場合、適切なエラー処理が行われることを確認 <期待する結果> ・戻り値がERRORであること。 ・loggerにエラーログが記録されていること。 2.データベース接続エラーが発生した場合、適切に例外がキャッチされ、ログが記録されることを確認 <期待する結果> ・戻り値がERRORであること。 ・loggerに例外メッセージが記録されていること。 | 1.セッションが未設定の場合 ・戻り値: ERROR ・エラーメッセージ: エラーが発生しました ・ログ内容: User session is missing. 2.データベース接続エラー ・戻り値: ERROR ・ログ内容: [ERROR] データベース接続中にエラーが発生しました (例外内容) | 成功 | 単体テストNo12.2 | 該当テストケースで特記事項なし。 | 完了 | <課題> ・MySQL停止操作の自動化案 ・手動操作が難しいため、自動化スクリプトを作成する。 ・Windows環境では PowerShell スクリプトを活用し、MySQL サービスの停止/起動を自動化する。 例: Stop-Service -Name "mysqld" Start-Sleep -Seconds 10 Start-Service -Name "mysqld" ・Linux環境では、システムスクリプトを作成し、systemctl または service コマンドで同様の操作を実現する。 ・エラー画面のHTML要素の検証が不足している。エラーメッセージやリンクの正確性をHTML要素レベルで検証することを検討する。 例: エラー画面のDOM構造における <div> 要素や <p> 要素の内容をチェック。 ・自動化テストの活用 ・Selenium などを使用し、エラー画面のDOM要素やリンクの動作をプログラムで検証。 ・定期的検証ツールを用いて、エラー画面のHTML構造の正確性を確認。 |
| 13.1 | DeleteBulletinboardAction | 掲示板削除処理 | | ・掲示板一覧にて、テスト対象の掲示板の「削除」リンクをクリックする操作。 | ・掲示板管理画面で、掲示板が削除できること。 ・戻り値が"success"である。 | 1.表示は: SUCCESS 2.掲示板削除後の内容 ・削除後タイトル: 2025年1月15日掲示板 ・掲示板本文: 1月15日に更新 ・掲示板削除フラグ: 削除する ・掲示板削除日: 2050-12-31 04:00:00 3.エラーや例外の発生: なし | 成功 | 単体テストNo13.1 | エラーや例外が発生しないことを確認済み。 | 完了 | ■追加内容 ・SQLStateが "08" で始まる場合、データベース接続エラーとして処理を追加。 |
| 13.2 | DeleteBulletinboardAction | DeleteBulletinboardAction - セッション未設定時およびDB接続エラーのハンドリング検証 | | 1.セッション未設定の場合 ・「掲示板管理」画面でセッションを削除した状態で、掲示板一覧の「削除」リンクをクリックする。 2.データベース接続エラー ・「掲示板作成」画面に必要項目を入力後、データベースを停止し「削除」ボタンをクリックする。 ・該当掲示板の「削除」リンクをクリックする。 | 1.セッションが未設定の場合、適切なエラー処理が行われることを確認 <期待する結果> ・戻り値がERRORであること。 ・loggerにエラーログが記録されていること。 2.データベース接続エラーが発生した場合、適切に例外がキャッチされ、ログが記録されることを確認 <期待する結果> ・戻り値がERRORであること。 ・loggerに例外メッセージが記録されていること。 | 1.セッションが未設定の場合 ・戻り値: ERROR ・エラーメッセージ: エラーが発生しました ・ログ内容: User session is missing. 2.データベース接続エラー ・戻り値: ERROR ・ログ内容: [ERROR] データベース接続中にエラーが発生しました (例外内容) | 成功 | 単体テストNo13.2 | 該当テストケースで特記事項なし。 | 完了 | <課題> ・MySQL停止操作の自動化案 ・手動操作が難しいため、自動化スクリプトを作成する。 ・Windows環境では PowerShell スクリプトを活用し、MySQL サービスの停止/起動を自動化する。 例: Stop-Service -Name "mysqld" Start-Sleep -Seconds 10 Start-Service -Name "mysqld" ・Linux環境では、システムスクリプトを作成し、systemctl または service コマンドで同様の操作を実現する。 ・エラー画面のHTML要素の検証が不足している。エラーメッセージやリンクの正確性をHTML要素レベルで検証することを検討する。 例: エラー画面のDOM構造における <div> 要素や <p> 要素の内容をチェック。 ・自動化テストの活用 ・Selenium などを使用し、エラー画面のDOM要素やリンクの動作をプログラムで検証。 ・定期的検証ツールを用いて、エラー画面のHTML構造の正確性を確認。 |
| 14.1 | CancelAction | 掲示板管理の「キャンセル」ボタン操作 | | ・掲示板管理画面が表示され、適切なセッション情報が存在すること。 | ・セッションが有効であること。 ・セッション変数 user_id が null でないことを確認など。 ・ブラウザで正しいURLが提供されていることを確認する。 ・HTTP POSTリクエストの送信を確認(パラメータ: action=cancel)。 ・上記確認後、cancelの文字列が返すことを確認する。 | 1.表示は: cancel 2.エラーや例外の発生: なし(ログにエラーメッセージが記録されていないことと確認) | 成功 | 単体テストNo14.1 | 期待値で不具合は確認されていない。 | 完了 | ■追加内容 ・リクエストパラメータやセッション情報(権限)を確認し、適切なエラー処理をBaseActionクラスへ押し付け統合化 ・BaseActionクラスのrequestパラメータを正しく確保しているか確認するデバッグ処理を追加。 ・権限後、リクエスト内部の権限のデバッグ処理を追加。 ・デバッグ時にセッションからユーザー情報を取得する処理を追加 ・追加したデバッグ処理は、通常運用時にコメントアウトする予定 ・権限テスト用デバッグのデバッグ処理を追加 |
| 14.2 | CancelAction | CancelAction - セッション未設定時およびDB接続エラーのハンドリング検証 | | 1.セッション未設定の状態 2.actionパラメータが null の状態で CancelAction を実行 3.actionパラメータが "invalid" の状態で CancelAction を実行 | 1.セッションが未設定の場合、ログに User session is missing. が記録される 2.エラーメッセージが null の状態で Invalid action parameter. Expected 'cancel', but received: null が記録される 3.エラー画面が表示され、ログに Invalid action parameter. Expected 'cancel', but received: invalid が記録される | 期待する結果と一致(ログ出力および画面表示を確認済み)。 | 成功 | 単体テストNo14.2 | -HttpServletRequest が null の場合の処理を BaseAction クラスに追加 -アクションパラメータが "cancel" 以外だった場合のログメッセージを汎用性のある内容に変更 | 完了 | |
| 15.1 | ListAction | キャンセルボタンクリック後の、掲示板管理画面一覧再読み込みする処理 | | 掲示板管理画面の画面(例: 掲示板作成画面)にてキャンセルボタンをクリックし、再読み込みの処理を実行すること。 | 期待する結果と一致(ログ出力および画面表示を確認済み)。 | 成功 | 単体テストNo15.1 | 期待値で不具合は確認されていない。 | 完了 | ■改善点 ・削除フラグの影響が明確なようにデバッグログ出力を改善 | |

■単体テストNo1.1

ユーザー認証処理(LoginAction のユーザー認証確認)

■目的

LoginAction クラスが UserDao の authenticate() メソッドを正しく呼び出し、指定されたユーザー情報で認証を実行し、その結果を取得できるかを確認する。

■対象行

115行目 : User authenticatedUser = UserDao.authenticate(user.getUser_name(), user.getPassword());

■テスト方法

- 1.対象行にブレークポイントを設置。
- 2.Eclipse のデバッグモードでテストを実行し、対象行の前後でauthenticatedUser の値を Eclipse の「変数」ウィンドウで確認。
- 3.確認結果を以下のように記録。

■期待されるテスト結果

- ・authenticatedUser の取得結果が以下を満たす場合に、正しい認証と判断する。
 - ・auth_type が「1」
 - ・delete_flag が「0」(有効なユーザー)。
 - ・user_name と user_id が入力したユーザー情報と一致。
 - ・ログに ユーザーID, ユーザー名, 権限タイプ が表示される。

■テスト実施内容

- ・テスト前(ステップオーバー前)
 - ・Eclipse「変数」ウィンドウの authenticatedUser は未設定状態 (null または変数が存在しない)。
- ・テスト後(ステップオーバー後)
 - ・Eclipse「変数」ウィンドウに authenticatedUser が表示され、以下のような値を持つ:
 - ・auth_type: 1
 - ・user_id: 30
 - ・user_name: tesutuser1

■デバッグログ出力

[2024-11-11 16:35:41.839] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.login.LoginAction - 認証結果: ユーザーID=30, ユーザー名=tesutuser1, 権限タイプ=1

以上

■単体テスト No1.2

認証結果の判定(LoginAction のユーザー認証結果判定)

■目的

`auth_type` が 1 である場合に、「管理者権限」のログが出力され,"admin" がリターンされることを確認する。

■対象行

```
152行目:if (authenticatedUser.getAuth_type() == 1) {
153行目:    logger.debug("管理者権限");
154行目:    logger.debug("認証結果: 管理者権限での処理を実行(adminを返す)");
155行目:    return "admin";
156行目:} else {
157行目:    logger.debug("一般ユーザー権限");
158行目:    logger.debug("認証結果: 一般ユーザー権限での処理を実行(userを返す)");
159行目:    return "user";
160行目:}
```

■テスト方法

- 1.対象行にブレークポイントを設置。
- 2.Eclipse のデバッグモードでテストを実行し、「変数」ウィンドウでauth_type` の値が 1 であることを確認
- 3.`auth_type` が 1 である場合にデバッグログの出力内容を確認
- 4.確認結果を以下のように記録。

■期待されるテスト結果

- `auth_type` が 1 の場合:
 - `auth_type` の取得結果が「1」であること
 - ログに「管理者権限」が出力されていること
 - メソッドが "admin" を返すこと
 - ログに「認証結果: 管理者権限での処理を実行(adminを返す)」が出力されること

■テスト実施内容

- ・テスト前(ステップオーバー前)
 - Eclipse「変数」ウィンドウのauth_typeは未設定状態(null または変数が存在しない)。
- ・テスト後(ステップオーバー後)
 - Eclipse「変数」ウィンドウに`auth_type` が表示され、期待通り「1」の値を持つことを確認
 - getAuth_type() 戻り値が期待通りの「1」であることを確認

■デバッグログ出力

[2024-11-12 11:20:36.284] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.login.LoginAction - 管理者権限

[2024-11-12 12:04:32.268] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.login.LoginAction - 認証結果: 管理者権限での処理を実行(adminを返す)

以上

■単体テストNo1.3

認証失敗時の処理(LoginActionのユーザー認証失敗時の処理)

■目的

ログイン画面にて無効なユーザー名やパスワードを入力し、認証失敗時の処理が実行されるかを確認する

■テスト対象行

テスト対象の以下のコードにブレークポイントを設置

```
164行目 :logger.debug("認証失敗");
165行目 :addActionError("ユーザー名かパスワードが不正です。");
166行目 :return LOGIN;
```

■テスト方法

- 1.対象行にブレークポイントを設置
- 2.ログイン画面にて「無効なユーザー名やパスワード」を入力し、ログインボタンをクリックする
- 3.Eclipseのデバッグモードでテストを実行し、処理が該当行で停止することを確認
- 4.デバッグログに「認証失敗」の出力内容を確認
- 5.認証失敗後、ログイン画面に戻り、「ユーザー名かパスワードが不正です。」のエラーメッセージが表示されることを確認
- 6.確認結果を以下のように記録

■期待される結果

- ・return LOGIN; が呼ばれ、ログイン画面に戻ることを確認する
- ・addActionError("ユーザー名かパスワードが不正です。"); が呼ばれた際、エラーメッセージが表示されることを確認する
→ ログイン画面上に「ユーザー名かパスワードが不正です。」のメッセージが表示されること
- ・デバッグログで「認証失敗」が出力されること

■テスト実施内容

- ・画面遷移とエラー表示の確認
 - ・return LOGIN; によりログイン画面に遷移し、ログイン画面に「ユーザー名かパスワードが不正です。」の表示を確認
- ・デバッグログ
 - ・以下のメッセージが記録されていることを確認
[2024-11-12 16:25:37.884] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.login.LoginAction - 認証失敗

以上

■単体テストNo2.1

ユーザー認証処理(LoginAction のユーザー認証確認)

■目的

LoginAction クラスが UserDao の authenticate() メソッドを正しく呼び出し、指定されたユーザー情報で認証を実行し、その結果を取得できるかを確認する。

■対象行

115行目 : User authenticatedUser = UserDao.authenticate(user.getUser_name(), user.getPassword());

■テスト方法

- 1.対象行にブレークポイントを設置。
- 2.Eclipse のデバッグモードでテストを実行し、対象行の前後でauthenticatedUser の値を Eclipse の「変数」ウィンドウで確認。
- 3.確認結果を以下のように記録。

■期待されるテスト結果

- ・authenticatedUser の取得結果が以下を満たす場合に、正しい認証と判断する。
 - ・auth_type が「0」。
 - ・delete_flag が「0」(有効なユーザー)。
 - ・user_name と user_id が入力したユーザー情報と一致。
 - ・ログに ユーザーID, ユーザー名, 権限タイプ が表示される。

■テスト実施内容

- ・テスト前(ステップオーバー前)
 - ・Eclipse「変数」ウィンドウの authenticatedUser は未設定状態 (null または変数が存在しない)。
- ・テスト後(ステップオーバー後)
 - ・Eclipse「変数」ウィンドウに authenticatedUser が表示され、以下のような値を持つ:
 - ・auth_type: 0
 - ・user_id: 36
 - ・user_name: tesutuser11

■デバッグログ出力

[2024-11-12 17:47:37.784] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.login.LoginAction - 認証結果: ユーザーID=36, ユーザー名=tesutuser11, 権限タイプ=0

以上

■単体テストNo2.2

認証結果の判定(LoginAction のユーザー認証結果判定)

■目的

`auth_type` が 0 である場合に、「一般ユーザー権限」のログが出力され,"user" がリターンされることを確認する。

■対象行

```
152行目:if (authenticatedUser.getAuth_type() == 1) {
153行目:    logger.debug("管理者権限");
154行目:    logger.debug("認証結果: 管理者権限での処理を実行(adminを返す)");
155行目:    return "admin";
156行目:} else {
157行目:    logger.debug("一般ユーザー権限");
158行目:    logger.debug("認証結果: 一般ユーザー権限での処理を実行(userを返す)");
159行目:    return "user";
160行目:}
```

■テスト方法

- 1.対象行にブレークポイントを設置。
- 2.Eclipse のデバッグモードでテストを実行し、「変数」ウィンドウでauth_type` の値が 0 であることを確認
- 3.`auth_type` が 0である場合にデバッグログの出力内容を確認
- 4.確認結果を以下のように記録。

■期待されるテスト結果

- `auth_type` が 0 の場合 :
 - `auth_type` の取得結果が「0」であること
 - ログに「一般ユーザー権限」が出力されていること
 - メソッドが "user" を返すこと
 - ログに「認証結果: 一般ユーザー権限での処理を実行(userを返す)」が出力されること

■テスト実施内容

- ・テスト前(ステップオーバー前)
 - Eclipse「変数」ウィンドウのauth_typeは未設定状態(null または変数が存在しない)。
- ・テスト後(ステップオーバー後)
 - Eclipse「変数」ウィンドウに`auth_type` が表示され、期待通り「0」の値を持つことを確認
 - getAuth_type() 戻り値が期待通りの「0」であることを確認

■デバッグログ出力

[2024-11-12 18:02:37.015] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.login.LoginAction - 一般ユーザー権限

[2024-11-12 18:02:37.963] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.login.LoginAction - 認証結果: 一般ユーザー権限での処理を実行(userを返す)

以上

■単体テストNo2.3

認証失敗時の処理(LoginActionのユーザー認証失敗時の処理)

■目的

ログイン画面にて無効なユーザー名やパスワードを入力し、認証失敗時の処理が実行されるかを確認する

■テスト対象行

テスト対象の以下のコードにブレークポイントを設置

164行目 :logger.debug("認証失敗");

165行目 :addActionError("ユーザー名かパスワードが不正です。");

166行目 :return LOGIN;

■テスト方法

1.対象行にブレークポイントを設置

2.ログイン画面にて「無効なユーザー名やパスワード」を入力し、ログインボタンをクリックする

3.Eclipseのデバッグモードでテストを実行し、処理が該当行で停止することを確認

4.デバッグログに「認証失敗」の出力内容を確認

5.認証失敗後、ログイン画面に戻り、「ユーザー名かパスワードが不正です。」のエラーメッセージが表示されることを確

6.確認結果を以下のように記録

■期待される結果

・return LOGIN; が呼ばれ、ログイン画面に戻ることを確認する

・addActionError("ユーザー名かパスワードが不正です。"); が呼ばれた際、エラーメッセージが表示されることを確認する
→ ログイン画面上に「ユーザー名かパスワードが不正です。」のメッセージが表示されること

・デバッグログで「認証失敗」が出力されること

■テスト実施内容

・画面遷移とエラー表示の確認

・return LOGIN; によりログイン画面に遷移し、ログイン画面に「ユーザー名かパスワードが不正です。」の表示を確認

・デバッグログ

・以下のメッセージが記録されていることを確認

[2024-11-12 18:09:52.767] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.login.LoginAction - 認証失敗

以上

■単体テストNo3.1

ユーザー認証処理 (UserDAOクラスのユーザー認証処理)

■目的

LoginActionクラスから取得したデータに基づき、ユーザー認証処理を行い、その結果に応じて異なる処理を実行できることを確認する。

■対象

UserDAOクラスのauthenticateメソッド

■前提条件

ユーザーが登録済みであり、正しいuser_nameとpasswordを入力している。

■テスト対象行

- ・34行目 :preparedStatement.setString(1, user_name);
- ・35行目 :preparedStatement.setString(2, password);
- ・40行目 :try (ResultSet rs = preparedStatement.executeQuery()) {
- ・42行目 :if (rs.next()) {
- ・49行目 :user.setUser_id(rs.getInt("user_id"));
- ・50行目 :user.setUser_name(rs.getString("user_name"));
- ・51行目 :user.setPassword(rs.getString("password"));
- ・52行目 :user.setAuth_type(rs.getInt("auth_type"));
- ・53行目 :user.setDelete_day(rs.getString("delete_day"));
- ・72行目 :return user;

■期待される結果

- ・34行目～35行目 : プリペアドステートメントにて各値が紐づけられているか確認。
- ・40行目 :preparedStatement が正しく初期化されているかを確認。
DBから取得した クエリ結果の行数を確認。クエリ結果が正しく取得できること。
取得したクエリ行の各列の値を確認するuser_name と password に対応するユーザー情報。データの整合性を確認する。
- ・42行目 : クエリの結果として空ではないResultSet が得られているかを確認。
正しいデータが含まれているか(user_name と password に対応するユーザー情報)
- ・49行目～53行目 :ResultSetに入っている各値がuserオブジェクトに紐づけられているかを確認。
- ・72行目 : 処理結果をuserオブジェクトとしてリターンできているかを確認。

■テスト実施内容

- 34行目 :preparedStatement.setString(1, user_name);
- ・目的: プリペアドステートメントにて各値が紐づけられているか確認する。
- ・テスト前 (ステップオーバー前)
確認内容:
-Eclipse「変数」ウィンドウの以下を確認。valueの値が「null」で取得できていない状態
preparedStatement → delegate → query → queryBindings → bindValues → [0] → valueの値が「null」
- ・テスト後 (ステップオーバー後)
確認内容:
-Eclipse「変数」ウィンドウの以下を確認。valueの値が、以下のような値を持つ:
preparedStatement → delegate → query → queryBindings → bindValues → [0] →
value xxxxadmin(id=119)

- 35行目 :preparedStatement.setString(2, password);
- ・目的: プリペアドステートメントにて各値が紐づけられているか確認する。

・テスト前（ステップオーバー前）

確認内容：
-Eclipse「変数」ウィンドウの以下を確認。valueの値が「null」で取得できていない状態
preparedStatement → delegate → query → queryBindings → bindValues → [1] → valueの値が「null」

・テスト後（ステップオーバー後）

確認内容：
-Eclipse「変数」ウィンドウの以下を確認。valueの値が、以下のような値を持つ：
preparedStatement → delegate → query → queryBindings → bindValues → [1] →
value "zE4c9m+q" (id=136)

●40行目 :try (ResultSet rs = preparedStatement.executeQuery()) {

・テスト前（ステップオーバー前）

目的 :preparedStatement が初期化前されている状態であることを確認
① -Eclipse「変数」ウィンドウの以下を確認。preparedStatementの「isClosed」の値が以下の通りであることを確認。
確認内容：
preparedStatement → isClosed false
※preparedStatement が正しく初期化されており、有効な状態であることを確認まだ閉じられておらず、操作可能な状態






② -Eclipse「変数」ウィンドウの以下を確認。「Value」の値が以下の通りであることを確認する。
確認内容：
preparedStatement → delegate → query → queryBindings → bindValues → [0] → valueの値が「"xxxxadmin"」であること
対応するパラメータ :user_name
preparedStatement → delegate → query → queryBindings → bindValues → [1] → valueの値が「"zE4c9m+q"」であること
対応するパラメータ :password

・テスト後（ステップオーバー後）

① -Eclipse「変数」ウィンドウの以下を確認。「executeQuery()」の項目が表示されるので、「isClosed」の値が以下の通りであることを確認。ResultSet が有効な状態であることを確認する。
目的 :ResultSet が有効な状態であることを確認する。
確認内容：
preparedStatement → executeQuery() → delegate → isClosedの値が「false」
※クエリ実行後、ResultSet が有効であることを確認済み。

② -Eclipse「変数」ウィンドウの以下を確認。「rows」の階層にDBから取得した クエリ結果の行数を確認。結果が返されていることを確認する。
目的 :クエリ結果の行数を確認する
確認内容：
preparedStatement → executeQuery() → delegate → rowData → rowsの階層に[0]の行が存在する。
※結果として、1行分のクエリ結果が取得できていることを確認済み。クエリが正しく実行され、結果が取得できている状態。

③ -Eclipse「変数」ウィンドウの以下を確認。
目的 :データの整合性を確認する
確認内容：
preparedStatement → executeQuery() → delegate → rowData → internalRowData 取得したクエリ行の各列の値を確認する。データの整合性を確認する。
※各列の値が期待通りであることを確認。
ユーザー名 : "xxxxadmin"
パスワード : "zE4c9m+q"

| 名前 | 値 |
|---|----------|
|   internalRowData | (id=307) |
|   [0] | (id=319) |
|  [0] | 50 |

| | |
|---------|----------|
| △ [0] | 50 |
| △ [1] | 54 |
| ▽ △ [1] | (id=320) |
| △ [0] | 117 |
| △ [1] | 101 |
| △ [2] | 104 |
| △ [3] | 97 |
| △ [4] | 114 |
| △ [5] | 97 |
| △ [6] | 97 |
| △ [7] | 100 |
| △ [8] | 109 |
| △ [9] | 105 |
| △ [10] | 110 |
| ▽ △ [2] | (id=321) |
| △ [0] | 122 |
| △ [1] | 69 |
| △ [2] | 52 |
| △ [3] | 99 |
| △ [4] | 57 |
| △ [5] | 109 |
| △ [6] | 43 |
| △ [7] | 113 |
| ▽ △ [3] | (id=322) |
| △ [0] | 49 |
| ▽ △ [4] | (id=323) |
| △ [0] | 49 |
| ▽ △ [5] | (id=324) |
| △ [0] | 50 |
| △ [1] | 57 |
| △ [2] | 57 |
| △ [3] | 57 |
| △ [4] | 45 |
| △ [5] | 49 |
| △ [6] | 50 |
| △ [7] | 45 |
| △ [8] | 51 |
| --- | -- |

| | |
|--------|----|
| △ [9] | 49 |
| △ [10] | 32 |
| △ [11] | 48 |
| △ [12] | 48 |
| △ [13] | 58 |
| △ [14] | 48 |
| △ [15] | 48 |
| △ [16] | 58 |
| △ [17] | 48 |
| △ [18] | 48 |
| △ [19] | 46 |
| △ [20] | 48 |
| △ [21] | 48 |
| △ [22] | 48 |
| △ [23] | 48 |
| △ [24] | 48 |
| △ [25] | 48 |

データ内容と解釈

以下は、取得したバイトデータをASCIIコードに変換し、テーブルカラムに対応させた内容

- 1.[0]: user_id(ユーザーID)
 - ・値: [50, 54]
 - ・解釈: ASCIIコードで 50 は '2'、54 は '6' を意味し、結合するとuser_id は '26' であると判定される。
※ユーザーIDのデータとして正しい形式。
- 2.[1]: user_name(ユーザー名)
 - ・値: [117, 101, 104, 97, 114, 97, 97, 100, 109, 105, 110]
 - ・解釈: ASCIIコードで各数値を文字に変換すると,'xxxxadmin' となり、ユーザー名として解釈される。
※ユーザー名として期待されるデータ形式で正しい。
- 3.[2]: password(パスワード)
 - ・値: [122, 69, 52, 99, 57, 109, 43, 113]
 - ・解釈: ASCIIコードで各バイトを変換すると,'zE4c9m+q' となり、パスワードのデータとして解釈される。
※セキュリティ上の理由でハッシュ化されているかは別途確認する必要があるものの、データとしては正しい。
- 4.[3]: auth_type(認証タイプ)
 - ・値: [49]
 - ・解釈: ASCIIコードで 49 は '1' に対応するため、auth_type が '1' であることを示す。
※認証タイプが'1' を示す設定として適切。
- 5.[4]: delete_flag(削除フラグ)
 - ・値: [49]
 - ・解釈: ASCIIコードで 49 は '1' に対応し、削除フラグが'1' であることを示す。
※削除フラグが'1'(有効)であることを示すデータとして適切。

6.[5]: delete_day(削除日)

- ・値: [50, 57, 57, 57, 45, 49, 50, 45, 51, 49, 32, 48, 48, 58, 48, 48, 58, 48, 48, 46, 48, 48, 48, 48, 48]
- ・解釈: ASCIIコードで変換すると2999-12-31 00:00:00.000000 となり、削除日が指定されている。
※ 削除日が非常に未来の日付として設定されている。

<検証結果>

デバッグ結果より、取得した各データがテストユーザーの情報と一致することを確認できた。

●42行目 :if (rs.next()) {

・テスト前(ステップオーバー前)

目的:クエリの結果として空ではないResultSet が得られているかを確認する。

-Eclipse「変数」ウィンドウの以下を確認。rowsの[0]の値が取得できていない状態

確認内容:

rs→delegate→rowData→rows→[0]

wasEmpty false

・テスト後(ステップオーバー後)

-Eclipse「変数」ウィンドウの以下を確認。rowsの[0]に'internalRowData'が表示されるので、値を確認する。

確認内容:

rs→delegate→rowData→rows→[0]→internalRowData→

| 名前 | 値 |
|---------------------|----------|
| ▼ ▲ internalRowData | (id=307) |
| ▼ ▲ [0] | (id=319) |
| ▲ [0] | 50 |
| ▲ [1] | 54 |
| ▼ ▲ [1] | (id=320) |
| ▲ [0] | 117 |
| ▲ [1] | 101 |
| ▲ [2] | 104 |
| ▲ [3] | 97 |
| ▲ [4] | 114 |
| ▲ [5] | 97 |
| ▲ [6] | 97 |
| ▲ [7] | 100 |
| ▲ [8] | 109 |
| ▲ [9] | 105 |
| ▲ [10] | 110 |
| ▼ ▲ [2] | (id=321) |
| ▲ [0] | 122 |
| ▲ [1] | 69 |
| ▲ [2] | 52 |
| ▲ [3] | 55 |

| | |
|---------|----------|
| △ [3] | 99 |
| △ [4] | 57 |
| △ [5] | 109 |
| △ [6] | 43 |
| △ [7] | 113 |
| ▽ △ [3] | (id=322) |
| △ [0] | 49 |
| ▽ △ [4] | (id=323) |
| △ [0] | 49 |
| ▽ △ [5] | (id=324) |
| △ [0] | 50 |
| △ [1] | 57 |
| △ [2] | 57 |
| △ [3] | 57 |
| △ [4] | 45 |
| △ [5] | 49 |
| △ [6] | 50 |
| △ [7] | 45 |
| △ [8] | 51 |
| △ [9] | 49 |
| △ [10] | 32 |
| △ [11] | 48 |
| △ [12] | 48 |
| △ [13] | 58 |
| △ [14] | 48 |
| △ [15] | 48 |
| △ [16] | 58 |
| △ [17] | 48 |
| △ [18] | 48 |
| △ [19] | 46 |
| △ [20] | 48 |
| △ [21] | 48 |
| △ [22] | 48 |
| △ [23] | 48 |
| △ [24] | 48 |
| △ [25] | 48 |

データ内容と解釈

以下は、取得したバイトデータをASCIIコードに変換し、テーブルカラムに対応させた内容

- 1.[0]: user_id(ユーザーID)
- ・値: [50, 54]
 - ・解釈: ASCIIコードで 50 は '2'、54 は '6' を意味し、結合するとuser_id は '26' であると判定される。
※ユーザーIDのデータとして正しい形式。
- 2.[1]: user_name(ユーザー名)
- ・値: [117, 101, 104, 97, 114, 97, 97, 100, 109, 105, 110]
 - ・解釈: ASCIIコードで各数値を文字に変換すると'xxxxadmin' となり、ユーザー名として解釈される。
※ユーザー名として期待されるデータ形式で正しい。
- 3.[2]: password(パスワード)
- ・値: [122, 69, 52, 99, 57, 109, 43, 113]
 - ・解釈: ASCIIコードで各バイトを変換すると,'zE4c9m+q' となり、パスワードのデータとして解釈される。
※セキュリティ上の理由でハッシュ化されているかは別途確認する必要があるものの、データとしては正しい。
- 4.[3]: auth_type(認証タイプ)
- ・値: [49]
 - ・解釈: ASCIIコードで 49 は '1' に対応するため、auth_type が '1' であることを示す。
※認証タイプが'1' を示す設定として適切。
- 5.[4]: delete_flag(削除フラグ)
- ・値: [49]
 - ・解釈: ASCIIコードで 49 は '1' に対応し、削除フラグが'1' であることを示す。
※削除フラグが'1'(有効)であることを示すデータとして適切。
- 6.[5]: delete_day(削除日)
- ・値: [50, 57, 57, 57, 45, 49, 50, 45, 51, 49, 32, 48, 48, 58, 48, 48, 58, 48, 48, 46, 48, 48, 48, 48, 48]
 - ・解釈: ASCIIコードで変換すると2999-12-31 00:00:00.000000 となり、削除日が指定されている。
※ 削除日が非常に未来の日付として設定されている。

<検証結果>

デバッグ結果より、取得した各データがテストユーザーの情報と一致することを確認できた。

- 49行目 :user.setUser_id(rs.getInt("user_id"));
目的: ResultSet から取得した user_id の値が正しく user オブジェクトに設定されるかを確認する。
- ・テスト前(ステップオーバー前)
 - Eclipse「変数」ウィンドウの以下を確認。確認内容:
user User (id=159) → user → user_id の値が「0」
※上記の通り、user_idが「0」で初期値の状態
user オブジェクトが初期化された直後であり,setUser_id() メソッド未実行のため、初期値0 であることが確認できる。
- ・テスト後(ステップオーバー後)
 - Eclipse「変数」ウィンドウの以下を確認。確認内容:
user User (id=159) → user → user_id の値が「26」
※上記の通り、user_idが「26」で入力フォームで入力したユーザーのIDと一致する。
ResultSet の rs.getInt("user_id") により、データベースから取得された値26 が setUser_id() を通じてフィールドにセットされていることが確認できる。

●50行目 :user.setUser_name(rs.getString("user_name"));

目的: ResultSet から取得した user_name の値が正しく user オブジェクトに設定されるかを確認する。

・テスト前(ステップオーバー前)

-Eclipse「変数」ウィンドウの以下を確認。

確認内容:

user User (id=159) → user → user_name の値が「null」

※上記の通り、user_nameが「null」で初期値の状態

user オブジェクトが初期化された直後であり、setUser_name() メソッド未実行のため、初期値がnull であることが確認できる。

・テスト後(ステップオーバー後)

-Eclipse「変数」ウィンドウの以下を確認。

確認内容:

user User (id=159) → user → user_name の値が「xxxxadmin」

※上記の通り、user_nameが「xxxxadmin」で入力フォームで入力したユーザー名と一致する。

ResultSet の rs.getString("user_name")により、データベースから取得された値「xxxxadmin」が user.setUser_nameを通じてフィールドにセットされていることが確認できる。

●51行目 :user.setPassword(rs.getString("password"));

目的: ResultSet から取得した password の値が正しく user オブジェクトに設定されるかを確認する。

・テスト前(ステップオーバー前)

-Eclipse「変数」ウィンドウの以下を確認。

確認内容:

user User (id=159) → user → password の値が「null」

※上記の通り、passwordが「null」で初期値の状態

user オブジェクトが初期化された直後であり、setPassword() メソッド未実行のため、初期値null であることが確認できる。

・テスト後(ステップオーバー後)

-Eclipse「変数」ウィンドウの以下を確認。

確認内容:

user User (id=159) → user → password の値が「zE4c9m+q」

※上記の通り、passwordが「zE4c9m+q」で入力フォームで入力したパスワードと一致する。

ResultSet の rs.getString("password") により、データベースから取得された値zE4c9m+q が user.setPassword() を通じてフィールドにセットされていることが確認できる。

●52行目 :user.setAuth_type(rs.getInt("auth_type"));

目的: ResultSet から取得した auth_type の値が正しく user オブジェクトに設定されるかを確認する。

・テスト前(ステップオーバー前)

-Eclipse「変数」ウィンドウの以下を確認。

確認内容:

user User (id=159) → user → auth_type の値が「0」

※上記の通り、auth_typeが「0」で初期値の状態

user オブジェクトが初期化された直後であり、setAuth_type() メソッド未実行のため、初期値0 であることが確認できる。

・テスト後(ステップオーバー後)

-Eclipse「変数」ウィンドウの以下を確認。

確認内容:

user User (id=159) → user → auth_type の値が「1」

※上記の通り、auth_typeが「1」で入力フォームで入力したユーザーの権限と一致する。

ResultSet の rs.getInt("auth_type") により、データベースから取得された値 が user.setAuth_type() を通じてフィールドにセットされていることが確認できる。

●53行目 :user.setDelete_day(rs.getString("delete_day"));

目的: ResultSet から取得したdelete_day の値が正しく user オブジェクトに設定されるかを確認する。

・テスト前（ステップオーバー前）

-Eclipse「変数」ウィンドウの以下を確認。

確認内容：

user User (id=159) → user → delete_day の値が「null」

※上記の通り、delete_dayが「null」で初期値の状態

user オブジェクトが初期化された直後であり、setDelete_day() メソッド未実行のため、初期値null であることが確認できる。

・テスト後（ステップオーバー後）

-Eclipse「変数」ウィンドウの以下を確認。

確認内容：

user User (id=159) → user → delete_day の値が「2999-12-31 00:00:00」

※上記の通り、delete_dayが「2999-12-31 00:00:00」で入力フォームで入力したユーザーの削除日と一致する。

ResultSet の rs.getString("delete_day") により、データベースから取得された値2999-12-31 00:00:00 が user.setDelete_day() を通じてフィールドにセットされていることが確認できる。

●72行目 :return user;

・テスト前（ステップオーバー前）

確認項目

1.user オブジェクトの状態確認

目的:userオブジェクトの値が期待したとおりにセットされていることを確認する。

確認内容：

-Eclipse「変数」ウィンドウの以下を確認する。

user User (id=159) → user → user_id の値が「26」であること

user User (id=159) → user → user_name の値が「xxxxadmin」であること

user User (id=159) → user → password の値が「zE4c9m+q」であること

2.sql 変数の内容確認

目的:SQL 文が正しいか、またパラメータが正しくバインドされているかを確認する。

確認内容：

-Eclipse「変数」ウィンドウの以下を確認する。

sql "SELECT * FROM users WHERE user_name=? AND password=?" (id=133) であること

バインドパラメータ

1. user_name="xxxxadmin"

2. password="zE4c9m+q"

・テスト後（ステップオーバー後）

確認項目

1.authenticate() の戻り値の確認

目的:authenticate() メソッドの戻り値として、期待通りのuser オブジェクトが返されているかを確認する。

確認内容：

-Eclipse「変数」ウィンドウの以下を確認する。

authenticate() User (id=164) → user_id の値が「26」であること

authenticate() User (id=164) → user_name の値が「xxxxadmin」であること

authenticate() User (id=164) → password の値が「zE4c9m+q」であること

以下のログ出力を確認

[2024-11-23 15:32:10.156] DEBUG http-nio-8080-exec-3 com.company.bulletinboard.dao.UserDAO - ユーザー認証成功

[2024-11-23 15:32:18.005] DEBUG http-nio-8080-exec-3 com.company.bulletinboard.dao.UserDAO - Userデータ: user_id=26, user_name=xxxxadmin, password=zE4c9m+q, auth_type=1, delete_flag=0, delete_day=2999-12-31 00:00:00

以上

■単体テストNo3.2

認証失敗処理 (UserDAOクラスのユーザー認証処理)

■目的

謝ったユーザー名と、パスワードが入力された際に例外がキャッチされ、最終的にuserがnullとして返されていること。

■前提条件

データベースに登録されていない、ユーザー名とパスワードを入力していること。

- ・テスト用のユーザー名 :admin999999
- ・テスト用のパスワード :zE4c9m+q

■テスト対象行

```
34行目 :preparedStatement.setString(1, user_name);
35行目 :preparedStatement.setString(2, password);
42行目 :if (rs.next()) {
64行目 :} catch (SQLException e) {
72行目 :return user;
```

■期待される結果

- ・34行目～35行目 : プリペアドステートメントにて各値が紐づけられていること。
- ・42行目 : 認証失敗時にrs.next()がfalseになること。(この条件が成立しない場合、userは生成されずnullのままとなるのが期待挙動。)
- ・64行目 : SQL実行時に例外が発生した場合に正しくキャッチされること。
- ・72行目 : userがnullで返されていること。

■テスト実施内容

- 34行目 :preparedStatement.setString(1, user_name);

・目的: プリペアドステートメントにて各値が紐づけられているか確認する。

・テスト前 (ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウの以下を確認。valueの値が「null」で取得できていない状態

preparedStatement → delegate → query → queryBindings → bindValues → [0] → valueの値が「null」

・テスト後 (ステップオーバー後)

確認内容:

-Eclipse「変数」ウィンドウの以下を確認。valueの値が、以下のような値を持つ:

preparedStatement → delegate → query → queryBindings → bindValues → [0] →
value "admin999999" (id=119)

- 35行目 :preparedStatement.setString(2, password);

・目的: プリペアドステートメントにて各値が紐づけられているか確認する。

・テスト前 (ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウの以下を確認。valueの値が「null」で取得できていない状態

preparedStatement → delegate → query → queryBindings → bindValues → [1] → valueの値が「null」

・テスト後 (ステップオーバー後)

確認内容:

-Eclipse「変数」ウィンドウの以下を確認。valueの値が、以下のような値を持つ:
preparedStatement → delegate → query → queryBindings → bindValues → [1] →
value "zE4c9m+q" (id=136)

●42行目:if (rs.next()) {

・目的:認証失敗時にrs.next()がfalseになることを確認する。

・テスト前(ステップオーバー前)

① -Eclipse「変数」ウィンドウの以下を確認。currentRow の値が以下であることを確認する。

確認内容:

rs (HikariProxyResultSet) → delegate ResultSetImpl (id=315) → connection ConnectionImpl (id=159) → currentRow の値が「-1」

※rs にデータが存在しない状態。結果セットが初期化されており、まだデータにアクセスしていない状態を示す。

② -Eclipse「変数」ウィンドウの以下を確認。テストケースで使⽤した入力データがバインドされた値と一致していることを確認する。

確認内容:

rs → statement → delegate → query → queryBindings → bindValues → [0] → valueの値が "admin999999" (id=119)

rs → statement → delegate → query → queryBindings → bindValues → [1] → valueの値が "zE4c9m+q" (id=136)

※上記の通り、フォームで入力した値と一致する。

③ -Eclipse「変数」ウィンドウの以下を確認。データベースとの接続が確立されていること、また接続が有効な状態であることを確認する。

確認内容:

rs → connection → delegate → ConnectionImpl (id=159)

※delegate: ConnectionImpl が存在し、接続が正しく確立されていると裏付けられる。

補足:ConnectionImpl は実際のデータベース接続を表すオブジェクトである。

HikariProxyConnection は、HikariCP がラップしているプロキシオブジェクトであり,delegate にラップされていない実際の接続 (ConnectionImpl) が格納される。

もし、delegate が null の場合、接続が適切に確立されていないか、すでに閉じられている可能性がある。

④ -Eclipse「変数」ウィンドウの以下を確認。データベースの接続が自動コミットモードで作成されるので、この設定が適切か確認する。

確認内容:

rs → statement → delegate → isAutoCommitの値が「true」

※上記の通り、isAutoCommitの値が「true」なのでデータベースの接続は自動コミットモードで作成されている。設定が適切な状況。

⑤ -Eclipse「変数」ウィンドウの以下を確認。接続が読み取り専用に設定されていないことを確認する。

確認内容:

rs → statement → delegate → isReadOnlyの値が「false」

※上記の通り、isReadOnlyの値が「false」なので、読取り専用に設定されていない状況。

⑥ -Eclipse「変数」ウィンドウの以下を確認。適切なトランザクション分離レベル(0 など) が設定されていることを確認する。

確認内容:

rs → statement → delegate → transactionIsolationの値が「0」

※上記の通り、transactionIsolationの値が「0」なので、適切なトランザクション分離レベルと裏付けられる。(0 はデフォルトレベル)

⑦ -Eclipse「変数」ウィンドウの以下を確認。接続がアクティブであることを確認する。

確認内容:

rs → statement → delegate → lastAccess の値が「4374008009700」

※上記の通り、lastAccess の値が「4374008009700」となっており、値が現在時刻に近いことを確認し、接続がアクティブであることを裏付けられる。

⑧ -Eclipse「変数」ウィンドウの以下を確認。プールエントリーが存在することを確認し、接続プールが正しく管理されていることを確認する。

確認内容:

⑧-1. rs → connection → poolEntry → connectionが「ConnectionImpl」

※「ConnectionImpl」は接続プールから借用された実際のデータベース接続を表す。結果セットが初期化されており、まだデータにアクセスしていない状態を示す。

⑧-2. rs → connection → poolEntry → evictの値が「true」

⑧-3. rs → connection → poolEntry → hikariPoolの値が「HikariPool」

※コネクションプールが正しく動作していることを示す。

⑧-4. rs → connection → poolEntry → lastAccessedの値が「1962452816800」

※この値は接続が最後にアクセスされた時刻を示す。値が現在時刻に近いことを確認し、接続がアクティブであることを裏付ける。

⑧-5. rs → connection → poolEntry → stateの値が「1」

※ 接続プール内での状態を示す。「1」は接続がアクティブな状態。

・テスト後(ステップオーバー後)

① -Eclipse「変数」ウィンドウの以下を確認。currentRow の値が以下であることを確認する。

確認内容:

rs → delegate → currentRow の値が「-1」

※結果セットが空なので、currentRow は初期値の -1 のままで変更されていない

② -Eclipse「変数」ウィンドウの以下を確認。currentRow の値が以下であることを確認する。

確認内容:

rs → delegate → onValidRow の値が「false」

※rs.next() が失敗(=結果セットが空)した場合、onValidRow は false になっており、これにより現在の行が有効でないことを示す。

③ -Eclipse「変数」ウィンドウの以下を確認。currentRow の値が以下であることを確認する。

確認内容:

rs → delegate → invalidRowReason が「Illegal operation on empty result set.」

※結果セットが空で、操作が無効とされた場合、このエラーメッセージが設定される。

④ -Eclipse「変数」ウィンドウの以下を確認。isClosed の値が以下であることを確認する。

確認内容:

rs → delegate → isClosed の値が「false」

※結果セットが閉じられておらず、rs がまだ有効で操作可能な状態である。

※※上記の結果を基に、ステップオーバー後は「認証失敗時にrs.next() が false になること」の証明の裏付けとなる。

●64行目:} catch (SQLException e) {

・目的:SQL実行時に例外が発生した場合に正しくキャッチされることを確認する。

・テスト前(ステップオーバー前)

① -Eclipse「変数」ウィンドウの以下を確認。originalSqlの値が以下であることを確認する。

理由:SQL 文が正しく構築されているか確認する。

確認内容:

preparedStatement → delegate → query → originalSqlの値が「SELECT * FROM users WHERE user_name=? AND password=?」

② -Eclipse「変数」ウィンドウの以下を確認。isClosedの値が以下であることを確認する。

理由: 接続状態を確認する。PreparedStatement がすでに閉じられている場合は、SQL 実行時に SQLException が発生する可能性がある為。

確認内容:

preparedStatement → delegate → isClosedの値が「false」

※上記の通り、値が「false」なので接続は開いている。

③ -Eclipse「変数」ウィンドウの以下を確認。connectionの各プロパティを確認する。

③-1. -Eclipse「変数」ウィンドウの以下を確認。isAutoCommitの値を確認する

理由: 自動コミットが有効でない場合、コミット漏れが原因でエラーやデッドロックの可能性がある為、確認する。通常SQL 操作で問題が起こる際にはこの値を確認することが推奨される。

確認内容: preparedStatement → connection → isAutoCommitの値が「true」

※上記の値が「true」の為、自動コミットモードが有効。

③-2. -Eclipse「変数」ウィンドウの以下を確認。transactionIsolationの値を確認する

理由: トランザクション分離レベルがSQL の整合性やパフォーマンスに影響を与える可能性があるため、適切なレベルが設定されていることを確認。

確認内容: preparedStatement → connection → transactionIsolationの値が「0」

※上記の値が「0: TRANSACTION_NONE」の為、適切な値。

③-3. -Eclipse「変数」ウィンドウの以下を確認。isReadOnlyの値を確認する

理由: 書き込み系の SQL (INSERT, UPDATE, DELETE) を実行する場合、isReadOnly が true だと例外が発生するため、接続が適切な状態かを確認

確認内容: preparedStatement → connection → isReadOnlyの値が「false」

※上記の値が「false: 書き込み操作が許可されている。」の為、適切な値。

③-4. -Eclipse「変数」ウィンドウの以下を確認。dbcatalogの値を確認する

理由: 特定のスキーマにアクセスする必要があるSQL 文で、この値が正しく設定されていないと、期待するデータにアクセスできない可能性がある。

確認内容: preparedStatement → connection → dbcatalogの値が「null」

※上記の値が「null」の為、適切な値。

③-5. -Eclipse「変数」ウィンドウの以下を確認。networkTimeoutの値を確認する

理由: 過剰に短い値が設定されていると、接続が切断される可能性があるため、接続の安定性を確認する。

確認内容: preparedStatement → connection → networkTimeoutの値が「0」

※上記の値が「0」の為、適切なタイムアウト値。

④ -Eclipse「変数」ウィンドウの以下を確認。proxyResultSetの値が以下であることを確認する。

理由: proxyResultSet が null の場合、まだ SQL 文が実行されていない状態であることを示す。SQL 文が正しく実行される前の状態を把握できる為。

確認内容: preparedStatement → proxyResultSetの値が「null」

※上記の通り、値が「null」なのでSQL 文が正しく実行される前の状態

・テスト後(ステップオーバー後)

① -Eclipse「変数」ウィンドウの以下を確認。「close() 戻り値」の状態を確認する。

理由: リソースの適切なクローズを確認(明示的な戻り値がなくてもエラーが出ていないか確認)。

確認内容: close() 戻り値 (明示的な戻り値がありません)

※上記の通り、明示的な戻り値はないが、エラーは発生していない状況。よって、例外発生時の処理に問題は無いと裏付けられる。

② -Eclipse「変数」ウィンドウの以下を確認。「user」の状態を確認する。

理由: 例外が発生した場合、user の状態が変化していないことが期待される。この値が適切でない場合、例外発生時の処理に問題がある可能性がある。

確認内容: userの値が「null」

※上記の通り、user が null のままであることから、例外発生時の処理に問題は無いと裏付けられる。

③ -Eclipse「変数」ウィンドウの以下を確認。「user_name」の値を確認する。

理由: 値が変化している場合、SQL 実行やエラー処理が元の入力値に影響を及ぼしている可能性がある。

値がそのままであれば、例外処理が他のデータに影響を与えないことを確認する為。

確認内容: user_name の値が「"admin999999" (id=135)」

※上記の通り、値が変化していないので例外処理がデータに影響を与えていないことが裏付けられる。

④ -Eclipse「変数」ウィンドウの以下を確認。「password」の値を確認する。

理由: 値が変化している場合、SQL 実行やエラー処理が元の入力値に影響を及ぼしている可能性がある。

値がそのままであれば、例外処理が他のデータに影響を与えないことを確認する為。

確認内容: passwordの値が「"zE4c9m+q" (id=140)」

※上記の通り、値が変化していないので例外処理がデータに影響を与えていないことが裏付けられる。

⑤ -Eclipse「変数」ウィンドウの以下を確認。「sql」の値を確認する。

理由: 例外発生後でもsql の状態が保持されているかを確認し、意図しないSQL の変更が行われていないことを保証する。

確認内容: sqlの値が「SELECT * FROM users WHERE user_name=? AND password=?」

※上記の通り、値が変化していないので例外処理に影響を与えていないことが裏付けられる。

●72行目: return user;

・目的: userがnullで返されていることを確認する。

・テスト前(ステップオーバー前)

① -Eclipse「変数」ウィンドウの以下を確認。「user」の状態を確認する。

理由: userの状態が「null」であることを確認する。前段の処理から変化が無いこと。

確認内容: userの値が「null」

※上記の通り、前段の例外処理64行目)から変化が無いので正常値と判断。

② -Eclipse「変数」ウィンドウの以下を確認。「user_name」の状態を確認する。

理由: 前段の処理から値に変化が無いことを確認する。処理の整合性を検証する為。

確認内容: user_nameの値が「admin999999」

※上記の通り、前段の例外処理64行目)から変化が無いので正常値と判断。

③ -Eclipse「変数」ウィンドウの以下を確認。「password」の状態を確認する。

理由: 前段の処理から値に変化が無いことを確認する。処理の整合性を検証する為。

確認内容: passwordの値が「zE4c9m+q」

※上記の通り、前段の例外処理64行目)から変化が無いので正常値と判断。

④ -Eclipse「変数」ウィンドウの以下を確認。「sql」の状態を確認する。

理由: 前段の処理から値に変化が無いことを確認する。処理の整合性を検証する為。

確認内容: sqlの値が「SELECT * FROM users WHERE user_name=? AND password=?」

※上記の通り、前段の例外処理64行目)から変化が無いので正常値と判断。

・テスト後(ステップオーバー後)

※ステップオーバー後に、LoginActionクラスの115行目authenticate() メソッドの呼び出し元に移動する。

① -Eclipse「変数」ウィンドウの以下を確認。「authenticate() 戻り値」の状態を確認する。

理由: メソッドの戻り値が期待する結果(null か、それ以外の値)と一致しているかを検証するため。

確認内容: authenticate() 戻り値の値が「null」

※上記の通り、入力情報に基づくユーザーデータがデータベースに存在しなかったと判断できる。意図した結果。

② -Eclipse「変数」ウィンドウの以下を確認。「this (LoginAction インスタンス)」の状態を確認する。

理由:LoginAction クラスの内部状態を確認し、DAO から渡された値が正しくクラスフィールドに反映されているかを検証するため。

確認内容: this → user → user_nameの値が「admin999999」

this → user → user_nameの値が「zE4c9m+q」

※上記の通り、UserDAOから渡された値が、LoginActionのクラスフィールドに反映されている。意図した結果。

以上

■単体テストNo4

ログアウト処理

■目的

「ログアウト」ボタンクリック後、現在のセッションを取得しそのセッションを、無効化できること。
セッション無効化後、ログイン画面へ遷移できること。

■テスト対象行

テスト対象の以下のコードにブレークポイントを設置

23行目 : HttpSession session = ServletActionContext.getRequest().getSession(false);

27行目 : session.invalidate();

33行目 : return "success";

■テスト方法

1.対象行にブレークポイントを設置

2.「ログアウト」ボタンをクリックする

3.Eclipseのデバッグモードでテストを実行し、処理が該当行で停止することを確認

4.Eclipse の「変数」ウィンドウで、sessionの `isValid` の値が false であることを確認

5.デバッグログの出力内容を確認

6.確認結果を以下のように記録

■期待されるテスト結果

・sessionの取得結果が以下を満たす場合に、正しい処理と判断する。

-isValid項目の値が「false」

- セッションIDがログに表示されていること

・ログに以下の内容が出力されること

「セッションが無効化されました。セッションID:xxxxxxx」

「ログアウト処理が成功しました。ログインページにリダイレクトします。」

■テスト実施内容

・テスト前(ステップオーバー前)

・Eclipse「変数」ウィンドウにsessionが表示され、以下のような値を持つ：

-isValid : true

・テスト後(ステップオーバー後)

・Eclipse「変数」ウィンドウにsessionが表示され、以下のような値を持つ：

-isValid : false

■デバッグログ出力

[2024-11-14 10:08:30.451] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.login.LogoutAction - セッションが無効化されました。セッションID: 834E01F6496DAC13CB43F1CCEE31C5A2

[2024-11-14 10:08:32.487] INFO http-nio-8080-exec-6 com.company.bulletinboard.action.login.LogoutAction - ログアウト処理が成功しました。ログインページにリダイレクトします。

以上

■単体テストNo5.1

掲示板管理画面への遷移処理

■目的

- ・「掲示板管理画面へ」リンクをクリック後、データベースのテーブルから掲示板データの一覧を取得できること。
- ・上記処理後、「SUCCESS」を返すことを確認する。

■テスト対象行

```
57行目 :if (sessionUser == null) {
129行目 :try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
142行目 :while (resultSet.next()) {
164行目 :User bulletinboard = new User();
166行目 :bulletinboard.setBulletinboard_id(resultSet.getInt("bulletinboard_id"));
169行目 :bulletinboard.setBulletinboard_title(resultSet.getString("bulletinboard_title"));
172行目 :bulletinboard.setBulletinboard_content(resultSet.getString("bulletinboard_content"));
175行目 :bulletinboard.setUser_id(resultSet.getInt("user_id"));
178行目 :bulletinboard.setBulletinboard_delete_flag(resultSet.getInt("bulletinboard_delete_flag"));
181行目 :bulletinboard.setBulletinboard_delete_day(resultSet.getString("bulletinboard_delete_day"));
80行目  :return SUCCESS;
```

■テスト実施内容

- 57行目 :if (sessionUser == null) {
- ・目的:セッションが正しく設定されているかを確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

```
sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「JP9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」
```

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

- ・テスト後(ステップオーバー後)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値:

```
sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「JP9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」
```

上記の通り、ステップオーバー後も情報が保持されており、意図した内容。

2. this → session の内容:

- sessionの値が`SessionMap`オブジェクトである。
- session内にキー`"loggedInUser"`が存在する(`session.containsKey("loggedInUser") == true`)。
- session.get("loggedInUser")の値が`sessionUser`オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

```
auth_type: 1
delete_day: "2999-12-31 00:00:00"
delete_flag: 0
password: "JP9T-LH2"
user_id: 30
user_name: "tesutuser1"
```

上記の通り、クラスフィールドにsessionUserオブジェクトと同じ内容が補完されている。正常動作。

3. ログに以下の内容が出力されている:

```
[2024-11-27 12:04:52.409] INFO http-nio-8080-exec-4 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - Session User: tesutuser1
```

- 129行目 :try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
- ・目的:接続の正常性を確認する。
- ・テスト前(ステップオーバー前)

確認内容:
-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

- 「sql」変数の確認。
 - sql → SELECT * FROM bulletinboard (id=209)
 - ※不正な値 (NULL や予期しない構文など) が含まれていない
- 「connection」オブジェクトの確認。接続がクローズされていないことの確認する。
 - connection → connectionオブジェクト自体が存在し、NULLで無いこと
 - connection → delegate → openStatementsの値が「CopyOnWriteArrayList<E> (id=213)」
 - ※openStatementsの値が空でなくアクセス可能な状態。
 - isAutoCommit → isAutoCommitの値が「true」であること
- 接続先データベース情報の確認
 - connection → delegate → origHostToConnectTo → 「localhost」 (id=257) であること
 - connection → delegate → origPortToConnectTo → ポート番号が「3306」であること
 - connection → delegate → database → データベース名が「bulletinboard_db」 (id=240) であること
 - connection → delegate → user → 「root」 (id=285) であること
- 接続有効性のログ
 - connection.isValid(timeout)メソッド が「true」を返しており、ログに「データベース接続は正常です。」というメッセージが表示されている。
 - ログ内容 :[2024-12-02 12:19:09.215] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

・テスト後 (ステップオーバー後)
確認内容 :PreparedStatement の生成が正しく行われ、後続処理に影響がないことを確認する
- Eclipse「変数」ウィンドウで以下を確認。

- 「sql」変数の確認
 - sql → SELECT * FROM bulletinboard" (id=151)
 - ※正しい SQL 文 ("SELECT * FROM bulletinboard") を保持していることを確認
- connection の有効性を確認
 - preparedStatement → isClosedの値が「false」であることを確認
 - ※ステートメントが閉じていない状態
- 「proxyResultSet」の確認
 - preparedStatement → proxyResultSetの値が「null」であることを確認
- 接続有効性のログ
 - connection.isValid(timeout)メソッド が「true」を返すことを確認
 - ログ内容 :[2024-12-03 13:00:24.288] DEBUG http-nio-8080-exec-3 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db
 - ※※ログメッセージに「データベース接続は正常です。」が出力されていることを確認。接続は有効な状態

●142行目 :while (resultSet.next()) {
・目的 :resultSet の状態やクエリの実行結果に問題がないかを確認するwhile 文内の処理に進む準備ができていることを確認する。
・テスト前 (ステップオーバー前)

確認内容:
-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

- 「resultSet」の有効性確認
 - resultSet → statement → delegate → isClosed が false
 - ※resultSet はまだ有効であり、操作可能な状態
 - while 文が開始される前にデバックログにて、以下のログを確認する。
 - [2024-12-24 12:17:28.630] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
 - ※ログに「初回評価 true」と出力されているので、少なくとも1件のデータが存在し、`resultSet.next()` が正常動作していることを確認。

- 「sql」変数の確認
 - sql → SELECT * FROM bulletinboard" (id=151)
 - ※正しい SQL 文 ("SELECT * FROM bulletinboard") を保持していることを確認

- 「preparedStatement」の有効性確認
 - preparedStatement → isClosed の値が「false」
 - ※ステートメントは有効で操作可能な状態
 - デバックログに以下のログが出力されること。
 - [2024-12-04 08:37:48.725] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - PreparedStatement: HikariProxyPreparedStatement@1982403014 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM bulletinboard
 - ※ログにステートメント情報 (HikariProxyPreparedStatement および SQL 文) が出力されているので、SQL 文が正常に紐づけられていることを確認。

- bulletinboards リストの状態確認
 - デバックログに以下のログが出力されること。
 - [2024-12-24 12:17:40.239] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0
 - ※bulletinboards リストの要素数が「0」であることから、データがまだ追加されていない状態であることを確認。

・テスト後 (ステップオーバー後)
確認内容 :PreparedStatement の生成が正しく行われ、後続処理に影響がないことを確認する
- Eclipse「変数」ウィンドウで以下を確認。

- resultSet.next() の評価結果確認

・デバックログに以下のログが出力されること。
[2024-12-24 12:17:28.630] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
※SQLクエリで取得したデータがresultSet に存在し、次の処理に進む準備が整っている

2.preparedStatement と resultSet の有効性確認
・preparedStatement → isClosed の値が「false」
・resultSet → delegate → isClosedの値が「false」
※データベース接続が維持されており,preparedStatement と resultSet がクローズされていないことを確認

3.bulletinboards リストの状態確認
・デバッグで以下のログを確認する。
[2024-12-24 12:17:40.239] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0
※リストが変更されていないことを確認。この時点ではモデル生成やデータ追加処理が未実行のため,bulletinboards のサイズが 0 のままである

4.「データベースから該当する掲示板データを正しく取得できていること確認する
・Eclipse「変数」ウィンドウの以下を確認する
debugData ArrayList<E> (id=175)
 [0] HashMap<K,V> (id=190)
 [0] HashMap\$Node<K,V> (id=209)
 key user_id (id=222)
 value null
 [1] HashMap\$Node<K,V> (id=202)
 key bulletinboard_title (id=224)
 value 2024年7月18日 日本 (id=201)
 [2] HashMap\$Node<K,V> (id=198)
 key bulletinboard_id (id=226)
 value Integer (id=196)
 [3] HashMap\$Node<K,V> (id=206)
 key bulletinboard_content (id=229)
 value 2024年7月18日 日本 掲示板、追記 (id=205)

※上記の値が、掲示板テーブルのデータと一致すること。

- 164行目 :User bulletinboard = new User();
- ・テスト前 (ステップオーバー前)
- ・目的:
 - ①SQL文の実行準備が正しく行われていることを確認する
 - データベース接続が有効であること。
 - 実行されるSQL文が期待どおりであること。
 - ②データ取得の準備が正しく行われていることを確認する
 - ResultSetが初期状態にあること(カーソルがまだデータ行に移動していない)。
 - 次のresultSet.next()呼び出しにより、データが正しく取得されることが期待できること。
 - ③初期状態のプロパティが正しいことを確認する
 - アクションオブジェクトのプロパティ(bulletinboardsなど)が初期化されていること。

確認内容:
-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.resultSet の状態確認
・resultSet → delegate → currentRow の値が「-1」
※カーソルが結果セットの先頭行の「前」にあり、このためcurrentRow は -1 を示す。これは、初回のresultSet.next()が呼び出されていない(初期状態)ことの裏付けとなる。正常な動き。
・デバックログを確認
[2024-12-24 12:17:28.630] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
※このログは、resultSet.next() が初回評価でtrue を返しているため、少なくとも1件のデータが正しく取得されていることを示している。のためcurrentRow == -1 は初期状態として正しいと判断できる。

2. preparedStatement の状態確認
・sql → SELECT * FROM bulletinboard" (id=151)
※ sql に期待通りのクエリ(SELECT * FROM bulletinboard) が格納されていることを確認。誤ったSQL文では無いこと。

3.接続状態の確認
・preparedStatement → isClosed の値が「 false」であることを確認。
※ステートメントが閉じておらず、接続が維持されていることを確認
・resultSet → delegate → isClosed の値が「false」であることを確認。
※ResultSetも閉じられていない状態である。

4. this の状態確認
・this → bulletinboards の値が ArrayList<E>(id=193) となっている
・また、デバックログが以下の内容となっている。

[2024-12-24 12:17:40.239] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0
※ログにも明示されており、初期状態として期待通りでる。

- 5.エラーログ確認
- 以下の通り、現状のデバッグログにはエラーや例外はなく、問題が発生していないことが確認できる。
[2024-12-24 12:17:28.630] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
[2024-12-24 12:17:38.705] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - PreparedStatement: HikariProxyPreparedStatement@1842907940 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM bulletinboard
[2024-12-24 12:17:40.239] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0

- テスト後(ステップオーバー後)
- 目的:
 - ①User モデルのインスタンスが正しく生成され、初期化状態が期待どおりであることを確認する。
 - ②その他のオブジェクト状態がステップオーバー後に変化がないことを確認する。

確認内容:
- Eclipse「変数」ウィンドウで以下を確認。
1.bulletinboard オブジェクトの生成確認
•bulletinboard の階層を展開し、以下のプロパティを確認する。
- bulletinboard_contentの値が「null」
- bulletinboard_creation_dayの値が「null」
- bulletinboard_delete_dayの値が「null」
- bulletinboard_delete_flagの値が「0」
- bulletinboard_idの値が「0」
- bulletinboard_titleの値が「null」
※上記の通り、各プロパティがモデルの初期状態に一致している。
※User モデルが初期状態であり、まだデータが設定されていない状態を示す。

- 2.resultSet の状態確認
- resultSet → delegate → currentRow の値が「-1」
※カーソルが結果セットの先頭行の「前」にあり、このためcurrentRow は -1 を示す。これは、初回のresultSet.next()が呼び出されていない(初期状態)こととの裏付けとなる。正常な動き。
 - デバッグログを確認
[2024-12-24 12:17:28.630] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
※このログで、resultSet.next() が初回評価でtrue を返しているため、少なくとも1件のデータが正しく取得されていることを示している。のためcurrentRow == -1 は初期状態として正しいと判断できる。
※結果として、ステップオーバー後に変化無しと裏付けられる。

- 3.接続状態の確認
- preparedStatement → isClosed の値が「false」であることを確認。
※ステートメントが閉じておらず、接続が維持されていることを確認
 - resultSet → delegate → isClosed の値が「false」であることを確認。
※ResultSetも閉じられていない状態である。
※結果として、ステップオーバー後に変化無しと裏付けられる。

4. this の状態確認
- this → bulletinboards の値が ArrayList<E>(id=168) となっている
 - また、デバッグログが以下の内容となっている。
[2024-12-24 12:17:40.239] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0
※ログにも明示されており、初期状態として期待通りでる。
※結果として、ステップオーバー後に変化無しと裏付けられる。

- 5.エラーログ確認
- 以下の通り、現状のデバッグログにはエラーや例外はなく、問題が発生していないことが確認できる。
[2024-12-24 12:17:28.630] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
[2024-12-24 12:17:38.705] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - PreparedStatement: HikariProxyPreparedStatement@1842907940 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM bulletinboard
[2024-12-24 12:17:40.239] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0
※結果として、ステップオーバー後に変化無しと裏付けられる。

- 166行目 :bulletinboard.setBulletinboard_id(resultSet.getInt("bulletinboard_id"));
- テスト前(ステップオーバー前)
- 目的:モデルの初期状態が期待通りであることを確認する(カーソル位置、モデル、接続、ログ)
- 1.resultSet のカーソル位置の確認
•resultSet → delegate → currentRow の値が「-1」
※currentRow は -1 を示す。これは、resultSet のカーソルはまだ「最初の行の前」にあることを確認できる。カーソルの移動する準備が整っている状態。正常な動き。
- デバッグログを確認
[2024-12-24 12:17:28.630] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
※このログで、resultSet.next() が初回評価でtrue を返しているため、少なくとも1件のデータが正しく取得されていることを示している。のためcurrentRow == -1 は初期状態として正しいと判断できる。

2.bulletinboard オブジェクトの生成確認

- bulletinboard の階層を展開し、以下のプロパティを確認する。
 - bulletinboard_contentの値が「null」
 - bulletinboard_creation_dayの値が「null」
 - bulletinboard_delete_dayの値が「null」
 - bulletinboard_delete_flagの値が「0」
 - bulletinboard_idの値が「0」
 - bulletinboard_titleの値が「null」

※上記の通り、各プロパティがモデルの初期状態に一致している。
※モデルが正しく生成されており、データがセットされる準備が整っていることが確認できる。

3.接続状態の確認

- preparedStatement → isClosed の値が「 false」であることを確認。
 - ※ステートメントが閉じておらず、接続が維持されていることを確認
- resultSet → delegate → isClosed の値が「false」である を確認。
 - ※ResultSetも閉じられていない状態である。

4. this の状態確認

- this → bulletinboards の値が ArrayList<E>(id=168) となっている
 - また、デバッグログが以下の内容となっている。
 - [2024-12-24 12:17:40.239] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0
- ※bulletinboards.size() == 0 は、まだデータがbulletinboards リストに追加されていない初期状態を示しており、正常。

5.エラーログ確認

- 以下の通り、現状のデバッグログにはエラーや例外はなく、問題が発生していないことが確認できる。
 - [2024-12-24 12:17:28.630] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
 - [2024-12-24 12:17:38.705] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - PreparedStatement: HikariProxyPreparedStatement@1842907940 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM bulletinboard
 - [2024-12-24 12:17:40.239] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0
- ※現状の実行状態に問題がないことを示している。

- テスト後(ステップオーバー後)
 - 目的:resultSet に入っている bulletinboard_id をモデルにセットされたかどうかの確認する。
 - 確認内容:
 - Eclipse「変数」ウィンドウで以下を確認。
 - bulletinboard の階層を展開し、以下のプロパティを確認する。
 - bulletinboard → bulletinboard_content の値が「null」
 - bulletinboard_creation_day の値が「null」
 - bulletinboard_delete_day の値が「null」
 - bulletinboard_delete_flag の値が「0」
 - bulletinboard_id の値が「34」
 - bulletinboard_title の値が「null」
- ※上記の通り、bulletinboard_id の値が正常に取得できている。
- デバッグログを確認
 - [2024-12-24 11:57:57.366] DEBUG http-nio-8080-exec-9 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboard_id: = 34

- 169行目 :bulletinboard.setBulletinboard_title(resultSet.getString("bulletinboard_title"));
 - テスト前(ステップオーバー前)
 - 目的:resultSet に入っている bulletinboard_titleの値が モデルにセットされていないことを確認する。
 - 確認内容:
 - Eclipse「変数」ウィンドウで以下を確認。
 - bulletinboard の階層を展開し、以下のプロパティを確認する。
 - bulletinboard → bulletinboard_content の値が「null」
 - bulletinboard_creation_day の値が「null」
 - bulletinboard_delete_day の値が「null」
 - bulletinboard_delete_flag の値が「0」
 - bulletinboard_id の値が「34」
 - bulletinboard_title の値が「null」
- ※上記の通り、bulletinboard_title の値が「null」で、値がモデルにセットされていないことを確認。
- デバッグログを確認
 - [2024-12-24 12:17:28.630] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
 - [2024-12-24 12:17:38.705] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - PreparedStatement: HikariProxyPreparedStatement@1842907940 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM bulletinboard
 - [2024-12-24 12:17:40.239] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0
- ※この段階でのエラーログは出力されていない。

- テスト後(ステップオーバー後)
- 目的:resultSet に入っている bulletinboard_title をモデルにセットされたかどうかの確認する。

- ・確認内容:
 - Eclipse「変数」ウィンドウで以下を確認。
- ・bulletinboard の階層を展開し、以下のプロパティを確認する。
 - bulletinboard → bulletinboard_content の値が「null」
 - bulletinboard_creation_day の値が「null」
 - bulletinboard_delete_day の値が「null」
 - bulletinboard_delete_flag の値が「0」
 - bulletinboard_id の値が「34」
 - bulletinboard_title の値が「2024年7月18日本日」

※上記の通り、bulletinboard_title の値が正常に取得できている。

- ・デバッグログを確認
 - [2024-12-24 12:49:46.303] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboard_title: 2024年7月18日本日

- 172行目 :bulletinboard.setBulletinboard_content(resultSet.getString("bulletinboard_content"));
- ・テスト前（ステップオーバー前）
- ・目的 :resultSet に入っている bulletinboard_contentの値が モデルにセットされていないことを確認する。

- ・確認内容:
 - Eclipse「変数」ウィンドウで以下を確認。
- ・bulletinboard の階層を展開し、以下のプロパティを確認する。
 - bulletinboard → bulletinboard_content の値が「null」
 - bulletinboard_creation_day の値が「null」
 - bulletinboard_delete_day の値が「null」
 - bulletinboard_delete_flag の値が「0」
 - bulletinboard_id の値が「34」
 - bulletinboard_title の値が「2024年7月18日本日」

※上記の通り、bulletinboard_content の値が「null」で、値がモデルにセットされていないことを確認。

- ・デバッグログを確認
 - [2024-12-24 12:54:51.375] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
 - [2024-12-24 12:55:02.776] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - PreparedStatement: HikariProxyPreparedStatement@1606618620 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM bulletinboard
 - [2024-12-24 12:55:07.190] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0
- ※この段階でのエラーログは出力されていない。

- ・テスト後（ステップオーバー後）
- ・目的 :resultSet に入っている bulletinboard_content をモデルにセットされたかどうかの確認する。

- ・確認内容:
 - Eclipse「変数」ウィンドウで以下を確認。
- ・bulletinboard の階層を展開し、以下のプロパティを確認する。

bulletinboard → bulletinboard_content の値が「2024年7月18日本日掲示板、追記」

- bulletinboard_creation_day の値が「null」
- bulletinboard_delete_day の値が「null」
- bulletinboard_delete_flag の値が「0」
- bulletinboard_id の値が「34」
- bulletinboard_title の値が「2024年7月18日本日」

※上記の通り、bulletinboard_contentの値が正常に取得できている。

- ・デバッグログを確認
 - [2024-12-24 13:05:54.288] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboard_content: 2024年7月18日本日掲示板、追記

- 175行目 :bulletinboard.setUser_id(resultSet.getInt("user_id"));
- ・テスト前（ステップオーバー前）
- ・目的 :resultSet に入っている user_idの値が モデルにセットされていないことを確認する。

- ・確認内容:
 - Eclipse「変数」ウィンドウで以下を確認。
- ・bulletinboard の階層を展開し、以下のプロパティを確認する。

bulletinboard → user_idの値が「0」

※上記の通り、user_id の値が「0」で、値がモデルにセットされていないことを確認。

- ・デバッグログを確認
 - [2024-12-24 12:54:51.375] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
 - [2024-12-24 12:55:02.776] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - PreparedStatement: HikariProxyPreparedStatement@1606618620 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM bulletinboard
 - [2024-12-24 12:55:07.190] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0
- ※この段階でのエラーログは出力されていない。

- ・テスト後（ステップオーバー後）
- ・目的 :resultSet に入っている user_id をモデルにセットされたかどうかの確認する。

- ・確認内容:
 - Eclipse「変数」ウィンドウで以下を確認。

•bulletinboard の階層を展開し、以下のプロパティを確認する。
bulletinboard → user_id の値が「0」
※上記の通り、bulletinboard_contentの値が正常に取得できている。
•デバッグログを確認
[2024-12-24 13:17:06.499] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - user_id: 0

●178行目 :bulletinboard.setBulletinboard_delete_flag(resultSet.getInt("bulletinboard_delete_flag"));
•テスト前（ステップオーバー前）
•目的 :resultSet に入っている bulletinboard_delete_flagの値が モデルにセットされていないことを確認する。
•確認内容 :
- Eclipse「変数」ウィンドウで以下を確認。
•bulletinboard の階層を展開し、以下のプロパティを確認する。
bulletinboard → bulletinboard_content の値が「null」
bulletinboard_creation_day の値が「null」
bulletinboard_delete_day の値が「null」
bulletinboard_delete_flag の値が「0」
bulletinboard_id の値が「34」
bulletinboard_title の値が「2024年7月18日本日」
※上記の通り、bulletinboard_delete_flagの値が「0」で、値がモデルにセットされていないことを確認。
•デバッグログを確認
[2024-12-24 12:54:51.375] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
[2024-12-24 12:55:02.776] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - PreparedStatement: HikariProxyPreparedStatement@1606618620 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM bulletinboard
[2024-12-24 12:55:07.190] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0
※この段階でのエラーログは出力されていない。

•テスト後（ステップオーバー後）
•目的 :resultSet に入っている bulletinboard_delete_flag をモデルにセットされたかどうかの確認する。
•確認内容 :
- Eclipse「変数」ウィンドウで以下を確認。
•bulletinboard の階層を展開し、以下のプロパティを確認する。
bulletinboard → bulletinboard_content の値が「2024年7月18日本日掲示板、追記」
bulletinboard_creation_day の値が「null」
bulletinboard_delete_day の値が「null」
bulletinboard_delete_flag の値が「0」
bulletinboard_id の値が「34」
bulletinboard_title の値が「2024年7月18日本日」
※上記の通り、bulletinboard_delete_flag の値が正常に取得できている。
•デバッグログを確認
[2024-12-24 13:21:43.159] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboard_delete_flag: 0

●181行目 :bulletinboard.setBulletinboard_delete_day(resultSet.getString("bulletinboard_delete_day"));
•テスト前（ステップオーバー前）
•目的 :resultSet に入っている bulletinboard_delete_dayの値が モデルにセットされていないことを確認する。
•確認内容 :
- Eclipse「変数」ウィンドウで以下を確認。
•bulletinboard の階層を展開し、以下のプロパティを確認する。
bulletinboard → bulletinboard_content の値が「null」
bulletinboard_creation_day の値が「null」
bulletinboard_delete_day の値が「null」
bulletinboard_delete_flag の値が「0」
bulletinboard_id の値が「34」
bulletinboard_title の値が「2024年7月18日本日」
※上記の通り、bulletinboard_delete_dayの値が「0」で、値がモデルにセットされていないことを確認。
•デバッグログを確認
[2024-12-24 12:54:51.375] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - resultSet.next() 初回評価: true
[2024-12-24 12:55:02.776] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - PreparedStatement: HikariProxyPreparedStatement@1606618620 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM bulletinboard
[2024-12-24 12:55:07.190] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboards.size(): 0
※この段階でのエラーログは出力されていない。

•テスト後（ステップオーバー後）
•目的 :resultSet に入っている bulletinboard_delete_dayの値がモデルにセットされたかどうかの確認する。
•確認内容 :
- Eclipse「変数」ウィンドウで以下を確認。
•bulletinboard の階層を展開し、以下のプロパティを確認する。
bulletinboard → bulletinboard_content の値が「2024年7月18日本日掲示板、追記」

```
        bulletinboard_creation_day の値が「null」
        bulletinboard_delete_day の値が「2999-12-31 00:00:00」
        bulletinboard_delete_flag の値が「0」
        bulletinboard_id の値が「34」
        bulletinboard_title の値が「2024年7月18日本日」
※上記の通り、bulletinboard_delete_day の値が正常に取得できている。
・デバッグログを確認
    [2024-12-24 13:25:58.384] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - bulletinboard_delete_day: 2999-12-31 00:00:00
※前述の処理をテーブルの件数(13件)分ループ処理を行い、80行目に進む
```

- 80行目 :return SUCCESS;
- ・テスト前(ステップオーバー前)
- ・目的:グローバル変数の「this」にモデルの値が格納されていないことを確認する、
- ・確認内容:グローバル変数の「this」にデータベースから取得したデータが「bulletinboards」に格納されていることを確認する
 - Eclipse「変数」ウィンドウで以下を確認。
- ・this の階層を展開し、bulletinboardsの以下のプロパティを確認する。

```
this      MoveBulletinboardManagementAction (id=119)
bulletinboards  ArrayList<E> (id=174)
    [0]          User (id=205)
    [1]          User (id=237)
    [2]          User (id=268)
    [3]          User (id=300)
    [4]          User (id=332)
    [5]          User (id=363)
    [6]          User (id=394)
    [7]          User (id=425)
    [8]          User (id=456)
    [9]          User (id=487)
    [10]         User (id=518)
    [11]         User (id=562)
    [12]         User (id=593)
```

※この時点でデータベースから取得した掲示板情報13テーブル)がthisのbulletinboardsに格納されていことを確認。

- ・テスト後(ステップオーバー後)
- ・目的:ステップオーバー後、BaseActionクラスに処理が渡されることを確認する。
- ・確認内容:ステップオーバー後、BaseActionクラスの72行目に処理が遷移することを確認
 - 以降の処理でBaseActionクラスとフレームワーク後の処理後に、掲示板管理画面が表示されるが、テスト範囲外の為、割愛する。

以上

■単体テストNo5.2

MoveBulletinboardManagementActionクラスのエラー処理

■目的

- ①セッション未設定時のエラー処理が正しく動作すること。
- ② データベース接続エラーの処理が正しく動作すること。
- ③クエリ実行エラーの処理が正しく動作すること。(リスクがあるので中止)

■テスト対象行

＜①セッション未設定時のエラー処理＞

```
62行目 :if (sessionUser == null) {  
63行目 :String errorMessage = "User session is missing.";  
64行目 :addActionError(errorMessage);  
65行目 :logger.error("Error in MoveBulletinboardManagementAction: " + errorMessage);  
66行目 :return ERROR;
```

＜② データベース接続エラーの処理＞

```
204行目 :} catch (SQLException e) {  
205行目 :logger.error("データベース接続中にエラーが発生しました", e);  
206行目 :throw e;
```

■テスト実施内容

＜①セッション未設定時のエラー処理＞

- 管理者権限にて、管理メニューにログインする
- 管理メニュー画面にてセッションを削除する
管理メニュー画面にて右クリックし、「検証」をクリックする
アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、「掲示板管理画面へ」リンクをクリックする
- 「掲示板管理画面へ」リンクをクリック後、以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

＜② データベース接続エラーの処理＞

- 管理メニュー画面にログインする
- 管理メニューにて管理者権限にて、以下のコマンドを実効しMySQLサーバーのサービスを停止させる
コマンド:net stop mysql80

※以下、操作履歴

C:\WINDOWS\system32>net stop mysql80

MySQL80 サービスを停止中です..

MySQL80 サービスは正常に停止されました。

- 上記操作実行後、改めて「掲示板管理画面へ」リンクをクリックする。以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

- 以下、エラーログ

```
[2025-01-01 15:46:33.176] ERROR http-nio-8080-exec-7 com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction - データベース接続中にエラーが発生しました
    at com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction.getAllBulletinboards(MoveBulletinboardManagementAction.java:94) [classes/:?]
    at com.company.bulletinboard.action.admin.bulletinboard.MoveBulletinboardManagementAction.mainProc(MoveBulletinboardManagementAction.java:76) [classes/:?]
```

- テスト実施後、MySQLサーバーのサービスを起動し、掲示板管理画面へ正常に遷移できることを確認する

コマンド :net start mysql80

※以下、操作履歴

C:\WINDOWS\system32>net start mysql80

MySQL80 サービスを開始します.

MySQL80 サービスは正常に開始されました。

以上

■単体テストNo6.1

ユーザー管理画面への遷移処理

■目的

- ・「ユーザー管理画面へ」リンクをクリック後、データベースのテーブルからユーザーデータの一覧を取得できること。
- ・上記処理後、「SUCCESS」を返すことを確認する。

■テスト対象行

```
53行目 :if (sessionUser == null) {
90行目 :PreparedStatement preparedStatement = connection.prepareStatement(sql) {
142行目 :while (resultSet.next()) {
164行目 :User bulletinboard = new User();
166行目 :bulletinboard.setBulletinboard_id(resultSet.getInt("bulletinboard_id"));
169行目 :bulletinboard.setBulletinboard_title(resultSet.getString("bulletinboard_title"));
172行目 :bulletinboard.setBulletinboard_content(resultSet.getString("bulletinboard_content"));
175行目 :bulletinboard.setUser_id(resultSet.getInt("user_id"));
178行目 :bulletinboard.setBulletinboard_delete_flag(resultSet.getInt("bulletinboard_delete_flag"));
181行目 :bulletinboard.setBulletinboard_delete_day(resultSet.getString("bulletinboard_delete_day"));
80行目  :return SUCCESS;
```

■テスト実施内容

●53行目 :if (sessionUser == null) {

- ・目的:セッションが正しく設定されているかを確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「JP9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

- ・テスト後(ステップオーバー後)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値:

sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「JP9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」

上記の通り、ステップオーバー後も情報が保持されており、意図した内容。

2. this → session の内容:

- sessionの値が「SessionMap」オブジェクトである。
- session内にキー「loggedInUser」が存在する(「session.containsKey("loggedInUser") == true」)。
- session.get("loggedInUser")の値が「sessionUser」オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

auth_type: 1
delete_day: "2999-12-31 00:00:00"
delete_flag: 0
password: "JP9T-LH2"
user_id: 30
user_name: "tesutuser1"

上記の通り、クラスフィールドにsessionUserオブジェクトと同じ内容が補完されている。正常動作。

3. ログに以下の内容が出力されている:

[2025-01-04 11:41:09.156] INFO http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - Session User: tesutuser1

●93行目 :PreparedStatement preparedStatement = connection.prepareStatement(sql) {

- ・目的:接続の正常性を確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認。

- ・sql → SELECT * users (id=157)
※不正な値(NULLや予期しない構文など)が含まれていない

2.「connection」オブジェクトの確認。接続がクローズされていないことの確認する。

- ・connection → connectionオブジェクト自体が存在し、NULLで無いこと
- ・connection → delegate → openStatementsの値が「CopyOnWriteArrayList<E> (id=213)」
※openStatementsの値が空でなくアクセス可能な状態。
- ・isAutoCommit → isAutoCommitの値が「true」であること

3.接続先データベース情報の確認

- ・connection → delegate → origHostToConnectTo → 「localhost」(id=202)」であること
- ・connection → delegate → origPortToConnectTo → ポート番号が「3306」であること
- ・connection → delegate → database → データベース名が「bulletinboard_db」(id=188)」であること
- ・connection → delegate → user → 「root」(id=236)」であること

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返しており、ログに「データベース接続は正常です。」というメッセージが表示されている。

ログ内容:[2025-01-04 11:54:30.302] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - データベース接続は正常です。ホスト localhost, ポート: 3306, データベース: bulletinboard_db

- ・テスト後(ステップオーバー後)

確認内容:PreparedStatement の生成が正しく行われ、後続処理に影響がないことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

- 1.「sql」変数の確認
 - ・sql → SELECT * FROM users" (id=157)
 - ※正しいSQL 文("SELECT * FROM users")を保持していることを確認
- 2. connection の有効性を確認
 - ・preparedStatement → isClosedの値が「false」であることを確認
 - ※ステートメントが閉じていない状態
- 3.「proxyResultSet」の確認
 - ・preparedStatement → proxyResultSetの値が「null」であることを確認
- 4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返すことを確認

ログ内容:[2025-01-04 12:06:14.385] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - データベース接続は正常です。ホスト localhost, ポート: 3306, データベース: bulletinboard_db

※※ログメッセージに「データベース接続は正常です。」が出力されていることを確認。接続は有効な状態

●142行目 :while (resultSet.next()) {

・目的:resultSet の状態やクエリの実行結果に問題がないかを確認するwhile 文内の処理に進む準備ができていることを確認する。

・テスト前 (ステップオーバー前)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「resultSet」の有効性確認

・resultSet → statement → delegate → isClosed が false

※resultSet はまだ有効であり、操作可能な状態

・while 文が開始される前にデバックログにて、以下のログを確認する。

[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true

※ログに「初回評価 true」と出力されているので、少なくとも件のデータが存在し、resultSet.next() が正常動作していることを確認。

2.「sql」変数の確認

・sql → SELECT * FROM users" (id=157)

※正しいSQL 文("SELECT * FROM users")を保持していることを確認

3.「preparedStatement」の有効性確認

・preparedStatement → isClosed の値が「false」

※ステートメントは有効で操作可能な状態

・デバックログに以下のログが出力されること。

[2025-01-04 12:42:49.100] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - PreparedStatement: HikariProxyPreparedStatement@922532973 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM users

※ログにステートメント情報 HikariProxyPreparedStatement および SQL 文) が出力されているので、SQL 文が正常に紐づけられていることを確認。

4.users リストの状態確認

・デバックログに以下のログが出力されること。

[2025-01-04 12:42:49.087] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0

※users リストの要素数が「0」であることから、データがまだ追加されていない状態であることを確認。

・テスト後 (ステップオーバー後)

確認内容 :PreparedStatement の生成が正しく行われ、後続処理に影響がないことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

1. resultSet.next() の評価結果確認

・デバックログに以下のログが出力されること。

[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true

※SQLクエリで取得したデータがresultSet に存在し、次の処理に進む準備が整っている

2.preparedStatement と resultSet の有効性確認

・preparedStatement → isClosed の値が「false」

・resultSet → delegate → isClosedの値が「false」

※データベース接続が維持されており、preparedStatement と resultSet がクローズされていないことを確認

3.users リストの状態確認

・デバッグで以下のログを確認する。

[2025-01-04 13:08:11.865] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0

※リストが変更されていないことを確認。この時点ではモデル生成やデータ追加処理が未実行のためusers のサイズが 0 のままである

4.「データベースから該当する掲示板データを正しく取得できていること確認する

・Eclipse「変数」ウィンドウの 以下を確認する

```
debugData      ArrayList<E> (id=191)
    [0]          HashMap<K,V> (id=284)
        [0]      HashMap$Node<K,V> (id=297)
            key      password (id=321)
            value     n5J%v&db (id=296)
        [1]      HashMap$Node<K,V> (id=300)
            key      auth_type (id=324)
            value     Integer (id=299)
        [2]      HashMap$Node<K,V> (id=291)
            key      user_id (id=326)
            value     Integer (id=289)
        [3]      HashMap$Node<K,V> (id=294)
            key      user_name (id=327)
            value     nishioka444 (id=293)
        [4]      HashMap$Node<K,V> (id=305)
            key      delete_day (id=329)
            value     2025-12-31 00:00:00 (id=304)
            hash      0
            value     (id=331)
        [5]      HashMap$Node<K,V> (id=302)
            key      delete_flag (id=334)
            value     Integer (id=299)
```

※上記の値が、掲示板テーブルのデータと一致すること。

●164行目 :User bulletinboard = new User();

・テスト前 (ステップオーバー前)

・目的:

- ①SQL文の実行準備が正しく行われていることを確認する
 - データベース接続が有効であること。
 - 実行されるSQL文が期待どおりであること。
- ②データ取得の準備が正しく行われていることを確認する
 - ResultSetが初期状態にあること (カーソルがまだデータ行に移動していない)。
 - 次のresultSet.next()呼び出しにより、データが正しく取得されることが期待できること。
- ③初期状態のプロパティが正しいことを確認する
 - アクションオブジェクトのプロパティ(usersなど) が初期化されていること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.resultSet の状態確認

・resultSet → delegate → currentRow の値が「-1」

※カーソルが結果セットの先頭行の「前」にあり、このためcurrentRow は -1 を示す。これは、初回のresultSet.next()が呼び出されていない(初期状態)ことの裏付けとなる。正常な動き。

・デバッグログを確認

[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true

※このログは、resultSet.next() が初回評価でtrue を返しているため、少なくとも1件のデータが正しく取得されていることを示している。のためcurrentRow == -1 は初期状態として正しいと判断できる。

2. preparedStatement の状態確認

・sql → SELECT * FROM users" (id=157)

※ sql に期待通りのクエリ(SELECT * FROM users) が格納されていることを確認。誤ったSQL文では無いこと。

3.接続状態の確認

・preparedStatement → isClosed の値が「 false」であることを確認。

※ステートメントが閉じておらず、接続が維持されていることを確認

・resultSet → delegate → isClosed の値が「false」であることを確認。

※ResultSetも閉じられていない状態である。

4. this の状態確認

・this → users の値が ArrayList<E>(id=186) となっている

・また、デバッグログが以下の内容となっている。

[2025-01-04 13:08:11.865] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0

※ログにも明示されており、初期状態として期待通りでる。

5.エラーログ確認

・以下の通り、現状のデバッグログにはエラーや例外はなく、問題が発生していないことが確認できる。

[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true

[2025-01-04 13:08:10.634] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - PreparedStatement: HikariProxyPreparedStatement@2137757793 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM users

[2025-01-04 13:08:11.865] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0

・テスト後 (ステップオーバー後)

・目的:

- ①User モデルのインスタンスが正しく生成され、初期化状態が期待どおりであることを確認する。
- ②その他のオブジェクト状態がステップオーバー後に変化がないことを確認する。

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

1.user オブジェクトの生成確認

・user の階層を展開し、以下のプロパティを確認する。

- auth_typeの値が「0」

- delete_dayの値が「null」

- delete_flagの値が「null」

- bulletinboard_delete_flagの値が「0」

- passwordの値が「null」

- user_idの値が「0」

- user_nameの値が「null」

※上記の通り、各プロパティがモデルの初期状態に一致している。

※User モデルが初期状態であり、まだデータが設定されていない状態を示す。

2.resultSet の状態確認

・resultSet → delegate → currentRow の値が「-1」

※カーソルが結果セットの先頭行の「前」にあり、このためcurrentRow は -1 を示す。これは、初回のresultSet.next()が呼び出されていない(初期状態)ことの裏付けとなる。正常な動き。

・デバッグログを確認

[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true

※このログで、resultSet.next() が初回評価でtrue を返しているため、少なくとも1件のデータが正しく取得されていることを示している。のためcurrentRow == -1 は初期状態として正しいと判断できる。

※結果として、ステップオーバー後に変化無しと裏付けられる。

3.接続状態の確認

・preparedStatement → isClosed の値が「 false」であることを確認。

※ステートメントが閉じておらず、接続が維持されていることを確認

・resultSet → delegate → isClosed の値が「false」であることを確認。

※ResultSetも閉じられていない状態である。

※結果として、ステップオーバー後に変化無しと裏付けられる。

4. this の状態確認

・this → users の値が ArrayList<E>(id=186) となっている

・また、デバッグログが以下の内容となっている。

[2025-01-04 13:08:11.865] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0

※ログにも明示されており、初期状態として期待通りでる。

※結果として、ステップオーバー後に変化無しと裏付けられる。

5.エラーログ確認

・以下の通り、現状のデバッグログにはエラーや例外はなく、問題が発生していないことが確認できる。

[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true

[2025-01-04 13:08:10.634] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - PreparedStatement: HikariProxyPreparedStatement@2137757793 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM users

[2025-01-04 13:08:11.865] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0
※結果として、ステップオーバー後に変化無しと裏付けられる。

●165行目 :user.setUser_id(resultSet.getInt("user_id"));

・テスト前 (ステップオーバー前)

・目的: モデルの初期状態が期待通りであることを確認する (カーソル位置、モデル、接続、ログ)

1.resultSet のカーソル位置の確認

・resultSet → delegate → currentRow の値が「-1」

※currentRow は -1 を示す。これは、resultSet のカーソルはまだ「最初の行の前」にあることを確認できる。カーソルの移動する準備が整っている状態。正常な動き。

・デバッグログを確認

[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true

※このログで、resultSet.next() が初回評価で true を返しているため、少なくとも1件のデータが正しく取得されていることを示している。のため currentRow == -1 は初期状態として正しいと判断できる。

2.user オブジェクトの生成確認

・user の階層を展開し、以下のプロパティを確認する。

- auth_typeの値が「0」

- delete_dayの値が「null」

- delete_flagの値が「null」

- passwordの値が「null」

- user_idの値が「0」

- user_nameの値が「null」

※上記の通り、各プロパティがモデルの初期状態に一致している。

※モデルが正しく生成されており、データがセットされる準備が整っていることが確認できる。

3.接続状態の確認

・preparedStatement → isClosed の値が「 false」であることを確認。

※ステートメントが閉じておらず、接続が維持されていることを確認

・resultSet → delegate → isClosed の値が「false」であることを確認。

※ResultSetも閉じられていない状態である。

4. this の状態確認

・this → users の値が ArrayList<E>(id=186) となっている

・また、デバッグログが以下の内容となっている。

[2025-01-04 13:08:11.865] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0

※users.size() == 0 は、まだデータがusers リストに追加されていない初期状態を示しており、正常。

5.エラーログ確認

・以下の通り、現状のデバッグログにはエラーや例外はなく、問題が発生していないことが確認できる。

[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true

[2025-01-04 13:08:10.634] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - PreparedStatement: HikariProxyPreparedStatement@2137757793 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM users

[2025-01-04 13:08:11.865] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0

※現状の実行状態に問題がないことを示している。

・テスト後 (ステップオーバー後)

・目的: resultSet に入っている user_id をモデルにセットされたかどうかの確認する。

・確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

・user の階層を展開し、以下のプロパティを確認する。

- auth_typeの値が「0」

- delete_dayの値が「null」

- delete_flagの値が「null」

- passwordの値が「null」

- user_idの値が「15」

- user_nameの値が「null」

※上記の通り、user_id の値が正常に取得できている。

・デバッグログを確認

[2025-01-04 13:37:49.613] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - user_id: 15

●168行目 :user.setUser_name(resultSet.getString("user_name"));

・テスト前 (ステップオーバー前)

・目的: resultSet に入っている user_name の値が モデルにセットされていないことを確認する。

・確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

・user の階層を展開し、以下のプロパティを確認する。

- auth_typeの値が「0」

- delete_dayの値が「null」

- delete_flagの値が「null」

- passwordの値が「null」

- user_idの値が「15」

- user_nameの値が「null」

※上記の通り、user_name の値が「null」で、値がモデルにセットされていないことを確認。

・デバッグログを確認

[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true

[2025-01-04 13:08:10.634] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - PreparedStatement: HikariProxyPreparedStatement@2137757793 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM users

[2025-01-04 13:08:11.865] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0

※この段階でのエラーログは出力されていない。

・テスト後 (ステップオーバー後)

・目的: resultSet に入っている user_name をモデルにセットされたかどうかの確認する。

・確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

・user の階層を展開し、以下のプロパティを確認する。

- auth_typeの値が「0」

- delete_dayの値が「null」

- delete_flagの値が「null」

- passwordの値が「null」

- user_idの値が「15」
- user_nameの値が「nishioka444」
※上記の通り、user_name の値が正常に取得できている。

・デバッグログを確認
[2025-01-04 13:54:58.388] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - user_name: nishioka444

●171行目 :user.setPassword(resultSet.getString("password"));
・テスト前 (ステップオーバー前)
・目的:resultSet に入っている passwordの値が モデルにセットされていないことを確認する。

・確認内容:
- Eclipse「変数」ウィンドウで以下を確認。
・userの階層を展開し、以下のプロパティを確認する。
- auth_typeの値が「0」
- delete_dayの値が「null」
- delete_flagの値が「null」
- passwordの値が「null」
- user_idの値が「15」
- user_nameの値が「nishioka444」

※上記の通り、password の値が「null」で、値がモデルにセットされていないことを確認。

・デバッグログを確認
[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true
[2025-01-04 13:08:10.634] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - PreparedStatement: HikariProxyPreparedStatement@2137757793 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM users
[2025-01-04 13:08:11.865] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0
※この段階でのエラーログは出力されていない。

・テスト後 (ステップオーバー後)
・目的:resultSet に入っている password をモデルにセットされたかどうかの確認する。

・確認内容:
- Eclipse「変数」ウィンドウで以下を確認。
・user の階層を展開し、以下のプロパティを確認する。
- auth_typeの値が「0」
- delete_dayの値が「null」
- delete_flagの値が「null」
- passwordの値が「n5J%v&db」
- user_idの値が「15」
- user_nameの値が「nishioka444」

※上記の通り、passwordの値が正常に取得できている。

・デバッグログを確認
[2025-01-04 14:00:07.967] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - password: n5J%v&db

●174行目 :user.setAuth_type(resultSet.getInt("auth_type"));
・テスト前 (ステップオーバー前)
・目的:resultSet に入っている auth_typeの値が モデルにセットされていないことを確認する。

・確認内容:
- Eclipse「変数」ウィンドウで以下を確認。
・user の階層を展開し、以下のプロパティを確認する。

auth_type → auth_typeの値が「0」

※上記の通り、auth_type の値が「0」で、値がモデルにセットされていないことを確認。

・デバッグログを確認
[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true
[2025-01-04 13:08:10.634] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - PreparedStatement: HikariProxyPreparedStatement@2137757793 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM usersHikariProxyPreparedStatement@1606618620 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM bulletinbo
[2025-01-04 13:08:11.865] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0
※この段階でのエラーログは出力されていない。

・テスト後 (ステップオーバー後)
・目的:resultSet に入っている auth_type をモデルにセットされたかどうかの確認する。

・確認内容:
- Eclipse「変数」ウィンドウで以下を確認。
・user の階層を展開し、以下のプロパティを確認する。

user → auth_type の値が「1」

※上記の通り、auth_typeの値が正常に取得できている。

・デバッグログを確認
[2025-01-04 14:04:06.088] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - auth_type: 1

●177行目 :user.setDelete_flag(resultSet.getInt("delete_flag"));
・テスト前 (ステップオーバー前)
・目的:resultSet に入っている delete_flagの値が モデルにセットされていないことを確認する。

・確認内容:
- Eclipse「変数」ウィンドウで以下を確認。
・user の階層を展開し、以下のプロパティを確認する。
- auth_typeの値が「0」
- delete_dayの値が「null」
- delete_flagの値が「null」
- passwordの値が「n5J%v&db」
- user_idの値が「15」
- user_nameの値が「nishioka444」

※上記の通り、delete_flagの値が「null」で、値がモデルにセットされていないことを確認。

・デバッグログを確認
[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true
[2025-01-04 13:08:10.634] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - PreparedStatement: HikariProxyPreparedStatement@2137757793 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM users
[2025-01-04 13:08:11.865] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0
※この段階でのエラーログは出力されていない。

・テスト後 (ステップオーバー後)
・目的:resultSet に入っている delete_flag をモデルにセットされたかどうかの確認する。

・確認内容:

- Eclipse「変数」ウィンドウで以下を確認。
・user の階層を展開し、以下のプロパティを確認する。
- auth_typeの値が「0」
- delete_dayの値が「null」
- delete_flagの値が「1」
- passwordの値が「n5J%v&db」
- user_idの値が「15」
- user_nameの値が「nishioka444」

※上記の通り、delete_flag の値が正常に取得できている。

・デバッグログを確認
[2025-01-04 14:10:17.243] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - delete_flag: 1

●180行目 :user.setDelete_day(resultSet.getString("delete_day"));
・テスト前 (ステップオーバー前)
・目的:resultSet に入っているdelete_dayの値が モデルにセットされていないことを確認する。

・確認内容:
- Eclipse「変数」ウィンドウで以下を確認。
・user の階層を展開し、以下のプロパティを確認する。
- auth_typeの値が「0」
- delete_dayの値が「null」
- delete_flagの値が「1」
- passwordの値が「n5J%v&db」
- user_idの値が「15」
- user_nameの値が「nishioka444」

※上記の通り、delete_dayの値が「null」で、値がモデルにセットされていないことを確認。

・デバッグログを確認
[2025-01-04 12:44:58.466] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - resultSet.next() 初回評価: true
[2025-01-04 13:08:10.634] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - PreparedStatement: HikariProxyPreparedStatement@2137757793 wrapping com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM users
[2025-01-04 13:08:11.865] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - users.size(): 0
※この段階でのエラーログは出力されていない。

・テスト後 (ステップオーバー後)
・目的:resultSet に入っているdelete_dayの値がモデルにセットされたかどうかの確認する。

・確認内容:
- Eclipse「変数」ウィンドウで以下を確認。
・userの階層を展開し、以下のプロパティを確認する。
- auth_typeの値が「0」
- delete_dayの値が「2025-12-31 00:00:00」
- delete_flagの値が「1」
- passwordの値が「n5J%v&db」
- user_idの値が「15」
- user_nameの値が「nishioka444」

※上記の通り、delete_day の値が正常に取得できている。

・デバッグログを確認
[2025-01-04 14:16:11.665] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - delete_day: 2025-12-31 00:00:00
※前述の処理をテーブルの件数(12件)分ループ処理を行い、80行目に進む

●74行目 :return SUCCESS;
・テスト前 (ステップオーバー前)
・目的:グローバル変数の「this」にモデルの値が格納されていないことを確認する、
・確認内容:グローバル変数の「this」にデータベースから取得したデータが「users」に格納されていることを確認する
- Eclipse「変数」ウィンドウで以下を確認。
・this の階層を展開し、usersの以下のプロパティを確認する。

this MoveUserManagementAction (id=883)
users ArrayList<E> (id=833)
[0] User (id=855)
[1] User (id=888)
[2] User (id=921)
[3] User (id=954)
[4] User (id=987)
[5] User (id=1020)
[6] User (id=1053)
[7] User (id=1086)
[8] User (id=1119)
[9] User (id=1152)
[10] User (id=1185)
[11] User (id=1218)

※この時点でデータベースから取得したユーザー情報(12テーブル)がthisのusersに格納されていることを確認。

・テスト後 (ステップオーバー後)
・目的: ステップオーバー後、BaseActionクラスに処理が渡されることを確認する。
・確認内容:ステップオーバー後、BaseActionクラスの72行目に処理が遷移することを確認
以降の処理でBaseActionクラスとフレームワーク後の処理後に、ユーザー管理画面が表示されるが、テスト範囲外の為、割愛する。

以上

■単体テストNo6.2

MoveUserManagementActionクラスのエラー処理

■目的

- ①セッション未設定時のエラー処理が正しく動作すること。
- ② データベース接続エラーの処理が正しく動作すること。
- ③クエリ実行エラーの処理が正しく動作すること。(リスクがあるので中止)

■テスト対象行

＜①セッション未設定時のエラー処理＞

```
55行目 :if (sessionUser == null) {  
56行目 :String errorMessage = "User session is missing.";  
57行目 :addActionError(errorMessage);  
58行目 :logger.error("Error in MoveBulletinboardManagementAction: " + errorMessage);  
59行目 :return ERROR;
```

＜② データベース接続エラーの処理＞

```
200行目 :} catch (SQLException e) {  
201行目 :logger.error("データベース接続中にエラーが発生しました", e);  
202行目 :throw e;
```

■テスト実施内容

＜①セッション未設定時のエラー処理＞

- 管理者権限にて、管理メニューにログインする
- 管理メニュー画面にてセッションを削除する
管理メニュー画面にて右クリックし、「検証」をクリックする
アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、「ユーザー管理画面へ」リンクをクリックする
- 「ユーザー管理画面へ」リンクをクリック後、以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

＜② データベース接続エラーの処理＞

- 管理メニュー画面にログインする
- 管理メニューにて管理者権限にて、以下のコマンドを実効しMySQLサーバーのサービスを停止させる
コマンド :net stop mysql80
※以下、操作履歴

C:\WINDOWS\system32>net stop mysql80

MySQL80 サービスを停止中です..

MySQL80 サービスは正常に停止されました。

- 上記操作実行後、改めて「ユーザー管理画面へ」リンクをクリックする。以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

- 以下、エラーログ

```
[2025-01-05 09:38:17.185] ERROR http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.MoveUserManagementAction - データベース接続中にエラーが発生しました
    at com.company.bulletinboard.action.admin.user.MoveUserManagementAction.getAllUsers(MoveUserManagementAction.java:90) [classes/:?]
    at com.company.bulletinboard.action.admin.user.MoveUserManagementAction.mainProc(MoveUserManagementAction.java:69) [classes/:?]
```

- テスト実施後、MySQLサーバーのサービスを起動し、掲示板管理画面へ正常に遷移できることを確認する

コマンド:net start mysql80

※以下、操作履歴

C:\WINDOWS\system32>net start mysql80

MySQL80 サービスを開始します.

MySQL80 サービスは正常に開始されました。

以上

■単体テストNo8

管理メニューへ戻る処理

■目的

各管理画面（掲示板管理画面、投稿管理画面、ユーザー管理画面）から「管理メニュー画面に戻る」リンクをクリックした際に、正しく管理メニュー画面(ManagementMenu.jsp) に遷移することを確認する。

■テスト実施手順

- 1.掲示板管理画面から遷移の確認
 - ・掲示板管理画面にアクセス。
 - ・「管理メニュー画面に戻る」リンクをクリック。
 - ・ManagementMenu.jsp に遷移することを確認。
- 2.投稿管理画面から遷移の確認
 - ・投稿管理画面にアクセス。
 - ・「管理メニュー画面に戻る」リンクをクリック。
 - ・ManagementMenu.jsp に遷移することを確認。
- 3.ユーザー管理画面から遷移の確認
 - ・ユーザー管理画面にアクセス。
 - ・「管理メニュー画面に戻る」リンクをクリック。
 - ・ManagementMenu.jsp に遷移することを確認。

■テスト実施内容

- <1.掲示板管理画面から遷移の確認>
- 1-1. 掲示板管理画面にアクセスする。

掲示板管理

管理メニューに戻る

掲示板作成

| | | |
|--------------------------------|--------------------|--------------------|
| 2024年7月18日本日 | 編集 | 削除 |
| 2024年7月18日その② | 編集 | 削除 |
| 2024年7月18日その3 | 編集 | 削除 |
| 2024年7月22日 | 編集 | 削除 |
| 2024年9月1日 ※編集テスト | 編集 | 削除 |
| 2024年9月25日の掲示板 | 編集 | 削除 |
| 2024年9月25日 ※ドロップダウンリストで削除しない | 編集 | 削除 |
| 2024年9月25日 ※ドロップダウンリストで削除するを選択 | 編集 | 削除 |

せる

| | | |
|---------------------------------|--------------------|--------------------|
| 2024年9月25日 ※作成画面から掲示板IDフィールドを削除 | 編集 | 削除 |
| 2024年9月26日 ※削除フラグ：削除しない | 編集 | 削除 |
| 2024年9月26日 ※削除フラグ：削除する | 編集 | 削除 |
| testtitle | 編集 | 削除 |
| 営業部掲示板 | 編集 | 削除 |

を確認する。

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

1-2.「管理メニュー画面に戻る」リンクをクリックする。

掲示板管理

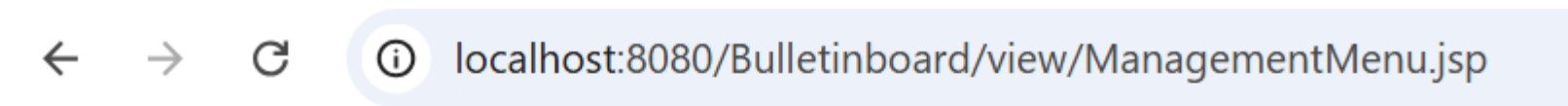
[管理メニューに戻る](#)

掲示板作成

| | | |
|---------------------------------|--------------------|--------------------|
| 2024年7月18日本日 | 編集 | 削除 |
| 2024年7月18日その② | 編集 | 削除 |
| 2024年7月18日その3 | 編集 | 削除 |
| 2024年7月22日 | 編集 | 削除 |
| 2024年9月1日 ※編集テスト | 編集 | 削除 |
| 2024年9月25日の掲示板 | 編集 | 削除 |
| 2024年9月25日 ※ドロップダウンリストで削除しない | 編集 | 削除 |
| 2024年9月25日 ※ドロップダウンリストで削除するを選択 | 編集 | 削除 |
| 2024年9月25日 ※作成画面から掲示板IDフィールドを削除 | 編集 | 削除 |
| 2024年9月26日 ※削除フラグ：削除しない | 編集 | 削除 |
| 2024年9月26日 ※削除フラグ：削除する | 編集 | 削除 |
| testtitle | 編集 | 削除 |

| | | |
|--------|----|----|
| 営業部掲示板 | 編集 | 削除 |
|--------|----|----|

1-3. ManagementMenu.jsp に遷移することを確認



管理者メニュー

[掲示板管理画面へ](#)

[投稿管理画面へ](#)

[ユーザー管理画面へ](#)

[ログアウト](#)

<2.投稿管理画面から遷移の確認>
※未実装の為、割愛

<3.ユーザー管理画面から遷移の確認>
3-1. ユーザー管理画面にアクセスする。

ユーザー管理

[管理メニューに戻る](#)

ユーザー作成

| ユーザーID | ユーザー名 | 権限 | 操作 | |
|--------|-------------|----|----|----|
| 15 | nishioka444 | 1 | 編集 | 削除 |
| 24 | uehara1 | 0 | 編集 | 削除 |
| 25 | uehara23 | 0 | 編集 | 削除 |
| 26 | ueharaadmin | 1 | 編集 | 削除 |

| | | | | |
|----|-------------|---|--------------------|--------------------|
| 30 | tesutuser1 | 1 | 編集 | 削除 |
| 31 | testuser2 | 1 | 編集 | 削除 |
| 32 | testuser3 | 1 | 編集 | 削除 |
| 33 | tesutuser4 | 0 | 編集 | 削除 |
| 34 | testuser5 | 1 | 編集 | 削除 |
| 35 | tesutuser1 | 0 | 編集 | 削除 |
| 36 | tesutuser11 | 0 | 編集 | 削除 |
| 38 | uehara10001 | 0 | 編集 | 削除 |

3-2.「管理メニュー画面に戻る」リンクをクリックする。

ユーザー管理

[管理メニューに戻る](#)

ユーザー作成

| ユーザーID | ユーザー名 | 権限 | 操作 | |
|--------|-------------|----|--------------------|--------------------|
| 15 | nishioka444 | 1 | 編集 | 削除 |
| 24 | uehara1 | 0 | 編集 | 削除 |
| 25 | uehara23 | 0 | 編集 | 削除 |
| 26 | ueharaadmin | 1 | 編集 | 削除 |
| 30 | tesutuser1 | 1 | 編集 | 削除 |
| 31 | testuser2 | 1 | 編集 | 削除 |
| 32 | testuser3 | 1 | 編集 | 削除 |
| 33 | tesutuser4 | 0 | 編集 | 削除 |
| 34 | testuser5 | 1 | 編集 | 削除 |
| 35 | tesutuser1 | 0 | 編集 | 削除 |
| 36 | tesutuser11 | 0 | 編集 | 削除 |
| 38 | uehara10001 | 0 | 編集 | 削除 |

3-3. ManagementMenu.jsp に遷移することを確認

← → ↻ ⓘ localhost:8080/Bulletinboard/view/ManagementMenu.jsp

管理者メニュー

[掲示板管理画面へ](#)

[投稿管理画面へ](#)

[ユーザー管理画面へ](#)

[ログアウト](#)

以上

■単体テストNo9.1

掲示板作成画面へ遷移する処理

■目的

掲示板管理画面から「掲示板作成」ボタンをクリックした際に、正しく掲示板作成画面(CreateBulletinboardScreen.jsp) に遷移することを確認する。

■テスト実施手順

掲示板管理画面から遷移の確認

- ・掲示板管理画面にアクセス。
- ・「掲示板作成」ボタンをクリック。
- ・CreateBulletinboardScreen.jsp に遷移することを確認。

■テスト実施内容

< 掲示板管理画面から掲示板作成画面への遷移確認 >

- 掲示板管理画面にアクセスする。

掲示板管理

[管理メニューに戻る](#)

掲示板作成

| | | |
|---------------------------------|--------------------|--------------------|
| 2024年7月18日本日 | 編集 | 削除 |
| 2024年7月18日その② | 編集 | 削除 |
| 2024年7月18日その3 | 編集 | 削除 |
| 2024年7月22日 | 編集 | 削除 |
| 2024年9月1日 ※編集テスト | 編集 | 削除 |
| 2024年9月25日の掲示板 | 編集 | 削除 |
| 2024年9月25日 ※ドロップダウンリストで削除しない | 編集 | 削除 |
| 2024年9月25日 ※ドロップダウンリストで削除するを選択 | 編集 | 削除 |
| 2024年9月25日 ※作成画面から掲示板IDフィールドを削除 | 編集 | 削除 |
| 2024年9月26日 ※削除フラグ：削除しない | 編集 | 削除 |
| 2024年9月26日 ※削除フラグ：削除する | 編集 | 削除 |
| testtitle | 編集 | 削除 |
| 営業部掲示板 | 編集 | 削除 |

- <② データベース接続エラーの処理>
- 管理メニュー画面にログインする
 - 管理メニューにて管理者権限にて、以下のコマンドを実効し、MySQLサーバーのサービスを停止させる
コマンド:net stop mysql80
※以下、操作履歴
C:\WINDOWS\system32>net stop mysql80
MySQL80 サービスを停止中です..
MySQL80 サービスは正常に停止されました。
 - 上記操作実行後、改めて「ユーザー管理画面へ」リンクをクリックする。以下の画面へ遷移することを確認する。
 - 「掲示板作成」ボタンをクリックする。

掲示板管理

[管理メニューに戻る](#)

掲示板作成

| | | |
|---------------------------------|--------------------|--------------------|
| 2024年7月18日本日 | 編集 | 削除 |
| 2024年7月18日その② | 編集 | 削除 |
| 2024年7月18日その3 | 編集 | 削除 |
| 2024年7月22日 | 編集 | 削除 |
| 2024年9月1日 ※編集テスト | 編集 | 削除 |
| 2024年9月25日の掲示板 | 編集 | 削除 |
| 2024年9月25日 ※ドロップダウンリストで削除しない | 編集 | 削除 |
| 2024年9月25日 ※ドロップダウンリストで削除するを選択 | 編集 | 削除 |
| 2024年9月25日 ※作成画面から掲示板IDフィールドを削除 | 編集 | 削除 |
| 2024年9月26日 ※削除フラグ：削除しない | 編集 | 削除 |
| 2024年9月26日 ※削除フラグ：削除する | 編集 | 削除 |
| testtitle | 編集 | 削除 |
| 営業部掲示板 | 編集 | 削除 |

掲示板作成

キャンセル

掲示板タイトル:

掲示板本文:

掲示板削除フラグ:

掲示板削除日:

作成

以上

■単体テストNo9.2

MoveCreateBulletinboardActionの、異常系処理

■目的

セッション未設定時のエラー処理が正しく動作すること。

■テスト実施内容

- 管理者権限にて、掲示板管理画面に遷移する
- 掲示板管理画面にてセッションを削除する
掲示板管理画面にて右クリックし、「検証」をクリックする
アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、「掲示板作成」ボタンをクリックする
- 「掲示板作成」ボタンをクリック後、以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

以上

■単体テストNo10.1

掲示板テーブルへのデータ登録処理

■目的

- ・掲示板作成画面の正常な入力データが、bulletinboardテーブルに登録されることを確認する。
- ・上記処理後、「SUCCESS」を返すことを確認する。

■テスト対象行

```
97行目 :if (sessionUser == null || sessionUser.getUser_id() <= 0) {
111行目 :PreparedStatement ps = connection.prepareStatement(sql)) {
155行目 :ps.setString(1, bulletinboard.getBulletinboard_title());
158行目 :ps.setString(2, bulletinboard.getBulletinboard_content());
161行目 :ps.setInt(3, bulletinboard.getBulletinboard_delete_flag());
164行目 :ps.setString(4, bulletinboard.getBulletinboard_delete_day());
167行目 :ps.setInt(5, bulletinboard.getUser_id());
173行目 :return SUCCESS;
```

■テスト実施内容

- 97行目 :if (sessionUser == null || sessionUser.getUser_id() <= 0) {
- ・目的:セッションが正しく設定されているかを確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jp9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

- ・テスト後(ステップオーバー後)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値:

sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jp9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」

上記の通り、ステップオーバー後も情報が保持されており、意図した内容。

2. this → session の内容:

- sessionの値が SessionMap`オブジェクトである。
- session内にキー`"loggedInUser"`が存在する(`session.containsKey("loggedInUser") == true`)。
- session.get("loggedInUser")の値が`sessionUser`オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

```
auth_type: 1
delete_day: "2999-12-31 00:00:00"
delete_flag: 0
password: "jP9T-LH2"
user_id: 30
user_name: "tesutuser1"
```

上記の通り、クラスフィールドにsessionUser`オブジェクトと同じ内容が補完されている。正常動作。

3. ログに以下の内容が出力されている:

```
[2025-01-05 13:25:42.015] INFO  http-nio-8080-exec-4 com.company.bulletinboard.action.admin.bulletinboard.InsertBulletinboardAction - Session User: tesutuser1
```

- 111行目 :PreparedStatement preparedStatement = connection.prepareStatement(sql)) {

- ・目的:接続の正常性を確認する。

- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認。

- ・sql → INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?,?)
※不正な値(NULLや予期しない構文など)が含まれていない

2.「connection」オブジェクトの確認。接続がクローズされていないことの確認する。

- ・connection → connectionオブジェクト自体が存在し、NULLで無いこと

- ・connection → delegate → openStatementsの値が「CopyOnWriteArrayList<E> (id=251)」

- ※openStatementsの値が空でなくアクセス可能な状態。

- ・isAutoCommit → isAutoCommitの値が「true」であること

3.接続先データベース情報の確認

- ・connection → delegate → origHostToConnectTo → 「"localhost" (id=256)」であること

- ・connection → delegate → origPortToConnectTo → ポート番号が「3306」であること

- ・connection → delegate → database → データベース名が「"bulletinboard_db" (id=241)」であること

- ・connection → delegate → user → 「"root" (id=284) 」であること

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返しており、ログに「データベース接続は正常です。」というメッセージが表示されている。

ログ内容:[2025-01-05 13:54:13.224] DEBUG http-nio-8080-exec-9 com.company.bulletinboard.action.admin.bulletinboard.InsertBulletinboardAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

- ・テスト後(ステップオーバー後)

確認内容:PreparedStatement の生成が正しく行われ、後続処理に影響がないことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

1.「sql」変数の確認

- ・sql → INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?,?)

- ※正しい SQL 文("INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?,?)")を保持していることを確認

2. connection の有効性を確認

- ・preparedStatement → isClosedの値が「false」であることを確認

- ※ステートメントが閉じていない状態

3.「proxyResultSet」の確認

- ・preparedStatement → proxyResultSetの値が「null」であることを確認

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返すことを確認

ログ内容:[2025-01-05 13:54:13.224] DEBUG http-nio-8080-exec-9 com.company.bulletinboard.action.admin.bulletinboard.InsertBulletinboardAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

※※ログメッセージに「データベース接続は正常です。」が出力されていることを確認。接続は有効な状態

- 155行目 :ps.setString(1, bulletinboard.getBulletinboard_title());

・目的:ps の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

・sql → INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?,?)

※正しい SQL 文 ("INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?,?)")を保持していることを確認

2.「ps」の有効性確認

・ps → isClosed の値が「false」

※ステートメントは有効で操作可能な状態

3.フォームデータがpsステートメントに紐づけされていないことを確認

・ps → delegate → query → queryBindings → bairdValues → [0] → valueの値が「null」であることを確認。

※フォームデータがステートメントに紐づけられていない状態

・テスト後(ステップオーバー後)

確認内容:psステートメントにフォームデータが紐づけられたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

・ps → delegate → query → queryBindings → bairdValues → [0] → valueの値が「"2025年1月掲示板"」であることを確認。

※フォームデータがステートメントに紐づけられていることを確認。

以下、デバッグログ

[2025-01-05 14:30:49.848] DEBUG http-nio-8080-exec-10 com.company.bulletinboard.action.admin.bulletinboard.InsertBulletinboardAction - bulletinboard_title: 2025年1月掲示板

●158行目:ps.setString(2, bulletinboard.getBulletinboard_content());

・目的:ps の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

・sql → INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?,?)

※正しい SQL 文 ("INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?,?)")を保持していることを確認

2.「ps」の有効性確認

・ps → isClosed の値が「false」

※ステートメントは有効で操作可能な状態

3.フォームデータがpsステートメントに紐づけされていないことを確認

・ps → delegate → query → queryBindings → bairdValues → [1] → valueの値が「null」であることを確認。

※フォームデータがステートメントに紐づけられていない状態

・テスト後(ステップオーバー後)

確認内容:psステートメントにフォームデータが紐づけられたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

・ps → delegate → query → queryBindings → bairdValues → [1] → valueの値が「"1月掲示板"」であることを確認。

※フォームデータがステートメントに紐づけられていることを確認。

以下、デバッグログ

[2025-01-05 14:45:38.248] DEBUG http-nio-8080-exec-10 com.company.bulletinboard.action.admin.bulletinboard.InsertBulletinboardAction - bulletinboard_content: 1月掲示板

- 161行目 :ps.setInt(3, bulletinboard.getBulletinboard_delete_flag());
- ・目的:ps の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- ・sql → INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?)
- ※正しい SQL 文 ("INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?)")を保持していることを確認

2.「ps」の有効性確認

- ・ps → isClosed の値が「false」
- ※ステートメントは有効で操作可能な状態

3.フォームデータがpsステートメントに紐づけされていないことを確認

- ・ps → delegate → query → queryBindings → bairdValues → [2] → valueの値が「null」であることを確認。
- ※フォームデータがステートメントに紐づけられていない状態

- ・テスト後(ステップオーバー後)

確認内容:psステートメントにフォームデータが紐づけられたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

- ・ps → delegate → query → queryBindings → bairdValues → [2] → valueの値が「0」であることを確認。
- ※フォームデータがステートメントに紐づけられていることを確認。
- 以下、デバッグログ
- [2025-01-05 14:49:35.775] DEBUG http-nio-8080-exec-10 com.company.bulletinboard.action.admin.bulletinboard.InsertBulletinboardAction - bulletinboard_delete_flag: 0

- 164行目 :ps.setString(4, bulletinboard.getBulletinboard_delete_day());
- ・目的:ps の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- ・sql → INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?)
- ※正しい SQL 文 ("INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?)")を保持していることを確認

2.「ps」の有効性確認

- ・ps → isClosed の値が「false」
- ※ステートメントは有効で操作可能な状態

3.フォームデータがpsステートメントに紐づけされていないことを確認

- ・ps → delegate → query → queryBindings → bairdValues → [3] → valueの値が「null」であることを確認。
- ※フォームデータがステートメントに紐づけられていない状態

- ・テスト後(ステップオーバー後)

確認内容:psステートメントにフォームデータが紐づけられたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

- ・ps → delegate → query → queryBindings → bairdValues → [3] → valueの値が「2050-12-31 04:00:00」であることを確認。
- ※フォームデータがステートメントに紐づけられていることを確認。
- 以下、デバッグログ

[2025-01-05 14:51:46.599] DEBUG http-nio-8080-exec-10 com.company.bulletinboard.action.admin.bulletinboard.InsertBulletinboardAction - bulletinboard_delete_day: 2050-12-31 04:00:00

●167行目 :ps.setInt(5, bulletinboard.getUser_id());

・目的:ps の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

・sql → INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?)

※正しい SQL 文 ("INSERT INTO bulletinboard(bulletinboard_title, bulletinboard_content, bulletinboard_delete_flag, bulletinboard_delete_day, user_id) VALUES(?,?,?,?)")を保持していることを確認

2.「ps」の有効性確認

・ps → isClosed の値が「false」

※ステートメントは有効で操作可能な状態

3.フォームデータがpsステートメントに紐づけされていないことを確認

・ps → delegate → query → queryBindings → bainedValues → [4] → valueの値が「null」であることを確認。

※フォームデータがステートメントに紐づけられていない状態

・テスト後(ステップオーバー後)

確認内容:psステートメントにフォームデータが紐づけられたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

・ps → delegate → query → queryBindings → bainedValues → [4] → valueの値が「"30"」であることを確認。

※フォームデータがステートメントに紐づけられていることを確認。

以下、デバッグログ

[2025-01-05 14:53:59.257] DEBUG http-nio-8080-exec-10 com.company.bulletinboard.action.admin.bulletinboard.InsertBulletinboardAction - user_id: 30

●173行目 :return SUCCESS;

※return result;の部分については、アプリケーション全体の統合テストや、インターセプターの動作確認の範囲に含まれる為、範囲外とする。

以上

■単体テストNo10.2

InsertBulletinboardActionクラスのエラー処理

■目的

- ①セッション未設定時のエラー処理が正しく動作すること。
- ② データベース接続エラーの処理が正しく動作すること。
- ③クエリ実行エラーの処理が正しく動作すること。(リスクがあるので中止)

■テスト対象行

<①セッション未設定時のエラー処理>

```
104行目 :if (sessionUser == null) {  
105行目 :String errorMessage = "User session is missing.";  
106行目 :addActionError(errorMessage);  
107行目 :logger.error("Error in MoveBulletinboardManagementAction: " + errorMessage);  
108行目 :return ERROR;
```

<② データベース接続エラーの処理>

```
200行目 :} catch (SQLException e) {  
201行目 :logger.error("データベース接続中にエラーが発生しました", e);  
202行目 :throw e;
```

■テスト実施内容

<①セッション未設定時のエラー処理>

- 管理者権限にて、掲示板作成画面に遷移する
- 掲示板作成画面にてセッションを削除する
管理メニュー画面にて右クリックし、「検証」をクリックする
アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、「作成」ボタンをクリックする
- 「作成」ボタンをクリック後、以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

エラー処理後に以下のログが出力される。

エラーログ:[2025-01-06 12:48:06.665] ERROR http-nio-8080-exec-4 com.company.bulletinboard.action.admin.bulletinboard.InsertBulletinboardAction - Error in InsertBulletinboardAction: User session is missing.

<② データベース接続エラーの処理>

- 掲示板作成画面に遷移する
- 掲示板作成画面のフォームに必須項目を入力する
- コマンドプロンプトにて管理者権限の状態で、以下のコマンドを実効しMySQLサーバーのサービスを停止させる
コマンド:net stop mysql80
※以下、操作履歴
C:\WINDOWS\system32>net stop mysql80
MySQL80 サービスを停止中です..
MySQL80 サービスは正常に停止されました。
- 上記操作実行後、改めて掲示板作成画面にて「作成」ボタンをクリックする。以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

- 以下、エラーログ

[2025-01-06 13:00:20.861] ERROR http-nio-8080-exec-6 com.company.bulletinboard.action.admin.bulletinboard.InsertBulletinboardAction - データベース接続中にエラーが発生しました
at com.company.bulletinboard.action.admin.bulletinboard.InsertBulletinboardAction.mainProc(InsertBulletinboardAction.java:118) [classes/:?]

- テスト実施後、MySQLサーバーのサービスを起動する。

コマンド:net start mysql80
※以下、操作履歴
C:\WINDOWS\system32>net start mysql80
MySQL80 サービスを開始します。
MySQL80 サービスは正常に開始されました。

- エラー画面にてF5キーを押下し、画面のリロード後に掲示板管理画面に遷移できることを確認する。
また、作成した掲示板が一覧に表示されることを確認する。

掲示板管理

[管理メニューに戻る](#)

掲示板作成

| | | |
|--------------|--------------------|--------------------|
| 2024年7月18日本日 | 編集 | 削除 |
| 2024年7月18日 | 編集 | 削除 |

| | | |
|---------------------------------|--------------------|--------------------|
| 2024年7月18日その② | 編集 | 削除 |
| 2024年7月18日その3 | 編集 | 削除 |
| 2024年7月22日 | 編集 | 削除 |
| 2024年9月1日 ※編集テスト | 編集 | 削除 |
| 2024年9月25日の掲示板 | 編集 | 削除 |
| 2024年9月25日 ※ドロップダウンリストで削除しない | 編集 | 削除 |
| 2024年9月25日 ※ドロップダウンリストで削除するを選択 | 編集 | 削除 |
| 2024年9月25日 ※作成画面から掲示板IDフィールドを削除 | 編集 | 削除 |
| 2024年9月26日 ※削除フラグ：削除しない | 編集 | 削除 |
| 2024年9月26日 ※削除フラグ：削除する | 編集 | 削除 |
| testtitle | 編集 | 削除 |
| 営業部掲示板 | 編集 | 削除 |
| 2025年1月掲示板 | 編集 | 削除 |
| 2025年1月 本日二回目 | 編集 | 削除 |
| 2025年1月 三回目 | 編集 | 削除 |
| 2025年1月6日掲示板 | 編集 | 削除 |

以上

■単体テストNo11.1

掲示板編集画面の表示処理

■目的

- ・掲示板編集画面の入力フォームに、編集対象の既存データが表示できることを確認する。
- ・上記処理後、「SUCCESS」を返すことを確認する。

■テスト対象行

```
61行目 :if (sessionUser == null) {
144行目 :PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
159行目 :bulletinboard.setBulletinboard_id(resultSet.getInt("bulletinboard_id"));
162行目 :bulletinboard.setBulletinboard_title(resultSet.getString("bulletinboard_title"));
165行目 :bulletinboard.setBulletinboard_content(resultSet.getString("bulletinboard_content"));
168行目 :bulletinboard.setBulletinboard_delete_flag(resultSet.getInt("bulletinboard_delete_flag"));
171行目 :bulletinboard.setBulletinboard_delete_day(resultSet.getString("bulletinboard_delete_day"));
174行目 :return SUCCESS;
```

■事前準備

※掲示板テーブルに以下のデータが予め、登録されていることとする。

bulletinboard_id:57

bulletinboard_title:2025年1月6日掲示板

bulletinboard_content:1月6日に作成

user_id:30

bulletinboard_delete_flag:0

bulletinboard_delete_day: 2050-12-31 04:00:00.000000

■テスト実施内容

●61行目 :if (sessionUser == null) {

- ・目的:セッションが正しく設定されているかを確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

sessionUser → auth_typeの値が「1」

sessionUser → delete_dayの値が「2999-12-31 00:00:00」

sessionUser → delete_flagの値が「0」

sessionUser → passwordの値が「JP9T-LH2」

sessionUser → user_idの値が「30」

sessionUser → user_nameの値が「tesutuser1」

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

- ・テスト後(ステップオーバー後)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値:

sessionUser → auth_typeの値が「1」

sessionUser → delete_dayの値が「2999-12-31 00:00:00」

sessionUser → delete_flagの値が「0」

sessionUser → passwordの値が「JP9T-LH2」

sessionUser → user_idの値が「30」

sessionUser → user_nameの値が「tesutuser1」
上記の通り、ステップオーバー後も情報が保持されており、意図した内容。期待通りの結果。
2. this → session の内容:
- sessionの値が`SessionMap`オブジェクトである。
- session内にキー`"loggedInUser"`が存在する(`session.containsKey("loggedInUser") == true`)。
- session.get("loggedInUser")の値が`sessionUser`オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認
auth_type: 1
delete_day: "2999-12-31 00:00:00"
delete_flag: 0
password: "jP9T-LH2"
user_id: 30
user_name: "tesutuser1"
上記の通り、クラスフィールドにsessionUser`オブジェクトと同じ内容が補完されている。正常動作。期待通り。
3. ログに以下の内容が出力されている:
[2025-01-08 12:22:34.163] INFO http-nio-8080-exec-9 com.company.bulletinboard.action.admin.bulletinboard.EditBulletinboardAction - Session User: tesutuser1

●144行目:PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
・目的:接続の正常性を確認する。
・テスト前(ステップオーバー前)
確認内容:
-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。
1.「sql」変数の確認。
・sql → "SELECT * FROM bulletinboard WHERE bulletinboard_id = ?" (id=205)
※不正な値(NULLや予期しない構文など)が含まれていない。SQL文が事前に妥当な構文として検証されていること(エラーが発生しないこと)
2.「connection」オブジェクトの確認。接続がクローズされていないことの確認する。
・connection → connectionオブジェクト自体が存在し、NULLで無いこと
・connection → delegate → openStatementsの値が「CopyOnWriteArrayList<E> (id=241)」
※openStatementsの値が空でなくアクセス可能な状態。
・isAutoCommit → isAutoCommitの値が「true」であること
3.接続先データベース情報の確認
・connection → delegate → origHostToConnectTo → 「"localhost" (id=246)」であること
・connection → delegate → origPortToConnectTo → ポート番号が「3306」であること
・connection → delegate → database → データベース名が「"bulletinboard_db" (id=231)」であること
・connection → delegate → user → 「"root" (id=276) 」であること
4.接続有効性のログ
connection.isValid(timeout)メソッド が「true」を返しており、ログに「データベース接続は正常です。」というメッセージが表示されている。
ログ内容:[2025-01-08 12:28:15.035] DEBUG http-nio-8080-exec-9 com.company.bulletinboard.action.admin.bulletinboard.EditBulletinboardAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

・テスト後(ステップオーバー後)
確認内容:PreparedStatement の生成が正しく行われ、後続処理に影響がないことを確認する
- Eclipse「変数」ウィンドウで以下を確認。
1.「sql」変数の確認
・sql → SELECT * FROM bulletinboard WHERE bulletinboard_id = ?
※正しい SQL 文("ISELECT * FROM bulletinboard WHERE bulletinboard_id = ?")を保持していることを確認
2. connection の有効性を確認
・preparedStatement → isClosedの値が「false」であることを確認
※ステートメントが閉じていない状態
3.「proxyResultSet」の確認

・preparedStatement → proxyResultSetの値が「null」であることを確認

4. 接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返すことを確認

ログ内容:[2025-01-08 12:28:15.035] DEBUG http-nio-8080-exec-9 com.company.bulletinboard.action.admin.bulletinboard.EditBulletinboardAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

※※ログメッセージに「データベース接続は正常です。」が出力されていることを確認。接続は有効な状態

●159行目 : bulletinboard.setBulletinboard_id(resultSet.getInt("bulletinboard_id"));

・目的 : preparedStatement の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

期待値 : sqlの値が「SELECT * FROM bulletinboard WHERE bulletinboard_id = ?」であること。
preparedStatementのisCloseの値が「false」であること。
bulletinboard_idの値が「null」であること

確認内容 :

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1. 「sql」変数の確認

・sql → SELECT * FROM bulletinboard WHERE bulletinboard_id = ?

※正しい SQL 文 (SELECT * FROM bulletinboard WHERE bulletinboard_id = ?) を保持していることを確認。期待通り。

2. 「preparedStatement」の有効性確認

・preparedStatement → isClosed の値が「false」

※ステートメントは有効で操作可能な状態。期待通り。

3. 既存データがモデルのbulletinboardに紐づけされていないことを確認

・this → bulletinboard → bulletinboard_idの値が「null」であることを確認。

※既存データがモデルに紐づけられていない状態。期待通り。

・テスト後(ステップオーバー後)

期待値 : bulletinboard_idの値が「57」であること

確認内容 : 既存データがモデルのbulletinboardに紐づけされたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

・this → bulletinboard → bulletinboard_idの値が「57」であることを確認。

※既存データがモデルに紐づけられたことを確認。期待通りの結果。

以下、デバッグログ

[2025-01-08 13:34:11.002] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.EditBulletinboardAction - bulletinboard_id = 57

●162行目 : bulletinboard.setBulletinboard_title(resultSet.getString("bulletinboard_title"));

・目的 : preparedStatement の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

期待値 : sqlの値が「SELECT * FROM bulletinboard WHERE bulletinboard_id = ?」であること。
preparedStatement の isClosed の値が「false」であること。
bulletinboard_titleの値が「null」であること。

確認内容 :

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1. 「sql」変数の確認

・sql → SELECT * FROM bulletinboard WHERE bulletinboard_id = ?

※正しい SQL 文 ("SELECT * FROM bulletinboard WHERE bulletinboard_id = ?") を保持していることを確認。期待通り。

2.「preparedStatement」の有効性確認

- ・preparedStatement → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待通り。

3.既存データがモデルのbulletinboardに紐づけされていないことを確認。

- ・this → bulletinboard → bulletinboard_titleの値が「null」であることを確認。期待通り。
※既存データがモデルに紐づけられていない状態

- ・テスト後（ステップオーバー後）

確認内容: 既存データがモデルのbulletinboardに紐づけされたことを確認する

期待値: bulletinboard_titleの値が「2025年1月6日掲示板」であること

- Eclipse「変数」ウィンドウで以下を確認。

- ・this → bulletinboard → bulletinboard_titleの値が「2025年1月6日掲示板」であることを確認。
※既存データがモデルに紐づけられたことを確認。期待通り。
以下、デバッグログ
[2025-01-08 13:39:26.327] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.EditBulletinboardAction - bulletinboard_title: = 2025年1月6日掲示板

- 165行目 : bulletinboard.setBulletinboard_content(resultSet.getString("bulletinboard_content"));

- ・目的: preparedStatement の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

- ・テスト前（ステップオーバー前）

- ・期待値: sql の値が「SELECT * FROM bulletinboard WHERE bulletinboard_id = ?」であること。
preparedStatement の isClosed の値が「false」であること。
bulletinboard_contentの値が「null」であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- ・sql → SELECT * FROM bulletinboard WHERE bulletinboard_id = ?
※正しい SQL 文 ("SELECT * FROM bulletinboard WHERE bulletinboard_id = ?") を保持していることを確認。期待通りの結果。

2.「preparedStatement」の有効性確認

- ・preparedStatement → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待通りの結果。

3.既存データがモデルのbulletinboardに紐づけされていないことを確認

- ・this → bulletinboard → bulletinboard_contentの値が「null」であることを確認。
※既存データがモデルに紐づけられていない状態。期待値通り。

- ・テスト後（ステップオーバー後）

確認内容: 既存データがモデルのbulletinboardに紐づけされたことを確認する

期待値: bulletinboard_contentの値が「1月6日に作成」であること。

- ・this → bulletinboard → bulletinboard_contentの値が「1月6日に作成」であることを確認。
※既存データがモデルに紐づけられたことを確認。
※フォームデータがステートメントに紐づけられていることを確認。期待通り。
以下、デバッグログ
[2025-01-08 13:43:02.505] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.EditBulletinboardAction - bulletinboard_content: = 1月6日に作成

- 168行目 : bulletinboard.setBulletinboard_delete_flag(resultSet.getInt("bulletinboard_delete_flag"));

- ・目的: preparedStatement の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

- ・テスト前(ステップオーバー前)
- ・期待値: sql の値が、「SELECT * FROM bulletinboard WHERE bulletinboard_id = ?」であること。
 preparedStatementの値がisClosed の値が「false」であること。
 bulletinboard_delete_flagの値が「0」であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- ・sql → SELECT * FROM bulletinboard WHERE bulletinboard_id = ?
 ※正しい SQL 文("SELECT * FROM bulletinboard WHERE bulletinboard_id = ?")を保持していることを確認。期待通りの結果。

2.「preparedStatement」の有効性確認

- ・preparedStatement → isClosed の値が「false」
 ※ステートメントは有効で操作可能な状態。期待値通り。

3.既存データがモデルのbulletinboardに紐づけされていないことを確認

- ・this → bulletinboard → bulletinboard_delete_flagの値が「0」であることを確認。
 ※既存データがモデルに紐づけられていない状態。期待通り。

・テスト後(ステップオーバー後)

- ・期待値: bulletinboard_delete_flagの値が「0」であること。

確認内容: 既存データがモデルのbulletinboardに紐づけされたことを確認する

- ・this → bulletinboard → bulletinboard_delete_flagの値が「0」であることを確認。
 ※既存データがモデルに紐づけられたことを確認。
 ※フォームデータがステートメントに紐づけられていることを確認。

 以下、デバッグログ
 [2025-01-08 13:49:09.740] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.EditBulletinboardAction - bulletinboard_delete_flag: = 0
 ※期待通りの結果。

- 171行目 :bulletinboard.setBulletinboard_delete_day(resultSet.getString("bulletinboard_delete_day"));
- ・目的:preparedStatement の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。
- ・期待値: sql の値が「SELECT * FROM bulletinboard WHERE bulletinboard_id = ?」であること。
 preparedStatementのisClosed の値が「false」であること。
 bulletinboard_delete_dayの値が「null」であること。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- ・sql → SELECT * FROM bulletinboard WHERE bulletinboard_id = ?
 ※正しい SQL 文("SELECT * FROM bulletinboard WHERE bulletinboard_id = ?")を保持していることを確認。期待通りの結果。

2.「preparedStatement」の有効性確認

- ・preparedStatement → isClosed の値が「false」
 ※ステートメントは有効で操作可能な状態。期待値通り。

3.既存データがモデルのbulletinboardに紐づけされていないことを確認

- ・this → bulletinboard → bulletinboard_delete_dayの値が「null」であることを確認。
 ※既存データがモデルに紐づけられていない状態。期待値通り。

・テスト後(ステップオーバー後)

- 期待値: bulletinboard_delete_dayの値が「2050-12-31 04:00:00」であること。

確認内容: 既存データがモデルのbulletinboardに紐づけされたことを確認する

・this → bulletinboard →bulletinboard_delete_dayの値が「2050-12-31 04:00:00」であることを確認。

※既存データがモデルに紐づけられたことを確認。

※フォームデータがステートメントに紐づけられていることを確認。

以下、デバッグログ

[2025-01-08 13:53:30.238] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.admin.bulletinboard.EditBulletinboardAction - bulletinboard_delete_day: = 2050-12-31 04:00:00

※上記は期待通りの結果。

●174行目 :return SUCCESS;

※return SUCCESS;の部分については、アプリケーション全体の統合テストや、インターセプターの動作確認の範囲に含まれる為、範囲外とする。

以上

■単体テストNo11.2

EditBulletinboardAction - セッション未設定時およびDB接続エラーのハンドリング検証

■目的

- ① セッション未設定時に適切なエラー画面が表示され、ログが記録されること。
- ② データベース接続エラー時に適切なエラー画面が表示され、ログが記録されること。

■テスト対象行

＜①セッション未設定時のエラー処理＞

71行目 :if (sessionUser == null) {

＜② データベース接続エラーの処理＞

200行目 :} catch (SQLException e) {

201行目 :logger.error("データベース接続中にエラーが発生しました", e);

202行目 :throw e;

■期待する結果

・セッション未設定時

- 1.エラー画面が表示されること(HTML要素: <h1> に「エラーが発生しました」、<a> に「トップページへ」が含まれる)。
- 2.ログに「User session is missing.」が記録されること。

・データベース接続エラー時

- 1.エラー画面が表示されること(HTML要素: 同上)。
- 2.ログに「データベース接続中にエラーが発生しました」が記録されること。

■テスト実施内容

＜①セッション未設定時のエラー処理＞

- 管理者権限にて、掲示板管理画面に遷移する
- 掲示板管理画面にてセッションを削除する
管理メニュー画面にて右クリックし、「検証」をクリックする
アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、該当掲示板の「編集」リンクをクリックする
- 「編集」ボタンをクリック後、以下のエラー画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

エラー処理後に以下のログが出力される。

エラーログ :[2025-01-12 11:39:59.847] ERROR http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.EditBulletinboardAction - Error in MoveBulletinboardManagementAction: User session is missing.

＜② データベース接続エラーの処理＞

- 掲示板管理画面に遷移する
- コマンドプロンプトにて管理者権限の状態で、以下のコマンドを実効しMySQLサーバーのサービスを停止させる
コマンド:net stop mysql80
※以下、操作履歴
C:\WINDOWS\system32>net stop mysql80
MySQL80 サービスを停止中です..
MySQL80 サービスは正常に停止されました。
- 上記操作実行後、改めて掲示板管理画面にて該当掲示板の「編集」リンクをクリックする。以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

- 以下、エラーログ
[2025-01-12 11:48:31.195] ERROR http-nio-8080-exec-10 com.company.bulletinboard.action.admin.bulletinboard.EditBulletinboardAction - データベース接続中にエラーが発生しました
at com.company.bulletinboard.action.admin.bulletinboard.EditBulletinboardAction.getBulletinboardById(EditBulletinboardAction.java:104) [classes/:?]
at com.company.bulletinboard.action.admin.bulletinboard.EditBulletinboardAction.mainProc

- テスト実施後、MySQLサーバーのサービスを起動する。
コマンド:net start mysql80
※以下、操作履歴
C:\WINDOWS\system32>net start mysql80
MySQL80 サービスを開始します。
MySQL80 サービスは正常に開始されました。
- エラー画面にてF5キーを押下し、画面のリロード後に掲示板編集画面に遷移できることを確認する。
また、編集フォームに該当掲示板のデータがフォームに表示されることを確認する。

掲示板編集

キャンセル

掲示板タイトル: 2025年1月6日掲示板

掲示板本文: 1月6日に作成

掲示板削除フラグ: 削除しない ▼

掲示板削除日: 2050-12-31 04:00:00

更新

以上

■単体テストNo12.1

掲示板編集画面の更新処理

■目的

- ・掲示板編集画面にて既存データを編集後、正常に更新処理が実行できることを確認する。
- ・上記処理後、「SUCCESS」を返すことを確認する。

■テスト対象行

```
122行目 :if (sessionUser == null) {
178行目 :PreparedStatement ps = connection.prepareStatement(sql);
181行目 :ps.setString(1, bulletinboard.getBulletinboard_title());
184行目 :ps.setString(2, bulletinboard.getBulletinboard_content());
187行目 :ps.setInt(3, bulletinboard.getBulletinboard_delete_flag());
190行目 :ps.setString(4, bulletinboard.getBulletinboard_delete_day());
193行目 :ps.setInt(5, bulletinboard.getBulletinboard_id());
212行目 :return SUCCESS;
```

■事前準備

※掲示板テーブルに以下のデータが予め、登録されていることとする。

bulletinboard_id:57

bulletinboard_title:2025年1月6日掲示板

bulletinboard_content:1月6日に作成

user_id:30

bulletinboard_delete_flag:0

bulletinboard_delete_day: 2050-12-31 04:00:00.000000

■更新処理の内容

<変更前>

掲示板タイトル: 2025年1月6日掲示板

掲示板本文: 1月6日に作成

掲示板削除フラグ: 削除しない

掲示板削除日: 2050-12-31 04:00:00

<変更後>

掲示板タイトル: 2025年1月15日掲示板

掲示板本文: 1月15日に変更

掲示板削除フラグ: 削除する

掲示板削除日: 2099-12-31 04:00:00

■テスト実施内容

- 122行目 :if (sessionUser == null) {

- ・目的: セッションが正しく設定されているかを確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

sessionUser → auth_typeの値が「1」

sessionUser → delete_dayの値が「2999-12-31 00:00:00」

sessionUser → delete_flagの値が「0」

sessionUser → passwordの値が「jP9T-LH2」

sessionUser → user_idの値が「30」

sessionUser → user_nameの値が「tesutuser1」

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

- ・テスト後(ステップオーバー後)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値:

sessionUser → auth_typeの値が「1」

sessionUser → delete_dayの値が「2999-12-31 00:00:00」

sessionUser → delete_flagの値が「0」

sessionUser → passwordの値が「jP9T-LH2」

sessionUser → user_idの値が「30」

sessionUser → user_nameの値が「tesutuser1」

上記の通り、ステップオーバー後も情報が保持されており、意図した内容。期待通りの結果。

2. this → session の内容:

- sessionの値が`SessionMap`オブジェクトである。

- session内にキー`"loggedInUser"`が存在する(`session.containsKey("loggedInUser") == true`)。

- session.get("loggedInUser")の値が`sessionUser`オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

auth_type: 1

delete_day: "2999-12-31 00:00:00"

delete_flag: 0

password: "jP9T-LH2"

user_id: 30

user_name: "tesutuser1"

上記の通り、クラスフィールドにsessionUser`オブジェクトと同じ内容が補完されている。正常動作。期待通り。

3. ログに以下の内容が出力されている:

[2025-01-15 10:53:08.221] INFO http-nio-8080-exec-2 com.company.bulletinboard.action.admin.bulletinboard.UpdateBulletinboardAction - Session User: tesutuser1

●178行目 :PreparedStatement ps = connection.prepareStatement(sql);

・目的: 接続の正常性を確認する。

・テスト前 (ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認。

・sql → "UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?"

※不正な値(NULLや予期しない構文など)が含まれていない。SQL文が事前に妥当な構文として検証されていること(エラーが発生しないこと)」

2.「connection」オブジェクトの確認。接続がクローズされていないことの確認する。

・connection → connectionオブジェクト自体が存在し、NULLで無いこと

・connection → delegate → openStatementsの値が「CopyOnWriteArrayList<E> (id=241)」

※openStatementsの値が空でなくアクセス可能な状態。

・isAutoCommit → isAutoCommitの値が「true」であること

3.接続先データベース情報の確認

・connection → delegate → origHostToConnectTo → 「"localhost" (id=290)」であること

・connection → delegate → origPortToConnectTo → ポート番号が「3306」であること

・connection → delegate → database → データベース名が「"bulletinboard_db" (id=241)」であること

・connection → delegate → user → 「"root" (id=320) 」であること

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返しており、ログに「データベース接続は正常です。」というメッセージが表示されている。

ログ内容 :[2025-01-15 10:57:17.089] DEBUG http-nio-8080-exec-2 com.company.bulletinboard.action.admin.bulletinboard.UpdateBulletinboardAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

・テスト後 (ステップオーバー後)

確認内容 :ps の生成が正しく行われ、後続処理に影響がないことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

1.「sql」変数の確認

・sql → UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?

※正しい SQL 文("UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?")を保持していることを確認

2. connection の有効性を確認

- ps → isClosedの値が「false」であることを確認
- ※ステートメントが閉じていない状態

3.「proxyResultSet」の確認

- preparedStatement → proxyResultSetの値が「null」であることを確認

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返すことを確認

ログ内容:[2025-01-15 10:57:17.089] DEBUG http-nio-8080-exec-2 com.company.bulletinboard.action.admin.bulletinboard.UpdateBulletinboardAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

※※ログメッセージに「データベース接続は正常です。」が出力されていることを確認。接続は有効な状態

- 181行目 :ps.setString(1, bulletinboard.getBulletinboard_title());
 - ・目的: プリペAREDステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。
 - ・テスト前 (ステップオーバー前)
- 期待値:
- sqlの値が「UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?」であること。
 - ps の isClosed の値が「false」であること。
 - bulletinboard_titleの値が「2025年1月15日掲示板」であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- sql → UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?
- ※正しい SQL 文("UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?")を保持していることを確認。期待通り。

2.「ps」の有効性確認

- ps → isClosed の値が「false」
- ※ステートメントは有効で操作可能な状態。期待通り。

3.フォームの入力値が、モデルのbulletinboardに紐づけされていることを確認。

- this → bulletinboard →bulletinboard_titleの値が「2025年1月15日掲示板」であることを確認。期待通り。
- ※上記の通り、フォームの入力値がモデルに紐づけられている状態。

・テスト後 (ステップオーバー後)

確認内容:ステップオーバー後もbulletinboardに紐づけされた値に、変化が無ことを確認する

期待値:bulletinboard_titleの値が「2025年1月15日掲示板」であること(フォームの入力値)

- Eclipse「変数」ウィンドウで以下を確認。

- this → bulletinboard →bulletinboard_titleの値が「2025年1月15日掲示板」であることを確認。
- ※ステップオーバー前と変わりが無いことを確認。期待通り。

以下、デバッグログ

[2025-01-15 11:19:08.302] DEBUG http-nio-8080-exec-2 com.company.bulletinboard.action.admin.bulletinboard.UpdateBulletinboardAction - bulletinboard_title: = 2025年1月15日掲示板

- 184行目 :ps.setString(2, bulletinboard.getBulletinboard_content());
 - ・目的: プリペAREDステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。
 - ・テスト前 (ステップオーバー前)
- ・期待値:
- sql の値が「UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?」であること。
 - ps の isClosed の値が「false」であること。
 - bulletinboard_contentの値が「1月15日に変更」であること。
- 確認内容:
- Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。
- 1.「sql」変数の確認
- sql → UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?
 - ※正しい SQL 文("UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?")を保持していることを確認。期待通りの結果。

2.「ps」の有効性確認

- ・ps → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待通りの結果。

3.フォームの入力値がモデルのbulletinboardに紐づけされていることを確認

- ・this → bulletinboard →bulletinboard_contentの値が「1月15日に変更」であることを確認。
※入力値がモデルに紐づけられている状態。期待値通り。

- ・テスト後（ステップオーバー後）

確認内容: フォームの入力値がモデルのbulletinboardに紐づけされていることを確認する

期待値: bulletinboard_contentの値が「1月15日に変更」であること。

- ・this → bulletinboard →bulletinboard_contentの値が「1月15日に変更」であることを確認。
※フォームデータがモデルに紐づけられていることを確認。期待通り。
以下、デバッグログ

[2025-01-15 11:33:25.628] DEBUG http-nio-8080-exec-2 com.company.bulletinboard.action.admin.bulletinboard.UpdateBulletinboardAction - bulletinboard_content: = 1月15日に変更

- 187行目 :ps.setInt(3, bulletinboard.getBulletinboard_delete_flag());

- ・目的: プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

- ・テスト前（ステップオーバー前）

- ・期待値: sql の値が、「UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?」であること。
psの値がisClosed の値が「false」であること。
bulletinboard_delete_flagの値が「0」であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- ・sql → UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?
※正しいSQL 文(" UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?")を保持していることを確認。期待通りの結果。

2.「ps」の有効性確認

- ・ps → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待値通り。

3.フォームデータがモデルのbulletinboardに紐づけされていないことを確認

- ・this → bulletinboard →bulletinboard_delete_flagの値が「0」であることを確認。
※フォームデータがモデルに紐づけられていない状態。期待通り。

- ・テスト後（ステップオーバー後）

- ・期待値: bulletinboard_delete_flagの値が「1」であること。

確認内容: セレクトボタンのデータがモデルのbulletinboardに紐づけされたことを確認する

- ・this → bulletinboard →bulletinboard_delete_flagの値が「1」であることを確認。
※セレクトボタンのデータがステートメントに紐づけられていることを確認。
以下、デバッグログ

[2025-01-15 11:37:17.837] DEBUG http-nio-8080-exec-2 com.company.bulletinboard.action.admin.bulletinboard.UpdateBulletinboardAction - bulletinboard_delete_flag: = 1

※期待通りの結果。

- 190行目 :ps.setString(4, bulletinboard.getBulletinboard_delete_day());

- ・目的: プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

- ・期待値: sql の値が「UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?」であること。
psのisClosed の値が「false」であること。
bulletinboard_delete_dayの値が「2099-12-31 04:00:00」であること。

- ・テスト前（ステップオーバー前）

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

•sql → UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?
※正しいSQL文("UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?")を保持していることを確認。期待通りの結果。

2.「ps」の有効性確認

•ps → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待値通り。

3.フォームデータがモデルのbulletinboardに紐づけされていないことを確認

•this → bulletinboard →bulletinboard_delete_dayの値が「2099-12-31 04:00:00」であることを確認。
※フォームデータがモデルに紐づけられていない状態。期待値通り。

•テスト後(ステップオーバー後)

期待値: bulletinboard_delete_dayの値が「2099-12-31 04:00:00」であること。

確認内容: 入力データがモデルのbulletinboardに紐づけされたことを確認する

•this → bulletinboard →bulletinboard_delete_dayの値が「2099-12-31 04:00:00」であることを確認。
※フォームデータがステートメントに紐づけられていることを確認。
以下、デバッグログ
[2025-01-15 11:42:41.877] DEBUG http-nio-8080-exec-2 com.company.bulletinboard.action.admin.bulletinboard.UpdateBulletinboardAction - bulletinboard_delete_day: = 2099-12-31 04:00:00
※上記は期待通りの結果。

●193行目: bulletinboard.setBulletinboard_id(resultSet.getInt("bulletinboard_id"));

•目的: プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

•テスト前(ステップオーバー前)

期待値: sqlの値が「UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?」であること。
psのisCloseの値が「false」であること。
bulletinboard_idの値が「57」であること

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

•sql → UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?
※正しいSQL文(UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?)を保持していることを確認。期待通り。

2.「ps」の有効性確認

•ps → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待通り。

3.該当掲示板のIDがモデルのbulletinboardに紐づけられていることを確認

•this → bulletinboard →bulletinboard_idの値が「57」であることを確認。
※掲示板IDがモデルに紐づけられている状態。期待通り。

•テスト後(ステップオーバー後)

期待値: bulletinboard_idの値が「57」であること

確認内容: 該当掲示板の掲示欄IDがbulletinboardに紐づけされたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

•this → bulletinboard →bulletinboard_idの値が「57」であることを確認。
※既存データがモデルに紐づけられたことを確認。期待通りの結果。
以下、デバッグログ
[2025-01-15 11:51:06.186] DEBUG http-nio-8080-exec-2 com.company.bulletinboard.action.admin.bulletinboard.UpdateBulletinboardAction - bulletinboard_id: = 57

●213行目 :return SUCCESS;
※return SUCCESS;の部分については、アプリケーション全体の統合テストや、インターセプターの動作確認の範囲に含まれる為、範囲外とする。

以上

■単体テストNo12.2

UpdateBulletinboardAction - セッション未設定時およびDB接続エラーのハンドリング検証

■目的

- ① セッション未設定時に適切なエラー画面が表示され、ログが記録されること。
- ② データベース接続エラー時に適切なエラー画面が表示され、ログが記録されること。

■テスト対象行

＜①セッション未設定時のエラー処理＞

121行目 :if (sessionUser == null) {

＜② データベース接続エラーの処理＞

228行目 :if (sqlState != null && sqlState.startsWith("08")) {

229行目 :logger.error("データベース接続中にエラーが発生しました", e);

230行目 :addActionError("データベース接続中にエラーが発生しました。管理者にお問い合わせください");

■期待する結果

・セッション未設定時

- 1.エラー画面が表示されること(HTML要素: <h1> に「エラーが発生しました」、<a> に「トップページへ」が含まれる)。
- 2.ログに「User session is missing.」が記録されること。

・データベース接続エラー時

- 1.エラー画面が表示されること(HTML要素: 同上)。
- 2.ログに「データベース接続中にエラーが発生しました」が記録されること。

■テスト実施内容

＜①セッション未設定時のエラー処理＞

- 管理者権限にて、掲示板編集画面に遷移する
- 掲示板編集画面にて、入力フォームに編集内容を入力後、セッションを削除する
管理メニュー画面にて右クリックし、「検証」をクリックする
アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、該当掲示板の「編集」リンクをクリックする
- 「編集」ボタンをクリック後、以下のエラー画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

エラー処理後に以下のログが出力される。

エラーログ :[2025-01-17 16:02:46.344] ERROR http-nio-8080-exec-9 com.company.bulletinboard.action.admin.bulletinboard.UpdateBulletinboardAction - Error in MoveBulletinboardManagementAction: User session is missing.

＜② データベース接続エラーの処理＞

- 掲示板管理画面に遷移する

- コマンドプロンプトにて管理者権限の状態で、以下のコマンドを実効しMySQLサーバーのサービスを停止させる
コマンド:net stop mysql80
※以下、操作履歴
C:\WINDOWS\system32>net stop mysql80
MySQL80 サービスを停止中です..
MySQL80 サービスは正常に停止されました。
- 上記操作実行後、改めて掲示板編集画面にて「更新」ボタンをクリックする。以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

- 以下、エラーログ
[2025-01-17 16:06:12.935] ERROR http-nio-8080-exec-8 com.company.bulletinboard.action.admin.bulletinboard.UpdateBulletinboardAction - データベース接続中にエラーが発生しました
at com.company.bulletinboard.action.admin.bulletinboard.UpdateBulletinboardAction.mainProc(UpdateBulletinboardAction.java:139) [classes/:?]

- テスト実施後、MySQLサーバーのサービスを起動する。
コマンド:net start mysql80
※以下、操作履歴
C:\WINDOWS\system32>net start mysql80
MySQL80 サービスを開始します。
MySQL80 サービスは正常に開始されました。
- エラー画面にてF5キーを押下し、画面のリロード後に掲示板編集画面に遷移できることを確認する。
また、編集フォームに該当掲示板のデータがフォームに表示されることを確認する。

掲示板編集

キャンセル

掲示板タイトル: 2025年1月16日掲示板

掲示板本文: 1月16日に変更

掲示板削除フラグ: 削除しない ▼

掲示板削除日: 2100-12-31 04:00:00

更新

以上

■単体テストNo13.1

掲示板削除処理

■目的

- ・掲示板管理画面で、掲示板が削除できることを確認する。
- ・上記処理後、「SUCCESS」を返すことを確認する。

■テスト対象行

```
49行目 :if (sessionUser == null) {  
105行目 :PreparedStatement ps = connection.prepareStatement(sql);  
108行目 :ps.setInt(1, bulletinboard_id);  
118行目 :return SUCCESS;
```

■事前準備

※掲示板テーブルに以下のデータが予め、登録されていることとする。

bulletinboard_id:55

bulletinboard_title:2025年1月 本日二回目

bulletinboard_content:二回目作成

user_id:30

bulletinboard_delete_flag:0

bulletinboard_delete_day: 2050-12-31 04:00:00.000000

■削除処理の内容

<削除内容>

掲示板タイトル: 2025年1月 本日二回目

掲示板本文: 二回目作成

掲示板削除フラグ: 削除しない

掲示板削除日: 2050-12-31 04:00:00

■テスト実施内容

- 49行目 :if (sessionUser == null) {

- ・目的: セッションが正しく設定されているかを確認する。
- ・テスト前 (ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

sessionUser → auth_typeの値が「1」

sessionUser → delete_dayの値が「2999-12-31 00:00:00」

sessionUser → delete_flagの値が「0」

sessionUser → passwordの値が「jP9T-LH2」

sessionUser → user_idの値が「30」

sessionUser → user_nameの値が「tesutuser1」

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

- ・テスト後 (ステップオーバー後)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値:

sessionUser → auth_typeの値が「1」

sessionUser → delete_dayの値が「2999-12-31 00:00:00」

sessionUser → delete_flagの値が「0」

sessionUser → passwordの値が「jP9T-LH2」

sessionUser → user_idの値が「30」

sessionUser → user_nameの値が「tesutuser1」

上記の通り、ステップオーバー後も情報が保持されており、意図した内容。期待通りの結果。

2. this → session の内容:

- sessionの値が`SessionMap`オブジェクトである。
- session内にキー`"loggedInUser"`が存在する(`session.containsKey("loggedInUser") == true`)。
- session.get("loggedInUser")の値が`sessionUser`オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

```
auth_type: 1
delete_day: "2999-12-31 00:00:00"
delete_flag: 0
password: "jP9T-LH2"
user_id: 30
user_name: "tesutuser1"
```

上記の通り、クラスフィールドにsessionUser`オブジェクトと同じ内容が補完されている。正常動作。期待通り。

3. ログに以下の内容が出力されている:

```
ログ内容:[2025-01-18 19:05:32.181] INFO  http-nio-8080-exec-1 com.company.bulletinboard.action.admin.bulletinboard.DeleteBulletinboardAction - Session User: tesutuser1
[2025-01-18 19:05:32.868] INFO  http-nio-8080-exec-1 com.company.bulletinboard.action.admin.bulletinboard.DeleteBulletinboardAction - Session UserID: 30
[2025-01-18 19:05:33.451] INFO  http-nio-8080-exec-1 com.company.bulletinboard.action.admin.bulletinboard.DeleteBulletinboardAction - Session User_AuthType: 1
```

●178行目 :PreparedStatement ps = connection.prepareStatement(sql);

- ・目的: 接続の正常性を確認する。
- ・テスト前 (ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認。

・sql → "DELETE FROM bulletinboard WHERE bulletinboard_id = ?"

※不正な値(NULLや予期しない構文など)が含まれていない。SQL文が事前に妥当な構文として検証されていること(エラーが発生しないこと)

2.「connection」オブジェクトの確認。接続がクローズされていないことの確認する。

- ・connection → connectionオブジェクト自体が存在し、NULLで無いこと
- ・connection → delegate → openStatementsの値が「CopyOnWriteArrayList<E> (id=242)」
※openStatementsの値が空でなくアクセス可能な状態。
- ・isAutoCommit → isAutoCommitの値が「true」であること

3.接続先データベース情報の確認

- ・connection → delegate → origHostToConnectTo → 「"localhost" (id=265)」であること
- ・connection → delegate → origPortToConnectTo → ポート番号が「3306」であること
- ・connection → delegate → database → データベース名が「"bulletinboard_db" (id=232)」であること
- ・connection → delegate → user → 「"root" (id=299) 」であること

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返しており、ログに「データベース接続は正常です。」というメッセージが表示されている。

ログ内容:[2025-01-18 19:05:45.724] DEBUG http-nio-8080-exec-1 com.company.bulletinboard.action.admin.bulletinboard.DeleteBulletinboardAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

・テスト後 (ステップオーバー後)

確認内容:ps の生成が正しく行われ、後続処理に影響がないことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

1.「sql」変数の確認

・sql → DELETE FROM bulletinboard WHERE bulletinboard_id = ?

※正しいSQL文("DELETE FROM bulletinboard WHERE bulletinboard_id = ?")を保持していることを確認

2. connection の有効性を確認

- ・ps → isClosedの値が「false」であることを確認
※ステートメントが閉じていない状態

3.「proxyResultSet」の確認

- ・preparedStatement → proxyResultSetの値が「null」であることを確認

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返すことを確認

ログ内容:[2025-01-18 19:05:45.724] DEBUG http-nio-8080-exec-1 com.company.bulletinboard.action.admin.bulletinboard.DeleteBulletinboardAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db
※※ログメッセージに「データベース接続は正常です。」が出力されていることを確認。接続は有効な状態

- 181行目:ps.setString(1, bulletinboard.getBulletinboard_title());
- ・目的:プリペAREDステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。
- ・テスト前(ステップオーバー前)
- 期待値:sqlの値が「DELETE FROM bulletinboard WHERE bulletinboard_id = ?」であること。
ps の isClosed の値が「false」であること。
bulletinboard_idの値が「55」であること。

確認内容:
-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。
1.「sql」変数の確認
・sql → DELETE FROM bulletinboard WHERE bulletinboard_id = ?
※正しいSQL 文("DELETE FROM bulletinboard WHERE bulletinboard_id = ?")を保持していることを確認。期待通り。

2.「ps」の有効性確認
・ps → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待通り。

3.削除対象の値が、モデルのbulletinboardに紐づけされいることを確認。
・this → bulletinboard →bulletinboard_idの値が「55」であることを確認。期待通り。
※上記の通り、値がモデルに紐づけられている状態。

・テスト後(ステップオーバー後)
確認内容:ステップオーバー後もbulletinboardに値に、変化が無ことを確認する
期待値:bulletinboard_idの値が「55」であること。
- Eclipse「変数」ウィンドウで以下を確認。
・this → bulletinboard →bulletinboard_idの値が「55」であることを確認。
※ステップオーバー前と変わりが無いことを確認。期待通り。
以下、デバッグログ
[2025-01-18 19:14:14.422] DEBUG http-nio-8080-exec-1 com.company.bulletinboard.action.admin.bulletinboard.DeleteBulletinboardAction - bulletinboard_id: = 55
[2025-01-18 19:14:14.422] INFO http-nio-8080-exec-1 com.company.bulletinboard.action.admin.bulletinboard.DeleteBulletinboardAction - 掲示板の削除処理が成功しました。

●213行目:return SUCCESS;
※return SUCCESS;の部分については、アプリケーション全体の統合テストや、インターセプターの動作確認の範囲に含められる為、範囲外とする。

以上

■単体テストNo13.2

DeleteBulletinboardAction - セッション未設定時およびDB接続エラーのハンドリング検証

■目的

- ① セッション未設定時に適切なエラー画面が表示され、ログが記録されること。
- ② データベース接続エラー時に適切なエラー画面が表示され、ログが記録されること。

■テスト対象行

＜①セッション未設定時のエラー処理＞

55行目 :if (sessionUser == null) {

＜② データベース接続エラーの処理＞

141行目 :if (sqlState != null && sqlState.startsWith("08")) {

142行目 :logger.error("データベース接続中にエラーが発生しました", e);

143行目 :addActionError("データベース接続中にエラーが発生しました。管理者にお問い合わせください");

■期待する結果

・セッション未設定時

- 1.エラー画面が表示されること(HTML要素: <h1> に「エラーが発生しました」、<a> に「トップページへ」が含まれる)。
- 2.ログに「User session is missing.」が記録されること。

・データベース接続エラー時

- 1.エラー画面が表示されること(HTML要素: 同上)。
- 2.ログに「データベース接続中にエラーが発生しました」が記録されること。

■テスト実施内容

＜①セッション未設定時のエラー処理＞

- 管理者権限にて、掲示板管理画面に遷移する
- 掲示板一覧にて、セッションを削除する
 - 管理メニュー画面にて右クリックし、「検証」をクリックする
 - アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、該当掲示板の「削除」リンクをクリックする
- 「削除」リンクをクリック後、以下のエラー画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

エラー処理後に以下のログが出力される。

エラーログ :[2025-01-20 16:43:16.295] ERROR http-nio-8080-exec-10 com.company.bulletinboard.action.admin.bulletinboard.DeleteBulletinboardAction - Error in MoveBulletinboardManagementAction: User session is missing.

＜② データベース接続エラーの処理＞

- 掲示板管理画面に遷移する
- コマンドプロンプトにて管理者権限の状態、以下のコマンドを実効しMySQLサーバーのサービスを停止させる
コマンド:net stop mysql80
※以下、操作履歴
C:\WINDOWS\system32>net stop mysql80
MySQL80 サービスを停止中です..
MySQL80 サービスは正常に停止されました。
- 上記操作実行後、改めて掲示管理画面にて、該当掲示板の「削除」リンクをクリックする。以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

- 以下、エラーログ

[2025-01-20 16:46:54.335] ERROR http-nio-8080-exec-10 com.company.bulletinboard.action.admin.bulletinboard.DeleteBulletinboardAction - データベース接続中にエラーが発生しました
at com.company.bulletinboard.action.admin.bulletinboard.DeleteBulletinboardAction.mainProc(DeleteBulletinboardAction.java:74) [classes/:?]

- テスト実施後、MySQLサーバーのサービスを起動する。

コマンド:net start mysql80
※以下、操作履歴
C:\WINDOWS\system32>net start mysql80
MySQL80 サービスを開始します。
MySQL80 サービスは正常に開始されました。

- エラー画面にてF5キーを押下し、画面のリロード後に掲示板一覧に遷移できることを確認する。
また、該当掲示板が削除されていることを確認する。

以上

■単体テストNo14.1

掲示板管理のキャンセルボタン処理

■目的

- ・掲示板管理画面で、「キャンセル」ボタンの操作ができることを確認する。
- ・リクエストが正しく送信されているかを確認する。
- ・上記処理後、「cancel」の文字列を返すことを確認する。

■テスト対象行

53行目:if (this.request != null){
45行目:if (request == null) {
85行目:if (sessionUser == null) {
118行目:if ("cancel".equals(action)) {
122行目:return "cancel";

■テスト実施内容

●53行目:if (this.request != null){

- ・目的:アクションの開始時にリクエストのパラメータが適切であることを確認する。
—————メイン処理に到達する前に、パラメータが正しくセットされているかどうかを確認すること。

・期待値:リクエストパラ

—————セッションカ

—————必要なパラ

・テスト前(ステップオー

確認内容:

-Eclipse「変数」ウインド

request→request→parameterMap→[0]→のkeyの値が「action」であること

request→request→parameterMap→[0]→value→[0]の値が「cancel」であること

request→request→requestedSessionIdの値が「21603BA73D8E9361A930500900EACDE7」であること

request→session→attributes→[0]→valueの値が以下であること

auth_typeの値が「1」

delete_dayの値が「2999-12-31 00:00:00」

delete_flagの値が「0」

passwordの値が「jp9T-LH2」

user_idの値が「30」

user_nameの値が「tesutuser1」

request→session→idの値が「21603BA73D8E9361A930500900EACDE7」であること

※上記の通り、リクエスト内のrequestedSessionId がセッションに設定されたsession id と一致する。これにより、リクエストとセッションが正しくリンクしていることが確認できる。

・テスト前(ステップオーバー後)

確認内容:

-Eclipse「変数」ウインドウで以下を確認。request階層の以下の項目が、ステップオーバー前と変化が無いことを確認する。

request→request→parameterMap→[0]→のkeyの値が「action」であること

request→request→parameterMap→[0]→value→[0]の値が「cancel」であること

request→request→requestedSessionIdの値が「21603BA73D8E9361A930500900EACDE7」であること

request→session→attributes→[0]→valueの値が以下であること

auth_typeの値が「1」

delete_dayの値が「2999-12-31 00:00:00」

リクエストパラメータの確認は、BaseActionクラスへ処理を統合。単体テスト対象外。統合テストにて実施予定

delete_flagの値が「0」
passwordの値が「jp9T-LH2」
user_idの値が「30」
user_nameの値が「tesutuser1」
request→session→idの値が「21603BA73D8E9361A930500900EACDE7」であること
※上記の通り、ステップオーバー前の値と一致する。期待通り

・デバッグログ

[2025-01-20 19:58:55.122] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction -HttpServletRequest is not null in setServletRequest
[2025-01-20 19:58:56.805] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction -Session ID: 21603BA73D8E9361A930500900EACDE7
[2025-01-20 19:58:58.146] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction -Request parameters:
[2025-01-20 19:59:02.052] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction -action: cancel
[2025-01-20 19:59:12.322] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction -Set action parameter: cancel
[2025-01-20 19:59:14.484] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction -CancelAction mainProc started
[2025-01-20 19:59:14.500] INFO http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction -Start Cancel action -Session ID: 21603BA73D8E9361A930500900EACDE7

- 45行目 :if (request == null) {
- ・目的 :BaseActionクラスのrequestフィールドを正しく継承しているか確認。
request内の詳細情報が期待通りの内容になっているか確認。余計なパラメータが入っていないこと。
- ・テスト前(ステップオーバー前)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。クラスフィールドのrequestが「null」では無いことを確認する。
 - ・this → request(request StrutsRequestWrapper(id=150)) → request(RequestFacade(id=186)) → request(Request(id=191))
- ※上記の通り、「Request(id=191)」が『null』では無いことを確認。期待通りの結果。

- Eclipse「変数」ウィンドウで以下を確認。request内の詳細情報が期待通りの内容になっているかを確認する。
- ・this → request(request StrutsRequestWrapper(id=150)) → request(RequestFacade(id=186)) → request(Request(id=191)) →
parameterMap ParameterMap<K,V> (id=178)
[0] Collections\$UnmodifiableMap\$UnmodifiableEntrySet\$UnmodifiableEntry<K,V> (id=276)
key action (id=280)
hash -1422950858
value (id=284)
[0] a
[1] c
[2] t
[3] i
[4] o
[5] n
value String[1] (id=281)
[0] cancel (id=147)
hash -1367724422
value (id=291)
[0] c
[1] a
[2] n
[3] c
[4] e

※上記の通り、requestの値が「cancel」となっているの期待通り。
余計なパラメータは入っていない。

・テスト後(ステップオーバー後)

確認内容: 前述のステップオーバー前の確認項目の値と変化が無いことを確認する。

-Eclipse「変数」ウィンドウで以下を確認。クラスフィールドのrequestが「null」では無いことを確認する。

・this → request(request StrutsRequestWrapper(id=150)) → request(RequestFacade(id=186)) → request(Request(id=191))

※上記の通り、「Request(id=191)」が『null』でステップオーバー前と変化が無いことを確認。期待通りの結果。

余計なパラメータは入っていない。

-Eclipse「変数」ウィンドウで以下を確認。request内の詳細情報がステップオーバー前の内容と変化が無いことを確認する。

・this → request(request StrutsRequestWrapper(id=150)) → request(RequestFacade(id=186)) → request(Request(id=191)) →

parameterMap ParameterMap<K,V> (id=178)

[0] Collections\$UnmodifiableMap\$UnmodifiableEntrySet\$UnmodifiableEntry<K,V> (id=276)

key action (id=280)

hash -1422950858

value (id=284)

[0] a

[1] c

[2] t

[3] i

[4] o

[5] n

value String[1] (id=281)

[0] cancel (id=147)

hash -1367724422

value (id=291)

[0] c

[1] a

[2] n

[3] c

[4] e

[5] l

※上記の通り、requestの値が「cancel」となっているのステップオーバー前の内容と変化が無い。

余計なパラメータは入っていない。

■デバックログ

[2025-01-27 15:50:14.628] INFO http-nio-8080-exec-8 com.company.bulletinboard.action.CancelAction - HttpServletRequest in BaseAction is available.

[2025-01-27 15:50:17.613] INFO http-nio-8080-exec-8 com.company.bulletinboard.action.CancelAction - Request class: org.apache.struts2.dispatcher.StrutsRequestWrapper

[2025-01-27 15:50:24.383] INFO http-nio-8080-exec-8 com.company.bulletinboard.action.CancelAction - Request contains 1 parameters.

[2025-01-27 15:50:32.259] INFO http-nio-8080-exec-8 com.company.bulletinboard.action.CancelAction - Parameter 'action' value: cancel

●85行目 :if (sessionUser == null) {

・目的: セッションが正しく設定されているかを確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jp9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

・テスト後（ステップオーバー後）

確認内容：

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値：

sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jp9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」

上記の通り、ステップオーバー後も情報が保持されており、意図した内容。期待通りの結果。

2. this → session の内容：

- sessionの値が`SessionMap`オブジェクトである。
- session内にキー`"loggedInUser"`が存在する(`session.containsKey("loggedInUser") == true`)。
- session.get("loggedInUser")の値が`sessionUser`オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

auth_type: 1
delete_day: "2999-12-31 00:00:00"
delete_flag: 0
password: "jp9T-LH2"
user_id: 30
user_name: "tesutuser1"

上記の通り、クラスフィールドにsessionUser`オブジェクトと同じ内容が補完されている。正常動作。期待通り。

3. ログに以下の内容が出力されている：

ログ内容:[2025-01-27 17:18:34.201] INFO http-nio-8080-exec-10 com.company.bulletinboard.action.CancelAction - Session User: tesutuser1
[2025-01-27 17:18:35.656] INFO http-nio-8080-exec-10 com.company.bulletinboard.action.CancelAction - Session UserID: 30
[2025-01-27 17:18:37.241] INFO http-nio-8080-exec-10 com.company.bulletinboard.action.CancelAction - Session User_AuthType: 1

●118行目 :if ("cancel".equals(action)) {

・目的:this が正しく CancelAction クラスのインスタンスになっているか確認。
適切なオブジェクト参照が設定されているか確認。

・テスト前（ステップオーバー前）

期待値： action の値が "cancel" であること。
action="cancel" (id=xxx)として表示されていること
value の配列内容で "cancel" の各文字 ([0] c, [1] a, [2] n, [3] c, [4] e, [5] l) が正しく格納されていること。

確認内容：

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.action の値を確認

- this → actionの値が「cancel(id=176)」
※action の値が "cancel" である。期待通り。

2.value の配列内容を確認

- this → action → valueの値が以下の値となっている

| value | (id=227) |
|-------|----------|
| [0] | c |
| [1] | a |
| [2] | n |
| [3] | c |
| [4] | e |
| [5] | l |

※上記の通り文字列の「cansel」となる

- テスト後（ステップオーバー後）

確認内容:ステップオーバー後もvalueの値に、変化が無ことを確認する

期待値:valueの値が「cancel」であること。

- Eclipse「変数」ウィンドウで以下を確認。

- this → action → valueの値が以下の値となっている

| value | (id=227) |
|-------|----------|
| [0] | c |
| [1] | a |
| [2] | n |
| [3] | c |
| [4] | e |
| [5] | l |

※ステップオーバー前と変わりが無いことを確認。期待通り。

以下、デバッグログ

[2025-01-27 17:21:53.190] INFO http-nio-8080-exec-10 com.company.bulletinboard.action.CancelAction - Cancel button clicked

[2025-01-27 17:21:54.092] INFO http-nio-8080-exec-10 com.company.bulletinboard.action.CancelAction - After Cancel action - Session ID: 85B6A5F6E87DD88DC31E1993AB5AA8C7

- 122行目 :return "cancel";

※return "cancel";の部分については、アプリケーション全体の統合テストや、インターセプターの動作確認の範囲に含められる為、範囲外とする。

Struts2 のフレームワーク内部処理(リダイレクトや画面遷移)は別途システム統合テストの範囲とし、ここではカバレッジの対象外とする。

以上

■単体テストNo14.2

CancelAction - セッション未設定時およびDB接続エラーのハンドリング検証

■目的

- ① セッション未設定時に適切なエラー画面が表示され、ログが記録されること。
- ② action/パラメータの値が「null」の場合に適切なエラー画面が表示され、ログが記録されること。
- ③ action/パラメータの値が「invalid」の場合に適切なエラー画面が表示され、ログが記録されること。

■テスト対象行

<①セッション未設定時のエラー処理>

84行目 :if (sessionUser == null) {

<② action パラメータが null の場合のエラーハンドリング>

116行目 :if ("cancel".equals(action)) {

124行目 :logger.error("Invalid action parameter. Expected 'cancel', but received: " + action);

126行目 :return ERROR; // "error" 結果を返す

<③ action パラメータが "cancel" 以外 (invalid) の場合のエラーハンドリング>

116行目 :if ("cancel".equals(action)) {

124行目 :logger.error("Invalid action parameter. Expected 'cancel', but received: " + action);

126行目 :return ERROR; // "error" 結果を返す

■期待する結果

・セッション未設定時

- 1.エラー画面が表示されること(HTML要素: <h1> に「エラーが発生しました」、<a> に「トップページへ」が含まれる)。
- 2.ログに「User session is missing.」が記録されること。

・ action/パラメータの値が「null」の場合:

- 1.エラー画面が表示されること(HTML要素: 同上)。
- 2.ログに「Invalid action parameter. Expected 'cancel', but received: null」が記録されること。

・ action/パラメータの値が「invalid」の場合:

- 1.エラー画面が表示されること(HTML要素: 同上)。
- 2.ログに「Invalid action parameter. Expected 'cancel', but received: invalid」が記録されること。

■テスト実施内容

<①セッション未設定時のエラー処理>

- 管理者権限にて、掲示板管理画面に遷移する
- 「掲示板作成」ボタンをクリックし、掲示板作成画面に遷移する。遷移後、セッションを削除する
掲示板作成画面にて右クリックし、「検証」をクリックする
アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、掲示板作成画面の「キャンセル」ボタンをクリックする
- 「キャンセル」ボタンをクリック後、以下のエラー画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ戻る](#)

[トップページへ](#)

エラー処理後に以下のログが出力される。

エラーログ:[2025-01-20 16:43:16.295] ERROR http-nio-8080-exec-10 com.company.bulletinboard.action.admin.bulletinboard.DeleteBulletinboardAction - Error in MoveBulletinboardManagementAction: User session is missing.

<② action パラメータが null の場合のエラーハンドリング>

- 掲示板管理画面に遷移する
- 「掲示板作成」ボタンをクリックし、掲示板作成画面に遷移する。
- 掲示板作成画面にて、「キャンセル」ボタンをクリックする。
- Eclipseのデバッグモードになるので、116行目までステップオーバー(F6キーを押下)する。
- 116行目で、Eclipseの「変数」ウィンドウからthis → actionの値を「cancel」から「null」に値を変更する。値を変更後、ステップオーバー(F6キーを押下)する。
- 124行目の「logger.error("Invalid action parameter. Expected 'cancel', but received: " + action);」の例外処理に移動するので、ステップオーバー(F6キーを押下)する。
- 126行目の「return ERROR;」の行に移動するのでステップオーバー(F6キーを押下)する。
- 126行目のステップオーバー後、「OgnlRuntime.jav」ファイルの処理に移るが、「F8」キーで処理を進める。
- 上記操作実行後、以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

エラー処理後に以下のログが出力される。

- 以下、エラーログ

[2025-02-16 13:00:07.464] ERROR http-nio-8080-exec-10 com.company.bulletinboard.action.CancelAction - Invalid action parameter. Expected 'cancel', but received: null

<③ action パラメータが "cancel" 以外 (invalid) の場合のエラーハンドリング>

- 掲示板管理画面に遷移する
- 「掲示板作成」ボタンをクリックし、掲示板作成画面に遷移する。
- 掲示板作成画面にて、「キャンセル」ボタンをクリックする。
- Eclipseのデバッグモードになるので、116行目までステップオーバー(F6キーを押下)する。
- 116行目で、Eclipseの「変数」ウィンドウからthis → actionの値を「cancel」から「invalid」に値を変更する。値を変更後、ステップオーバー(F6キーを押下)する。
- 124行目の「logger.error("Invalid action parameter. Expected 'cancel', but received: " + action);」の例外処理に移動するので、ステップオーバー(F6キーを押下)する。
- 126行目の「return ERROR;」の行に移動するのでステップオーバー(F6キーを押下)する。
- 126行目のステップオーバー後、「OgnlRuntime.jav」ファイルの処理に移るが、「F8」キーで処理を進める。
- 上記操作実行後、以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

エラー処理後に以下のログが出力される。

- 以下、エラーログ

[2025-02-16 13:14:40.031] ERROR http-nio-8080-exec-6 com.company.bulletinboard.action.CancelAction - Invalid action parameter. Expected 'cancel', but received: invalid

以上

■単体テストNo15.1

ListAction - キャンセルボタンクリック後の、掲示板管理画面一覧を再読み込みする処理

■目的

処理がCancelActionクラスからリダイレクトされ正常に遷移し、掲示板管理画面が表示されること。
合わせて掲示板一覧が最新の状態で表示されること。

■テスト対象行

34行目 :BoardDAO dao = new BoardDAO();
39行目 :bulletinboards = dao.getAllBoards();
62行目 :return SUCCESS;

■期待する結果

- ・BoardDAOクラスのインスタンスが正常に生成されること。
- ・BoardDAOクラスの「getAllBoards()」メソッドが呼び出され、データベースから掲示板の全レコードを取得し、それをbulletinboardsリストに格納できること。
- ・掲示板データ取得後、最新のデータが掲示板に反映されること。

■テスト実施内容

34行目 :BoardDAO dao = new BoardDAO();
・目的 :BoardDAOクラスの「dao」インスタンスが正常に生成されることを確認する。
・テスト前 (ステップオーバー前)

確認内容:
Eclipse「変数」ウィンドウの「名前」列に「dao」の項目は表示されていない。BoardDAOクラスのインスタンスは生成されていない状態。

・テスト後 (ステップオーバー後)
確認内容:
ステップオーバー後、Eclipse「変数」ウィンドウの「名前」列に「dao」で、「値」列に「BoardDAO」の項目が表示される。BoardDAOクラスのdaoインスタンスが正常に生成された状態。

処理後に以下のログが出力される。
デバッグログ :[2025-02-21 13:06:52.436] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.ListAction - Fetching all boards from database

39行目 :bulletinboards = dao.getAllBoards();
・目的:
・テスト前 (ステップオーバー前)
確認内容:BoardDAOクラスのdaoインスタンスから「getAllBoards()」メソッドが呼び出される。
その後、データベースから掲示板の全レコードを取得しそれをbulletinboardsリストに格納できることを確認する。
Eclipse「変数」ウィンドウの「bulletinboards」の値が「null」であることを確認する。

・テスト後 (ステップオーバー後)
確認内容:ステップオーバー後、Eclipse「変数」ウィンドウの「bulletinboards」の値が「null」から「ArrayList<E> (id=247)」に更新されたことを確認する。
「ArrayList<E> (id=247)」階層に登録されている掲示板の情報が4件分、格納されていることを確認する。
以下は1件分の内容を抜粋。
bulletinboards ArrayList<E> (id=247)
[0] User (id=258)
bulletinboard_content 2024年7月18日本日掲示板、追記 (id=273)
bulletinboard_creation_day null
bulletinboard_delete_day 2999-12-31 00:00:00 (id=274)
bulletinboard_delete_flag 0
bulletinboard_id 34
bulletinboard_title 2024年7月18日本日 (id=275)
※上記の通り、掲示板情報が正常に取得できている状況。
※以下、デバッグログでも取得した掲示板データの整合を確認済み1件分のみ抜粋。
デバッグログ :[[2025-02-21 13:11:23.101] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.ListAction - Board Data: [ACTIVE] User{bulletinboard_id=34, bulletinboard_title='2024年7月18日本日', bulletinboard_content='2024年7月18日本日掲示板、追記, bulletinboard_delete_flag=0, bulletinboard_delete_day='2999-12-31 00:00:00'}
[2025-02-21 13:09:46.002] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.ListAction - Number of bulletin boards retrieved: 14
[2025-02-21 13:14:08.226] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.ListAction - ListAction mainProc finished

62行目 :return SUCCESS;
確認内容:ステップオーバー後、フレームワークの処理に進みF8キー押下後、以下の掲示板一覧が最新状態で表示されたことを確認。

掲示板管理

[管理メニューに戻る](#)

| | |
|----------------------------|---------------------------------------|
| 掲示板作成 | |
| 2024年7月18日本日 | 編集 削除 |
| 2024年7月18日その② | 編集 削除 |
| 2024年7月18日その3 | 編集 削除 |
| 2024年7月22日 | 編集 削除 |
| 2024年9月1日 ※編集テスト | 編集 削除 |
| 2024年9月25日の掲示板 | 編集 削除 |
| 2024年9月25日 ※ドラッグ＆ドロップで並び替え | 編集 削除 |

| | | |
|---------------------------------|--------------------|--------------------|
| 2024年9月25日 ※ドロップダウンリストで削除しない | 編集 | 削除 |
| 2024年9月25日 ※ドロップダウンリストで削除するを選択 | 編集 | 削除 |
| 2024年9月25日 ※作成画面から掲示板IDフィールドを削除 | 編集 | 削除 |
| 2024年9月26日 ※削除フラグ：削除しない | 編集 | 削除 |
| 2024年9月26日 ※削除フラグ：削除する | 編集 | 削除 |
| testtitle | 編集 | 削除 |
| 営業部掲示板 | 編集 | 削除 |
| 2025年1月掲示板 | 編集 | 削除 |

以上

■単体テストNo15.2

ListAction - BoardDAO のインスタンス生成エラー発生時および .getAllBoardsメソッド呼び出し後のハンドリング検証

■対象行

34行目 :BoardDAO dao = new BoardDAO();

39行目 :bulletinboards = dao.getAllBoards();

■目的

①BoardDAOのインスタンス生成エラー発生時に、例外処理が正常に実行されることを確認する。

②getAllBoardsメソッドが正常に呼ばれた後、取得した掲示板リストが`null` になった場合に、例外処理が正しく動作することを確認する。

■期待する結果

・BoardDAOクラスのインスタンが「null」だった場合に例外処理が発生し、エラー画面に遷移できること。

また、例外処理のデバックログが出力されること

・getAllBoardsメソッドが正常に呼ばれた後、取得した掲示板リストが`null` になった場合にエラー画面に遷移できること。

また、例外処理のデバックログが出力されること

■テスト実施内容

<①BoardDAOのインスタンス生成エラー発生時に、例外処理が正常に実行されることを確認する>

・テスト前①(セッションIDの事前確認)

確認内容:

事前にWEBブラウザの開発者ツールの「Cookies」から JSESSIONID を確認。

掲示板作成画面の「キャンセル」ボタンをクリック前に以下の値を確認する。

| 名前 | 値 |
|------------|----------------------------------|
| JSESSIONID | A87E53C708096970BC8229BAA292C7F6 |

・テスト前②(ステップオーバー前:

確認内容:

②Eclipse「変数」ウィンドウの「名前」列に「dao」の項目が表示されていないことを確認する BoardDAOクラスのインスタンスは生成されていない状態)

・テスト後①(ステップオーバー後:

確認内容:

Eclipseにて34行目をステップオーバー後、Eclipse「変数」ウィンドウの「名前」列に「dao」で、「値」列に「BoardDAO」の項目が表示される。BoardDAOクラスのdaoインスタンスが正常に生成された状態で以下の操作を行う。

<操作>

・Eclipseの「変数」ウィンドウの「dao」の値を「null」に変更する。

・値を変更「後、F8 を押して処理を再開する。

・65行目のエラー処理(bulletinboa() catch (Exception e) {})に進むので、F6キーを押下する。

・F6キーでそのまま進み「return ERROR;」の行でエラーを返す。

・「OgnlRuntime.java」の処理に進むが、F8キーを押下し、処理を進める。

・ブラウザの画面で以下のエラー処理の画面に遷移する。

・エラー画面

エラーが発生しました

ご不便をおかけして申し訳ありません。

- 掲示板データの取得に失敗しました。管理者にお問い合わせください。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

・エラーログ

[2025-03-10 12:58:28.422] ERROR http-nio-8080-exec-7 com.company.bulletinboard.action.ListAction - Error in ListAction mainProc at com.company.bulletinboard.action.ListAction.mainProc(ListAction.java:39) [classes/:?]

[2025-03-10 12:59:24.559] DEBUG http-nio-8080-exec-7 com.opensymphony.xwork2.conversion.impl.InstantiatingNullHandler - Entering nullPropertyValue [target=[com.company.bulletinboard.action.ListAction@4b03334b, com.opensymphony.xwork2.DefaultTextProvider@623e219f], property=struts]

[2025-03-10 12:59:24.577] DEBUG http-nio-8080-exec-7 com.opensymphony.xwork2.ognl.SecurityMemberAccess - Checking access for [target: com.company.bulletinboard.action.ListAction@4b03334b, member: public java.util.Locale com.opensymphony.xwork2.ActionSupport.getLocale(), property: locale]

※上記の通り、例外処理時のデバックログが出力されることを確認。

・テスト後②(エラー発生後の「JSESSIONID」を確認)

| 名前 | 値 |
|------------|----------------------------------|
| JSESSIONID | A87E53C708096970BC8229BAA292C7F6 |

※上記の通り、テスト前の「JSESSIONID」と変化が無いことを確認済み。

＜②getAllBoardsメソッドが正常に呼ばれた後、取得した掲示板リストが`null` になった場合に、例外処理が正しく動作することを確認する。＞

・テスト前①(セッションIDの事前確認)

確認内容:
事前にWEBブラウザの開発者ツールの「Cookies」から JSESSIONID を確認。
掲示板作成画面の「キャンセル」ボタンをクリック前に以下の値を確認する。

| 名前 | ▲ | 値 |
|------------|---|----------------------------------|
| JSESSIONID | | 0B8F4277FA78A887CCA02949F1DF6024 |

・テスト前②(ステップオーバー前)

確認内容:
Eclipse「変数」ウィンドウの「名前」列に「dao」の項目は表示されていない。BoardDAOクラスのインスタンスは生成されていない状態。

・テスト後①(ステップオーバー後)

確認内容:
ステップオーバー後、Eclipse「変数」ウィンドウのthis →「名前」列の「bulletinboards」の「値」が「ArrayList<E> (id=245)」となり、掲示板データが全て取得できている状態。
※結果として、getAllBoardsメソッドが正常に呼び出されている状況。
前述の状況で以下の操作を行う。

・操作
1. Eclipseの「変数」ウィンドウの「this」→「bulletinboards」の値を「ArrayList<E> (id=245)」から「null」に変更する。
 ※NullPointerException を意図的に発生させる、
2. 「F8」を押して処理を再開。
3. WEBブラウザの画面がエラー画面に遷移することを確認。

・エラー画面

エラーが発生しました

ご不便をおかけして申し訳ありません。

- 掲示板データの取得に失敗しました。管理者にお問い合わせください。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

・デバッグログ

[2025-03-10 14:17:58.431] ERROR http-nio-8080-exec-2 com.company.bulletinboard.action.ListAction - Error in ListAction mainProc
at com.company.bulletinboard.action.ListAction.mainProc(ListAction.java:41) [classes/:?]

[2025-03-10 14:17:58.499] DEBUG http-nio-8080-exec-2 com.opensymphony.xwork2.conversion.impl.InstantiatingNullHandler - Entering nullPropertyValue [target=[com.company.bulletinboard.action.ListAction@632ec00b, com.opensymphony.xwork2.DefaultTextProvider@4a633355], property=templateDir]

[2025-03-10 14:17:58.501] DEBUG http-nio-8080-exec-2 com.opensymphony.xwork2.ognl.SecurityMemberAccess - Checking access for [target: com.company.bulletinboard.action.ListAction@632ec00b, member: public java.util.Collection com.opensymphony.xwork2.ActionSupport.getActionErrors(), property: actionErrors]

[2025-03-10 14:17:58.502] DEBUG http-nio-8080-exec-2 com.opensymphony.xwork2.conversion.impl.InstantiatingNullHandler - Entering nullPropertyValue [target=[com.company.bulletinboard.action.ListAction@632ec00b, com.opensymphony.xwork2.DefaultTextProvider@4a633355], property=theme]

[2025-03-10 14:17:58.541] DEBUG http-nio-8080-exec-2 com.opensymphony.xwork2.ognl.SecurityMemberAccess - Checking access for [target: com.company.bulletinboard.action.ListAction@632ec00b, member: public java.util.Locale com.opensymphony.xwork2.ActionSupport.getLocale(), property: locale]

※上記の通り、デバッグログが出力される。

・テスト後②(エラー発生後の「JSESSIONID」を確認)

| 名前 | ▲ | 値 |
|------------|---|----------------------------------|
| JSESSIONID | | 0B8F4277FA78A887CCA02949F1DF6024 |

※上記の通り、テスト前の「JSESSIONID」と変化が無いことを確認済み。

以上

■単体テストNo16.1

ユーザー作成画面へ遷移する処理

■目的

ユーザー管理画面から「ユーザー作成」ボタンをクリックした際に、正しく掲示板作成画面(CreateUserScreen.jsp) に遷移することを確認する。

■テスト実施手順

ユーザー管理画面から遷移の確認

- ・ユーザー管理画面にアクセス。
- ・「ユーザー作成」ボタンをクリック。
- ・CreateUserScreen.jsp に遷移することを確認。

■テスト実施内容

<ユーザー管理画面からユーザー作成画面への遷移確認>

- ユーザー管理画面にアクセスする。

ユーザー管理

[管理メニューに戻る](#)

ユーザー作成

| ユーザーID | ユーザー名 | 権限 | 操作 | |
|--------|-------------|----|--------------------|--------------------|
| 15 | nishioka444 | 1 | 編集 | 削除 |
| 24 | uehara1 | 0 | 編集 | 削除 |
| 25 | uehara23 | 0 | 編集 | 削除 |
| 26 | ueharaadmin | 1 | 編集 | 削除 |
| 30 | tesutuser1 | 1 | 編集 | 削除 |
| 31 | testuser2 | 1 | 編集 | 削除 |
| 32 | testuser3 | 1 | 編集 | 削除 |
| 33 | tesutuser4 | 0 | 編集 | 削除 |
| 34 | testuser5 | 1 | 編集 | 削除 |
| 35 | tesutuser1 | 0 | 編集 | 削除 |
| 36 | tesutuser11 | 0 | 編集 | 削除 |
| 38 | uehara10001 | 0 | 編集 | 削除 |

- 「ユーザー板作成」ボタンをクリックする。

ユーザー管理

[管理メニューに戻る](#)

ユーザー作成

| ユーザーID | ユーザー名 | 権限 | 操作 | |
|--------|-------------|----|--------------------|--------------------|
| 15 | nishioka444 | 1 | 編集 | 削除 |
| 24 | uehara1 | 0 | 編集 | 削除 |
| 25 | uehara23 | 0 | 編集 | 削除 |
| 26 | ueharaadmin | 1 | 編集 | 削除 |
| 30 | tesutuser1 | 1 | 編集 | 削除 |
| 31 | testuser2 | 1 | 編集 | 削除 |
| 32 | testuser3 | 1 | 編集 | 削除 |
| 33 | tesutuser4 | 0 | 編集 | 削除 |
| 34 | testuser5 | 1 | 編集 | 削除 |
| 35 | tesutuser1 | 0 | 編集 | 削除 |
| 36 | tesutuser11 | 0 | 編集 | 削除 |
| 38 | uehara10001 | 0 | 編集 | 削除 |

- CreateUserScreen.action に遷移することを確認

←

→

↺

ⓘ

localhost:8080/Bulletinboard/

CreateUserScreen.action

ユーザー作成

キャンセル

ユーザー名:

パスワード:

権限:

一般ユーザー権限

ユーザー削除フラグ:

削除しない

削除しない

削除日:

作成

以上

■単体テストNo16.2

MoveCreateUserActionの、異常系処理

■目的

セッション未設定時のエラー処理が正しく動作すること。

■テスト実施内容

- 管理者権限にて、ユーザー管理画面に遷移する
- ユーザー管理画面にてセッションを削除する
 - ユーザー管理画面にて右クリックし、「検証」をクリックする
 - アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、「ユーザー作成」ボタンをクリックする
- 「ユーザー作成」ボタンをクリック後、以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

・エラーログ

[2025-03-28 10:41:19.832] INFO http-nio-8080-exec-7 com.company.bulletinboard.interceptor.BaseAction - 処理開始 :com.company.bulletinboard.action.admin.user.MoveCreateUserAction
[2025-03-28 10:41:19.832] ERROR http-nio-8080-exec-7 com.company.bulletinboard.action.admin.user.MoveCreateUserAction - Error in MoveBulletinboardManagementAction: User session is missing.
[2025-03-28 10:41:19.832] INFO http-nio-8080-exec-7 com.company.bulletinboard.interceptor.BaseAction - 処理終了 :com.company.bulletinboard.action.admin.user.MoveCreateUserAction

以上

■単体テストNo17.1

usersテーブルへのデータ登録処理

■目的

- ・ユーザー作成画面の正常な入力データが、usersテーブルに登録されることを確認する。
- ・上記処理後、「SUCCESS」を返すことを確認する。

■テスト対象行

```
150行目 :if (sessionUser == null) {
203行目 :PreparedStatement ps = connection.prepareStatement(sql)) {
206行目 :ps.setInt(1, user.getUser_id());
209行目 :ps.setString(2, user.getUser_name());
212行目 :ps.setString(3, user.getPassword());
215行目 :ps.setInt(4, user.getAuth_type());
218行目 :ps.setInt(5, user.getDelete_flag());
221行目 :ps.setString(6, user.getDelete_day());
240行目 :return SUCCESS;
```

■テスト実施内容

- 150行目 :if (sessionUser == null) {
- ・目的: セッションが正しく設定されているかを確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jP9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

- ・テスト後(ステップオーバー後)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値:

sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jP9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」

上記の通り、ステップオーバー後も情報が保持されており、意図した内容。

2. this → session の内容:

- sessionの値が「SessionMap」オブジェクトである。
- session内にキー「loggedInUser」が存在する(`session.containsKey("loggedInUser") == true`)。
- session.get("loggedInUser")の値が「sessionUser」オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

```
auth_type: 1
delete_day: "2999-12-31 00:00:00"
delete_flag: 0
password: "jP9T-LH2"
user_id: 30
user_name: "tesutuser1"
```

上記の通り、クラスフィールドにsessionUser`オブジェクトと同じ内容が補完されている。正常動作。

3. ログに以下の内容が出力されている:

```
[2025-03-28 13:38:43.258] INFO  http-nio-8080-exec-7 com.company.bulletinboard.action.admin.user.InsertUserAction - Session User: tesutuser1
```

●203行目 :PreparedStatement ps = connection.prepareStatement(sql)) {

- ・目的: 接続の正常性を確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認。

- ・sql → INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,,?)
※不正な値(NULLや予期しない構文など)が含まれていない

2.「connection」オブジェクトの確認。接続がクローズされていないことの確認する。

- ・connection → connectionオブジェクト自体が存在し、NULLで無いこと
- ・connection → delegate → openStatementsの値が「CopyOnWriteArrayList<E> (id=251)」
※openStatementsの値が空でなくアクセス可能な状態。
- ・isAutoCommit → isAutoCommitの値が「true」であること

3.接続先データベース情報の確認

- ・connection → delegate → origHostToConnectTo → 「localhost」(id=233)」であること
- ・connection → delegate → origPortToConnectTo → ポート番号が「3306」であること
- ・connection → delegate → database → データベース名が「bulletinboard_db」(id=190)」であること
- ・connection → delegate → user → 「root」(id=262) 」であること

4.接続有効性のログ

connection.isValid(timeout)メソッドが「true」を返しており、ログに「データベース接続は正常です。」というメッセージが表示されている。

ログ内容:[2025-03-28 13:39:38.724] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.admin.user.InsertUserAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

- ・テスト後(ステップオーバー後)

確認内容:PreparedStatement の生成が正しく行われ、後続処理に影響がないことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

1.「sql」変数の確認

- ・sql →
INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,,?)
※正しい SQL 文 ("

INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,,?)")を保持していることを確認

2. connection の有効性を確認

- ・preparedStatement → isClosedの値が「false」であることを確認
※ステートメントが閉じていない状態

3.「proxyResultSet」の確認

- ・preparedStatement → proxyResultSetの値が「null」であることを確認

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返すことを確認

ログ内容:[2025-03-28 13:39:38.724] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.admin.user.InsertUserAction - データベース接続は正常です。ホスト localhost, ポート: 3306, データベース: bulletinboard_db
※※ログメッセージに「データベース接続は正常です。」が出力されていることを確認。接続は有効な状態

●206行目 :ps.setInt(1, user.getUser_id());

・目的:ps の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

・sql → INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,,?)

※正しい SQL 文 ("INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,,?)")を保持していることを確認

2.「ps」の有効性確認

・ps → isClosed の値が「false」

※ステートメントは有効で操作可能な状態

3.フォームデータがpsステートメントに紐づけされていないことを確認

・ps → delegate → query → queryBindings → bairdValues → [0] → valueの値が「null」であることを確認。

※フォームデータがステートメントに紐づけられていない状態

・テスト後(ステップオーバー後)

確認内容:psステートメントにフォームデータが紐づけられたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

・ps → delegate → query → queryBindings → bairdValues → [0] → valueの値が「"0"」であることを確認。

※ユーザー作成の途中なので、この時点ではユーザーIDは割り当てられない。なので正常動作。

以下、デバッグログ

[2025-03-28 13:46:14.637] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.admin.user.InsertUserAction - User ID: = 0

●209行目 :ps.setString(2, user.getUser_name());

・目的:ps の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

・sql → INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,,?)

※正しい SQL 文 ("INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,,?)")を保持していることを確認

2.「ps」の有効性確認

・ps → isClosed の値が「false」

※ステートメントは有効で操作可能な状態

3.フォームデータがpsステートメントに紐づけされていないことを確認

・ps → delegate → query → queryBindings → bairdValues → [1] → valueの値が「null」であることを確認。

※フォームデータがステートメントに紐づけられていない状態

・テスト後(ステップオーバー後)

確認内容:psステートメントにフォームデータが紐づけられたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

・ps → delegate → query → queryBindings → bairdValues → [1] → valueの値が「"testuser0328"」であることを確認。

※フォームデータがステートメントに紐づけられていることを確認。

以下、デバッグログ

[2025-03-28 13:50:22.188] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.admin.user.InsertUserAction - User Name: = testuser0328

●212行目 :ps.setString(3, user.getPassword());

・目的:ps の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

・sql → INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,,?)

※正しい SQL 文("INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,,?)")を保持していることを確認

2.「ps」の有効性確認

・ps → isClosed の値が「false」

※ステートメントは有効で操作可能な状態

3.フォームデータがpsステートメントに紐づけされていないことを確認

・ps → delegate → query → queryBindings → bairdValues → [2] → valueの値が「null」であることを確認。

※フォームデータがステートメントに紐づけられていない状態

・テスト後(ステップオーバー後)

確認内容:psステートメントにフォームデータが紐づけられたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

・ps → delegate → query → queryBindings → bairdValues → [2] → valueの値が「"jP9T-LH2"」であることを確認。

※フォームデータがステートメントに紐づけられていることを確認。

以下、デバッグログ

[2025-03-28 13:54:02.288] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.admin.user.InsertUserAction - Password: = jP9T-LH2

●215行目 :ps.setInt(4, user.getAuth_type());

・目的:ps の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

・sql → INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,,?)

※正しい SQL 文("INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,,?)")を保持していることを確認

2.「ps」の有効性確認

・ps → isClosed の値が「false」

※ステートメントは有効で操作可能な状態

3.フォームデータがpsステートメントに紐づけされていないことを確認

- ・ps → delegate → query → queryBindings → baindValues → [3] → valueの値が「null」であることを確認。
※フォームデータがステートメントに紐づけられていない状態

・テスト後(ステップオーバー後)

確認内容:psステートメントにフォームデータが紐づけられたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

- ・ps → delegate → query → queryBindings → baindValues → [3] → valueの値が「" 1 "」であることを確認。
※フォームデータがステートメントに紐づけられていることを確認。(この場合は管理者権限の「」が割り当てられてる)
以下、デバッグログ

[2025-03-28 13:58:06.727] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.admin.user.InsertUserAction - Auth Type: = 1

●218行目 :ps.setInt(5, user.getDelete_flag());

・目的:ps の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- ・sql → INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,?)
※正しい SQL 文 ("INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,?)")を保持していることを確認

2.「ps」の有効性確認

- ・ps → isClosed の値が「false」
※ステートメントは有効で操作可能な状態

3.フォームデータがpsステートメントに紐づけされていないことを確認

- ・ps → delegate → query → queryBindings → baindValues → [4] → valueの値が「null」であることを確認。
※フォームデータがステートメントに紐づけられていない状態

・テスト後(ステップオーバー後)

確認内容:psステートメントにフォームデータが紐づけられたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

- ・ps → delegate → query → queryBindings → baindValues → [4] → valueの値が「"0"」であることを確認。
※Delete_flagが「0」で意図した値が設定されている
以下、デバッグログ

[2025-03-28 14:00:45.440] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.admin.user.InsertUserAction - Delete Flag: = 0

●221行目 :ps.setString(6, user.getDelete_day());

・目的:ps の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- ・sql → INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,?)
※正しい SQL 文 ("INSERT INTO users(user_id, user_name, password, auth_type, delete_flag, delete_day) VALUES(?,?,?,?,?)")を保持していることを確認

2.「ps」の有効性確認

- ・ps → isClosed の値が「false」
※ステートメントは有効で操作可能な状態

3.フォームデータがpsステートメントに紐づけされていないことを確認

- ・ps → delegate → query → queryBindings → bairdValues → [5] → valueの値が「null」であることを確認。
※フォームデータがステートメントに紐づけられていない状態

- ・テスト後（ステップオーバー後）

確認内容 :psステートメントにフォームデータが紐づけられたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

- ・ps → delegate → query → queryBindings → bairdValues → [5] → valueの値が「"2050-12-31 00:00:00"」であることを確認。

※データ削除日が割り当てられている。正常動作。

以下、デバッグログ

[2025-03-28 14:05:25.940] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.admin.user.InsertUserAction - Delete Day: = 2050-12-31 00:00:00

- 240行目 :return SUCCESS;

※return result;の部分については、アプリケーション全体の統合テストや、インターセプターの動作確認の範囲に含まれる為、範囲外とする。

以上

■単体テストNo17.2

InsertUserActionクラスのエラー処理

■目的

- ①セッション未設定時のエラー処理が正しく動作すること。
- ② データベース接続エラーの処理が正しく動作すること。
- ③クエリ実行エラーの処理が正しく動作すること。(リスクがあるので中止)

■テスト対象行

<①セッション未設定時のエラー処理>

```
150行目 :if (sessionUser == null) {  
151行目 :String errorMessage = "User session is missing.";  
152行目 :addActionError(errorMessage);  
153行目 :logger.error("Error in InsertBulletinboardAction: " + errorMessage);  
154行目 :return ERROR;
```

<② データベース接続エラーの処理>

```
246行目 :} catch (SQLException e) {  
248行目 :logger.error("データベース接続中にエラーが発生しました", e);  
250行目 :return ERROR;
```

■テスト実施内容

<①セッション未設定時のエラー処理>

- 管理者権限にて、ユーザー作成画面に遷移する
- ユーザー作成画面にてセッションを削除する
 - 管理メニュー画面にて右クリックし、「検証」をクリックする
 - アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、「作成」ボタンをクリックする
- 「作成」ボタンをクリック後、以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

エラー処理後に以下のログが出力される。

エラーログ :[2025-03-28 14:37:12.488] ERROR http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.InsertUserAction - Error in InsertBulletinboardAction: User session is missing.

<② データベース接続エラーの処理>

- 掲示板作成画面に遷移する
- 掲示板作成画面のフォームに必須項目を入力する
- コマンドプロンプトにて管理者権限の状態で、以下のコマンドを実効しMySQLサーバーのサービスを停止させる
 コマンド:net stop mysql80
 ※以下、操作履歴
 C:\WINDOWS\system32>net stop mysql80
 MySQL80 サービスを停止中です..
 MySQL80 サービスは正常に停止されました。
- 上記操作実行後、改めて掲示板作成画面にて「作成」ボタンをクリックする。以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

- 以下、エラーログ
[2025-03-28 14:40:59.471] ERROR http-nio-8080-exec-9 com.company.bulletinboard.action.admin.user.InsertUserAction - 接続を閉じる際にエラーが発生しました
 at com.company.bulletinboard.action.admin.user.InsertUserAction.mainProc(InsertUserAction.java:166) [classes/:?]

- テスト実施後、MySQLサーバーのサービスを起動する。
 コマンド:net start mysql80
 ※以下、操作履歴
 C:\WINDOWS\system32>net start mysql80
 MySQL80 サービスを開始します。
 MySQL80 サービスは正常に開始されました。
- エラー画面にてF5キーを押下し、画面のリロード後に掲示板管理画面に遷移できることを確認する。
 また、作成した掲示板が一覧に表示されることを確認する。

ユーザー管理

[管理メニューに戻る](#)

ユーザー作成

| ユーザーID | ユーザー名 | 権限 | 操作 | |
|--------|-------------|----|--------------------|--------------------|
| 15 | nishioka444 | 1 | 編集 | 削除 |
| 24 | uehara1 | 0 | 編集 | 削除 |
| 25 | uehara23 | 0 | 編集 | 削除 |

| | | | | |
|----|--------------|---|--------------------|--------------------|
| 26 | ueharaadmin | 1 | 編集 | 削除 |
| 30 | tesutuser1 | 1 | 編集 | 削除 |
| 31 | testuser2 | 1 | 編集 | 削除 |
| 32 | testuser3 | 1 | 編集 | 削除 |
| 33 | tesutuser4 | 0 | 編集 | 削除 |
| 34 | testuser5 | 1 | 編集 | 削除 |
| 35 | tesutuser1 | 0 | 編集 | 削除 |
| 36 | tesutuser11 | 0 | 編集 | 削除 |
| 38 | uehara10001 | 0 | 編集 | 削除 |
| 40 | testuser0328 | 1 | 編集 | 削除 |
| 41 | tesutuser13 | 0 | 編集 | 削除 |

以上

■単体テストNo18.1

ユーザー編集画面の表示処理

■目的

ユーザー編集画面の入力フォームに、編集対象の既存データが表示できることを確認する。
・上記処理後、「SUCCESS」を返すことを確認する。

■テスト対象行

```
52行目 :if (sessionUser == null) {  
88行目 :PreparedStatement preparedStatement = connection.prepareStatement(sql)) {  
102行目 :user.setUser_id(resultSet.getInt("user_id"));  
105行目 :user.setUser_name(resultSet.getString("user_name"));  
108行目 :user.setPassword(resultSet.getString("password"));  
111行目 :user.setAuth_type(resultSet.getInt("auth_type"));  
114行目 :user.setDelete_flag(resultSet.getInt("delete_flag"));  
117行目 :user.setDelete_day(resultSet.getString("delete_day"));  
129行目 :return SUCCESS;
```

■事前準備

※掲示板テーブルに以下のデータが予め、登録されていることとする。

```
user_id:41  
user_name:tesutuser13  
password:zE4c9m+q  
user_id:30  
auth_type:0  
delete_flag: 0  
delete_day:2050-12-31 00:00:00.000000
```

■テスト実施内容

●52行目 :if (sessionUser == null) {

- ・目的:セッションが正しく設定されているかを確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

```
sessionUser → auth_typeの値が「1」  
sessionUser → delete_dayの値が「2999-12-31 00:00:00」  
sessionUser → delete_flagの値が「0」  
sessionUser → passwordの値が「jP9T-LH2」  
sessionUser → user_idの値が「30」  
sessionUser → user_nameの値が「tesutuser1」
```

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

- ・テスト後(ステップオーバー後)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値:

```
sessionUser → auth_typeの値が「1」
```

sessionUser → delete_dayの値が「2999-12-31 00:00:00」

sessionUser → delete_flagの値が「0」

sessionUser → passwordの値が「jP9T-LH2」

sessionUser → user_idの値が「30」

sessionUser → user_nameの値が「tesutuser1」

上記の通り、ステップオーバー後も情報が保持されており、意図した内容。期待通りの結果。

2. this → session の内容:

- sessionの値が`SessionMap`オブジェクトである。

- session内にキー`"loggedInUser"`が存在する(`session.containsKey("loggedInUser") == true`)。

- session.get("loggedInUser")の値が`sessionUser`オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

auth_type: 1

delete_day: "2999-12-31 00:00:00"

delete_flag: 0

password: "jP9T-LH2"

user_id: 30

user_name: "tesutuser1"

上記の通り、クラスフィールドにsessionUser`オブジェクトと同じ内容が補完されている。正常動作。期待通り。

3. ログに以下の内容が出力されている:

[2025-03-29 07:18:46.869] INFO http-nio-8080-exec-6 com.company.bulletinboard.action.admin.user.EditUserAction - Session User: tesutuser1

●88行目:PreparedStatement preparedStatement = connection.prepareStatement(sql)) {

・目的:接続の正常性を確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認。

・sql → "SELECT * FROM bulletinboard WHERE bulletinboard_id = ?" (id=205)

※不正な値(NULLや予期しない構文など)が含まれていない。SQL文が事前に妥当な構文として検証されていること(エラーが発生しないこと)」

2.「connection」オブジェクトの確認。接続がクローズされていないことの確認する。

・connection → connectionオブジェクト自体が存在し、NULLで無いこと

・connection → delegate → openStatementsの値が「CopyOnWriteArrayList<E> (id=241)」

※openStatementsの値が空でなくアクセス可能な状態。

・isAutoCommit → isAutoCommitの値が「true」であること

3.接続先データベース情報の確認

・connection → delegate → origHostToConnectTo → 「"localhost" (id=246)」であること

・connection → delegate → origPortToConnectTo → ポート番号が「3306」であること

・connection → delegate → database → データベース名が「"bulletinboard_db" (id=231)」であること

・connection → delegate → user → 「"root" (id=276) 」であること

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返しており、ログに「データベース接続は正常です。」というメッセージが表示されている。

ログ内容:[2025-03-29 07:30:52.896] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.user.EditUserAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

・テスト後(ステップオーバー後)

確認内容:PreparedStatement の生成が正しく行われ、後続処理に影響がないことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

1.「sql」変数の確認

・sql → SELECT * FROM users WHERE user_id = ?
※正しい SQL 文 ("SELECT * FROM users WHERE user_id = ?") を保持していることを確認

2. connection の有効性を確認

・preparedStatement → isClosedの値が「false」であることを確認
※ステートメントが閉じていない状態

3. 「proxyResultSet」の確認

・preparedStatement → proxyResultSetの値が「null」であることを確認

4. 接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返すことを確認

ログ内容: [2025-03-29 07:30:52.896] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.user.EditUserAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

※※ログメッセージに「データベース接続は正常です。」が出力されていることを確認。接続は有効な状態

●102行目 :user.setUser_id(resultSet.getInt("user_id"));

・目的:preparedStatement の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

期待値: sqlの値が「SELECT * FROM users WHERE user_id = ?」であること。
preparedStatementのisCloseの値が「false」であること。
user_idの値が「null」であること

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1. 「sql」変数の確認

・sql → SELECT * FROM users WHERE user_id = ?
※正しい SQL 文 (SELECT * FROM users WHERE user_id = ?) を保持していることを確認。期待通り。

2. 「preparedStatement」の有効性確認

・preparedStatement → isClosed の値が「false」

※ステートメントは有効で操作可能な状態。期待通り。

3. 既存データがモデルのuserに紐づけされていないことを確認

・this → user → user_idの値が「0」であることを確認。
※既存データがモデルに紐づけられていない状態。期待通り。

・テスト後(ステップオーバー後)

期待値: user_idの値が「41」であること

確認内容: 既存データがモデルのuserに紐づけされたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

・this → user → user_idの値が「41」であることを確認。

※テスト対象の既存データがモデルに紐づけられたことを確認。期待通りの結果。
以下、デバッグログ

[2025-03-29 07:38:49.533] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.user.EditUserAction - User ID: = 41

●105行目 :user.setUser_name(resultSet.getString("user_name"));

・目的:preparedStatement の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

期待値: sqlの値が「SELECT * FROM users WHERE user_id = ?」であること。
preparedStatement の isClosed の値が「false」であること。

user_name"の値が「null」であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

•sql →SELECT * FROM users WHERE user_id = ?

※正しいSQL文("SELECT * FROM users WHERE user_id = ?")を保持していることを確認。期待通り。

2.「preparedStatement」の有効性確認

•preparedStatement → isClosed の値が「false」

※ステートメントは有効で操作可能な状態。期待通り。

3.既存データがモデルのuserに紐づけされていないことを確認。

•this → user → user_nameの値が「null」であることを確認。期待通り。

※既存データがモデルに紐づけられていない状態

•テスト後(ステップオーバー後)

確認内容: 既存データがモデルのuserに紐づけされたことを確認する

期待値: user_nameの値が「tesutuser13」であること

- Eclipse「変数」ウィンドウで以下を確認。

•this → user → user_nameの値が「tesutuser13」であることを確認。

※既存データがモデルに紐づけられたことを確認。期待通り。

以下、デバッグログ

[2025-03-29 07:43:59.449] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.user.EditUserAction - User Name: = tesutuser13

●108行目 :user.setPassword(resultSet.getString("password"));

•目的: preparedStatement の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

•テスト前(ステップオーバー前)

•期待値: sql の値が「SELECT * FROM users WHERE user_id = ?」であること。

preparedStatement の isClosed の値が「false」であること。

passwordの値が「null」であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

•sql → SELECT * FROM users WHERE user_id = ?

※正しいSQL文("SELECT * FROM users WHERE user_id = ?")を保持していることを確認。期待通りの結果。

2.「preparedStatement」の有効性確認

•preparedStatement → isClosed の値が「false」

※ステートメントは有効で操作可能な状態。期待通りの結果。

3.既存データがモデルのuserに紐づけされていないことを確認

•this → user → passwordの値が「null」であることを確認。

※既存データがモデルに紐づけられていない状態。期待値通り。

•テスト後(ステップオーバー後)

確認内容: 既存データがモデルのuserに紐づけされたことを確認する

期待値: passwordの値が「zE4c9m+q」であること。

- this → user → passwordの値が「zE4c9m+q」であることを確認。
※既存データがモデルに紐づけられたことを確認。
- ※フォームデータがステートメントに紐づけられていることを確認。期待通り。
以下、デバッグログ

[2025-03-29 07:50:13.201] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.user.EditUserAction - Password: = zE4c9m+q

- 111行目 :user.setAuth_type(resultSet.getInt("auth_type"));
- 目的:preparedStatement の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。
- テスト前(ステップオーバー前)
- 期待値: sql の値が、「SELECT * FROM users WHERE user_id = ?」であること。
preparedStatementの値がisClosed の値が「false」であること。
bulletinboard_delete_flagの値が「0」であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- sql → SELECT * FROM users WHERE user_id = ?
※正しい SQL 文 ("SELECT * FROM users WHERE user_id = ?")を保持していることを確認。期待通りの結果。

2.「preparedStatement」の有効性確認

- preparedStatement → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待値通り。

3.既存データがモデルのuserに紐づけされていないことを確認

- this → user → auth_typeの値が「0」であることを確認。
※既存データがモデルに紐づけられていない状態。期待通り。

- テスト後(ステップオーバー後)
- 期待値: auth_typeの値が「0」であること。

確認内容: 既存データがモデルのuserに紐づけされたことを確認する

- this → user → auth_typeの値が「0」であることを確認。
※既存データがモデルに紐づけられたことを確認。
- ※フォームデータがステートメントに紐づけられていることを確認。
以下、デバッグログ

[2025-03-29 07:53:50.131] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.user.EditUserAction - Auth Type: = 0

※期待通りの結果。

- 114行目 :user.setDelete_flag(resultSet.getInt("delete_flag"));
- 目的:preparedStatement の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。
- 期待値: sql の値が「SELECT * FROM users WHERE user_id = ?」であること。
preparedStatementのisClosed の値が「false」であること。
delete_flagの値が「null」であること。
- テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- sql → SELECT * FROM users WHERE user_id = ?

※正しい SQL 文 ("SELECT * FROM users WHERE user_id = ?") を保持していることを確認。期待通りの結果。

2. 「preparedStatement」の有効性確認

- ・preparedStatement → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待値通り。

3. 既存データがモデルのuserに紐づけされていないことを確認

- ・this → user → delete_flagの値が「0」であることを確認。
※既存データがモデルに紐づけられていない状態。期待値通り。

- ・テスト後（ステップオーバー後）

期待値: delete_flagの値が「0」であること。

確認内容: 既存データがモデルのuserに紐づけされたことを確認する

- ・this → user → delete_flagの値が「0」であることを確認。

※既存データがモデルに紐づけられたことを確認。

※フォームデータがステートメントに紐づけられていることを確認。

以下、デバッグログ

[2025-03-29 07:58:35.453] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.user.EditUserAction - Delete Flag: = 0

※上記は期待通りの結果。

- 117行目 :user.setDelete_day(resultSet.getString("delete_day"));

- ・目的:preparedStatement の状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

- ・期待値: sql の値が「SELECT * FROM users WHERE user_id = ?」であること。
preparedStatementのisClosed の値が「false」であること。
delete_flagの値が「null」であること。

- ・テスト前（ステップオーバー前）

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1. 「sql」変数の確認

- ・sql → SELECT * FROM users WHERE user_id = ?

※正しい SQL 文 ("SELECT * FROM users WHERE user_id = ?") を保持していることを確認。期待通りの結果。

2. 「preparedStatement」の有効性確認

- ・preparedStatement → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待値通り。

3. 既存データがモデルのuserに紐づけされていないことを確認

- ・this → user → delete_dayの値が「null」であることを確認。
※既存データがモデルに紐づけられていない状態。期待値通り。

- ・テスト後（ステップオーバー後）

期待値:

確認内容: 既存データがモデルのuserに紐づけされたことを確認する

- ・this → user → delete_dayの値が「2050-12-31 00:00:00」であることを確認。

※既存データがモデルに紐づけられたことを確認。

※フォームデータがステートメントに紐づけられていることを確認。

以下、デバッグログ

[2025-03-29 08:04:20.302] DEBUG http-nio-8080-exec-8 com.company.bulletinboard.action.admin.user.EditUserAction - Delete Day: = 2050-12-31 00:00:00

※上記は期待通りの結果。

●174行目 :return SUCCESS;

※return SUCCESS;の部分については、アプリケーション全体の統合テストや、インターセプターの動作確認の範囲に含められる為、範囲外とする。

以上

■単体テストNo18.2

EditUserAction - セッション未設定時およびDB接続エラーのハンドリング検証

■目的

- ① セッション未設定時に適切なエラー画面が表示され、ログが記録されること。
- ② データベース接続エラー時に適切なエラー画面が表示され、ログが記録されること。

■テスト対象行

＜①セッション未設定時のエラー処理＞

53行目 :if (sessionUser == null) {

＜② データベース接続エラーの処理＞

170行目 :} catch (SQLException e) {

178行目 :logger.error("データベース接続中にエラーが発生しました ", e);

179行目 :throw e;

■期待する結果

・セッション未設定時

- 1.エラー画面が表示されることHTML要素: <h1> に「エラーが発生しました」、<a> に「トップページへ」が含まれる）。
- 2.ログに「User session is missing.」が記録されること。

・データベース接続エラー時

- 1.エラー画面が表示されることHTML要素: 同上）。
- 2.ログに「データベース接続中にエラーが発生しました」が記録されること。

■テスト実施内容

＜①セッション未設定時のエラー処理＞

- 管理者権限にて、ユーザー管理画面に遷移する
- ユーザー管理画面にてセッションを削除する
 - 管理メニュー画面にて右クリックし、「検証」をクリックする
 - アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、該当ユーザーの「編集」リンクをクリックする
- 「編集」ボタンをクリック後、以下のエラー画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

エラー処理後に以下のログが出力される。

エラーログ :[2025-03-29 08:39:06.280] ERROR http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.EditUserAction - Error in MoveBulletinboardManagementAction: User session is missing.

＜② データベース接続エラーの処理＞

- ユーザー管理画面に遷移する
- コマンドプロンプトにて管理者権限の状態で、以下のコマンドを実効しMySQLサーバーのサービスを停止させる
 コマンド:net stop mysql80
 ※以下、操作履歴
 C:\WINDOWS\system32>net stop mysql80
 MySQL80 サービスを停止中です..
 MySQL80 サービスは正常に停止されました。
- 上記操作実行後、改めてユーザー管理画面にて該当ユーザーの「編集」リンクをクリックする。以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

- 以下、エラーログ

```
[2025-03-29 08:43:06.158] ERROR http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.EditUserAction - データベース接続中にエラーが発生しました
    at com.company.bulletinboard.action.admin.user.EditUserAction.getUserById(EditUserAction.java:87) [classes/:?]
    at com.company.bulletinboard.action.admin.user.EditUserAction.mainProc(EditUserAction.java:66) [classes/:?]
```

- テスト実施後、MySQLサーバーのサービスを起動する。
 コマンド:net start mysql80
 ※以下、操作履歴
 C:\WINDOWS\system32>net start mysql80
 MySQL80 サービスを開始します。
 MySQL80 サービスは正常に開始されました。

- エラー画面にてF5キーを押下し、画面のリロード後にユーザー編集画面に遷移できることを確認する。
 また、編集フォームに該当ユーザーのデータがフォームに表示されることを確認する。

ユーザー編集

キャンセル

ユーザー名:

パスワード:

権限: ▼

ユーザー削除フラグ:

削除しない ▼

削除日:

2050-12-31 00:00:00

更新

以上

■単体テストNo19.1

ユーザー編集画面の更新処理

■目的

- ・ユーザー編集画面にて既存データを編集後、正常に更新処理が実行できることを確認する。
- ・上記処理後、「SUCCESS」を返すことを確認する。

■テスト対象行

```
142行目 :if (sessionUser == null) {  
197行目 :PreparedStatement ps = connection.prepareStatement(sql);  
200行目 :ps.setString(1, user.getUser_name());  
203行目 :ps.setString(2, user.getPassword());  
206行目 :ps.setInt(3, user.getAuth_type());  
209行目 :ps.setInt(4, user.getDelete_flag());  
212行目 :ps.setString(5, user.getDelete_day());  
215行目 :ps.setInt(6, user.getUser_id());  
234行目 :return SUCCESS;
```

■事前準備

※usersテーブルに以下のデータが予め、登録されていることとする。

```
user_id:41  
user_name:tesutuser13  
password:zE4c9m+q  
auth_type:0  
delete_flag:0  
bulletinboard_delete_flag:0  
delete_day: 2050-12-31 00:00:00.000000
```

■更新処理の内容

<変更前>

```
ユーザー名 :tesutuser13  
パスワード :zE4c9m+q  
権限 :一般ユーザー権限  
ユーザー削除フラグ :削除しない  
削除日 :2050-12-31 00:00:00
```

<変更後>

```
ユーザー名 :tesutuser14  
パスワード :aE4c9m+h  
権限 :管理者権限  
ユーザー削除フラグ :削除する  
削除日 :2999-12-31 00:00:00
```

■テスト実施内容

- 142行目 :if (sessionUser == null) {

・目的:セッションが正しく設定されているかを確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

sessionUser → auth_typeの値が「1」

sessionUser → delete_dayの値が「2999-12-31 00:00:00」

sessionUser → delete_flagの値が「0」

sessionUser → passwordの値が「jP9T-LH2」

sessionUser → user_idの値が「30」

sessionUser → user_nameの値が「tesutuser1」
※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

・テスト後（ステップオーバー後）

確認内容：

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値：

sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jP9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」

上記の通り、ステップオーバー後も情報が保持されており、意図した内容。期待通りの結果。

2. this → session の内容：

- sessionの値がSessionMap`オブジェクトである。
- session内にキー`"loggedInUser"`が存在する(`session.containsKey("loggedInUser") == true`)。
- session.get("loggedInUser")の値がsessionUser`オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

auth_type: 1
delete_day: "2999-12-31 00:00:00"
delete_flag: 0
password: "jP9T-LH2"
user_id: 30
user_name: "tesutuser1"

上記の通り、クラスフィールドにsessionUser`オブジェクトと同じ内容が補完されている。正常動作。期待通り。

3. ログに以下の内容が出力されている：

[2025-03-29 15:44:57.064] INFO http-nio-8080-exec-6 com.company.bulletinboard.action.admin.user.UpdateUserAction - Session User: tesutuser1

●197行目:PreparedStatement ps = connection.prepareStatement(sql);

・目的: 接続の正常性を確認する。

・テスト前（ステップオーバー前）

確認内容：

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認。

・sql → "UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?"
※不正な値 (NULLや予期しない構文など) が含まれていない。SQL文が事前に妥当な構文として検証されていること(エラーが発生しないこと)

2.「connection」オブジェクトの確認。接続がクローズされていないことの確認する。

・connection → connectionオブジェクト自体が存在し、NULLで無いこと
・connection → delegate → openStatementsの値が「CopyOnWriteArrayList<E> (id=241)」
※openStatementsの値が空でなくアクセス可能な状態。
・isAutoCommit → isAutoCommitの値が「true」であること

3.接続先データベース情報の確認

・connection → delegate → origHostToConnectTo → 「"localhost" (id=290)」であること
・connection → delegate → origPortToConnectTo → ポート番号が「3306」であること
・connection → delegate → database → データベース名が「"bulletinboard_db(id:194)」であること
・connection → delegate → user → 「"root" (id=291) 」であること

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返しており、ログに「データベース接続は正常です。」というメッセージが表示されている。

ログ内容:[2025-03-29 15:47:02.825] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.admin.user.UpdateUserAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

・テスト後（ステップオーバー後）

確認内容:ps の生成が正しく行われ、後続処理に影響がないことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

1.「sql」変数の確認

・sql → UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?

※正しい SQL 文("UPDATE bulletinboard SET bulletinboard_title = ?, bulletinboard_content = ?, bulletinboard_delete_flag = ?, bulletinboard_delete_day = ? WHERE bulletinboard_id = ?")を保持していることを確認

2. connection の有効性を確認

・ps → isClosedの値が「false」であることを確認

※ステートメントが閉じていない状態

3.「proxyResultSet」の確認

・preparedStatement → proxyResultSetの値が「null」であることを確認

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返すことを確認

ログ内容:[2025-01-15 10:57:17.089] DEBUG http-nio-8080-exec-2 com.company.bulletinboard.action.admin.bulletinboard.UpdateBulletinboardAction - データベース接続は正常です。ホスト localhost, ポート: 3306, データベース: bulletinboard_db

※※ログメッセージに「データベース接続は正常です。」が出力されていることを確認。接続は有効な状態

●200行目:ps.setString(1, user.getUser_name());

・目的:プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

期待値: sqlの値が「UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?」であること。

ps の isClosed の値が「false」であること。

user_nameの値が「tesutuser14」であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

・sql → UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?

※正しい SQL 文("UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?")を保持していることを確認。期待通り。

2.「ps」の有効性確認

・ps → isClosed の値が「false」

※ステートメントは有効で操作可能な状態。期待通り。

3.フォームの入力値が、モデルのuserに紐づけされていることを確認。

・this → user → user_nameの値が「tesutuser14」であることを確認。期待通り。

※上記の通り、フォームの入力値がモデルに紐づけられている状態。

・テスト後(ステップオーバー後)

確認内容:ステップオーバー後もuserに紐づけされた値に、変化が無ことを確認する

期待値:user_nameの値が「tesutuser14」であること(フォームの入力値)

- Eclipse「変数」ウィンドウで以下を確認。

・this → user → user_nameの値が「tesutuser14」であることを確認。

※ステップオーバー前と変わりが無いことを確認。期待通り。

以下、デバッグログ

[2025-03-29 15:57:54.721] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.admin.user.UpdateUserAction - User Name: = tesutuser14

●203行目:ps.setString(2, user.getPassword());

・目的:プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

・期待値: sql の値が「UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?」であること。

ps の isClosed の値が「false」であること。

passwordの値が「aE4c9m+h」であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- sql →UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?
※正しいSQL文("UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?)を保持していることを確認。期待通りの結果。

2.「ps」の有効性確認

- ps → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待通りの結果。

3.フォームの入力値がモデルのuserに紐づけされていることを確認

- this → user → passwordの値が「aE4c9m+h」であることを確認。
※入力値がモデルに紐づけられている状態。期待値通り。

・テスト後(ステップオーバー後)

確認内容:フォームの入力値がモデルのuserに紐づけされていることを確認する

期待値: passwordの値が「aE4c9m+h」であること。

- this → user → passwordの値が「aE4c9m+h」であることを確認。
※フォームデータがモデルに紐づけられていることを確認。期待通り。
以下、デバッグログ
[2025-03-29 16:02:17.182] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.admin.user.UpdateUserAction - Password: = aE4c9m+h

●206行目 :ps.setInt(3, user.getAuth_type());

・目的:プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

- ・期待値: sql の値が、「UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?」であること。
psの値がisClosed の値が「false」であること。
auth_typeの値が「1」であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- sql → UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?
※正しいSQL文(" UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?")を保持していることを確認。期待通りの結果。

2.「ps」の有効性確認

- ps → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待値通り。

3.フォームデータがモデルのuserに紐づけされていないことを確認

- this → user → auth_typeの値が「1」であることを確認。
※フォームデータがモデルに紐づけられていない状態。期待通り。

・テスト後(ステップオーバー後)

- ・期待値: atuth_typeの値が「1」であること。

確認内容:セレクトボタンのデータがモデルのuserに紐づけされたことを確認する

- this → user → auth_typeの値が「1」であることを確認。
※セレクトボタンのデータがステートメントに紐づけられていることを確認。
以下、デバッグログ
[2025-03-29 16:06:04.423] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.admin.user.UpdateUserAction - Auth Type: = 1
※期待通りの結果。

●209行目 :ps.setInt(4, user.getDelete_flag());

・目的:プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・期待値: sql の値が「UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?」であること。
psのisClosed の値が「false」であること。
delete_flagの値が「1」であること。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

・sql → UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?
※正しい SQL 文("UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?")を保持していることを確認。期待通りの結果。

2.「ps」の有効性確認

・ps → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待値通り。

3.フォームデータがモデルのuserに紐づけされていることを確認

・this → user →delete_flagの値が「1」であることを確認。
※フォームデータがモデルに紐づけられている状態。期待値通り。

・テスト後(ステップオーバー後)

期待値: delete_flagの値が「1」であること。

確認内容: 入力データがモデルのuserに紐づけされたことを確認する

・this → user →delete_flagの値が「1」であることを確認。
※フォームデータがステートメントに紐づけられていることを確認。
以下、デバッグログ
[2025-03-29 16:17:53.932] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.admin.user.UpdateUserAction - Delete Flag: = 1
※上記は期待通りの結果。

●212行目 :ps.setString(5, user.getDelete_day());

・目的: プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

期待値: sqlの値が「UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?」であること。
psのisCloseの値が「false」であること。
delete_dayの値が「2999-12-31 00:00:00」であること

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

・sql → UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?
※正しい SQL 文 (UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?)を保持していることを確認。期待通り。

2.「ps」の有効性確認

・ps → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待通り。

3.フォームのデータがモデルuserに紐づけられていることを確認

・this → user → delete_dayの値が「2999-12-31 00:00:00」であることを確認。
※delete_dayの値がモデルに紐づけられている状態。期待通り。

・テスト後(ステップオーバー後)

期待値:delete_daytの値が「2999-12-31 00:00:00」であること

確認内容:フォームのdelete_dayの値がuserに紐づけされたことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

・this → user → delete_dayの値が「2999-12-31 00:00:00」であることを確認。
※既存データがモデルに紐づけられたことを確認。期待通りの結果。

以下、デバッグログ

[2025-03-29 16:23:17.169] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.admin.user.UpdateUserAction - Delete Day: = 2999-12-31 00:00:00

●215行目 :ps.setInt(6, user.getUser_id());

・目的: プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・期待値: sql の値が「UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?」であること。
 psのisClosed の値が「false」であること。
 user_idの値が「41」であること。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

・sql → UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?
 ※正しい SQL 文("UPDATE users SET user_name = ?, password = ?, auth_type = ?, delete_flag = ?, delete_day = ? WHERE user_id = ?")を保持していることを確認。期待通りの結果。

2.「ps」の有効性確認

・ps → isClosed の値が「false」
 ※ステートメントは有効で操作可能な状態。期待値通り。

3.フォームデータがモデルのuserに紐づけされていることを確認

・this → user → user_idの値が「41」であることを確認。
 ※フォームデータがモデルに紐づけられている状態。期待値通り。

・テスト後(ステップオーバー後)

期待値: user_idの値が「41」であること。

確認内容:ステップオーバー後もuser_idの値に変化が無いことを確認する

・this → user → user_idの値が「41」であることを確認。
 ※フォームデータがステートメントに紐づけられていることを確認。
 以下、デバッグログ

[2025-03-29 16:30:28.048] DEBUG http-nio-8080-exec-6 com.company.bulletinboard.action.admin.user.UpdateUserAction - User ID: = 41

※上記は期待通りの結果。

●234行目 :return SUCCESS;

※return SUCCESS;の部分については、アプリケーション全体の統合テストや、インターセプターの動作確認の範囲に含まれる為、範囲外とする。

以上

■単体テストNo19.2

UpdateUserAction - セッション未設定時およびDB接続エラーのハンドリング検証

■目的

- ① セッション未設定時に適切なエラー画面が表示され、ログが記録されること。
- ② データベース接続エラー時に適切なエラー画面が表示され、ログが記録されること。

■テスト対象行

＜①セッション未設定時のエラー処理＞

142行目 :if (sessionUser == null) {

＜② データベース接続エラーの処理＞

254行目 :if (sqlState != null && sqlState.startsWith("08")) {

255行目 :logger.error("データベース接続中にエラーが発生しました", e);

256行目 :addActionError("データベース接続中にエラーが発生しました。管理者にお問い合わせください");

■期待する結果

・セッション未設定時

- 1.エラー画面が表示されること(HTML要素: <h1> に「エラーが発生しました」、<a> に「トップページへ」が含まれる)。
- 2.ログに「User session is missing.」が記録されること。

・データベース接続エラー時

- 1.エラー画面が表示されること(HTML要素: 同上)。
- 2.ログに「データベース接続中にエラーが発生しました」が記録されること。

■テスト実施内容

＜①セッション未設定時のエラー処理＞

- 管理者権限にて、ユーザー編集画面に遷移する
- ユーザー編集画面にて、入力フォームに編集内容を入力後、セッションを削除する
管理メニュー画面にて右クリックし、「検証」をクリックする
アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、該当ユーザーの「更新」リンクをクリックする
- 「更新」ボタンをクリック後、以下のエラー画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

エラー処理後に以下のログが出力される。

[2025-03-30 10:37:38.236] ERROR http-nio-8080-exec-6 com.company.bulletinboard.action.admin.user.UpdateUserAction - Error in MoveBulletinboardManagementAction: User session is missing.

<② データベース接続エラーの処理>

- ユーザー編集画面に遷移する
- ユーザー編集画面にて、入力フォームに編集内容を入力
- コマンドプロンプトにて管理者権限の状態で、以下のコマンドを実効しMySQLサーバーのサービスを停止させる
コマンド:net stop mysql80
※以下、操作履歴
C:\WINDOWS\system32>net stop mysql80
MySQL80 サービスを停止中です..
MySQL80 サービスは正常に停止されました。
- 上記操作実行後、改めてユーザー編集画面にて「更新」ボタンをクリックする。以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

- 以下、エラーログ
[2025-03-30 10:49:27.190] ERROR http-nio-8080-exec-5 com.company.bulletinboard.action.admin.user.UpdateUserAction - データベース接続中にエラーが発生しました
at com.company.bulletinboard.action.admin.user.UpdateUserAction.mainProc(UpdateUserAction.java:160) [classes/:?]

- テスト実施後、MySQLサーバーのサービスを起動する。
コマンド:net start mysql80
※以下、操作履歴
C:\WINDOWS\system32>net start mysql80
MySQL80 サービスを開始します。
MySQL80 サービスは正常に開始されました。

- エラー画面にてF5キーを押下し、画面のリロード後にユーザー管理画面に遷移し、ユーザー一覧が表示される。
改めて、該当ユーザーの編集画面を表示し、編集内容が更新されていることを確認する。

ユーザー編集

キャンセル

ユーザー名:

パスワード:

権限:

ユーザー削除フラグ:

削除日: 2999-12-31 00:00:00

更新

以上

■単体テストN20.1

掲示板削除処理

■目的

- ・ユーザー管理画面で、ユーザーが削除できることを確認する。
- ・上記処理後、「SUCCESS」を返すことを確認する。

■テスト対象行

56行目 :if (sessionUser == null) {
112行目 :PreparedStatement ps = connection.prepareStatement(sql);
115行目 :ps.setInt(1, id);
121行目 :return SUCCESS;

■事前準備

※掲示板テーブルに以下のデータが予め、登録されていることとする。

user_id:41
user_name:tesutuser15
password:aE4c9m+h
auth_type:1
delete_flag: 1
delete_day:2999-12-31 00:00:00.000000

■削除処理の内容

<削除内容>
ユーザー名 :tesutuser15
パスワード:aE4c9m+h
権限:管理者権限
ユーザー削除フラグ:ユーザー削除フラグ
削除日:2999-12-31 00:00:00

■テスト実施内容

- 56行目 :if (sessionUser == null) {
- ・目的:セッションが正しく設定されているかを確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jp9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

- ・テスト後(ステップオーバー後)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値:

sessionUser → auth_typeの値が「1」

sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jP9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」

上記の通り、ステップオーバー後も情報が保持されており、意図した内容。期待通りの結果。

2. this → session の内容:

- sessionの値が`SessionMap`オブジェクトである。
- session内にキー`"loggedInUser"`が存在する(`session.containsKey("loggedInUser") == true`)。
- session.get("loggedInUser")の値が`sessionUser`オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

auth_type: 1
delete_day: "2999-12-31 00:00:00"
delete_flag: 0
password: "jP9T-LH2"
user_id: 30
user_name: "tesutuser1"

上記の通り、クラスフィールドにsessionUser`オブジェクトと同じ内容が補完されている。正常動作。期待通り。

3. ログに以下の内容が出力されている:

ログ内容:[2025-03-30 11:46:51.826] INFO http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - Session User: tesutuser1
[2025-03-30 11:46:52.975] INFO http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - Session UserID: 30
[2025-03-30 11:46:53.728] INFO http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - Session User_AuthType: 1

●112行目 :PreparedStatement ps = connection.prepareStatement(sql);

・目的: 接続の正常性を確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認。

・sql → "DELETE FROM users WHERE user_id = ?"

※不正な値(NULLや予期しない構文など)が含まれていない。SQL文が事前に妥当な構文として検証されていること(エラーが発生しないこと)

2.「connection」オブジェクトの確認。接続がクローズされていないことの確認する。

・connection → connectionオブジェクト自体が存在し、NULLで無いこと

・connection → delegate → openStatementsの値が「CopyOnWriteArrayList<E> (id=290)」

※openStatementsの値が空でなくアクセス可能な状態。

・isAutoCommit → isAutoCommitの値が「true」であること

3.接続先データベース情報の確認

・connection → delegate → origHostToConnectTo → 「"localhost" (id=297)」であること

・connection → delegate → origPortToConnectTo → ポート番号が「3306」であること

・connection → delegate → database → データベース名が「"bulletinboard_db" (id=248)」であること

・connection → delegate → user → 「"root" (id=355)」であること

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返しており、ログに「データベース接続は正常です。」というメッセージが表示されている。

[2025-03-30 11:57:05.189] DEBUG http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

・テスト後(ステップオーバー後)

確認内容:ps の生成が正しく行われ、後続処理に影響がないことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

1.「sql」変数の確認

・sql → DELETE FROM users WHERE user_id = ?

※正しい SQL 文 ("DELETE FROM users WHERE user_id = ?") を保持していることを確認

2. connection の有効性を確認

・ps → isClosedの値が「false」であることを確認

※ステートメントが閉じていない状態

3. 「proxyResultSet」の確認

・preparedStatement → proxyResultSetの値が「null」であることを確認

4. 接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返すことを確認

ログ内容 : [2025-03-30 11:57:05.189] DEBUG http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

※※ログメッセージに「データベース接続は正常です。」が出力されていることを確認。接続は有効な状態

●115行目 : ps.setInt(1, id);

・目的: プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前 (ステップオーバー前)

期待値: sqlの値が「DELETE FROM users WHERE user_id = ?」であること。

ps の isClosed の値が「false」であること。

bulletinboard_idの値が「55」であること。

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1. 「sql」変数の確認

・sql → DELETE FROM users WHERE user_id = ?

※正しい SQL 文 ("DELETE FROM users WHERE user_id = ?") を保持していることを確認。期待通り。

2. 「ps」の有効性確認

・ps → isClosed の値が「false」

※ステートメントは有効で操作可能な状態。期待通り。

3. 削除対象のユーザーIDの値が、thisのidに紐づけられているかの確認

・this → id → idの値が「41」であることを確認。期待通り。

・テスト後 (ステップオーバー後)

確認内容: ステップオーバー後もid値に、変化が無ことを確認する

期待値: idの値が「41」であること。

- Eclipse「変数」ウィンドウで以下を確認。

・this → 41 → idの値が「41」であることを確認。

※ステップオーバー前と変わりが無いことを確認。期待通り。

以下、デバッグログ

[2025-03-30 12:03:25.047] DEBUG http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - user_id: = 41

[2025-03-30 12:11:59.388] INFO http-nio-8080-exec-10 com.company.bulletinboard.interceptor.BaseAction - 処理終了 : com.company.bulletinboard.action.admin.user.DeleteUserAction

●213行目 : return SUCCESS;

※return SUCCESS;の部分については、アプリケーション全体の統合テストや、インターセプターの動作確認の範囲に含められる為、範囲外とする。

●削除処理の正常性確認

削除対象がテーブルから正常に削除されていることを確認済み。また、処理終了のログの出力されているので、問題なし。

mysql> select * from users;

| user_id | user_name | password | auth_type | delete_flag | delete_day |

| | | | | | | | | | |
|--|----|--------------|----------|--|---|--|---|----------------------------|--|
| | 15 | nishioka444 | n5J%v&db | | 1 | | 1 | 2025-12-31 00:00:00.000000 | |
| | 24 | xxxx1 | Q7iYxLP! | | 0 | | 0 | 2999-12-31 00:00:00.000000 | |
| | 25 | xxxx23 | xxxx23 | | 0 | | 0 | 2999-12-31 00:00:00.000000 | |
| | 26 | xxxxadmin | zE4c9m+q | | 1 | | 1 | 2999-12-31 00:00:00.000000 | |
| | 30 | tesutuser1 | jP9T-LH2 | | 1 | | 0 | 2999-12-31 00:00:00.000000 | |
| | 31 | testuser2 | test | | 1 | | 1 | 2999-12-31 00:00:00.000000 | |
| | 32 | testuser3 | test | | 1 | | 0 | 2999-12-31 00:00:00.000000 | |
| | 33 | tesutuser4 | test | | 0 | | 0 | 2999-12-31 00:00:00.000000 | |
| | 34 | testuser5 | test | | 1 | | 0 | 2999-12-31 00:00:00.000000 | |
| | 35 | tesutuser1 | zE4clm+q | | 0 | | 0 | 2050-12-31 00:00:00.000000 | |
| | 36 | tesutuser11 | jP9T-LH2 | | 0 | | 0 | 2050-12-31 00:00:00.000000 | |
| | 38 | xxxx10001 | zE4c9m+q | | 0 | | 0 | 2050-12-31 00:00:00.000000 | |
| | 40 | testuser0328 | jP9T-LH2 | | 1 | | 0 | 2050-12-31 00:00:00.000000 | |

※上記の通り削除対象のuser_id「41」がテーブル上から削除されている。

以上

■単体テストNo20.2

DeleteUserAction - セッション未設定時およびDB接続エラーのハンドリング検証

■目的

- ① セッション未設定時に適切なエラー画面が表示され、ログが記録されること。
- ② データベース接続エラー時に適切なエラー画面が表示され、ログが記録されること。

■事前準備

※掲示板テーブルに以下のデータが予め、登録されていることとする。

user_id:40
user_name:testuser0328
password:jP9T-LH2
auth_type:1
delete_flag: 0
delete_day:2050-12-31 00:00:00.000000

■削除処理の内容

<削除内容>
ユーザー名:testuser0328
パスワード:jP9T-LH2
権限:管理者権限
ユーザー削除フラグ:削除しない
削除日:2050-12-31 00:00:00

■テスト対象行

<①セッション未設定時のエラー処理>
56行目:if (sessionUser == null) {
<② データベース接続エラーの処理>
133行目:if (sqlState != null && sqlState.startsWith("08")) {
134行目:logger.error("データベース接続中にエラーが発生しました", e);
135行目:addActionError("データベース接続中にエラーが発生しました。管理者にお問い合わせください");

■期待する結果

- ・セッション未設定時
1.エラー画面が表示されることHTML要素: <h1> に「エラーが発生しました」、<a> に「トップページへ」が含まれる)。
2.ログに「User session is missing.」が記録されること。
- ・データベース接続エラー時
1.エラー画面が表示されることHTML要素: 同上)。
2.ログに「データベース接続中にエラーが発生しました」が記録されること。

■テスト実施内容

- <①セッション未設定時のエラー処理>
- 管理者権限にて、ユーザー管理画面に遷移する
 - ユーザー一覧にて、セッションを削除する
管理メニュー画面にて右クリックし、「検証」をクリックする
アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
 - クッキー削除後、該当ユーザーの「削除」リンクをクリックする

-「削除」リンクをクリック後、以下のエラー画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

エラー処理後に以下のログが出力される。

[2025-03-30 12:36:00.942] ERROR http-nio-8080-exec-8 com.company.bulletinboard.action.admin.user.DeleteUserAction - Error in DeleteUserAction: User session is missing.

＜② データベース接続エラーの処理＞

- ユーザー管理画面に遷移する
- コマンドプロンプトにて管理者権限の状態で、以下のコマンドを実効しMySQLサーバーのサービスを停止させる
コマンド:net stop mysql80
※以下、操作履歴
C:\WINDOWS\system32>net stop mysql80
MySQL80 サービスを停止中です..
MySQL80 サービスは正常に停止されました。
- 上記操作実行後、改めてユーザー管理画面にて、該当ユーザーの「削除」リンクをクリックする。以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

- 以下、エラーログ

[2025-03-30 12:40:25.953] ERROR http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - データベース接続中にエラーが発生しました
at com.company.bulletinboard.action.admin.user.DeleteUserAction.mainProc(DeleteUserAction.java:75) [classes/:?]

- テスト実施後、MySQLサーバーのサービスを起動する。
コマンド:net start mysql80
※以下、操作履歴
C:\WINDOWS\system32>net start mysql80
MySQL80 サービスを開始します。

MySQL80 サービスは正常に開始されました。

- エラー画面にてF5キーを押下し、画面のリロード後にユーザー一覧に遷移できることを確認する。
また、該当掲示板が削除されていることを確認する。

ユーザー管理

[管理メニューに戻る](#)

ユーザー作成

| ユーザーID | ユーザー名 | 権限 | 操作 | |
|--------|-------------|----|--------------------|--------------------|
| 15 | nishioka444 | 1 | 編集 | 削除 |
| 24 | uehara1 | 0 | 編集 | 削除 |
| 25 | uehara23 | 0 | 編集 | 削除 |
| 26 | ueharaadmin | 1 | 編集 | 削除 |
| 30 | tesutuser1 | 1 | 編集 | 削除 |
| 31 | testuser2 | 1 | 編集 | 削除 |
| 32 | testuser3 | 1 | 編集 | 削除 |
| 33 | tesutuser4 | 0 | 編集 | 削除 |
| 34 | testuser5 | 1 | 編集 | 削除 |
| 35 | tesutuser1 | 0 | 編集 | 削除 |
| 36 | tesutuser11 | 0 | 編集 | 削除 |
| 38 | uehara10001 | 0 | 編集 | 削除 |

※上記の通り、削除対象のユーザーは一覧に存在しない。

以上

■単体テストNo21.1

ユーザー管理のキャンセルボタン処理

■目的

- ・ユーザー管理画面で、「キャンセル」ボタンの操作ができることを確認する。
- ・リクエストが正しく送信されているかを確認する。
- ・上記処理後、「cancel」の文字列を返すことを確認する。

■テスト対象行

```
53行目:if (this.request != null){
58行目 :if (request == null) {
97行目 :if (sessionUser == null) {
151行目 :if ("usercancel".equals(action)) {
157行目 :return "usercancel";
```

■テスト実施内容

◆53行目:if (this.request != null){

・目的:アクションの開始時にリクエストのパラメータが適切であることを確認する。
—————メイン処理に到達する前に、パラメータが正しくセットされているかどうかを確認すること。

・期待値:リクエストパラ

—————セッションカ

—————必要なパラ

・テスト前(ステップオー

確認内容:

-Eclipse「変数」ウインド

```
request → request → parameterMap → [0] → のkeyの値が「action」であること
request → request → parameterMap → [0] → value → [0]の値が「cancel」であること
request → request → requestedSessionIdの値が「21603BA73D8E9361A930500900EACDE7」であること
request → session → attributes → [0] → valueの値が以下であること
    auth_typeの値が「1」
    delete_dayの値が「2999-12-31 00:00:00」
    delete_flagの値が「0」
    passwordの値が「jp9T-LH2」
    user_idの値が「30」
    user_nameの値が「tesutuser1」
```

request → session → idの値が「21603BA73D8E9361A930500900EACDE7」であること

※上記の通り、リクエスト内のrequestedSessionId がセッションに設定されたsession-id と一致する。これにより、リクエストとセッションが正しくリンクしていることが確認できる。

・テスト前(ステップオーバー後)

確認内容:

-Eclipse「変数」ウインドウで以下を確認。request階層の以下の項目が、ステップオーバー前と変化が無いことを確認する。

```
request → request → parameterMap → [0] → のkeyの値が「action」であること
request → request → parameterMap → [0] → value → [0]の値が「cancel」であること
request → request → requestedSessionIdの値が「21603BA73D8E9361A930500900EACDE7」であること
request → session → attributes → [0] → valueの値が以下であること
    auth_typeの値が「1」
```

リクエストパラメータの確認は、BaseActionクラスへ処理を統合。
単体テスト対象外。統合テストにて実施予定

delete_dayの値が「2999-12-31 00:00:00」
delete_flagの値が「0」
passwordの値が「jP9T-LH2」
user_idの値が「30」
user_nameの値が「tesutuser1」

request→session→idの値が「21603BA73D8E9361A930500900EACDE7」であること
※上記の通り、ステップオーバー前の値と一致する。期待通り

・デバッグログ

[2025-01-20 19:58:55.122] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction—HttpServletRequest is not null in setServletRequest
[2025-01-20 19:58:56.805] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction—Session ID: 21603BA73D8E9361A930500900EACDE7
[2025-01-20 19:58:58.146] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction—Request parameters:
[2025-01-20 19:59:02.052] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction—action: cancel
[2025-01-20 19:59:12.322] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction—Set action parameter: cancel
[2025-01-20 19:59:14.484] DEBUG http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction—CancelAction-mainProc started
[2025-01-20 19:59:14.500] INFO http-nio-8080-exec-4 com.company.bulletinboard.action.CancelAction—Start Cancel action—Session ID: 21603BA73D8E9361A930500900EACDE7

- 58行目 :if (request == null) {
- ・目的:BaseActionクラスのrequestフィールドを正しく継承しているか確認。
request内の詳細情報が期待通りの内容になっているか確認。余計なパラメータが入っていないこと。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。クラスフィールドのrequestが「null」では無いことを確認する。
・this → request(request StrutsRequestWrapper(id=165)) → request(RequestFacade(id=177)) → request(Request(id=179))
※上記の通り、「Request」が「null」では無いことを確認。期待通りの結果。

-Eclipse「変数」ウィンドウで以下を確認。request内の詳細情報が期待通りの内容になっているかを確認する。
・this → request(request StrutsRequestWrapper(id=165)) → request(RequestFacade(id=177)) → request(Request(id=179)) →
parameterMap ParameterMap<K,V> (id=230)
[0] Collections\$UnmodifiableMap\$UnmodifiableEntrySet\$UnmodifiableEntry<K,V> (id=247)
key action (id=251)
hash -1422950858
value (id=255)
[0] a
[1] c
[2] t
[3] i
[4] o
[5] n
value String[1] (id=252)
[0] usercancel (id=156)
hash 2032522757
value (id=258)
[0] u
[1] s
[2] e
[3] r

| | |
|-----|---|
| [4] | c |
| [5] | a |
| [6] | n |
| [7] | c |
| [8] | e |
| [9] | l |

※上記の通り、requestの値が「cancel」となっているので期待通り。
余計なパラメータは入っていない。

・テスト後(ステップオーバー後)

確認内容: 前述のステップオーバー前の確認項目の値と変化が無いことを確認する。

-Eclipse「変数」ウィンドウで以下を確認。クラスフィールドのrequestが「null」では無いことを確認する。

・this → request(request StrutsRequestWrapper(id=165)) → request(RequestFacade(id=177)) → request(Request(id=179))

※上記の通り、「Request」が「null」では無い、ステップオーバー前と変化が無いことを確認。期待通りの結果。
余計なパラメータは入っていない。

-Eclipse「変数」ウィンドウで以下を確認。request内の詳細情報がステップオーバー前の内容と変化が無いことを確認する。

・this → request(request StrutsRequestWrapper(id=165)) → request(RequestFacade(id=177)) → request(Request(id=179))

```
parameterMap  ParameterMap<K,V>  (id=230)
    [0]          Collections$UnmodifiableMap$UnmodifiableEntrySet$UnmodifiableEntry<K,V>  (id=290)
        key      action (id=251)
            hash  -1422950858
            value (id=255)
                [0]      a
                [1]      c
                [2]      t
                [3]      i
                [4]      o
                [5]      n
        value     String[1] (id=252)
            [0]     usercancel (id=156)
                hash 2032522757
                value (id=258)
                    [0]      u
                    [1]      s
                    [2]      e
                    [3]      r
                    [4]      c
                    [5]      a
                    [6]      n
                    [7]      c
                    [8]      e
                    [9]      l
```

※上記の通り、requestの値が「cancel」となっているのステップオーバー前の内容と変化が無い。
余計なパラメータは入っていない。

■デバックログ

[2025-03-30 14:24:42.338] INFO http-nio-8080-exec-8 com.company.bulletinboard.action.UserCancelAction - HttpServletRequest in BaseAction is available.

[2025-03-30 14:24:52.850] INFO http-nio-8080-exec-8 com.company.bulletinboard.action.UserCancelAction - Request class: org.apache.struts2.dispatcher.StrutsRequestWrapper

[2025-03-30 14:27:11.443] INFO http-nio-8080-exec-8 com.company.bulletinboard.action.UserCancelAction - Request contains 1 parameters.

[2025-03-30 14:27:15.494] INFO http-nio-8080-exec-8 com.company.bulletinboard.action.UserCancelAction - Parameter 'action' value: usercancel

※※上記のログの通り、BaseActionクラスのrequestフィールドを正しく継承できている。その裏付けとして、requestの値が「usercancel」となっているので期待通り。

●97行目: if (sessionUser == null) {

・目的: セッションが正しく設定されているかを確認する。

・テスト前(ステップオーバー前)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

sessionUser → auth_typeの値が「1」

sessionUser → delete_dayの値が「2999-12-31 00:00:00」

sessionUser → delete_flagの値が「0」

sessionUser → passwordの値が「jP9T-LH2」

sessionUser → user_idの値が「30」

sessionUser → user_nameの値が「tesutuser1」

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

・テスト後(ステップオーバー後)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値:

sessionUser → auth_typeの値が「1」

sessionUser → delete_dayの値が「2999-12-31 00:00:00」

sessionUser → delete_flagの値が「0」

sessionUser → passwordの値が「jP9T-LH2」

sessionUser → user_idの値が「30」

sessionUser → user_nameの値が「tesutuser1」

上記の通り、ステップオーバー後も情報が保持されており、意図した内容。期待通りの結果。

2. this → session の内容:

- sessionの値が「SessionMap」オブジェクトである。

- session内にキー「loggedInUser」が存在する(`session.containsKey("loggedInUser") == true`)。

- session.get("loggedInUser")の値が「sessionUser」オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

auth_type: 1

delete_day: "2999-12-31 00:00:00"

delete_flag: 0

password: "jP9T-LH2"

user_id: 30

user_name: "tesutuser1"

上記の通り、クラスフィールドにsessionUserオブジェクトと同じ内容が補完されている。正常動作。期待通り。

3. ログに以下の内容が出力されている:

ログ内容: [2025-03-30 14:32:51.932] INFO http-nio-8080-exec-8 com.company.bulletinboard.action.UserCancelAction - Session User: tesutuser1

[2025-03-30 14:32:52.229] INFO http-nio-8080-exec-8 com.company.bulletinboard.action.UserCancelAction - Session UserID: 30

- 151行目 :if ("usercancel".equals(action)) {
- ・目的: this が正しく CancelAction クラスのインスタンスになっているか確認。
適切なオブジェクト参照が設定されているか確認。
- ・テスト前(ステップオーバー前)
- 期待値: action の値が "usercancel" であること。
action="usercancel"(id=xxx)として表示されていること
value の配列内容で "usercancel" の各文字 ([0] u, [1] s, [2] e, [3] r, [4] c, [5] a, [6] n, [7] s, [8] e, [9] l) が正しく格納されていること。

確認内容:
-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。
1.action の値を確認
・this → actionの値が「usercancel」
※action の値が "usercancel" である。期待通り。

2.value の配列内容を確認
・this → action → valueの値が以下の値となっている

| value | (id=258) |
|-------|----------|
| [0] | u |
| [1] | s |
| [2] | e |
| [3] | r |
| [4] | c |
| [5] | a |
| [6] | n |
| [7] | c |
| [8] | e |
| [9] | l |

※上記の通り文字列の「usercancel」となる

・テスト後(ステップオーバー後)
確認内容: ステップオーバー後もvalueの値に、変化が無ことを確認する
期待値: valueの値が「usercancel」であること。
- Eclipse「変数」ウィンドウで以下を確認。
・this → action → valueの値が以下の値となっている

| value | (id=258) |
|-------|----------|
| [0] | u |
| [1] | s |
| [2] | e |
| [3] | r |
| [4] | c |
| [5] | a |
| [6] | n |
| [7] | c |

[8] e
[9] l

※ストップオーバー前と変わりが無いことを確認。期待通り。

以下、デバッグログ

[2025-03-30 14:42:27.737] INFO http-nio-8080-exec-8 com.company.bulletinboard.action.UserCancelAction - Cancel button clicked

[2025-03-30 14:42:32.562] INFO http-nio-8080-exec-8 com.company.bulletinboard.action.UserCancelAction - After Cancel action - Session ID: 107AE14433B7551ADA2D3F4D231FA685

●157行目 :return "usercancel";

※return "usercancel";の部分については、アプリケーション全体の統合テストや、インターセプターの動作確認の範囲に含まれる為、範囲外とする。

Struts2 のフレームワーク内部処理(リダイレクトや画面遷移)は別途システム統合テストの範囲とし、ここではカバレッジの対象外とする。

以上

■単体テストNo21.2

UserCancelAction - セッション未設定時およびDB接続エラーのハンドリング検証

■目的

- ① セッション未設定時に適切なエラー画面が表示され、ログが記録されること。
- ② actionパラメータの値が「null」の場合に適切なエラー画面が表示され、ログが記録されること。
- ③ actionパラメータの値が「invalid」の場合に適切なエラー画面が表示され、ログが記録されること。

■テスト対象行

<①セッション未設定時のエラー処理>

97行目 :if (sessionUser == null) {

<② action パラメータが null の場合のエラーハンドリング>

151行目 :if ("usercancel".equals(action)) {

160行目 :logger.error("Invalid action parameter. Expected 'cancel', but received: " + String.valueOf(action));

163行目 :return ERROR;

<③ action パラメータが "usercancel" 以外 (invalid) の場合のエラーハンドリング>

151行目 :if ("usercancel".equals(action)) {

160行目 :logger.error("Invalid action parameter. Expected 'cancel', but received: " + String.valueOf(action));

163行目 :return ERROR;

■期待する結果

・セッション未設定時

- 1.エラー画面が表示されること(HTML要素: <h1> に「エラーが発生しました」等が含まれる)。
- 2.ログに「User session is missing.」が記録されること。

・ actionパラメータの値が「null」の場合:

- 1.エラー画面が表示されること(HTML要素: 同上)。
- 2.ログに「Invalid action parameter. Expected 'usercancel', but received: null」が記録されること。

・ actionパラメータの値が「invalid」の場合:

- 1.エラー画面が表示されること(HTML要素: 同上)。
- 2.ログに「Invalid action parameter. Expected 'usercancel', but received: invalid」が記録されること。

■テスト実施内容

<①セッション未設定時のエラー処理>

- 管理者権限にて、ユーザー管理画面に遷移する
- 「ユーザー作成」ボタンをクリックし、ユーザー作成画面に遷移する。遷移後、セッションを削除する
ユーザー作成画面にて右クリックし、「検証」をクリックする
アプリケーションタブ→ Cookieを選択し、該当のクッキーを削除する
- クッキー削除後、ユーザー作成画面の「キャンセル」ボタンをクリックする
- 「キャンセル」ボタンをクリック後、以下のエラー画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

エラー処理後に以下のログが出力される。

エラーログ:[2025-03-30 15:16:19.190] ERROR http-nio-8080-exec-6 com.company.bulletinboard.action.UserCancelAction - Error in MoveUsesrManagementAction: User session is missing.

<② action パラメータが null の場合のエラーハンドリング>

- ユーザー管理画面に遷移する
- 「ユーザー作成」ボタンをクリックし、ユーザー作成画面に遷移する。
- ユーザー作成画面にて、「キャンセル」ボタンをクリックする。
- Eclipseのデバッグモードになるので、116行目までステップオーバー(F6キーを押下)する。
- 151行目で、Eclipseの「変数」ウィンドウからthis → actionの値を「usercancel」から「null」に値を変更する。値を変更後、ステップオーバー(F6キーを押下)する。
- 160行目の「logger.error("Invalid action parameter. Expected 'usercancel', but received: " + String.valueOf(action));」の例外処理に移動するので、ステップオーバー(F6キーを押下)する。
- 163行目の「return ERROR;」の行に移動するのでステップオーバー(F6キーを押下)する。
- 上記のステップオーバー後、「OgnlRuntime.jav」ファイルの処理に移るが、「F8」キーで処理を進める。
- 上記操作実行後、以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

エラー処理後に以下のログが出力される。

- 以下、エラーログ

[2025-03-30 15:29:01.753] ERROR http-nio-8080-exec-8 com.company.bulletinboard.action.UserCancelAction - Invalid action parameter. Expected 'usercancel', but received: null

<③ action パラメータが "usercancel" 以外 (invalid) の場合のエラーハンドリング>

- ユーザー管理画面に遷移する
- 「ユーザー作成」ボタンをクリックし、掲示板作成画面に遷移する。
- ユーザー作成画面にて、「キャンセル」ボタンをクリックする。
- Eclipseのデバッグモードになるので、151行目までステップオーバー(F6キーを押下)する。
- 151行目で、Eclipseの「変数」ウィンドウからthis → actionの値を「usercancel」から「invalid」に値を変更する。値を変更後、ステップオーバー(F6キーを押下)する。
- 160行目の「logger.error("Invalid action parameter. Expected 'usercancel', but received: " + String.valueOf(action));」の例外処理に移動するので、ステップオーバー(F6キーを押下)する。
- 163行目の「return ERROR;」の行に移動するのでステップオーバー(F6キーを押下)する。
- 上記のステップオーバー後、「OgnlRuntime.jav」ファイルの処理に移るが、「F8」キーで処理を進める。
- 上記操作実行後、以下の画面へ遷移することを確認する。

エラーが発生しました

ご不便をおかけして申し訳ありません。

トップページに戻る場合は、以下のリンクをクリックしてください。

[トップページへ](#)

エラー処理後に以下のログが出力される。

- 以下、エラーログ

[2025-03-30 15:39:58.582] ERROR http-nio-8080-exec-2 com.company.bulletinboard.action.UserCancelAction - Invalid action parameter. Expected 'usercancel', but received: invalid

以上

■単体テストNo22.1

UserListAction - キャンセルボタンクリック後の、ユーザー管理画面一覧を再読み込みする処理

■目的

処理がUserCancelActionクラスからリダイレクトされ正常に遷移し、ユーザー管理画面が表示されること。
合わせてユーザー一覧が最新の状態で表示されること。

■テスト対象行

37行目 :UserListDAO userdao = new UserListDAO();
42行目 :users = userdao.getAllUsers();
62行目 :return SUCCESS;

■期待する結果

- ・UserListDAOクラスのインスタンスが正常に生成されること。
- ・UserListDAOクラスの「getAllUsers();」メソッドが呼び出され、データベースからユーザーの全レコードを取得し、それをusersリストに格納できること。
- ・ユーザーデータ取得後、最新のデータがユーザー一覧に反映されること。

■テスト実施内容

37行目 :UserListDAO userdao = new UserListDAO();
・目的 :UserListDAOクラスの「userdao」インスタンスが正常に生成されることを確認する。
・テスト前 (ステップオーバー前)

確認内容 :
Eclipse「変数」ウィンドウの「名前」列に「userdao」の項目は表示されていない。UserListDAOクラスのインスタンスは生成されていない状態。

・テスト後 (ステップオーバー後)
確認内容 :
ステップオーバー後、Eclipse「変数」ウィンドウの「名前」列に「userdao」で、「値」列に「UserListDAO」の項目が表示される。UserListDAOクラスのuserdaoインスタンスが正常に生成された状態。

処理後に以下のログが出力される。
デバッグログ :[2025-03-30 16:05:46.633] DEBUG http-nio-8080-exec-9 com.company.bulletinboard.action.UserListAction - Fetching all users from database
[2025-03-30 16:01:56.332] DEBUG http-nio-8080-exec-9 com.company.bulletinboard.action.UserListAction - Creating UserListDAO instance

42行目 :users = userdao.getAllUsers();
・目的 :
・テスト前 (ステップオーバー前)
確認内容 :UserListDAOクラスのdaoインスタンスから「getAllUsers();」メソッドが呼び出される。
その後、データベースからユーザーの全レコードを取得しそれをusersリストに格納できることを確認する。
Eclipse「変数」ウィンドウのthis「users」の値が「null」であることを確認する。

・テスト後 (ステップオーバー後)
確認内容 : ステップオーバー後、Eclipse「変数」ウィンドウのthis「users」の値が「null」から「ArrayList<E> (id=236)」に更新されたことを確認する。
「ArrayList<E> (id=236)」階層に登録されているユーザーの情報が12件分、格納されていることを確認する。
以下は1件分の内容を抜粋。
bulletinboards ArrayList<E> (id=247)
[0] User (id=248)
auth_type 1
delete_day 2025-12-31 00:00:00 (id=261)
delete_flag 1
password n5J%v&db (id=262)
user_id 15
user_name nishioka444 (id=271)
※上記の通り、ユーザー情報が正常に取得できている状況。
※以下、デバッグログでも取得したユーザーデータの整合を確認済み1件分のみ抜粋。
デバッグログ :[2025-03-30 16:12:37.366] DEBUG http-nio-8080-exec-9 com.company.bulletinboard.action.UserListAction - User: User{user_id=15, user_name='nishioka444', password='n5J%v&db', auth_type=1, delete_flag=1, delete_day='2025-12-31 00:00:00'}rd_delete_day='2999-12-31 00:00:00'}
[2025-03-30 16:12:35.336] DEBUG http-nio-8080-exec-9 com.company.bulletinboard.action.UserListAction - Number of users retrieved: 12
[2025-03-30 16:13:17.712] DEBUG http-nio-8080-exec-9 com.company.bulletinboard.action.UserListAction - UserListAction mainProc finished[2025-03-30 16:13:17.712] DEBUG http-nio-8080-exec-9 com.company.bulletinboard.action.UserListAction - UserListAction mainProc finished

62行目 :return SUCCESS;
確認内容 : ステップオーバー後、フレームワークの処理に進みF8キー」押下後、以下の掲示板一覧が最新状態で表示されたことを確認。

ユーザー管理

[管理メニューに戻る](#)

ユーザー作成

| ユーザーID | ユーザー名 | 権限 | 操作 | |
|--------|-------------|----|--------------------|--------------------|
| 15 | nishioka444 | 1 | 編集 | 削除 |
| 24 | uehara1 | 0 | 編集 | 削除 |
| 25 | uehara23 | 0 | 編集 | 削除 |
| 26 | ueharaadmin | 1 | 編集 | 削除 |
| 30 | tesutuser1 | 1 | 編集 | 削除 |
| 31 | testuser2 | 1 | 編集 | 削除 |
| 32 | testuser3 | 1 | 編集 | 削除 |
| 33 | tesutuser4 | 0 | 編集 | 削除 |
| 34 | testuser5 | 1 | 編集 | 削除 |
| 35 | tesutuser1 | 0 | 編集 | 削除 |
| 36 | tesutuser11 | 0 | 編集 | 削除 |
| 38 | uehara10001 | 0 | 編集 | 削除 |

以上

■単体テストN22.2

UserListAction - UserListDAO のインスタンス生成エラー発生時および、getAllUsers()メソッド呼び出し後のハンドリング検証

■対象行

37行目 : UserListDAO userdao = new UserListDAO();

42行目 : users = userdao.getAllUsers();

■目的

①UserListDAOのインスタンス生成エラー発生時に、例外処理が正常に実行されることを確認する。

②getAllUsersメソッドが正常に呼ばれた後、取得したユーザーリストが`null` になった場合に、例外処理が正しく動作することを確認する。

■期待する結果

・UserListDAOクラスのインスタンが「null」だった場合に例外処理が発生し、エラー画面に遷移できること。

また、例外処理のデバックログが出力されること

・getAllUsersメソッドが正常に呼ばれた後、取得したユーザーリストが`null` になった場合にエラー画面に遷移できること。

また、例外処理のデバックログが出力されること

■テスト実施内容

<①UserListDAOのインスタンス生成エラー発生時に、例外処理が正常に実行されることを確認する>

・テスト前①(セッションIDの事前確認)

確認内容:

事前にWEBブラウザの開発者ツールの「Cookies」から JSESSIONID を確認。

ユーザー作成画面の「キャンセル」ボタンをクリック前に以下の値を確認する。

| 名前 | ▲ | 値 |
|------------|---|----------------------------------|
| JSESSIONID | | 093B785BCE12EAC81849060953CEE99C |

・テスト前②(ステップオーバー前)

確認内容:

②Eclipse「変数」ウィンドウの「名前」列に『userdao』の項目が表示されていないことを確認する (UserListDAOクラスのインスタンスは生成されていない状態)

・テスト後①(ステップオーバー後)

確認内容:

Eclipseにて37行目をステップオーバー後、Eclipse「変数」ウィンドウの「名前」列に『userdao』で、「値」列に『UserListDAO』の項目が表示される。UserListDAOクラスのuserdaoインスタンスが正常に生成された状態で以下の操作を行う。

<操作>

・Eclipseの「変数」ウィンドウのに『userdao』の値を『null』に変更する。

・値を変更「後」、F8 を押して処理を再開する。

・52行目のエラー処理{ catch (Exception e) {} }に進むので、F6キーを押下する。

・F6キーでそのまま進み「return ERROR;」の行でエラーを返す。

・「OgnlRuntime.java」の処理に進むが、F8キーを押下し、処理を進める。

・ブラウザの画面で以下のエラー処理の画面に遷移する。

・エラー画面

エラーが発生しました

ご不便をおかけして由、訳ありません。

ご利用をおかりして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

・エラーログ
[2025-03-30 16:34:27.936] ERROR http-nio-8080-exec-8 com.company.bulletinboard.action.UserListAction - Error in UserListAction mainProc
at com.company.bulletinboard.action.UserListAction.mainProc(UserListAction.java:42) [classes/:?]

※上記の通り、例外処理時のデバックログが出力されることを確認。

・テスト後②(エラー発生後の「JSESSIONID」を確認)

| 名前 | ▲ | 値 |
|------------|---|----------------------------------|
| JSESSIONID | | 093B785BCE12EAC81849060953CEE99C |

※上記の通り、テスト前の「JSESSIONID」と変化が無いことを確認済み。

<②getAllUsersメソッドが正常に呼ばれた後、取得したユーザーリストがnull` になった場合に、例外処理が正しく動作することを確認する。>

・テスト前①(セッションIDの事前確認)

確認内容:
事前にWEBブラウザの開発者ツールの「Cookies」から JSESSIONID を確認。

ユーザー作成画面の「キャンセル」ボタンをクリック前に以下の値を確認する。

| 名前 | ▲ | 値 |
|------------|---|----------------------------------|
| JSESSIONID | | 093B785BCE12EAC81849060953CEE99C |

・テスト前②(ステップオーバー前)

確認内容:
Eclipse「変数」ウィンドウの「名前」列に『userdao』の項目は表示されていない。BoardDAOクラスのインスタンスは生成されていない状態。

・テスト後①(ステップオーバー後)

確認内容:
ステップオーバー後、Eclipse「変数」ウィンドウのthis →「名前」列の『users』の「値」が『ArrayList<E> (id=288)』となり。ユーザーデータが全て取得できている状態。
※結果として、getAllUsersメソッドが正常に呼び出されている状況。

前述の状況で以下の操作を行う。

・操作

1. Eclipseの「変数」ウィンドウの「this」→「users」の値を「ArrayList<E> (id=288)」から『null』に変更する。
※NullPointerException を意図的に発生させる、
2. `F8` を押して処理を再開。
3. WEBブラウザの画面がエラー画面に遷移することを確認。

・エラー画面

エラーが発生しました

ご不便をおかけして申し訳ありません。

以下のリンクをクリックしてください。

[前のページへ戻る](#)

[一覧画面に戻る](#)

・デバッグログ

[2025-03-10 14:17:58.431] [2025-03-30 16:43:12.171] ERROR http-nio-8080-exec-9 com.company.bulletinboard.action.UserListAction - Error in UserListAction mainProc
at com.company.bulletinboard.action.UserListAction.mainProc(UserListAction.java:44) [classes/:?]

※上記の通り、デバッグログが出力される。

・テスト後②(エラー発生後の「JSESSIONID」を確認)

| 名前 | ▲ | 値 |
|------------|---|----------------------------------|
| JSESSIONID | | 093B785BCE12EAC81849060953CEE99C |

※上記の通り、テスト前の「JSESSIONID」と変化が無いことを確認済み。

以上

■単体テストNo23.1

GetThreadAction - ユーザーポータルログイン時のスレッド一覧表示と、スレッドに紐づく投稿件数取得処理の検証

■目的

- ・ユーザーポータルのログイン時に、スレッドデータの取得とその一覧が最新の状態で表示できること。
- ・上記処理後、スレッドデータに紐づく投稿件数が取得できること。また、スレッド一覧の該当スレッドに投稿件数が表示されること。

■テスト対象行

```
60行目: if (sessionUser == null) {
    → セッションユーザーの確認
107行目: PreparedStatement preparedStatement = connection.prepareStatement(sql);
    → SQLの準備
146行目: try (ResultSet resultSet = preparedStatement.executeQuery()) {
    → SQL実行と結果取得
154行目: User thread = new User();
    → スレッドオブジェクトの生成
156行目: thread.setThread_id(resultSet.getInt("thread_id"));
    → スレッドIDのセット
159行目: thread.setThread_title(resultSet.getString("thread_title"));
    → スレッドタイトルのセット
162行目: thread.setThread_delete_flag(resultSet.getInt("thread_delete_flag"));
    → 削除フラグのセット
165行目: thread.setThread_delete_day(resultSet.getString("thread_delete_day"));
    → 削除フラグのセット
169行目: threads.add(thread);
    → 削除フラグのセット
※※前述の処理後に、以下の行に処理が移る
88行目: int postCount = postDao.getPostCountByThreadId(thread.getThread_id());
    → スレッドデータ取得後に、該当のスレッドの投稿件数を取得
```

■事前準備

- ・threadsテーブルに以下のデータ(33件)が予め、登録されていることとする。※以下、1件分のみ記載

```
thread_id: 15
thread_title: 2024年7月18日のスレッド
bulletinboard_id: 34
user_id: 24
thread_delete_flag: 0
thread_delete_day: null
```

- ・posts テーブルに以下のデータ(111レコード)が予め、登録されていること。※1件分のみ記載

```
post_id: 5
post_content: 本日水曜日テスト投稿です。8月10日に編集。
thread_id: 27
user_id: 24
post_delete_flag: 0
post_delete_day: NULL
post_timestamp: 2024-08-10 08:44:03
```

■テスト実施内容

- 60行目: if (sessionUser == null) {
- ・目的: セッションが正しく設定されているかを確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

sessionUser → auth_typeの値が「0」
sessionUser → delete_dayの値が「2050-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jp9T-LH2」
sessionUser → user_idの値が「36」
sessionUser → user_nameの値が「tesutuser11」

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

・テスト後（ステップオーバー後）

確認内容：

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値：

sessionUser → auth_typeの値が「0」
sessionUser → delete_dayの値が「2050-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jp9T-LH2」
sessionUser → user_idの値が「36」
sessionUser → user_nameの値が「tesutuser11」

上記の通り、ステップオーバー後も情報が保持されており、意図した内容。期待通りの結果。

2. this → session の内容：

- sessionの値が`SessionMap`オブジェクトである。
- session内にキー`"loggedInUser"`が存在する(`session.containsKey("loggedInUser") == true`)。
- session.get("loggedInUser")の値が`sessionUser`オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

sessionUser → auth_typeの値が「0」
sessionUser → delete_dayの値が「2050-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jp9T-LH2」
sessionUser → user_idの値が「36」
sessionUser → user_nameの値が「tesutuser11」

上記の通り、クラスフィールドにsessionUser`オブジェクトと同じ内容が補完されている。正常動作。期待通り。

3. ログに以下の内容が出力されている：

[2025-04-01 11:16:52.606] INFO http-nio-8080-exec-5 com.company.bulletinboard.action.user.Portals.GetThreadAction - Session User: tesutuser11
[2025-04-01 11:16:53.399] INFO http-nio-8080-exec-5 com.company.bulletinboard.action.user.Portals.GetThreadAction - Session UserID: 36
[2025-04-01 11:16:54.125] INFO http-nio-8080-exec-5 com.company.bulletinboard.action.user.Portals.GetThreadAction - Session User_AuthType: 0

●107行目: PreparedStatement preparedStatement = connection.prepareStatement(sql);

・目的: 接続の正常性を確認する。

・テスト前（ステップオーバー前）

確認内容：

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認。

・sql → "SELECT * FROM threads"

※不正な値(NULLや予期しない構文など)が含まれていない。SQL文が事前に妥当な構文として検証されていること(エラーが発生しないこと)」

2.「connection」オブジェクトの確認。接続がクローズされていないことの確認する。

・connection → connectionオブジェクト自体が存在し、NULLで無いこと
・connection → delegate → openStatementsの値が「CopyOnWriteArrayList<E> (id=288)」
※openStatementsの値が空でなくアクセス可能な状態。
・isAutoCommit → isAutoCommitの値が「true」であること

3.接続先データベース情報の確認

・connection → delegate → origHostToConnectTo → 「"localhost" (id=291)」であること
・connection → delegate → origPortToConnectTo → ポート番号が「3306」であること

- ・connection → delegate → database → データベース名が`"bulletinboard_db(id:295)"`であること
- ・connection → delegate → user → `"root" (id=326)`であること

4. 接続有効性のログ

connection.isValid(timeout)メソッド が`true`を返しており、ログに「データベース接続は正常です。」というメッセージが表示されている。

ログ内容:[2025-04-01 11:26:49.511] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.user.Portal.GetThreadAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

・テスト後(ステップオーバー後)

確認内容:ps の生成が正しく行われ、後続処理に影響がないことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

1.「sql」変数の確認

・sql → SELECT * FROM threads

※正しいSQL 文("SELECT * FROM threads")を保持していることを確認

2. connection の有効性を確認

・preparedStatement → isClosedの値が`false`であることを確認

※ステートメントが閉じていない状態

3.「proxyResultSet」の確認

・preparedStatement → proxyResultSetの値が`null`であることを確認

4. 接続有効性のログ

connection.isValid(timeout)メソッド が`true`を返すことを確認

ログ内容:[2025-04-01 11:26:49.511] DEBUG http-nio-8080-exec-5 com.company.bulletinboard.action.user.Portal.GetThreadAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

※※ログメッセージに「データベース接続は正常です。」が出力されていることを確認。接続は有効な状態

●146行目: try (ResultSet resultSet = preparedStatement.executeQuery()) {

・目的:SQLの結果が正しく取得できているかを検証。プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

期待値: sqlの値が`SELECT * FROM threads`であること。
 preparedStatement の isClosed の値が`false`であること。
 ステップオーバー前なので、resultsの値が`null`であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

・sql → SELECT * FROM threads

※正しいSQL 文("SELECT * FROM threads")を保持していることを確認。期待通り。

2.「preparedStatement」の有効性確認

・preparedStatement → isClosed の値が`false`

※ステートメントは有効で操作可能な状態。期待通り。

3.resultset値が、`null`を確認。

・preparedStatement → delegate → resultSetの値が`null`であることを確認。期待通り。

・テスト後(ステップオーバー後)

期待値: sqlの値が`SELECT * FROM threads`であること。
 preparedStatement の isClosed の値が`false`であること。
 ステップオーバー後、resultsのcurrentRow の値が`-1`であること。また、thisRow の値が`null`であること。

確認内容:ステップオーバー後もuserに紐づけされた値に、変化が無ことを確認する

期待値:user_nameの値が`tesutuser14`であること(フォームの入力値)

- Eclipse「変数」ウィンドウで以下を確認。

・sql → SELECT * FROM threads

※正しいSQL 文("SELECT * FROM threads")を保持していることを確認。期待通り。

2.「preparedStatement」の有効性確認

- ・preparedStatement → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待通り。

3.resultset値を確認。

- ・preparedStatement → delegate → resultSet → db → valueの値が以下の値であることを確認する。

resultsのcurrentRow の値が「-1」

resultSetのthisRow の値が「null」

※補足

currentRow = -1 → これは ResultSet の初期状態。

thisRow = null → resultSetのcurrentRowで、まだ行データがセットされていない。

while (resultSet.next()) を繰り返して、currentRow が 0 以上になり、thisRow にデータが入る

続きはここから
20250401

154行目: User thread = new User();

- ・目的:userモデルの「thread」インスタンスが正常に生成されることを確認する。

・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

Eclipse「変数」ウィンドウの「名前」列に「thread」の項目は表示されていない。Userモデルのthreadインスタンスは生成されていない状態。

・テスト後(ステップオーバー後)

確認内容:

ステップオーバー後、Eclipse「変数」ウィンドウの「名前」列に「thread」で、「値」列に「User(id=218)」の項目が表示される。Userクラスのthreadインスタンスが正常に生成された状態。

- 156行目: thread.setThread_id(resultSet.getInt("thread_id"));

- ・目的:プリペAREDステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。

・テスト前(ステップオーバー前)

- ・期待値:
sql の値が、「SELECT * FROM threads」であること。
preparedStatementのisClosed の値が「false」であること。
thread_idの値が「0」であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- ・sql → SELECT * FROM threads

※正しい SQL 文(" SELECT * FROM threads")を保持していることを確認。期待通りの結果。

2.「preparedStatement」の有効性確認

- ・preparedStatement → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待値通り。

3.データがモデルインスタンスのhreadに紐づけされていないことを確認

- ・thread → thread_idの値が「0」であることを確認。

※データがモデルに紐づけられていない状態。期待通り。

・テスト後(ステップオーバー後)

- ・期待値: thread_idの値が「15」であること。

確認内容:SQLのクエリ結果がモデルのhreadインスタンスに紐づけられているかの確認

- ・thread → thread_idの値が「15」であることを確認。

※データがモデルに紐づけられていることを確認。

以下、デバッグログ

[2025-04-01 13:06:03.218] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - thread_id = 15

※期待通りの結果。

- 159行目: thread.setThread_title(resultSet.getString("thread_title"));
- ・目的: プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。
- ・テスト前 (ステップオーバー前)
- ・期待値: sql の値が、 「SELECT * FROM threads」であること。
 preparedStatementのisClosed の値が 「false」であること。
 thread_titleの値が 「null」であること。

確認内容:
-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。
1. 「sql」変数の確認
・sql → SELECT * FROM threads
 ※正しいSQL 文 (" SELECT * FROM threads")を保持していることを確認。期待通りの結果。

2. 「preparedStatement」の有効性確認
・preparedStatement → isClosed の値が 「false」
 ※ステートメントは有効で操作可能な状態。期待値通り。

3. データがモデルインスタンスのhreadに紐づけされていないことを確認
・thread → thread_titleの値が 「null」であることを確認。
 ※データがモデルに紐づけられていない状態。期待通り。

・テスト後 (ステップオーバー後)
・期待値: thread_titleの値が 「2024年7月18日のスレッド」であること。
確認内容: SQLのクエリ結果がモデルのhreadインスタンスに紐づけられているかの確認
・thread_titleの値が 「2024年7月18日のスレッド」であることを確認。
 ※データがモデルに紐づけられていることを確認。
 以下、デバッグログ
 [2025-04-01 13:12:49.064] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - thread_title = 2024年7月18日のスレッド
 ※期待通りの結果。

- 162行目: thread.setThread_delete_flag(resultSet.getInt("thread_delete_flag"));
- ・目的: プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。
- ・テスト前 (ステップオーバー前)
- ・期待値: sql の値が、 「SELECT * FROM threads」であること。
 preparedStatementのisClosed の値が 「false」であること。
 thread_delete_flagの値が 「0」であること。

確認内容:
-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。
1. 「sql」変数の確認
・sql → SELECT * FROM threads
 ※正しいSQL 文 (" SELECT * FROM threads")を保持していることを確認。期待通りの結果。

2. 「preparedStatement」の有効性確認
・preparedStatement → isClosed の値が 「false」
 ※ステートメントは有効で操作可能な状態。期待値通り。

3. データがモデルインスタンスのhreadに紐づけされていないことを確認
・thread → thread_delete_flagの値が 「0」であることを確認。
 ※データがモデルに紐づけられていない状態。期待通り。

・テスト後 (ステップオーバー後)
・期待値: thread_delete_flagの値が 「0」であること。

確認内容:SQLのクエリ結果がモデルのthreadインスタンスに紐づけられているかの確認

- thread_delete_flagの値が「0」であることを確認。
※データがモデルに紐づけられていることを確認。
以下、デバッグログ
[2025-04-01 13:20:10.407] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - thread_delete_flag = 0
- ※期待通りの結果。

- 165行目: thread.setThread_delete_day(resultSet.getString("thread_delete_day"));
- 目的:プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。
- テスト前(ステップオーバー前)
- 期待値:
 - sql の値が、「SELECT * FROM threads」であること。
 - preparedStatementのisClosed の値が「false」であること。
 - thread_delete_dayの値が「null」であること。

- 確認内容:
- Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。
 - 1.「sql」変数の確認
 - sql → SELECT * FROM threads
※正しいSQL 文(" SELECT * FROM threads")を保持していることを確認。期待通りの結果。

- 2.「preparedStatement」の有効性確認
- preparedStatement → isClosed の値が「false」
※ステートメントは有効で操作可能な状態。期待値通り。

- 3.データがモデルインスタンスのhreadに紐づけされていないことを確認
- thread → thread_delete_dayの値が「null」であることを確認。
※データがモデルに紐づけられていない状態。期待通り。

- テスト後(ステップオーバー後)
- 期待値: thread_delete_dayの値が「null」であること。

確認内容:SQLのクエリ結果がモデルのthreadインスタンスに紐づけられているかの確認

- thread_delete_dayの値が「null」であることを確認。
※そもそもデータベース上では値が「null」なので問題無し。
以下、デバッグログ
[2025-04-01 13:22:24.268] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - thread_delete_day = null
- ※期待通りの結果。

- 169行目: threads.add(thread);
 - 目的:threadオブジェクトに入っているスレッドデータをhreadリストに追加されたことを確認する。
 - テスト前(ステップオーバー前)
 - 期待値:Eclipseの「変数」ウィンドウのthreadsに値が「ArrayList<E> (id=217)」の状態、リストの初期化はされているが、値は入っていない状態を確認する。
- 確認内容:this → threadsに値の階層が存在しない。つまりデータが入っていないことを確認

- テスト後(ステップオーバー後)
 - 期待値:threadsリストにスレッドデータが追加されていることを確認する
- 確認内容:Eclipseの「変数」ウィンドウのthis → threads → [0]の階層を開き以下のデータが存在することを確認する
- | | |
|---------|-----------------------|
| threads | ArrayList<E> (id=217) |
| | [0] |
| | User (id=218) |
| | thread_delete_ |
| | thread_delete_1 |
| | thread_id |
| | thread_title |
- 0

15

2024年7月18日のスレッド (id=267)

※上記の通り、スレッドデータ件分がthreadsリストに保存できていることを確認。
※以降、取得したデータ文の処理(3件)がループ処理によって続くが、ここでは割愛する。但し、ループ処理による全てのデータ取得は確認済み。

・デバックログ
[2025-04-01 13:53:43.613] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - スレッドの総数: 33
[2025-04-01 13:53:45.418] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Resources are about to be closed: HikariProxyConnection@289319943 wrapping com.mysql.cj.jdbc.ConnectionImpl@7e277228
※上記の通り、全スレッドデータ33件分の取得が完了したことを確認済み。

88行目: int postCount = postDao.getPostCountByThreadId(thread.getThread_id());
・補足と処理の流れ:
メインメソッド(mainProc())から呼び出された「getAllThreads()」の処理が正常終了し、その結果が呼び出し元のメインメソッドmainProc()の77行目「getAllThreads();」に返ってくる。
その後、メインメソッド内の87行目の「for (User thread : threads) {」「各スレッドに対して投稿件数を取得し、それをスレッドオブジェクトに格納する。」に処理が移る。
ここでは、「getAllThreads();」で取得したスレッドデータを元に、そのスレッドデータに紐づく投稿件数を取得する。

- ・目的: スレッドデータに紐づく、投稿件数を取得する
- ・テスト前(ステップオーバー前)
- ・期待値: スレッドデータに紐づく投稿件数をカウントできること

確認内容:
①Eclipseの「変数」ウィンドウのthis → 「threads」リストにスレッドデータ33件分が存在することを確認する。

```
threads      ArrayList<E> (id=217)
[0]          User (id=218)
[1]          User (id=299)
[2]          User (id=318)
[3]          User (id=352)
[4]          User (id=372)
[5]          User (id=394)
[6]          User (id=412)
[7]          User (id=433)
[8]          User (id=455)
[9]          User (id=476)
[10]         User (id=498)
[11]         User (id=520)
[12]         User (id=542)
[13]         User (id=561)
[14]         User (id=580)
[15]         User (id=600)
[16]         User (id=616)
[17]         User (id=639)
[18]         User (id=663)
[19]         User (id=685)
[20]         User (id=707)
[21]         User (id=730)
[22]         User (id=751)
[23]         User (id=772)
[24]         User (id=794)
[25]         User (id=816)
[26]         User (id=837)
[27]         User (id=858)
[28]         User (id=879)
[29]         User (id=900)
[30]         User (id=921)
[31]         User (id=943)
[32]         User (id=964)
```

※上記の通り、threadリストにデータが存在する。期待通り。

- ②Eclipseの「変数」ウィンドウのthis → threads → [0] →「post_count」の値が「0」であることを確認。
- ③Eclipseの「変数」ウィンドウのthis → 名前列に「postDao」の項目が存在し、PostDAOクラスのインスタンとして生成されていることを確認する。

- ・テスト後(ステップオーバー後)
- ・期待値:ステップオーバー後、スレッドに紐づく投稿のカウンタ処理が実行され、カウンタ数が正常に取得できること。
- 確認内容:Eclipseの「変数」ウィンドウのthis → threads → [0]の階層を開き、「post_count」の値が「0」から「8」に変化したことを確認する。

また、以下のデバックログが出力されることを確認する。
[2025-04-01 14:30:34.503] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 15: 8
※上記のログの通り、スレッドID「15」に紐づく投稿の件数は「8」件だということが確認できる。

※以降、全スレッド(33件分)に紐づく投稿件数のカウンタを確認する。Eclipseの「変数」ウィンドウの内容は割愛。

・デバックログ

以下の通り、スレッド紐づく投稿件数取得処理が正常に実行されたことが裏付けられる。

[2025-04-01 14:30:34.503] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 15: 8
[2025-04-01 14:33:42.154] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 16: 0
[2025-04-01 14:35:30.437] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 17: 0
[2025-04-01 14:35:38.947] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 18: 0
[2025-04-01 14:35:47.394] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 19: 0
[2025-04-01 14:35:55.779] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 20: 5
[2025-04-01 14:36:05.158] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 21: 0
[2025-04-01 14:36:14.504] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 22: 0
[2025-04-01 14:36:24.651] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 23: 0
[2025-04-01 14:36:32.585] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 24: 0
[2025-04-01 14:36:41.956] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 25: 0
[2025-04-01 14:36:51.541] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 26: 7
[2025-04-01 14:37:01.437] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 27: 78
[2025-04-01 14:37:10.460] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 28: 5
[2025-04-01 14:37:18.987] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 29: 0
[2025-04-01 14:37:27.363] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 30: 2
[2025-04-01 14:37:35.770] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 31: 0
[2025-04-01 14:37:43.030] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 32: 0
[2025-04-01 14:37:52.577] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 33: 0
[2025-04-01 14:38:01.336] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 34: 0
[2025-04-01 14:38:08.555] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 35: 0
[2025-04-01 14:38:18.239] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 36: 0
[2025-04-01 14:38:27.172] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 37: 0
[2025-04-01 14:38:35.897] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 38: 0
[2025-04-01 14:38:44.376] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 39: 0
[2025-04-01 14:38:52.860] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 40: 0
[2025-04-01 14:39:02.816] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 41: 0
[2025-04-01 14:39:11.451] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 42: 0
[2025-04-01 14:39:20.199] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 43: 4
[2025-04-01 14:39:28.759] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 44: 0
[2025-04-01 14:39:37.336] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 45: 1
[2025-04-01 14:39:45.837] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 46: 0
[2025-04-01 14:39:54.656] DEBUG http-nio-8080-exec-7 com.company.bulletinboard.action.user.Portal.GetThreadAction - Set post count for thread_id 47: 1

・ユーザーポータル画面上での確認

以下の通り、画面でもスレッドの一覧とそれに紐づく投稿件数の取得結果が正常に表示された。

<ユーザーポータル画面のスレッド一覧

2024年7月18日のスレッド (投稿件数:8件)
2024年7月18日のスレッド4 ※9月27日のに編集 (投稿件数:0件)
2024年7月18日のスレッド5 (投稿件数:0件)
2024年7月18日のスレッド6 (投稿件数:0件)
2024年7月18日のスレッド7 (投稿件数:0件)
2024年7月19日のスレッド (投稿件数:5件)
2024年7月22日のスレッド (投稿件数:0件)
2024年7月22日のスレッド2 (投稿件数:0件)
2024年7月25日のスレッド (投稿件数:0件)
2024年7月25日のスレッドtest (投稿件数:0件)
2024年7月27日のスレッド (投稿件数:0件)
2024/07/27テストスレッド (投稿件数:7件)
2024/07/28日曜日 ※編集テスト (投稿件数:78件)
8月11日のテストです ※19:27 (投稿件数:5件)
8月11日のテスト2 (投稿件数:0件)
08/30 テストスレッド (投稿件数:2件)
08/30 テストスレッドその2 (投稿件数:0件)
08/30 テストスレッドその3 (投稿件数:0件)
2024/08/30 テストです (投稿件数:0件)
2024年9月1日のスレッド (投稿件数:0件)
2024/09/01日曜日その2 (投稿件数:0件)
2024/09/01日曜日その3 (投稿件数:0件)
2024/09/01日曜日その4 (投稿件数:0件)
2024/09/01日曜日その5 (投稿件数:0件)
2024/09/01日曜日その6 (投稿件数:0件)
2024/09/01日曜日その7 (投稿件数:0件)
2024/09/01日曜日その8 (投稿件数:0件)
2024/09/01日曜日その9 (投稿件数:0件)
xxのテストスレッド09/01 (投稿件数:4件)
xxのテストスレッド09/01 その2 (投稿件数:0件)
2024/09/10 月曜日 (投稿件数:1件)
2024/09/20金曜日 (投稿件数:0件)
本日はノー残業デー (投稿件数:1件)

以上

■単体テストN23.2

■目的

- ・ユーザー管理画面で、ユーザーが削除できることを確認する。
- ・上記処理後、「SUCCESS」を返すことを確認する。

■テスト対象行

```
60行目 :if (sessionUser == null) {
106行目 :PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
115行目 :ps.setInt(1, id);
121行目 :return SUCCESS;
```

■事前準備

※掲示板テーブルに以下のデータが予め、登録されていることとする。

```
user_id:41
user_name:tesutuser15
password:aE4c9m+h
auth_type:1
delete_flag: 1
delete_day:2999-12-31 00:00:00.000000
```

■削除処理の内容

```
<削除内容>
ユーザー名 :tesutuser15
パスワード:aE4c9m+h
権限:管理者権限
ユーザー削除フラグ:ユーザー削除フラグ
削除日:2999-12-31 00:00:00
```

■テスト実施内容

●56行目 :if (sessionUser == null) {

- ・目的:セッションが正しく設定されているかを確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。sessionUserの以下の項目が、意図した値かどうかを確認する。

```
sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「JP9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」
```

※テスト用のユーザー情報でログインし、sessionUser変数に保存されている。意図した内容。

- ・テスト後(ステップオーバー後)

確認内容:

- Eclipse「変数」ウィンドウで以下を確認。

1. sessionUserの値:

```
sessionUser → auth_typeの値が「1」
sessionUser → delete_dayの値が「2999-12-31 00:00:00」
```

sessionUser → delete_flagの値が「0」
sessionUser → passwordの値が「jP9T-LH2」
sessionUser → user_idの値が「30」
sessionUser → user_nameの値が「tesutuser1」
上記の通り、ステップオーバー後も情報が保持されており、意図した内容。期待通りの結果。

2. this → session の内容:

- sessionの値が`SessionMap`オブジェクトである。
- session内にキー`"loggedInUser"`が存在する(`session.containsKey("loggedInUser") == true`)。
- session.get("loggedInUser")の値が`sessionUser`オブジェクトと一致する。

this → session → [0] → value (Userオブジェクト) の以下の内容を確認

auth_type: 1
delete_day: "2999-12-31 00:00:00"
delete_flag: 0
password: "jP9T-LH2"
user_id: 30
user_name: "tesutuser1"

上記の通り、クラスフィールドにsessionUser`オブジェクトと同じ内容が補完されている。正常動作。期待通り。

3. ログに以下の内容が出力されている:

ログ内容:[2025-03-30 11:46:51.826] INFO http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - Session User: tesutuser1
[2025-03-30 11:46:52.975] INFO http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - Session UserID: 30
[2025-03-30 11:46:53.728] INFO http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - Session User_AuthType: 1

●112行目 :PreparedStatement ps = connection.prepareStatement(sql);

- ・目的:接続の正常性を確認する。
- ・テスト前(ステップオーバー前)

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認。

・sql → "DELETE FROM users WHERE user_id = ?"

※不正な値(NULLや予期しない構文など)が含まれていない。SQL文が事前に妥当な構文として検証されていること(エラーが発生しないこと)

2.「connection」オブジェクトの確認。接続がクローズされていないことの確認する。

- ・connection → connectionオブジェクト自体が存在し、NULLで無いこと
- ・connection → delegate → openStatementsの値が「CopyOnWriteArrayList<E> (id=290)」
※openStatementsの値が空でなくアクセス可能な状態。
- ・isAutoCommit → isAutoCommitの値が「true」であること

3.接続先データベース情報の確認

- ・connection → delegate → origHostToConnectTo → 「"localhost" (id=297)」であること
- ・connection → delegate → origPortToConnectTo → ポート番号が「3306」であること
- ・connection → delegate → database → データベース名が「"bulletinboard_db" (id=248)」であること
- ・connection → delegate → user → 「"root" (id=355)」であること

4.接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返しており、ログに「データベース接続は正常です。」というメッセージが表示されている。

[2025-03-30 11:57:05.189] DEBUG http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

- ・テスト後(ステップオーバー後)

確認内容:ps の生成が正しく行われ、後続処理に影響がないことを確認する

- Eclipse「変数」ウィンドウで以下を確認。

1.「sql」変数の確認

・sql → DELETE FROM users WHERE user_id = ?

※正しいSQL 文("DELETE FROM users WHERE user_id = ?")を保持していることを確認

2. connection の有効性を確認

- ps → isClosedの値が「false」であることを確認
 - ※ステートメントが閉じていない状態

3.「proxyResultSet」の確認

- preparedStatement → proxyResultSetの値が「null」であることを確認

4. 接続有効性のログ

connection.isValid(timeout)メソッド が「true」を返すことを確認

ログ内容:[2025-03-30 11:57:05.189] DEBUG http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - データベース接続は正常です。ホスト: localhost, ポート: 3306, データベース: bulletinboard_db

※※ログメッセージに「データベース接続は正常です。」が出力されていることを確認。接続は有効な状態

- 115行目 :ps.setInt(1, id);
- 目的: プリペアドステートメントの状態やクエリの実行前に問題がないかを確認する。後続の処理に進む準備ができていることを確認する。
- テスト前(ステップオーバー前)

期待値: sqlの値が「DELETE FROM users WHERE user_id = ?」であること。
ps の isClosed の値が「false」であること。
bulletinboard_idの値が「55」であること。

確認内容:

-Eclipse「変数」ウィンドウで以下を確認。以下のような値が正しく設定されていることを確認。

1.「sql」変数の確認

- sql → DELETE FROM users WHERE user_id = ?
 - ※正しいSQL 文("DELETE FROM users WHERE user_id = ?")を保持していることを確認。期待通り。

2.「ps」の有効性確認

- ps → isClosed の値が「false」
 - ※ステートメントは有効で操作可能な状態。期待通り。

3.削除対象のユーザーIDの値が、thisのidに紐づけられているかの確認

- this → id →idの値が「41」であることを確認。期待通り。

- テスト後(ステップオーバー後)

確認内容: ステップオーバー後もid値に、変化が無ことを確認する

期待値: idの値が「41」であること。

- Eclipse「変数」ウィンドウで以下を確認。

- this → 41 →idの値が「41」であることを確認。

※ステップオーバー前と変わりが無いことを確認。期待通り。
以下、デバッグログ

[2025-03-30 12:03:25.047] DEBUG http-nio-8080-exec-10 com.company.bulletinboard.action.admin.user.DeleteUserAction - user_id: = 41

[2025-03-30 12:11:59.388] INFO http-nio-8080-exec-10 com.company.bulletinboard.interceptor.BaseAction - 処理終了 :com.company.bulletinboard.action.admin.user.DeleteUserAction

- 213行目 :return SUCCESS;

※return SUCCESS;の部分については、アプリケーション全体の統合テストや、インターセプターの動作確認の範囲に含められる為、範囲外とする。

- 削除処理の正常性確認

削除対象がテーブルから正常に削除されていることを確認済み。また、処理終了のログの出力されているので、問題なし。

mysql> select * from users;

| user_id | user_name | password | auth_type | delete_flag | delete_day |
|---------|-------------|----------|-----------|-------------|----------------------------|
| 15 | nishioka444 | n5J%v&db | 1 | 1 | 2025-12-31 00:00:00.000000 |

| | | | | | | | | | | | | |
|--|----|--|--------------|--|----------|--|---|--|---|--|----------------------------|--|
| | 24 | | xxxx1 | | Q7iYxLP! | | 0 | | 0 | | 2999-12-31 00:00:00.000000 | |
| | 25 | | xxxx23 | | xxxx23 | | 0 | | 0 | | 2999-12-31 00:00:00.000000 | |
| | 26 | | xxxxadmin | | zE4c9m+q | | 1 | | 1 | | 2999-12-31 00:00:00.000000 | |
| | 30 | | tesutuser1 | | jP9T-LH2 | | 1 | | 0 | | 2999-12-31 00:00:00.000000 | |
| | 31 | | testuser2 | | test | | 1 | | 1 | | 2999-12-31 00:00:00.000000 | |
| | 32 | | testuser3 | | test | | 1 | | 0 | | 2999-12-31 00:00:00.000000 | |
| | 33 | | tesutuser4 | | test | | 0 | | 0 | | 2999-12-31 00:00:00.000000 | |
| | 34 | | testuser5 | | test | | 1 | | 0 | | 2999-12-31 00:00:00.000000 | |
| | 35 | | tesutuser1 | | zE4clm+q | | 0 | | 0 | | 2050-12-31 00:00:00.000000 | |
| | 36 | | tesutuser11 | | jP9T-LH2 | | 0 | | 0 | | 2050-12-31 00:00:00.000000 | |
| | 38 | | xxxx10001 | | zE4c9m+q | | 0 | | 0 | | 2050-12-31 00:00:00.000000 | |
| | 40 | | testuser0328 | | jP9T-LH2 | | 1 | | 0 | | 2050-12-31 00:00:00.000000 | |

※上記の通り削除対象のuser_id「41」がテーブル上から削除されている。

以上