



# *Transaction*



# ユーザオペレーション

- 予測不可能な利用者の操作
  - 二度押し
    - 決済処理の重複などの弊害
  - ブックマークなどによる遷移への割り込み
    - ウィザードスタイルのページ遷移が破綻
  - リンク先を新しいブラウザで表示
    - 同一処理が重複する可能性
  - ブラウザの戻るボタン
    - キャッシュされた古い情報が表示される可能性
    - アクションが再度実行される可能性

# トラブル原因はアクションの順序

- 全ては、ページとアクションが正しい順序で実行されない事に起因

# サーバサイドでの現象

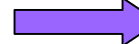
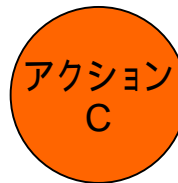
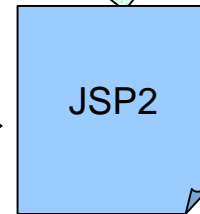
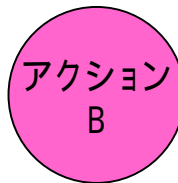
## ■ 本来の順序



クライアントサイド



サーバサイド



アクション  
フォワード



http response  
http request



想定遷移

# サーバサイドでの現象(二度押し)

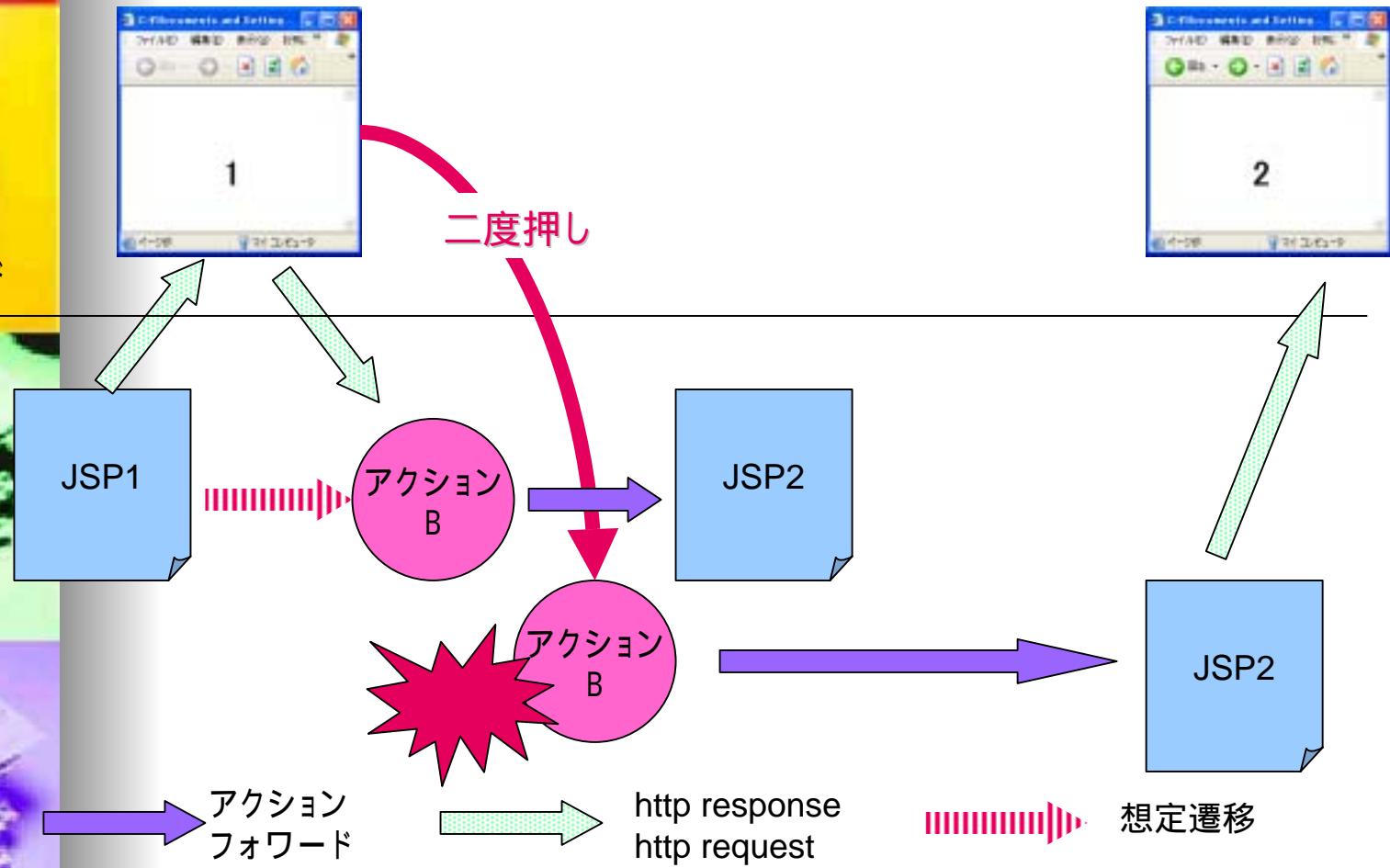
- アクションB実行後、アクションBが再実行



クライアントサイド



サーバトサイド



# サーバサイドでの現象(遷移割り込み)

- アクションAが**実行されないまま**、アクションBが**実行**



クライアントサイド

遷移割り込み



サーバトサイド

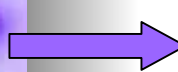
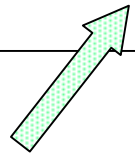
アクション  
A

JSP1

アクション  
B

JSP2

アクション  
C



アクション  
フォワード



http response  
http request

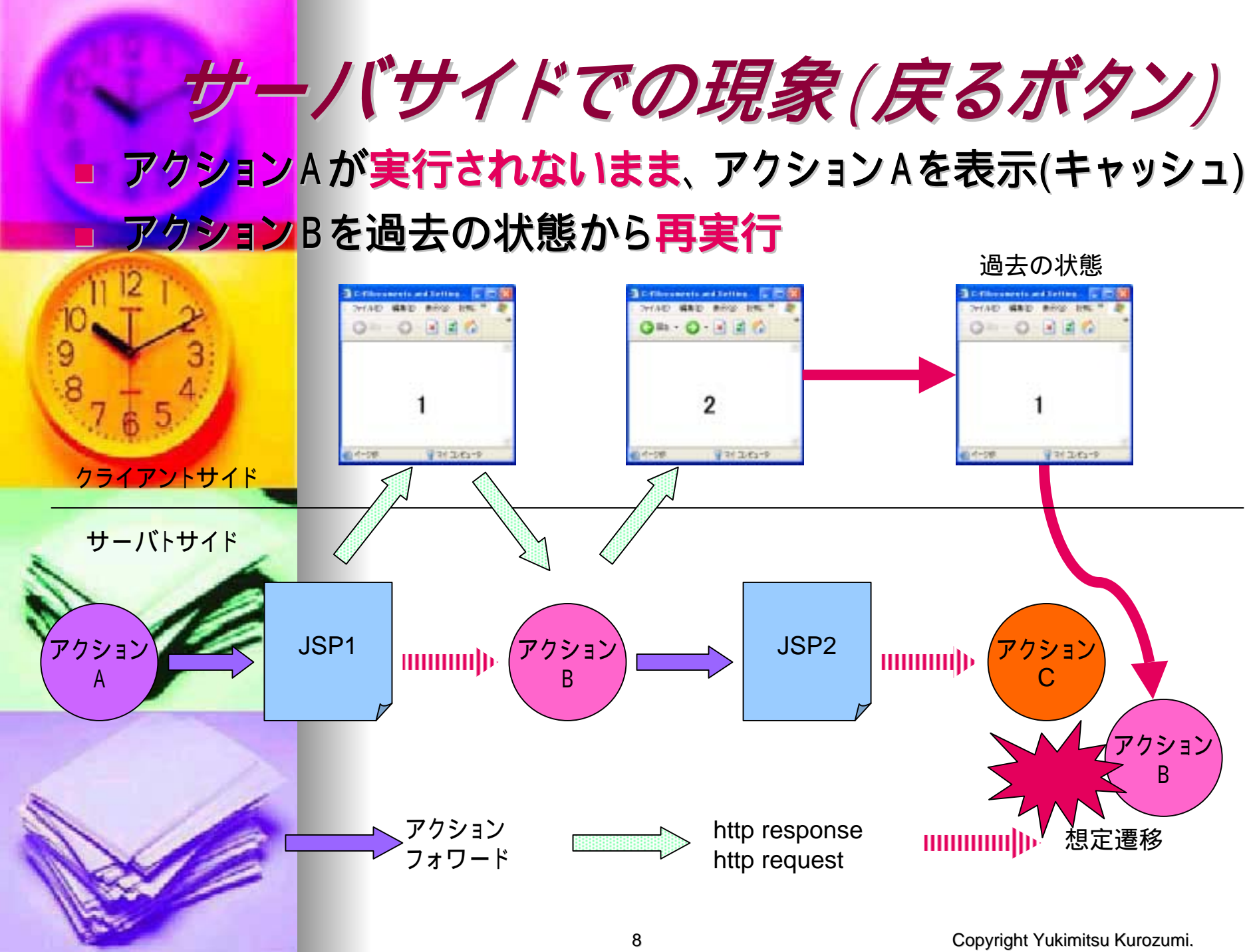


想定遷移



# サーバサイドでの現象(戻るボタン)

- アクションAが**実行されないまま**、アクションAを表示(キャッシュ)
- アクションBを過去の状態から**再実行**



# 予測不可能な操作を防止・検出する

- 全ては、ページとアクションが正しい順序で実行されない事に起因



- 完全な防止策は難しい
- 順序不正のアクセスを防止・検出する！！
- 方法は・・・
  - 次に実行されるべきアクションを保持する？
  - Refererにて、正しいページからの遷移かチェックする？
  - JavaScriptを駆使する？
  - *Struts Transaction Token* 機能を利用する



# *Struts Transaction Token* 機能

- *Struts*標準機能(v1.0, v.1.1rc1)
- *Transaction Token* にてセッション毎に遷移を管理
- 正しいページからのリクエストか？を検出可能

||

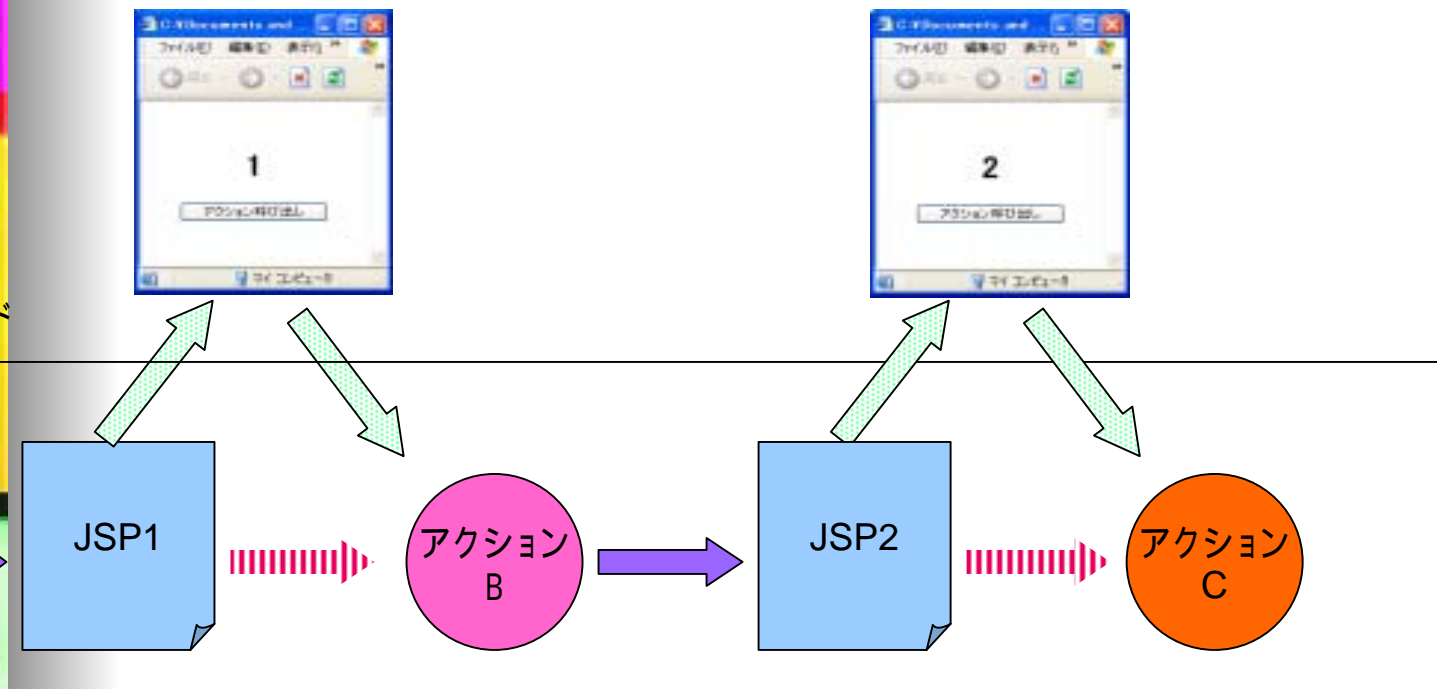
- 正しい処理順序か？の検出が可能

# Struts Transaction Token の仕組み

- アクションの終了直前に *Transaction Token* (ユニークなキー) をセッションに保存させる
- レスポンス生成時に JSP (Struts カスタムタグ) が、*Transaction Token* をページに埋め込む
- ブラウザはリクエストと一緒に埋め込まれた *Transaction Token* を送信する
- アクションの入りでセッションに保存されている *Transaction Token* とリクエストに付けられている *Transaction Token* を比較
  - 同じなら正しいページからのリクエスト
  - 無効なら不正なリクエスト

# Struts Transaction Token のしくみ

1. アクションの終了直前にTransaction Tokenをセッションに保存



**Token=T1** — Transaction Token  
毎回ユニークな値が自動生成される

時間 →

セッションオブジェクト



# Struts Transaction Tokenのしくみ

2. レスポンス生成時にJSPが、Transaction Tokenをページに埋め込む

Token T1が埋め込まれたページ



クライアントサイド

サーバサイド

アクション A

JSP1

アクション B

JSP2

アクション C

Token=T1

Strutsカスタムタグが自動的にTransaction Tokenをページに埋め込む

時間

セッションオブジェクト

アクション  
フォワード

http response  
http request

想定遷移

# Struts Transaction Tokenのしくみ

3. ブラウザはリクエストと一緒に埋め込まれたTransaction Tokenを送信

Token T1が埋め込まれたページ



Token T1と一緒に送信される



クライアントサイド

サーバサイド

アクション A

JSP1

アクション B

JSP2

アクション C

Token=T1

時間

セッションオブジェクト

アクション  
フォワード

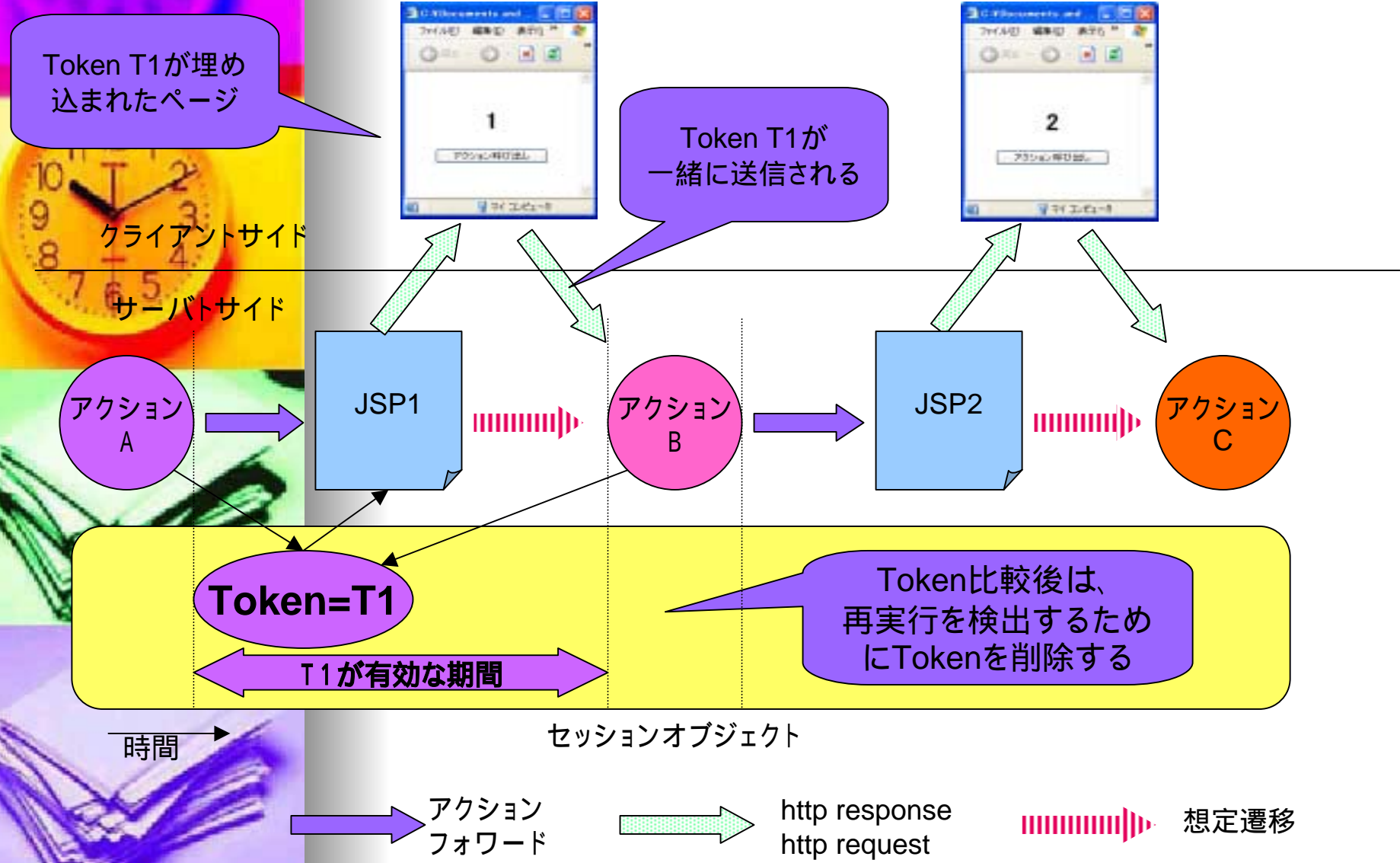
http response  
http request

想定遷移



# Struts Transaction Tokenのしくみ

## 5. 再実行検出のためにTokenを削除



# Struts Transaction Tokenのしくみ

6. 再度、アクションの終了直前にTransaction Tokenをセッションに保存

Token T1が埋め込まれたページ



Token T1と一緒に送信される



クライアントサイド

サーバサイド

アクション A

JSP1

アクション B

JSP2

アクション C

Token=T1

Token=T2

アクション終了直前に再度Tokenを保存する  
値は異なる

T1が有効な期間

時間

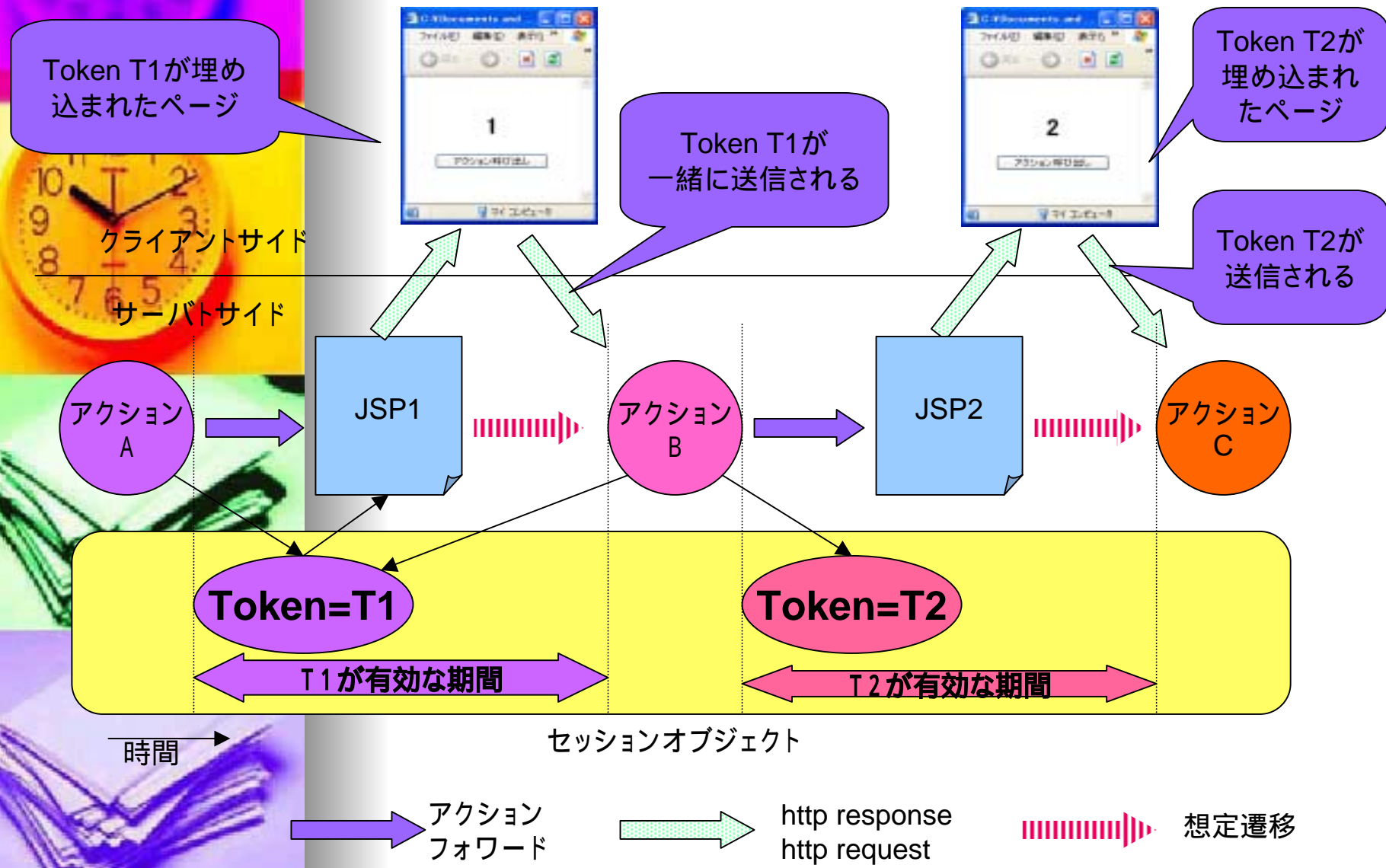
セッションオブジェクト

アクション  
フォワード

http response  
http request

想定遷移

# Struts Transaction Tokenのしくみ



# Struts Transaction Tokenのしくみ

## 二度押し・ポップアップリンクによる再実行検出

Token T1が埋め込まれたページ



Tokenが既に無効なので不正アクセス



二度押し

クライアントサイド

サーバサイド

アクション A

JSP1

アクション B

JSP2

アクション C

Token=T1

Token=T2

T1が有効な期間

T2が有効な期間

時間

セッションオブジェクト

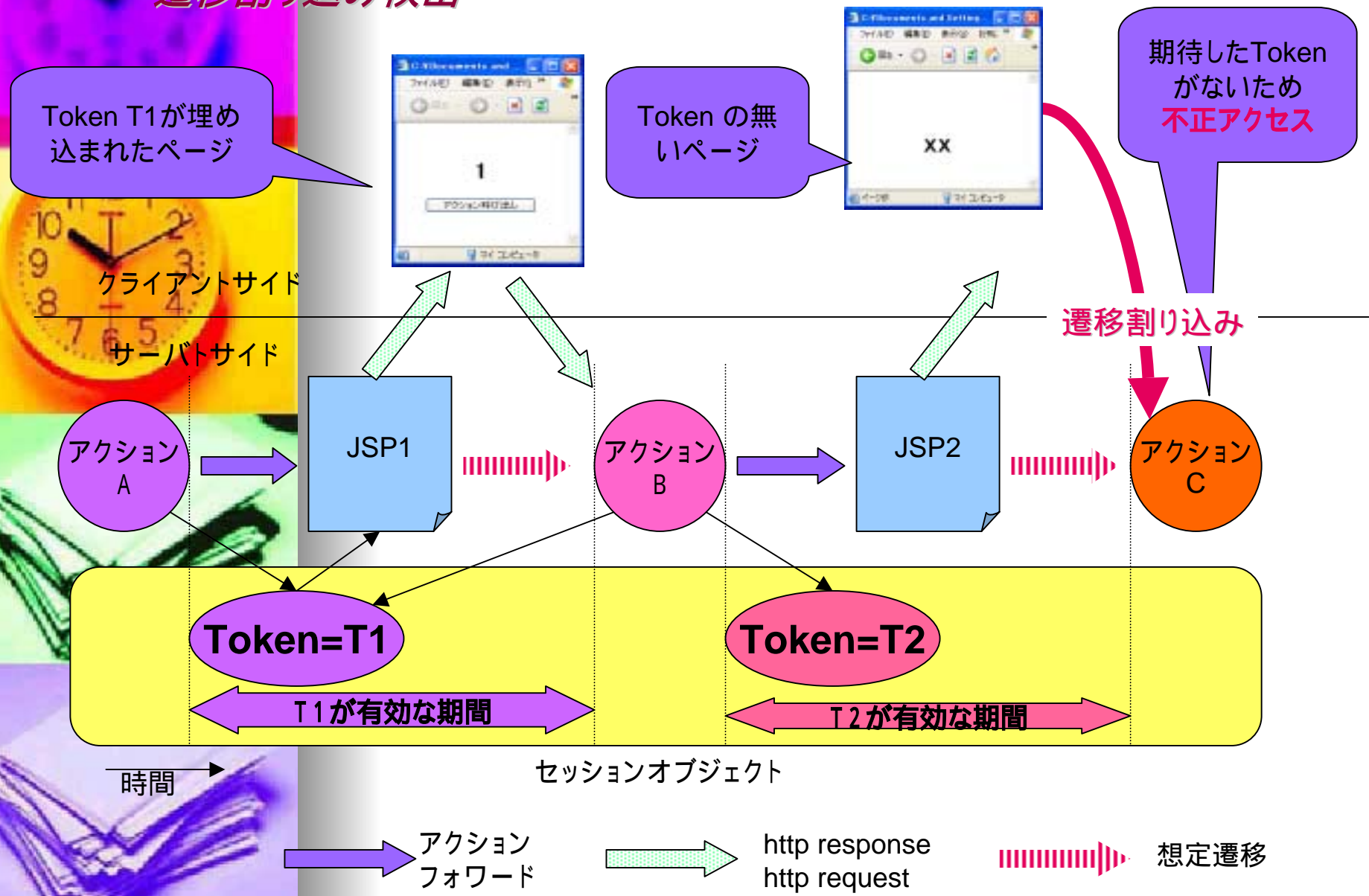
アクション  
フォワード

http response  
http request

想定遷移

# Struts Transaction Tokenのしくみ

## 遷移割り込み検出





# *Struts Transaction Token*記述方法

- JSPを呼ぶ前のアクションにて、トークンを設定
  - Action::saveToken( request )
- JSP内のStrutsカスタムタグでtransactionを指定
  - リンク
    - <html:link ... transaction="true"> にてリクエストパラメータが生成される
  - フォーム
    - <html:form> はsaveTokenが行われていれば、hiddenタグを自動生成
- アクションの最初にて、有効なリクエストか判定
  - Action::isTokenValid(request, true)
    - False なら不正アクセス
    - 第二引数は判定後、Tokenを削除するかどうか

# Struts Transaction Token記述方法

- アクションクラス

```
public class MyAction extends Action {
    public ActionForward execute (....) throws Exception {
        // 不正アクセス検出
        boolean b = isValid( request, true );
        if ( b == false ) {
            // 不正アクセス!!
        }

        // ビジネスロジック呼び出し
        ....
        ActionForward ret = mapping.findForward("success");

        // 次の呼び出しが JSP ならTokenを設定
        if ( ret.getPath().endsWith( ".jsp" ) ) {
            saveToken( request );
        }
    }
}
```

# *Struts Transaction Token* 記述方法

- J S P

.....

<!-- リンクは明示的にtransactionを指定 →

```
<html:link forward="nextpage" transaction="true">
```

.....

<!-- フォームは無指定でも自動的に対応 →

```
<html:form action="/nextaction">
```

....

```
</html:form>
```

.....