# 総合開発演習

# 総合開発演習の流れ

- チームでひとつのシステムを作り上げる。

  要求仕様と設計のヒントをもとに、チームでシステムを作り上げてください。

- システムを作り上げる中で技術や仕事の進め方を身につける。

  仕様はかなり難しく、スケジュールもタイトです。効率よく作業しないとプロジェクトは失敗に終わります。チーム全員で協力してプロジェクトを成功に導いてください。

- 研修の成果を最後に会社の上司や先輩の前で発表する。

  最終日は成果発表会です。会社の上司や先輩が大勢観覧に来ます。成果を発表するためには、成果を具体的に示す必要があります。

  「遅れた原因は何か？」「品質が低下した原因は何か？」「なぜそのような判断をしたのか？」などです。必ずその日に実施したことを残してください。

# 作成するシステム

- 高速バスの予約システム

- サブシステム

  利用者サブシステム

  　一般の人がバスの予約を行うシステム

  運用管理サブシステム

  　管理者がサービスを維持管理、運用を行うシステム

- 提供される情報

  要求は「要求一覧」および「業務フロー」を参照してください。

  仕様は「画面遷移図」と「画面設計書」に一部示されています。

  一部は仕様から自分たちで考えて作成する必要があります。

  色々な予約サイトを参考にして、使いやすいシステムを提案して作りましょう。

# 役割

- 受講者の役割

  開発チームのSEを担当します。各チームにはチームリーダーが1名います。

- 講師の役割

  お客様（仕様調整決定権を持つ）

  配られる要求や仕様はあいまいな点があります。また、かなりタイトなスケジュールであるため、工数削減のために仕様調整が必要になるかもしれません。その相談役としてお客様とコミュニケーションしてください。

  開発チームのマネージャ

  チームリーダーはマネージャに毎日状況を報告する必要があります。報告するタイミングはその日の終礼前後です。報告が間に合わない場合は、翌朝になることを報告しなければなりません。

  レビューア

  作成した設計書、ソースコードおよびテスト仕様書は必ずレビューする必要があります。何をレビューしてほしいかは、必ず電子ファイルに一覧を作成してから依頼してください。

  講師

  技術的な相談役です。何か技術的な質問があれば質問してください。

# 開発プロセス

- インクリメンタル開発

  総合開発演習はインクリメンタル開発をベースに開発していきます。

  本番環境の構築は講師により実施されるため、本番への配布開始タイミングは講師の指示に従ってください。

  インクリメンタル開発はメンバーの経験値が向上したり、PDCAサイクルを回してプロセスを洗練できます。反面、テストの工数が大きくかかります。

# 各イテレーションの作業

設計～配布の単位をイテレーションといいます。総合開発演習は３回のイテレーションで実施します。

第1イテレーションでは作成すべき成果物は少なく、第2、第3イテレーションと進んでいくにつれて、成果物が増えます。

| 成果物 | 第1イテレーション | 第2イテレーション | 第3イテレーション |
|---|---|---|---|
| 画面仕様 | 提供 | 提供 | ● |
| テーブル設計 | 提供 | ● | ● |
| 設計画面遷移図 | 提供 | ● | ● |
| 画面設計 | 提供 | ● | ● |
| 実装 | ● | ● | ● |
| 単体テスト仕様書 | ● | ● | ● |
| システムテスト仕様書 | 提供 | ● | ● |
| テスト実施 | ● | ● | ● |

# 各イテレーションで開発する機能の配分

- 前提

  既に完成している機能：【運用管理者】ログインを含む認証、バス一覧、バス登録

  ※デザインは一切設定されていないため、各チームで作成してください。路線一覧画面がまだないため、ログイン後の画面は仮でバス一覧になっています。

- 第1イテレーション（2.5日程度を目安）

  開発が必要な機能：　【運用管理者】バス削除、路線一覧、路線登録、路線削除

- 第2イテレーション（5日程度を目役）

  開発が必要な機能：　【利用者】トップ（路線検索）～予約完了

  【運用管理者】路線変更、会員一覧、会員予約状況

- 第3イテレーション（6.5日程度を目安）

  開発が必要な機能：　【利用者】会員登録、ログイン、予約一覧、予約キャンセル

  【その他】仕様に書かれていない要求

# WBSのヒント（要求理解・設計フェーズ）

- 要求理解

  要求一覧や画面設計書をもとに、作り上げるシステムの全体像を理解してください。不明点があれば、マネージャや顧客に確認してください。
- タスク分割と担当者の割り当て

  タスクを洗い出します。また、工数を見積もります。

  担当者に割り当てて、ガントチャートを作成します。
- ガントチャートの作成

  見積りをしてガントチャートにします。
- 設計作業（作成した成果物は全てcommonプロジェクトに保存）

  画面仕様の決定

  ユーザーの使い勝手を考慮して細かい画面仕様の変更を提案したり、チェック仕様を決めたりします。また、メッセージ一覧も入力してください。なお、必須チェック、文字種チェックは項目定義でわかるため、チェック仕様に記述する必要はありません。

  テーブル設計

  ER図作成、テーブル定義書作成、DDLの作成。DDLのSQL文はcommonプロジェクトのddl.sqlに、テストデータのSQL文はcommonプロジェクトのdata.sqlに記述してください。

  画面設計

  チェック仕様の作成、設計画面遷移図の作成、利用SQLの作成

  設計レビュー

  出来上がったものからマネージャーや顧客にレビューしてください。

# WBSのヒント（実装、テストフェーズ）

- プログラムの作成

  プログラムを作成します。

  出来上がった機能単位でマネージャーにレビューしてください。

  デザインは各チームで作成してください。
- 単体テスト仕様書の作成

  単体テスト仕様書を作成します。出来上がった機能単位でマネージャーにレビューしてください。
- テストデータの作成

  システムテスト用のテストデータを用意します。
- テスト仕様書（システムテスト）の作成

  システムテストの仕様書を作成します。出来上がったらマネージャーにレビューしてください。
- システムテストの実施

  システムテストを実施して不具合が出たら、その内容を必ず障害一覧に残します。一通りテストが完了したらプログラムを修正して再テストします。

  障害一覧を残しておかないと、システムの品質が向上していることを示すことができません。必ず残すようにしてください。

# 仕様調整

- 要求の調整

  プロジェクトの進捗によっては、要求の調整が必要になるかもしれません。ただし、調整には「根拠」が必要です。くれぐれも、感覚で提案しないようにしてください。

  根拠を作るには、計画と実績を残し、未来を予測する必要があります。

- 仕様の調整

  仕様は「画面設計書（利用者サブシステム）」と「画面設計書（運用管理者サブシステム）」で示されています。この仕様通りに完成させることが目的ではありません。コストをかけずにもっと良い仕様に出来れば、お客様に提案しましょう。逆にコスト的に抑える必要がある場合には、代替案を示しましょう。

  調整は要求のそれと同様に、「根拠」が必要です。くれぐれも、感覚で提案しないようにしてください。

# 1日の流れ

- 朝会（10分程度）

  連絡事項の共有

  目標の共有

  前日の振り返り（KPT）の確認

- 実務

  メンバーはリーダーに対して適宜報連相を実施

  リーダーは仕様確認、仕様変更、計画変更が必要な際は講師に相談すること。

- 夕会（30分程度）

  進捗の共有

  振り返り（KPT）

  講師への報告

# KPT

KPTは振り返りをまとめる手法です。 Keep、Problem、Tryで項目をあげていきます。

- KPTのポイント

  Problemは具体的な事象を挙げること。

  Tryは具体的な行動を挙げること。

  Tryが成功したらKeepに移動すること。

  TryとKeepには発生日を付加すること。

  TryとKeepは前日分を翌日にコピーしてから追加すること。

- 作業の仕方

  夕会で各人がProblemを洗い出し、その内容を全員で共有します。
  Problemの原因と対策を全員で議論してTryとして記載します。

  リーダーはマネージャーに進捗と合わせてKPTを報告します。

  翌日の朝会ではKPTの内容を確認し、忘れないようにします。

# 講師への依頼

講師１名が最大で4チームを担当します。レビューなどが発生する時期は、非常に混雑します。そのため、講師に報連相する際は、下記のルールを守ってください。

- 講師の机に予約票を用意します。これに記入して予約してください。
- 予約票には報連相したい時間帯とチーム名、名前、内容を記載します。
- 1度に予約できる時間帯は20分を上限とします。
- 誰も予約していない時間帯であれば、すぐに報連相しても構いません。

# ルール①

- 報連相は最重要。

  定期的にリーダやマネージャに「報告」すること。

  チーム内で「連絡」しあうこと。

  うまくいかないときは、早目にリーダに「相談」すること。それでも解決できない場合は講師に「相談」すること。

  とにかく、考え込まずにアウトプット！早め早めに報連相！

- 必ず予め計画を立て、進捗管理すること。

  計画が出来ないと、うまくいっているのかどうかすら、わからなくなります。更に最終発表で具体的に何をやったのかが示せなくなります。

- チーム間の情報共有はNG

  実際の業務では、同じ機能を複数のチームで開発することはありません。他のチームはいないものとして行動してください。

# ルール②

- 最終的に全ての種類のタスクを全員が1度は経験するように割り振ります。

  設計画面遷移図、コントローラ、JSP、マッパー、単体テスト、システムテストを一通り経験すること。

- チーム内でのサポート（積極的にやってよいこと）

  問題が発生したとき、その内容を伝える。

  考え方、やり方、情報源を教える。

- チーム内でのサポート（禁止事項）

  代わりに成果物を作成する。

  講師への相談なしにタスクの担当者を変更する。

- その他

  書かれていないこと、わからないこと、悩んでいること、うまくいかないことがあれば、**積極的に講師に相談してください。**

# 技術的なヒント

- サロゲートキーによるテーブル設計にするため、一意性の制約はUNIQUE INDEXを利用してください。

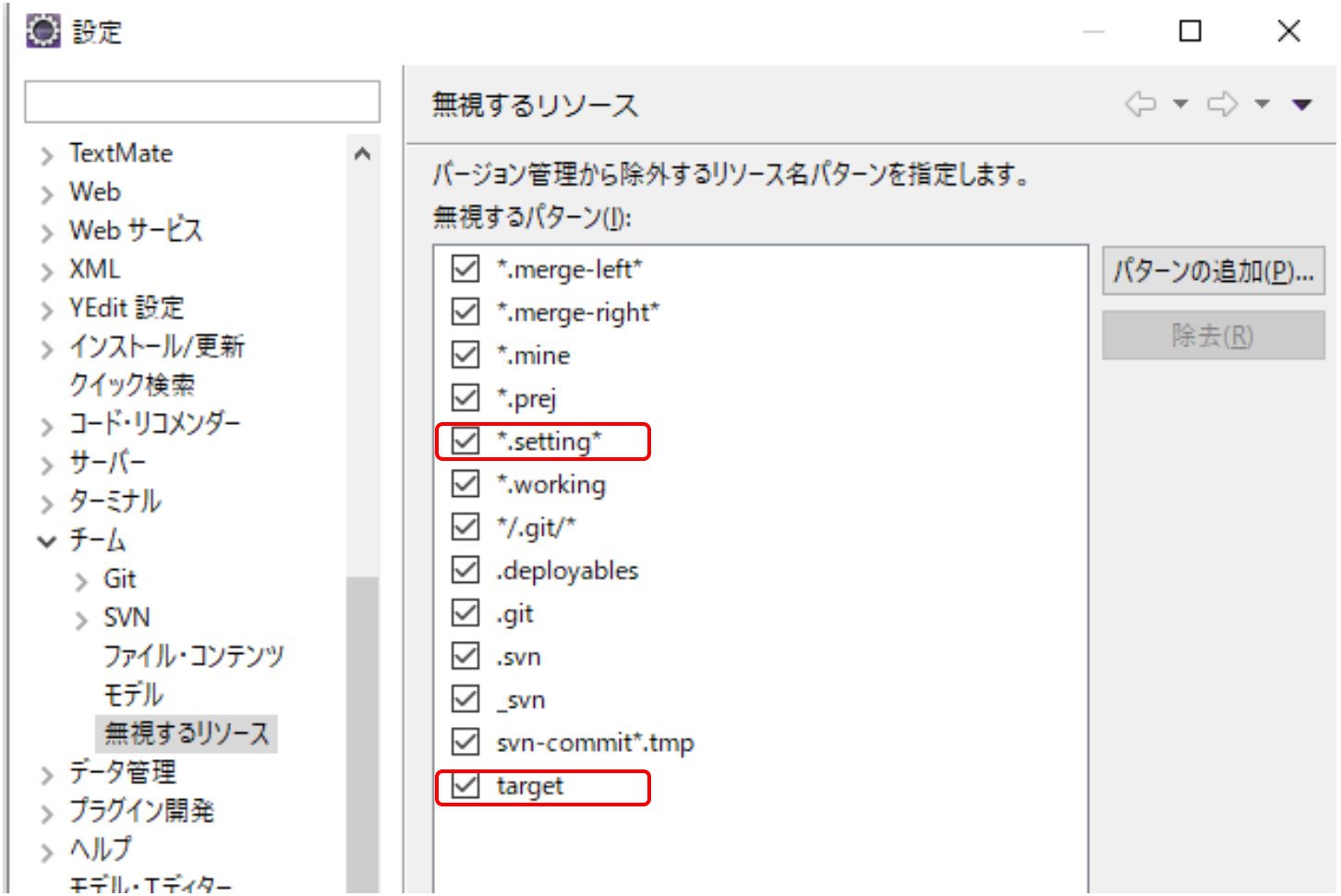
- 一意性の制約によるチェックは、どちらかによって実現してください。

  １．バリデータクラスを作成してチェックします。ただし、この方式の場合はほぼ同時に複数のユーザーが重複するデータを登録しようとしたとき、システムエラーになります。バスのナンバーはこの方式でチェックしています。

  ２．データを追加更新する処理で例外をキャッチして判定します。この方式は完璧に制御できますが、処理が難しくなります。

# EclipseのSubversion設定

[ウィンドウ]-[設定]を開き、[チーム]-[SVN]-[無視するリソース]を下記のように設定します。ワークスペースを作成したら、必ずこの設定をしてください。



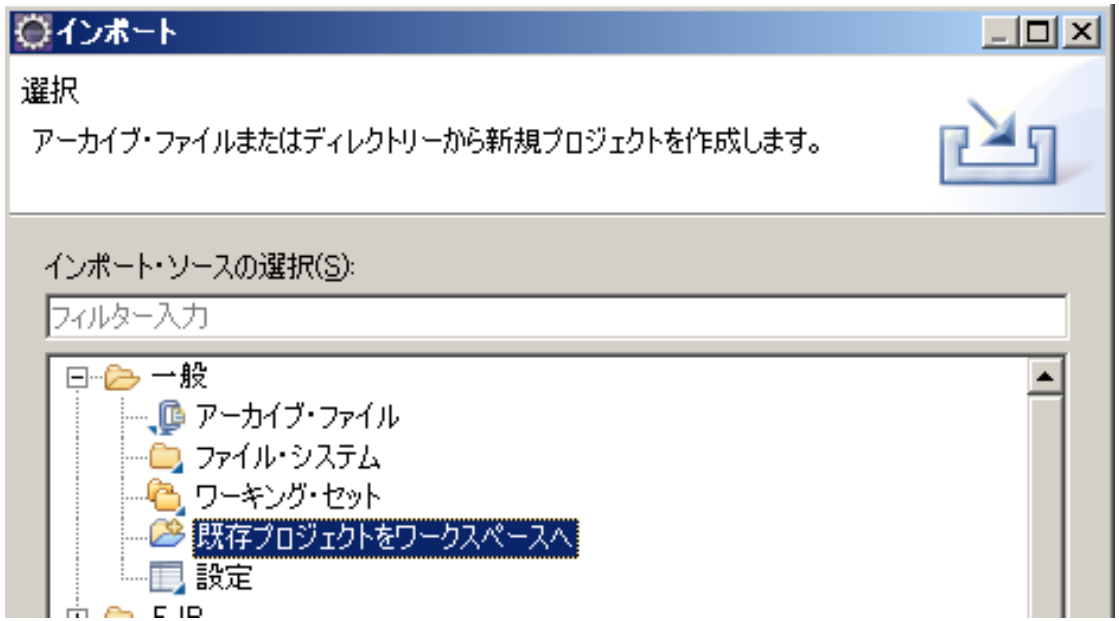
# プロジェクトの作成方法

ここからの作業は**各チーム1名だけ実施**すればよいです。他のメンバーはSVNからチェックアウトしてプロジェクトを作成します。

- ワークスペースに「../workspace_bus_startup」を指定してEclipseを起動します。
- Eclipseプロジェクトのインポート

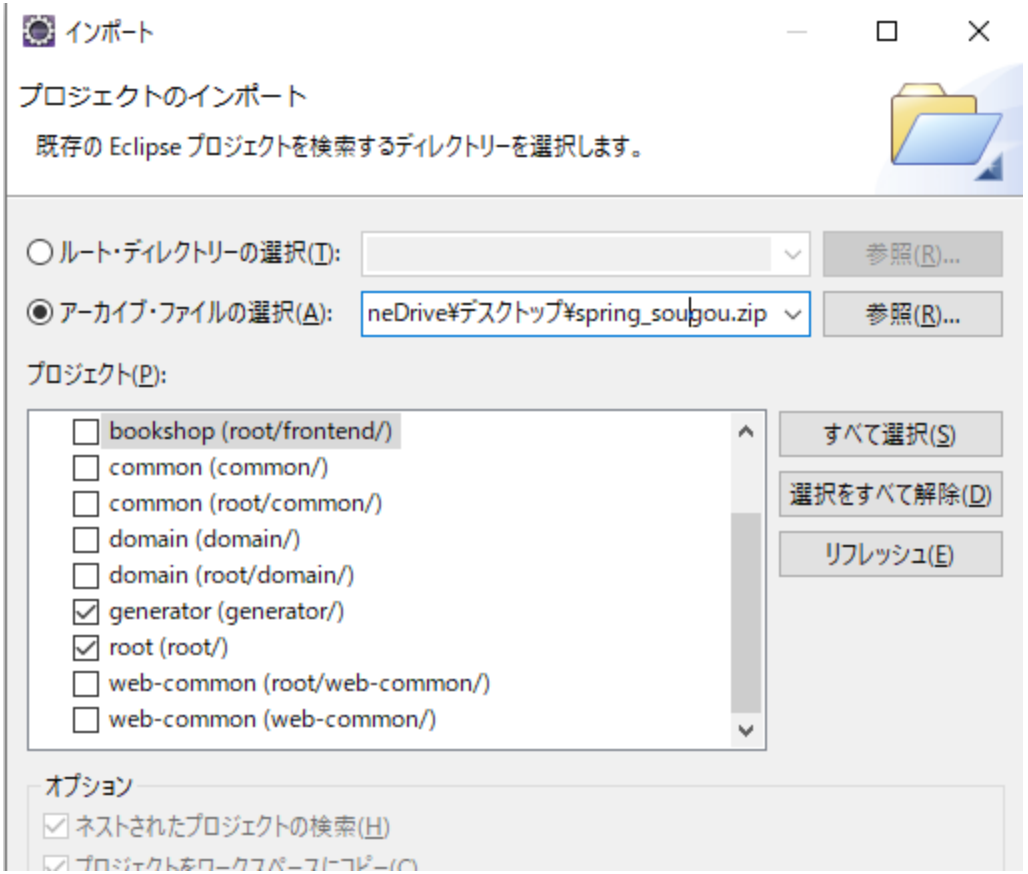
  Eclipseのメニューから、[ファイル]-[インポート]を選択します。

  一般の中の「既存プロジェクトをワークスペースへ」を選択して次へボタンを押下します。



# プロジェクトの作成方法

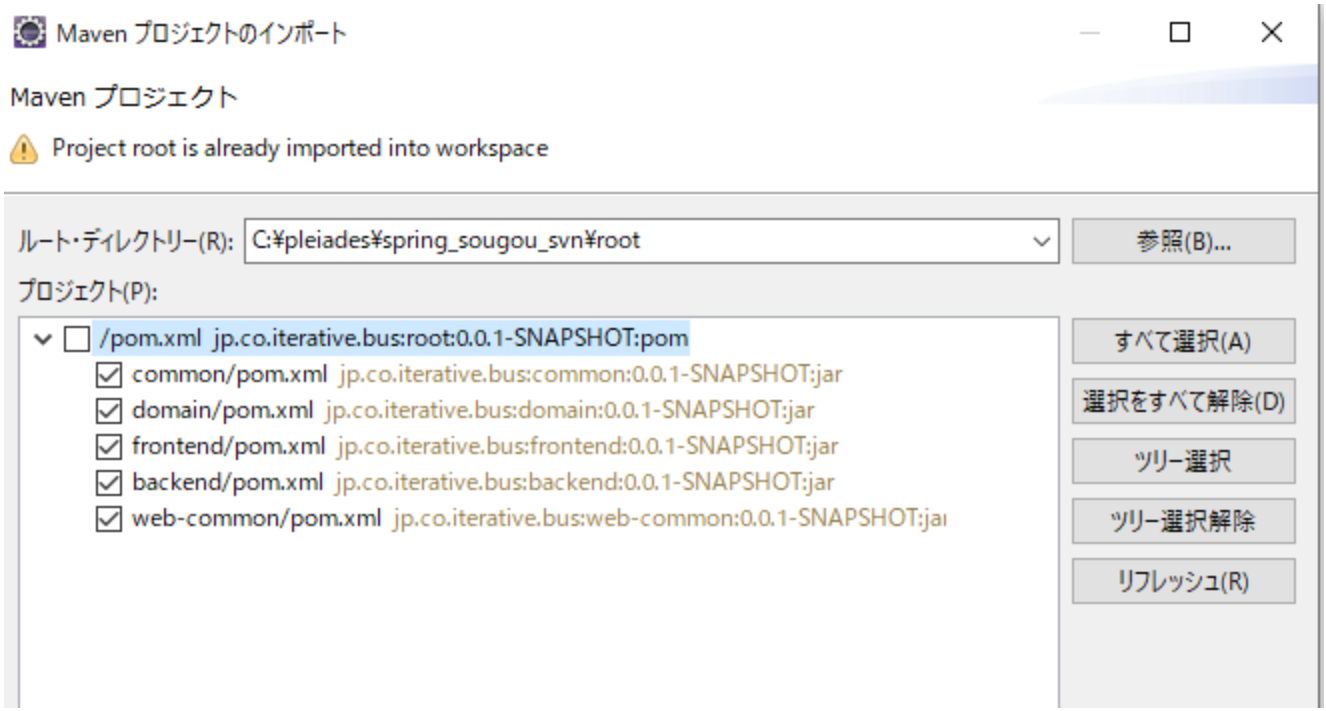
アーカイブファイルの選択に配布された「spring_sougou.zip」を指定し、プロジェクトで「generator」と「root」を選択して完了ボタンをクリックします。



# プロジェクトの作成方法

rootプロジェクトを右クリックして、[インポート]をクリックし、[Maven]-[既存Mavenプロジェクト]を選択します。

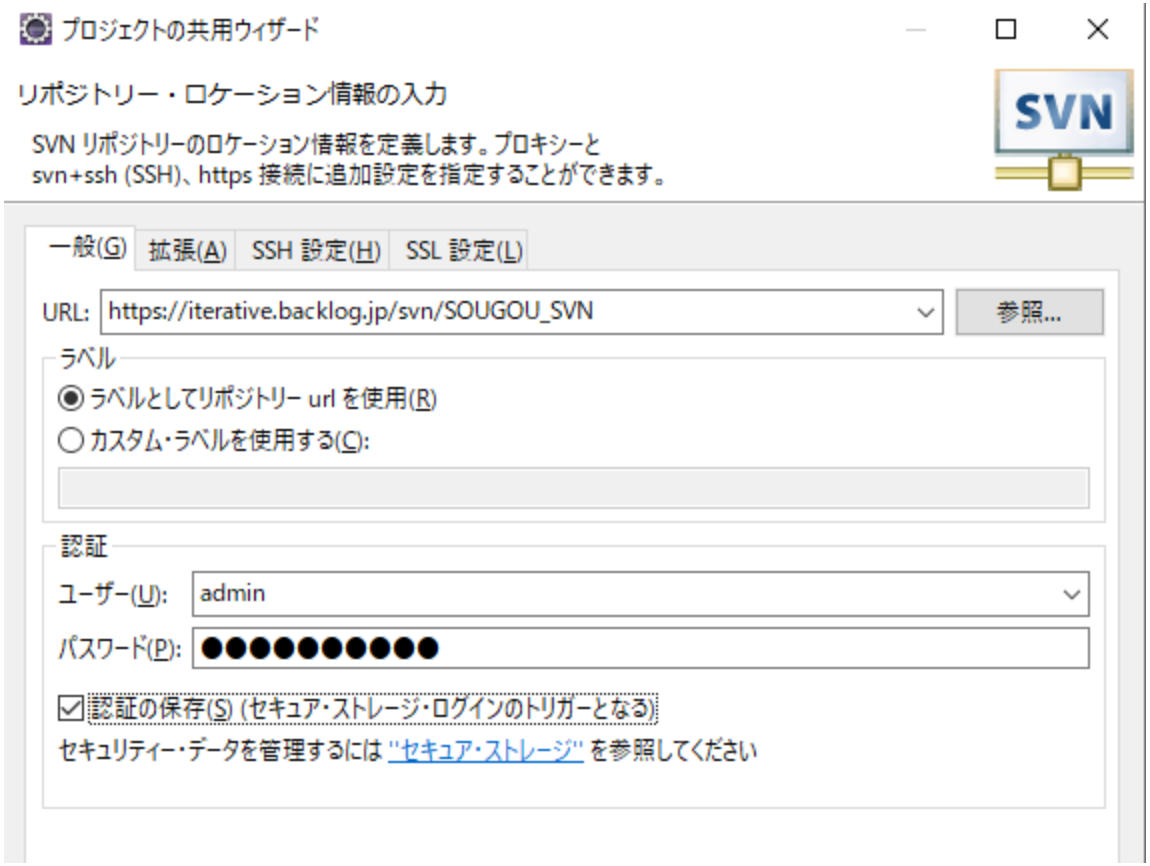
下記のように全てのプロジェクトをチェックした状態で、完了ボタンをクリックします。



# プロジェクトの作成方法

rootプロジェクトを右クリックして、[チーム]-[プロジェクトの共用]を選択します。リポジトリータイプでは「SVN」を選択して次へボタンをクリックします。

URL、ユーザー、パスワードを入力して、次へボタンをクリックします。

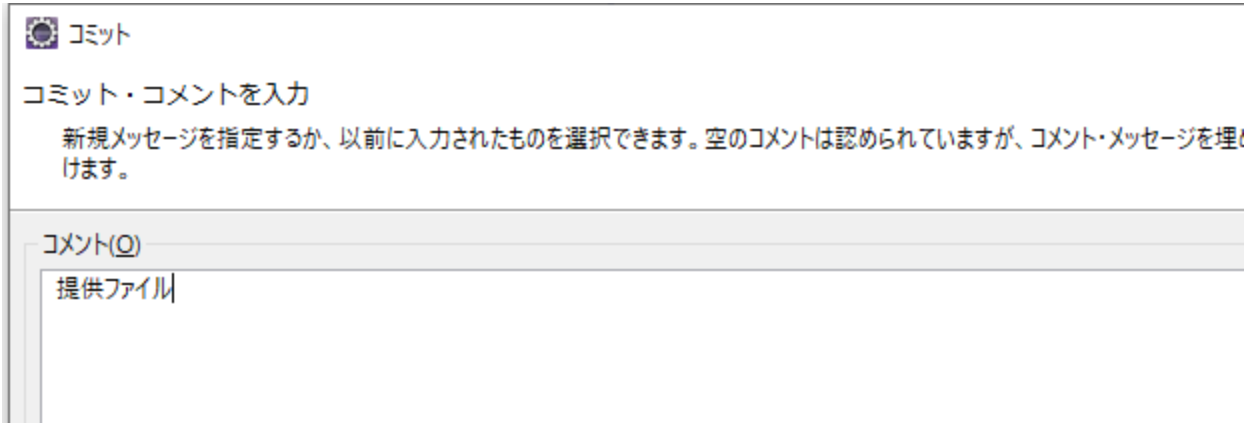


# プロジェクトの作成方法

シンプルモードを選んで完了ボタンをクリックします。

# プロジェクトの作成方法

コミットコメントを入力してコミットします。



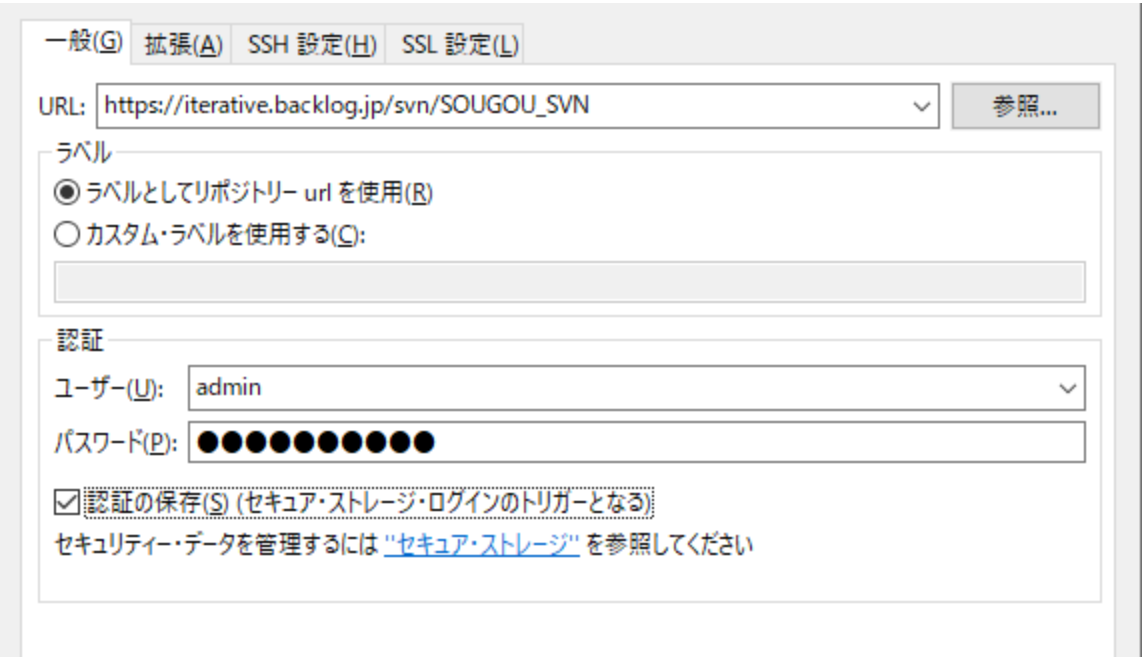
# SVNからプロジェクトを取り込む

ここで説明する作業は、チーム全員が行う必要があります。

- ワークスペースに「../workspace_bus」を指定してEclipseを起動します。
- Eclipseプロジェクトのインポート

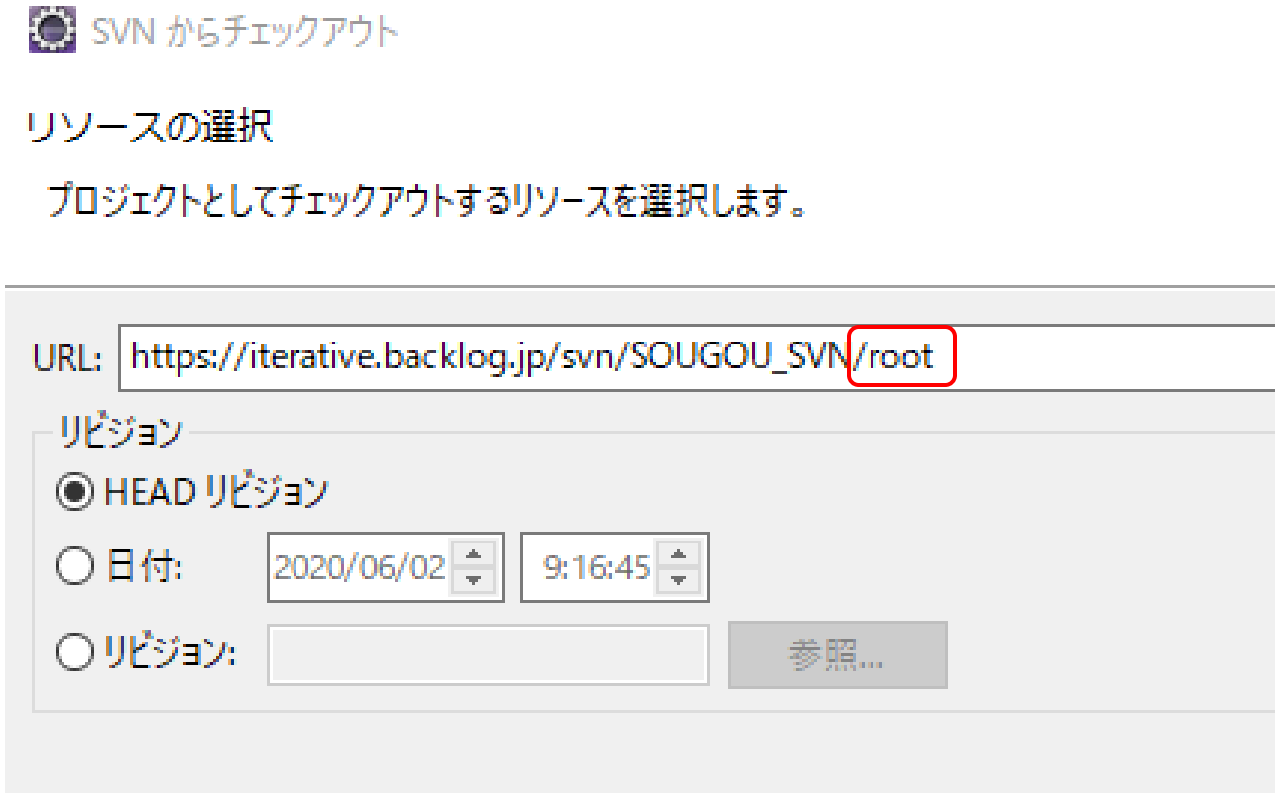
  Eclipseのメニューから、[ファイル]-[インポート]を選択します。

  一般の中の「SVN」内の「SVNからプロジェクト」を選択して次へボタンを押下します。下記リポジトリーロケーションの情報を登録します。



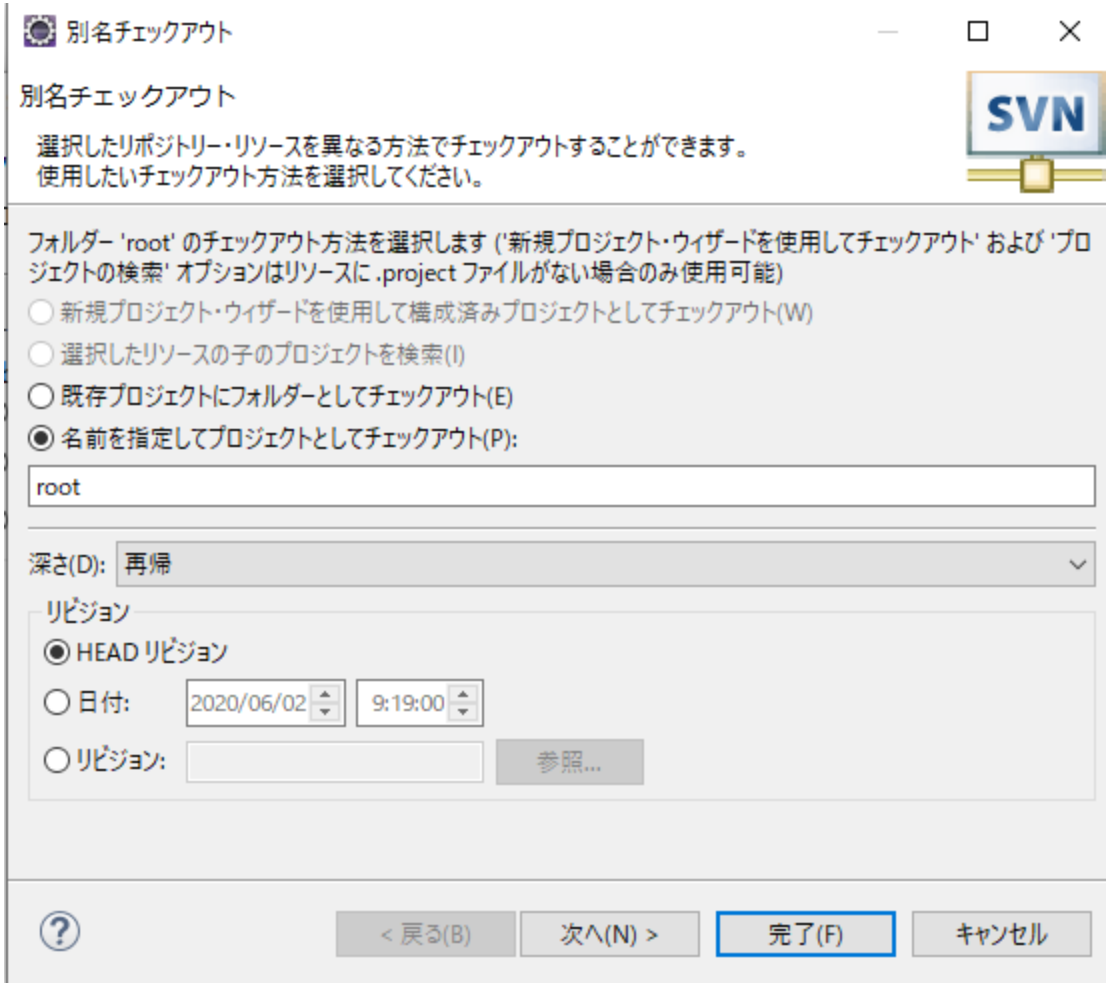
# SVNからプロジェクトを取り込む

SVNのURLに「/root」を付けて指定して完了ボタンをクリックします。



# SVNからプロジェクトを取り込む

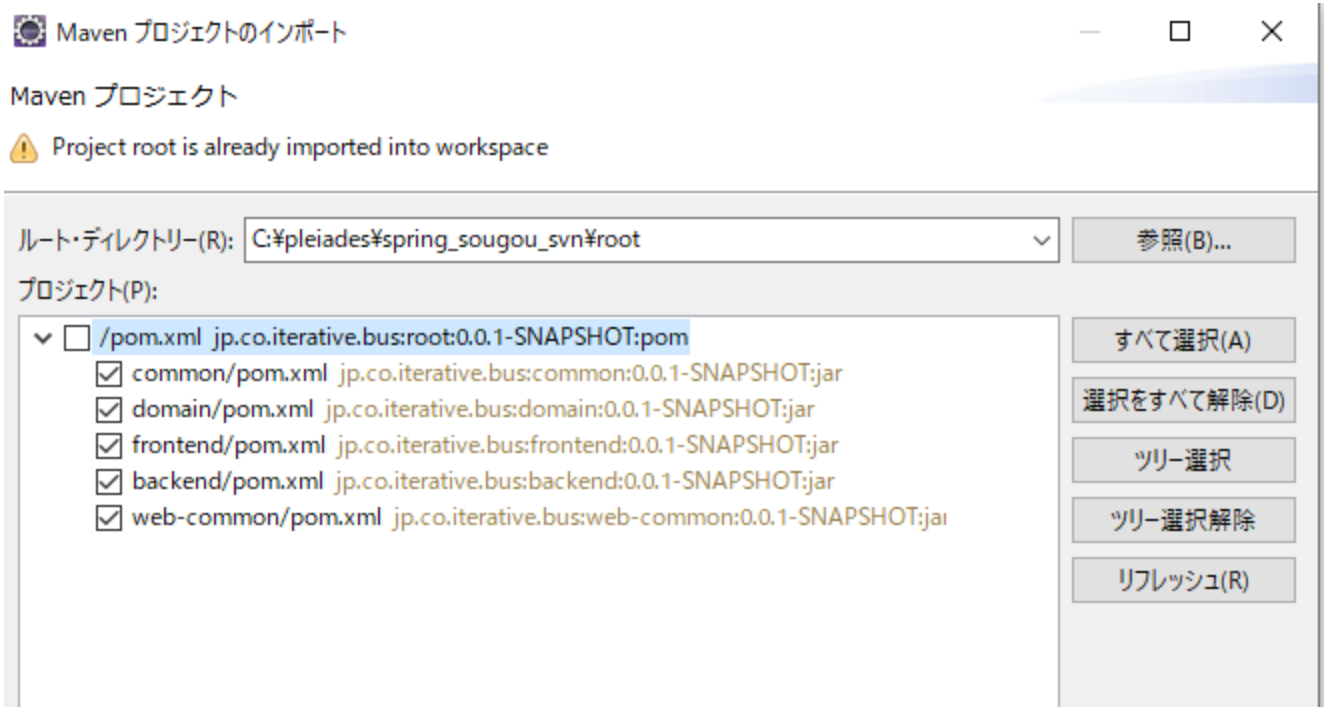
完了ボタンをクリックします。



# SVNからプロジェクトを取り込む

rootプロジェクトを右クリックして、[インポート]をクリックし、[Maven]-[既存Mavenプロジェクト]を選択します。

下記のように全てのプロジェクトをチェックした状態で、完了ボタンをクリックします。

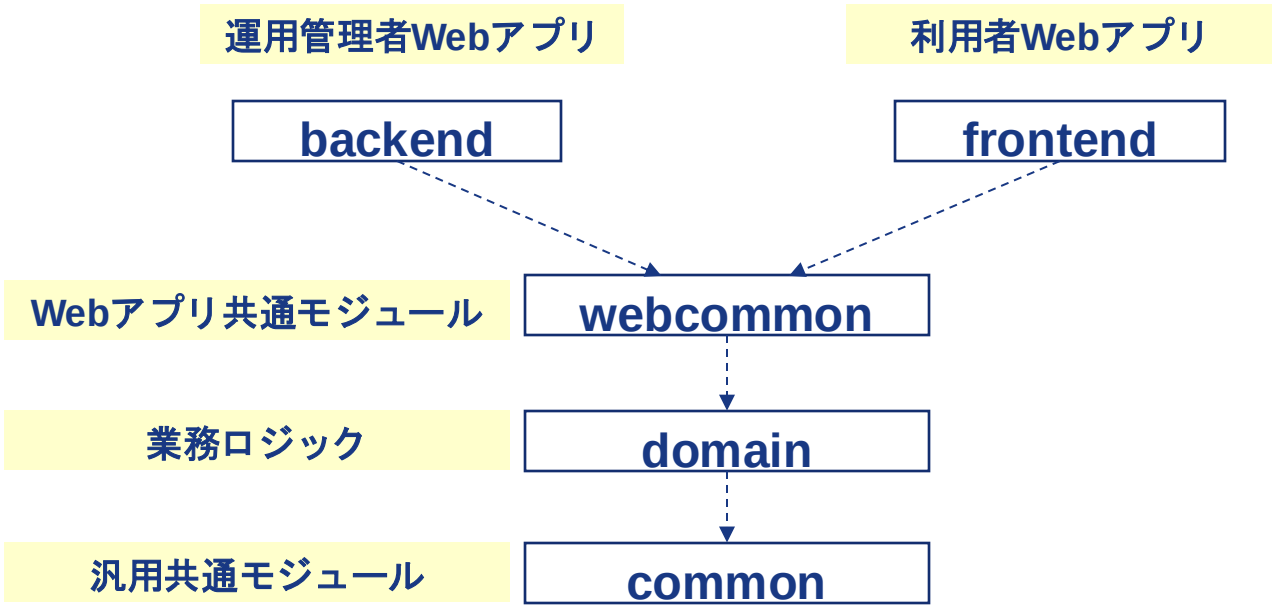


# SVNにコミットする際の注意

各プロジェクト（backendやcommonなど）には「target」というフォルダがあります。このフォルダにはバイトコードなどのファイルが保存され、他のPCと共有するとトラブルになる恐れがあります。そのため、SVNにはtargetフォルダ配下のファイルをコミットしないでください。

なお、「root」プロジェクトを同期化すれば、targetフォルダは差分として表示されません。

# プロジェクト構成




| プロジェクト名 | クラス群 | 説明 |
|---|---|---|
| backend/frontend | Controller,Form,JSP等 | サブシステムごとのアプリの画面のプログラムを配置します。 |
| webcommon | Controller,Form,JSP等 | サブシステムをまたいで共通化するプログラムを配置します。 |
| domain | Mapper,Entity,Example等 | データベースの処理や概念レベルの処理を配置します。 |
| common | Util等 | システム全体で利用する汎用的なプログラムを配置します。 |