


4時間で学ぶ、効率的な 自動テストスクリプトのメンテナンス ～Geb編～



テスト自動化研究会(STAR)

システムテスト実行の自動化

- GUI(画面)自動テストツール
- 画面操作を自動化し、テスト作業を効率化!
- Geb, Selenium, UFT, UWSCなど様々なツールがある



- ブラウザのテストツール
- オープンソース
- Selenium WebDriverの使いやすいラッパー

今から4時間で学ぶこと

1. Gebの基本的な使い方
2. Gebテストを効率よくメンテナンスする方法

Geb, Groovy, Spock

□ Geb

- Groovyベースの自動テストツール
- Selenium WebDriverの使いやすいラッパー
- JQuery互換の要素操作API

□ Groovy

- JVM (Java Virtual Machine)上で動作するスクリプト言語
- Rubyライクなシンプルで分かりやすい文法とAPI
- Javaのコードがほぼ100%修正せずに利用可能

□ Spock

- Groovyベースのテスト・フレームワーク
- Gebとの統合機能でGebを更に使いやすく

タイムテーブル

1. Gebの使い方

1-1. 入門課題	70分
1-2. 実践課題	30分

休憩

2. Gebテストを効率よくメンテナンスする

2-1. 概要説明	10分
2-2. 実践課題: ページオブジェクトデザインパターン	60分

休憩

2-3. 実践課題: システムのバージョンアップ	40分
--------------------------	-----

1. Gebの使い方

1-1. 入門課題 (70分)

入門課題

- Gebの基礎を学びます
- 5分程度のミニ課題×7
- 必要なもの
 - IntelliJ IDEA
 - Google Chrome
 - 課題プログラムインストールキット

入門課題その1

「動かしてみよう、Geb」

1. IntelliJ IDEAを起動します
2. `src/test/groovy/introwork/IntroWork1Spec.groovy`を開いてください

入門課題その1

「IntroWork1Spec.groovy」を実行し、成功することを確認してください

□ 手順

1. `src/test/groovy/introwork/IntroWork1Spec.groovy`を右クリックし、Run 'IntroWork1Spec'を選びます
2. テストが実行され、結果が緑になれば成功です

入門課題その1 解説

Spock

- テストの実行には、テスト・フレームワーク「Spock」を使っています

- `def "テストメソッド名"`
 - メインとなるテスト処理を記述します
 - `when:`
 - URLの移動や要素のクリックなど期待動作までのステップを記述します
 - `then:`
 - 要素のアサーションや期待動作を記述します

入門課題その1 解説

def "テストメソッド名"

```
def "LoginShouldSuccess"() {  
  when:  
  
  .....  
  // 期待動作までのステップを記述します  
  go url  
  
  then:  
  // アサーションや期待動作を記述します  
  .....  
}
```

テストメソッドの命名規則

内容を端的に表すものにします。Spockでは日本語も使用できます。手動のテストケース名を使用してもよいでしょう

入門課題その2

「クリックしてみよう」

1. `src/test/groovy/introwork/IntroWork2Spec.groovy` を右クリックし、Run 'IntroWork2Spec' を選びます
2. 「OK」ボタンが置かれたページが表示されます

入門課題その2

「OK」ボタンをクリックする処理を、
`IntroWork2Spec.groovy` に実装してください

入門課題その2

「クリックしてみよう」

□ 「OK」ボタンのidを調べます

1. introWork/introWork2.htmlを、Google Chromeから直接開きます
2. 「OK」ボタンを右クリックし「要素の検証」を選びます

□ Sleep処理を消して、クリック操作を記述します

```
$("#要素のid").click();
```

□ 書けたら実行してみます

補足 要素の指定の仕方

- Gebの要素の検索の仕方はjQuery互換です
 - \$("タグ名")
 - などのタグで検索します
 - \$(".クラス名")
 - タグのCSSクラス名で検索します
 - \$("タグ名", name : "名前")
 - nameに指定した名前を検索します
- 余裕があったら、nameでも指定してみましょう
 - \$("button", name : "okButton")

入門課題その2 解答例

```
def "OneCanClickOKButton"() {  
  when:  
  .....  
  go url  
  
  then:  
  $( "#ok_button" ).click()  
}
```

入門課題その3

「文字列を入力してみよう」

1. IntroWork3Spec.groovyをSpockテストとして実行します
2. テキスト入力欄が置かれたページが表示されます

入門課題その3 (5分)

テキスト入力欄の「Test」という文字列を消して、代わりに「Geb」という文字列を入力する処理を、IntroWork3Spec.groovyに実装してください

入門課題その3

「文字列を入力してみよう」

□ ヒント

- value("文字列")で文字列を入力できます
- value("文字列")は上書きなので元の文字列は自動的に消去されます

```
$( "#要素のid" ).value( "文字列" )
```


入門課題その3 解答例

```
def "OneCanSetValue"() {  
  when:  
  .....  
  go url  
  
  then:  
  $( "#subject" ).value( "Geb" )  
}
```

入門課題その4

「ラジオボタンを選択してみよう」

- IntroWork4Spec.groovyをSpockテストとして実行すると、課題ページが表示されます

入門課題その4 (5分)

ラジオボタンの「あり」の選択肢を選ぶ処理を、IntroWork4Spec.groovyに実装してください

- ヒント
 - ラジオボタンの選択は「click」で行います

入門課題その4 解答例

```
def "OneCanClickRadioButton"() {  
  when:  
    .....  
    go url  
  
  then:  
    $( "#on_radio" ).click()  
}
```

入門課題その5

「チェックボックスを選択してみよう」

- IntroWork5Spec.groovyをSpockテストとして実行すると、課題ページが表示されます

入門課題その5 (5分)

チェックボックスのチェックをオンにする処理を、IntroWork5Spec.groovyに実装してください

入門課題その5

「チェックボックスを選択してみよう」

□ ヒント

- value(*true*)でチェックを入れます
- value(*false*)でチェックを外せます

```
$("#要素のid").value(true)
```

入門課題その5 解答例

```
def "OneCanSetCheckbox"() {  
  when:  
  .....  
  go url  
  
  then:  
  $( "#allowed_check" ).value(true)  
}
```

入門課題その6

「プルダウンを選択してみよう」

- IntroWork6Spec.groovyをSpockテストとして実行すると、課題ページが表示されます

入門課題その6 (5分)

プルダウンの選択値を5にする処理を、
IntroWork6Spec.groovyに実装してください

入門課題その6

「プルダウンを選択してみよう」

□ ヒント

- \$("要素のid")でselect要素を取得します
- value("文字列")メソッドでoptionのvalueを指定します
- valueに対応したプルダウンが選択されます

```
$("#要素のid").value("文字列")
```


入門課題その6 解答例

```
def "OneCanSelectOptionByValue"() {  
  when:  
  .....  
  go url  
  
  then:  
  $( "#head_count" ).value( "5" )  
}
```

入門課題その7

「表示された値のチェックをしてみよう」

- IntroWork7Spec.groovyをSpockテストとして実行すると、課題ページが表示されます

入門課題その7 (5分)

表示された金額の値が「9000」であることをチェックする処理を、IntroWork7Spec.groovyに実装してください

入門課題7

「表示された値のチェックをしてみよう」

□ ヒント

- text()により表示されているテキストを取得します
- == で値が9000であることをチェックします
 - then節にはassertを明示的に書く必要はありません
 - Power Assertという機能で失敗を分かりやすく表示してくれます

```
$("#要素のid").text() == "値"
```

入門課題その7 解答例

```
def "OneCanGetAndCheckText"() {  
  when:  
    .....  
    go url  
  
  then:  
    $("total").text() == "9000"  
}
```

入門課題で学んだこと

- クリック
- 文字列入力
- ラジオボタン
- チェックボックス
- プルダウン
- 値チェック

1. Gebの使い方

1-2. 実践課題 (30分)

実践課題その1

- src/test/groovy/practicework/
PracticeWork1Spec.groovyをSpockテストとして実行
すると、「STARホテル宿泊予約画面」が表示されま
す

実践課題その1 (30分)

docs/TestCase.pdfの「実践課題その1」テストケースを、
PracticeWork1Spec.groovyに実装してください。

- 予約処理の自動化
- 確認画面の値チェックの自動化

□ ヒント

- reserveApp/index.htmlがHTMLファイルです。

実践課題その1 解答例

- answer/test/groovy/practicework/work1/
PracticeWork1Spec.groovy

休憩

タイムテーブル

1. Gebの使い方

1-1. 入門課題	70分
-----------	-----

1-2. 実践課題	30分
-----------	-----

休憩

2. Gebテストを効率よくメンテナンスする

2-1. 概要説明	10分
-----------	-----

2-2. 実践課題: ページオブジェクトデザインパターン	60分
------------------------------	-----

休憩

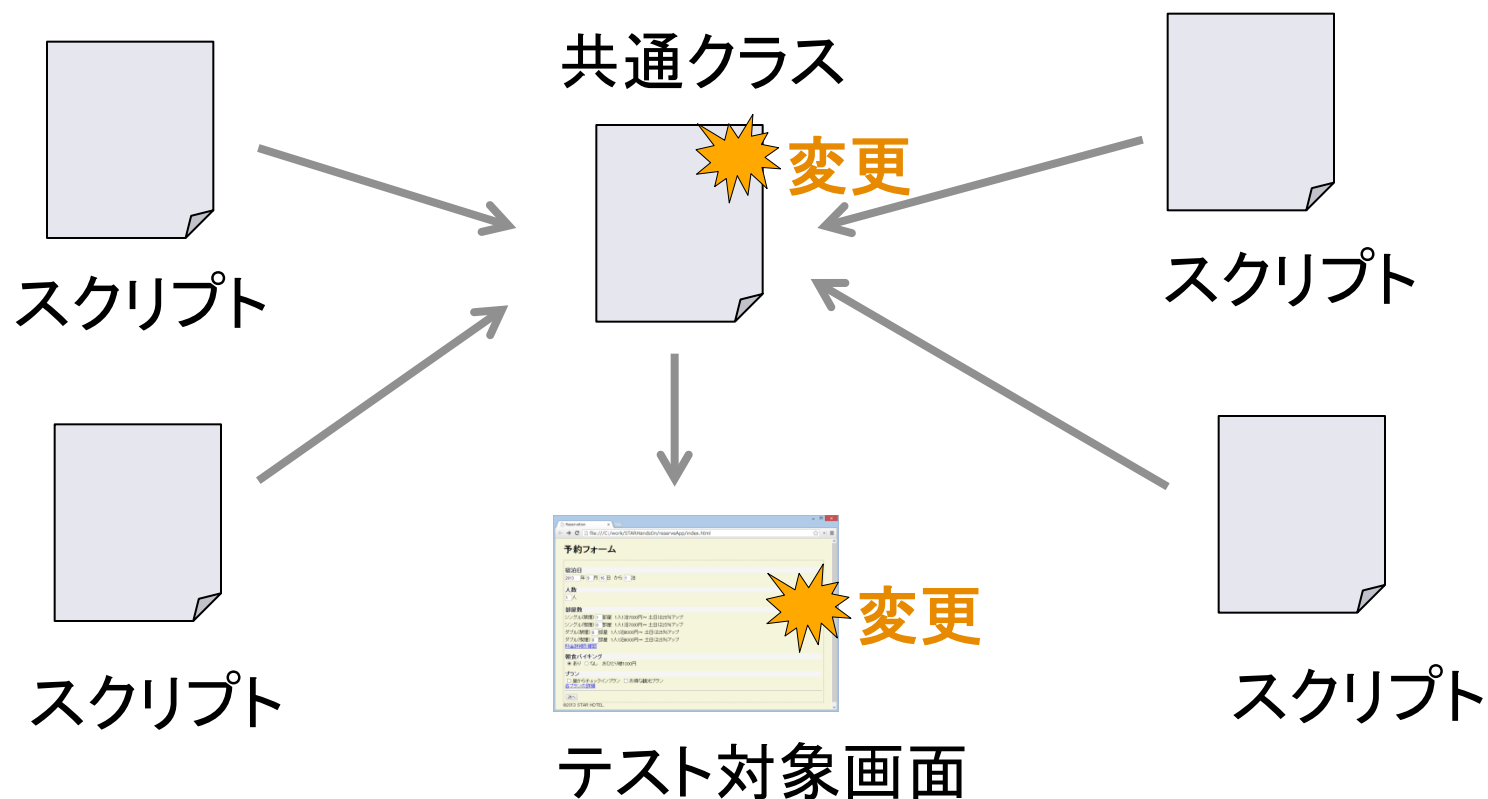
2-3. 実践課題: システムのバージョンアップ	40分
--------------------------	-----

2. Gebテストを効率よくメンテナンスする

2-1. 概要説明 (10分)

テストスクリプトの共通化

- プログラムの共通化をうまく行えば、修正は1か所で済みます

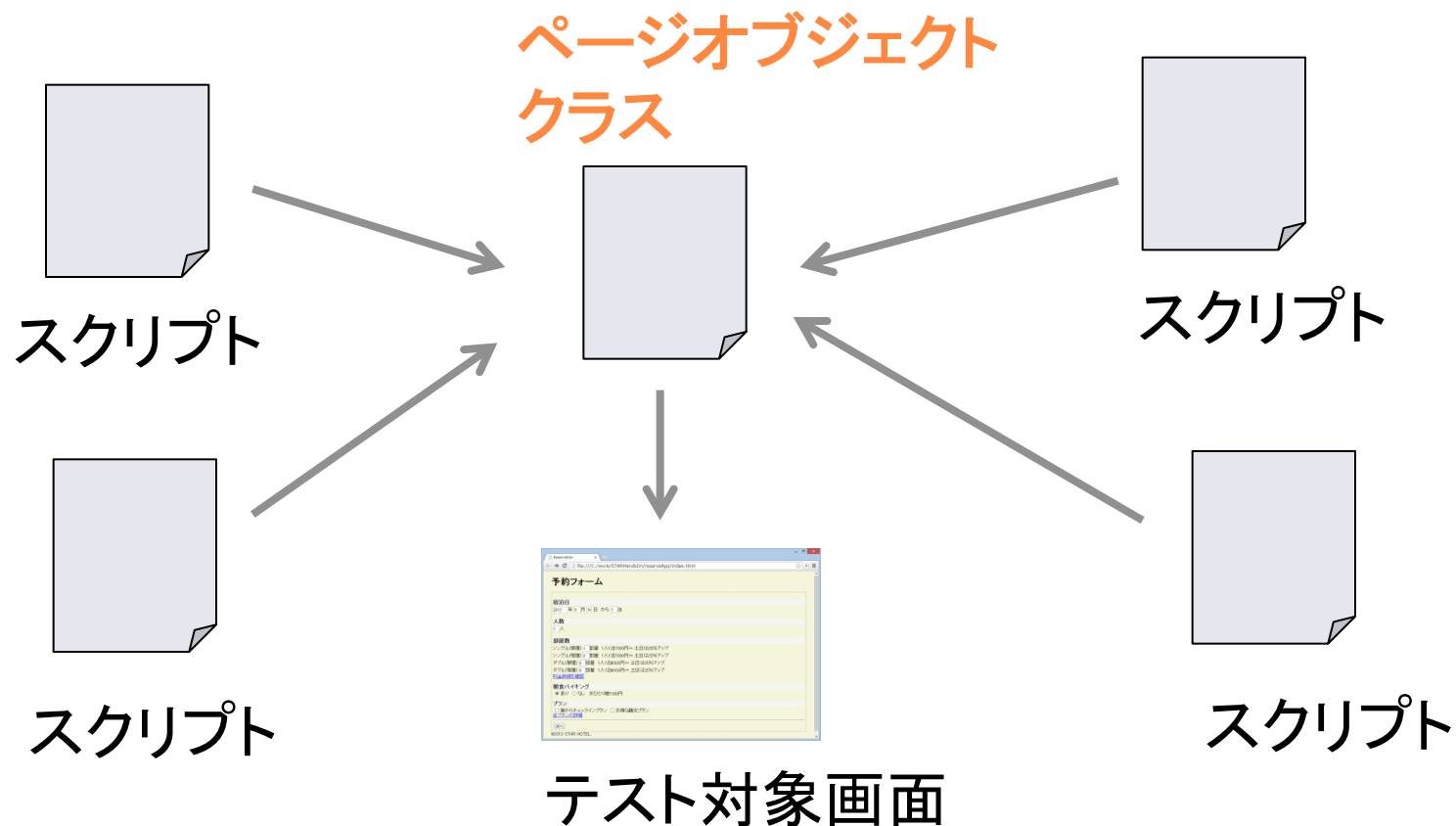


今から学ぶこと

- Gebで、共通化によってスクリプトのメンテナンスコストを抑える方法
- ページオブジェクトデザインパターン

ページオブジェクトデザインパターン

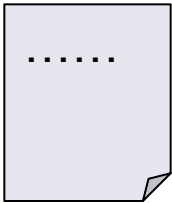
□ Gebプログラム共通化の デザインパターン



「STARホテル宿泊予約画面」 ページオブジェクトを使わない場合

```
$( "#reserve_term" ).value( "3" )
```

スクリプト



スクリプト



テスト対象画面

「STARホテル宿泊予約画面」の ページオブジェクトを使った場合

```
reserveInputGebPage.reserveTerm = "3"
```

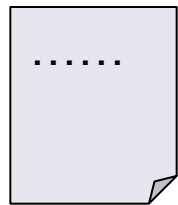
スクリプト

ページオブジェクトクラス

ReserveInputGebPage

+reserveTerm

+setReserveDate(year, month, day)



スクリプト



テスト対象画面

「STARホテル宿泊予約画面」の ページオブジェクトを使った場合

```
ReserveInputGebPage inputPage  
    = browser.to ReserveInputGebPage  
inputPage.reserveTerm = "3"
```

スクリプト

- idなどのHTML情報が、スクリプト中に現れません
- clickなどGebの処理もスクリプト中に現れません

2. Gebテストを効率よくメンテナンスする

2-2. 実践課題: ページオブジェクト デザインパターン (60分)

ページオブジェクトデザインパターン を実践

□ 実践課題その2

- 「実践課題その1」テストケースをページオブジェクトで書き換えます

□ 実践課題その3

- ページオブジェクトを使って新しいテストケースを実装します

実践課題その2

実践課題その2 (40分)

次の3つの実装を完成させてください。

- 1ページ目「予約入力画面」のページオブジェクトsrc/
test/groovy/practicework/pages/
ReserveInputGebPage.java

- 2ページ目「予約確認画面」のページオブジェクトsrc/
test/groovy/practicework/pages/
ReserveConfirmGebPage.java

- 「実践課題その1」テストケースをページオブジェクトで実装し直した、src/test/groovy/practicework/
PracticeWork2Spec.groovy

実践課題その2 ヒント

□ 朝食バイキングの値のsetメソッド

```
public void setBreakfast(boolean on) { ..... }  
reserveInputGebPage.breakfast = true
```

□ ページ遷移の確認

■ ページオブジェクト側の設定

```
static at = { title == "Reservation" }
```

■ スクリプト側での確認

```
ReserveConfirmGebPage reserveConfirmGebPage  
= browser.at ReserveConfirmGebPage
```

実践課題その2 解答例

- answer/test/groovy/practicework/work2以下
 - pages/ReserveInputGebPage.java
 - pages/ReserveConfirmGebPage.java
 - PracticeWork2Spec.groovy

実践課題その3

実践課題その3 (20分)

「実践課題その3」テストケースをページオブジェクトで実装した、`src/test/groovy/practicework/PracticeWork3Spec.groovy` を完成させてください。

実践課題その3 ヒント

- 確認画面の「昼からチェックインプラン」項目の有無を調べるメソッド

```
public boolean existsPlanB() { ..... }
```

- 要素の存在チェックをオフにする

```
planB(required: false) { ..... }
```

- 要素が存在するかどうかを調べる方法

```
$("要素のid").size() > 0
```


実践課題その3 解答例

- answer/test/groovy/practicework以下
 - work3/PracticeWork3Spec.groovy
 - work3/pages/ReserveConfirmPage.java

休憩

タイムテーブル

1. Gebの使い方

1-1. 入門課題	60分
1-2. 実践課題	40分

休憩

2. Gebテストを効率よくメンテナンスする

2-1. 概要説明	10分
2-2. 実践課題: ページオブジェクトデザインパターン	60分

休憩

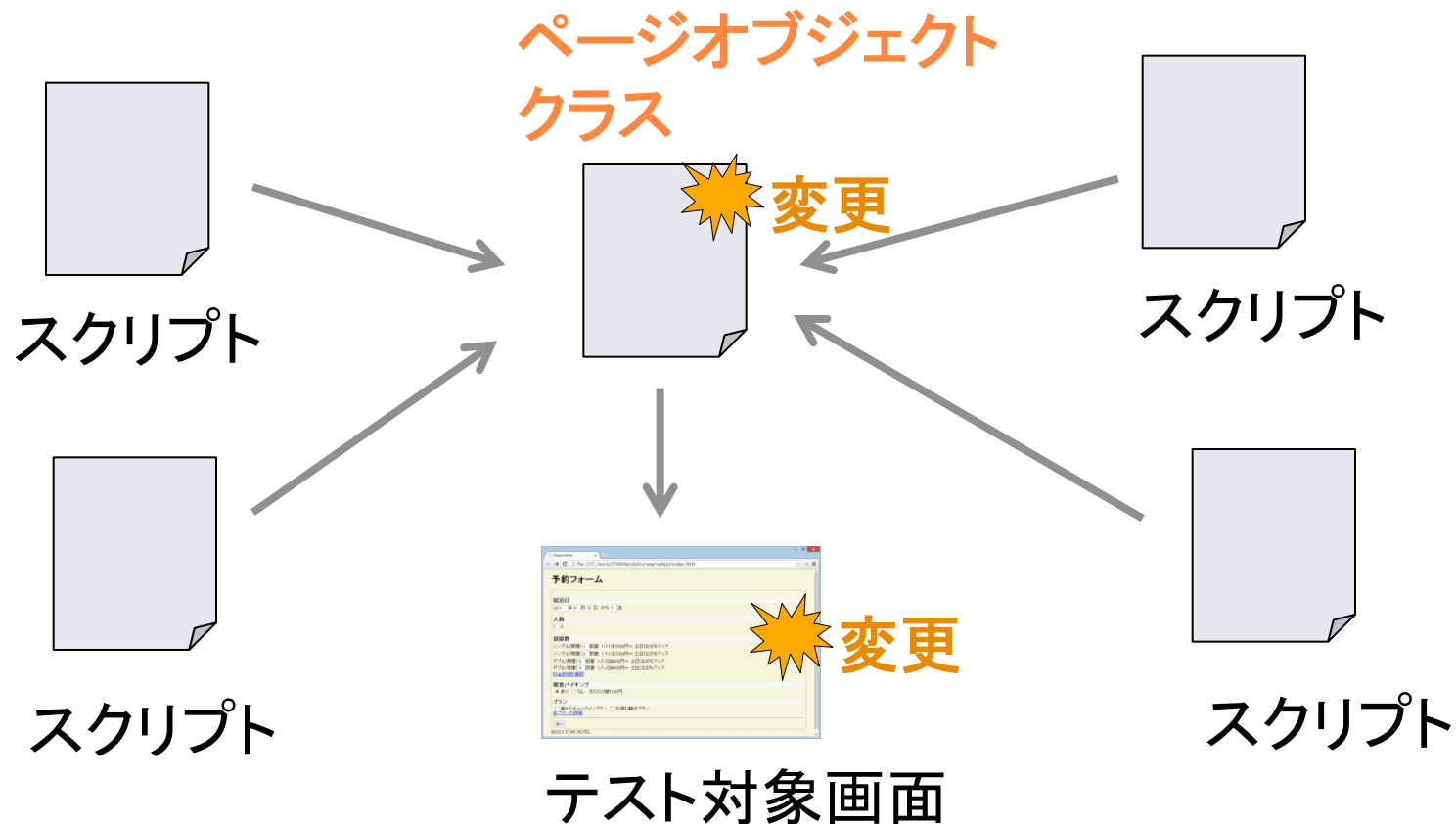
2-3. 実践課題: システムのバージョンアップ	40分
--------------------------	-----

2. Gebテストを効率よくメンテナンスする

2-3. 実践課題:システムのバージョンアップ (40分)

テスト対象画面が変更された時の 影響範囲

□ ページオブジェクトデザインパターン



実践課題その4

- 実際にテスト対象画面が変更されると、どんな修正が必要になるか、体感してみましょう。
- 「実践課題その2」で作成した、`src/test/groovy/practicework/pages/ReserveInputGebPage.groovy`を開きます
- URLを"`reserveApp/index.html`"から"`reserveApp_Renewal/index.html`"に書き換えます
- `PracticeWork2Spec.groovy`を実行し、失敗することを確認します。

実践課題その4

実践課題その4 (40分)

PracticeWork2Spec.groovyとPracticeWork3Spec.groovyのURLを"[reserveApp_Renewal/index.html](#)"に書き換えたテストが成功するよう、ページオブジェクトの内容を書き換えてください。

□ ヒント

- 書き換え前のページオブジェクトは、バックアップを取っておくのがお勧めです。

実践課題その4 ヒント

□ setReserveDateメソッドの実装

```
element.value(  
    year + "/" + month + "/" + day)
```

+

```
element << Keys.RETURN
```


実践課題その4 解答例

- answer/test/groovy/practicework/work4以下
 - pages/ReserveInputGebPage.groovy
 - pages/ReserveConfirmGebPage.groovy

発展課題

実践課題その5

□ 時間がある方はチャレンジ!

実践課題その5

「実践課題その5」テストケースをページオブジェクトで実装した、src/test/groovy/practicework/PracticeWork5Spec.groovy を完成させてください。

実践課題その5 ヒント

□ テキスト入力欄の値の取得

- value()は省略可能です

□ ラジオボタンの選択状態の取得

- 文字列 "true"と""で返ってくることに注意して下さい

```
$("要素のid").attr("checked") == "true"
```

実践課題その5 解答例

- answer/test/groovy/practicework以下
 - work5/PracticeWork5Spec.groovy
 - work5/pages/ReserveInputPage.java

学んだことのまとめ

- Gebの基礎を学びました
- ページオブジェクトデザインパターンを学びました
 - 変更されやすい画面情報を1ヶ所に集約して、効率よくメンテナンス

今回取り上げなかった話題

□ モジュール

- モジュールを使用すると、複数のページで再利用可能な断片を定義し、各ページで利用できます

□ Gebの設定

- GebConfig.groovyによって複数WebDriverの設定やデフォルトのBaseURL、各種のデフォルト値を設定します

□ JavaScriptやAjaxなど動的要素の取り扱い

- GebではJavaScriptやAjax, システムダイアログなどの動的要素の簡単な記述で扱えます

□ 他ツールとの連携

- Spockを初めとしたテスト・フレームワークやGradleなどのビルトツールとも簡単な設定で連携できます

お疲れさまでした
