

JSP基礎 スライド教材

動的Webページ作成から実践的な開発まで

対象レベル：初級～中級

学習時間：45-60分（詳細学習時）、20-30分（概要把握時）



2025年8月19日

≡ 目次




JSP基礎学習 - 14セッション構成

- 1 ▶ タイトル
- 2 ≡ 目次
- 3 ⓘ JSPとは
- 4 👤 JSPアーキテクチャ
- 5 ✂ 環境準備
- 6 </> JSP構文要素
- 7 📦 暗黙オブジェクト
- 8 ▶ アクション要素
- 9 ⚙ EL式とJSTL
- 10 🔄 フォーム処理とDB連携
- 11 👥 実践応用
- 12 👍 ベストプラクティス
- 13 ? よくある質問
- 14 📄 参考リンク・資料




JSPとは

JSP (JavaServer Pages) は、Javaを使用した動的Webページを作成するサーバーサイド技術です。HTMLにJavaコードを埋め込み、実行時に動的にWebページを生成します。

JSPの主な特徴

-  **HTMLへのJava埋め込み**
HTMLコード内にJavaロジックを直接記述
-  **自動サーブレット変換**
JSPコンテナがサーブレットに自動変換・コンパイル
-  **MVCのView層**
Model-View-Controllerパターンでのプレゼンテーション層

技術的位置付け

-  **サーバーサイド実行**
Webサーバー上でJavaコードを実行
-  **データベース連携**
JDBCを使用したDB操作が可能
-  **サーブレットとの連携**
サーブレットで処理、JSPで表示の分離

💡 動的Webページ生成の仕組み

リクエスト時にJavaコードを実行し、HTMLとして出力。データベースから取得した情報を動的に表示できます。

JSPアーキテクチャ

JSPライフサイクル

1

変換 (Translation)

JSPファイルをサーブレットのJavaソースコードに変換

2

コンパイル (Compilation)

Javaソースコードをクラスファイルにコンパイル

3

初期化 (Initialization)

jspInit()メソッドを実行してリソース初期化

4

実行 (Execution)

_jspService()メソッドでリクエスト処理・レスポンス生成

5

破棄 (Destruction)

jspDestroy()メソッドでリソース解放

MVCパターンでのJSP

Model (モデル)

JavaBeans、Entity、DTO
ビジネスロジック・データ処理

View (ビュー)

★ JSPが担当
プレゼンテーション層・UI表示

Controller (コントローラー)

サーブレット、フレームワーク
リクエスト制御・処理の振り分け

⚙️ JSPコンテナの役割

- JSPの自動変換・コンパイル
- リクエスト・レスポンス管理
- セッション・スコープ管理

環境準備

セットアップ手順

1

JDK8以降のインストール

Oracle JDKまたはOpenJDKを公式サイトからダウンロード・インストール

2

Apache Tomcat 9.0以降

サーブレットコンテナとしてTomcatをダウンロード・解凍・設定

3

IDE環境の構築

Eclipse、IntelliJ IDEA、VSCodeなど開発環境の準備

4

最初のJSPプロジェクト作成

Dynamic Web Projectの作成とHello World JSPページの実装

必要なツール



JDK 8+

Java開発キット



Apache Tomcat

Webアプリケーションサーバー



IDE

Eclipse / IntelliJ IDEA / VSCode



Webブラウザー

Chrome / Firefox / Edge

web.xmlの基本設定

```
<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

JSP構文要素

⚙️ ディレクティブ (Directive)

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
```

JSPページの設定・属性を定義

</> スクリプトレット (Scriptlet)

```
<% String name = "太郎"; %>
```

Javaコードを直接記述

= 式 (Expression)

```
<%= name %>
```

変数や式の値をHTMLに出力

! 宣言 (Declaration)

```
<%! private int counter = 0; %>
```

メソッドや変数をクラスレベルで宣言

💬 JSPコメント

```
<%-- JSPコメント --%>
```

サーバーサイドコメント (HTMLに出力されない)

主要ディレクティブ

page: ページ属性の設定

include: 静的ファイルのインクルード

taglib: カスタムタグライブラリの使用

暗黙オブジェクト

主要な暗黙オブジェクト



request

HttpServletRequest - リクエスト情報、パラメータ取得



response

HttpServletResponse - レスポンス設定、リダイレクト



session

HttpSession - セッションデータの保存・取得



application

ServletContext - アプリケーション全体で共有するデータ



out

JspWriter - HTML出力ストリームへの書き込み

その他のオブジェクト



pageContext

他の暗黙オブジェクトへのアクセス、スコープ管理



config

ServletConfig - サーブレット初期化パラメータ



page

現在のJSPインスタンスへの参照 (this)



exception

エラーページでのみ利用可能な例外情報

スコープの範囲

page: 現在のページ内

request: 1回のリクエスト内

session: ユーザーセッション内

application: アプリ全体

アクション要素

jsp:useBean

```
<jsp:useBean id="user" class="com.example.User" scope="session" />
```

JavaBeansオブジェクトの作成・取得

↓ jsp:setProperty

```
<jsp:setProperty name="user" property="name" value="太郎" />
```

JavaBeansのプロパティに値を設定

↑ jsp:getProperty

```
<jsp:getProperty name="user" property="name" />
```

JavaBeansのプロパティ値を取得・出力

+ jsp:include

```
<jsp:include page="header.jsp" />
```

他のJSPページを動的にインクルード

↪ jsp:forward

```
<jsp:forward page="welcome.jsp" />
```

他のページへのフォワード処理

⚙️ jsp:param

```
<jsp:include page="menu.jsp">  
  <jsp:param name="selected" value="home" />  
</jsp:include>
```

インクルード・フォワード時のパラメータ渡し

アクション要素の特徴

- XML形式の構文
- 実行時に処理される
- 再利用性と保守性を向上

EL式とJSTL

Expression Language (EL)

\$ 基本構文

```
${user.name} ${param.id} ${sessionScope.cart}
```

スコープ内のオブジェクトやプロパティへの簡単アクセス

演算子

```
${price * quantity} ${name eq 'admin'} ${not empty list}
```

算術、比較、論理演算子を使用可能

JSTLコアタグ

? 条件分岐

```
<c:if test="${user != null}">...</c:if>
```

if文、choose-when-otherwiseで条件処理

JSTLループ・操作

🔄 繰り返し

```
<c:forEach items="${users}" var="user">
  ${user.name}
</c:forEach>
```

リストや配列の繰り返し処理

📄 変数操作

```
<c:set var="message" value="こんにちは" />
```

変数の設定、削除、スコープ管理

🔗 URL処理

```
<c:url value="/user" var="userUrl">...</c:url>
```

URL生成、パラメータ付加、エンコード

JSTLタグライブラリ宣言

```
<%@ taglib
uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>
```

フォーム処理とDB連携




HTMLフォームデータの取得

-  リクエストパラメータ
request.getParameter()でフォーム値を取得
-  バリデーション
入力値の検証、エラーハンドリング
-  セッション管理
ユーザー状態の保持、ログイン情報
-  Cookie操作
クライアント側でのデータ永続化

🛡 セキュリティ対策

- XSS対策: HTMLエスケープ処理
- CSRF対策: トークン検証

JDBCでのDB連携

-  接続確立
DriverManager.getConnection()
-  データ取得
PreparedStatementでSELECT文実行
-  データ更新
INSERT, UPDATE, DELETE文の実行
-  トランザクション
commit, rollbackでデータ整合性保持
-  リソース解放
Connection, Statement, ResultSetのclose()

ベストプラクティス

- コネクションプールの使用
- SQLインジェクション対策

実践応用

ユーザー管理システムの構築

- 1 ログイン・ログアウト機能
セッション管理、認証処理、リダイレクト制御
- 2 ユーザー一覧表示
JSTLでのリスト処理、ページング対応、検索機能
- 3 ユーザー登録・更新
フォームバリデーション、データベース更新、エラーハンドリング
- 4 ユーザー削除
確認ダイアログ、論理削除、関連データの整合性管理

実装のポイント

- MVCパターンでの役割分担
- コンポーネントの再利用性を意識

主要機能と技術要素

- 🛡️ 認証と許可
セッションベース認証、権限制御
- ❗ 入力検証
サーバーサイドバリデーション、エラー表示
- ☰ ページング
LIMIT・OFFSET、ページナビゲーション
- 🔍 検索機能
動的SQL生成、部分一致検索
- 🔔 メッセージ表示
成功・エラーメッセージの管理
- 📱 レスポンシブデザイン
CSSフレームワーク、モバイル対応

最新開発トレンド

- Ajaxでの非同期処理
- RESTful APIとの連携

ベストプラクティス

設計指針



ビジネスロジックの分離

JSPは表示層のみ、処理ロジックはサーブレットやJavaBeansで



カスタムタグの活用

再利用可能なコンポーネントとしてタグライブラリを作成



コンポーネントの再利用

ヘッダー、フッター、メニューは共通コンポーネント化



EL式とJSTL優先

スクリプトレットよりEL式やJSTLを使用して可読性向上

⚠ 避けるべきパターン

- JSP内に直接ビジネスロジックを記述
- 大量のJavaコードをスクリプトレットで記述

セキュリティとパフォーマンス



XSS対策

ユーザー入力には必ずHTMLエスケープ、JSTLでの
escapeXml



SQLインジェクション対策

PreparedStatementの使用、パラメータバインディング



CSRF対策

トークンを使用したリクエスト検証の実装



パフォーマンス最適化

JSPプリコンパイル、コネクションプールの使用



メモリ管理

セッション使用の最小化、リソースの適切な解放

👍 推奨アプローチ

- コードレビューでの品質管理
- 単体テストでの品質保証
- 継続的なリファクタリング

よくある質問（FAQ）

❓ Q1. 文字化けが発生します。解決方法を教えてください。

pageディレクティブでcontentTypeとpageEncodingをUTF-8に設定し、web.xmlでも文字エンコーディングフィルタを設定してください。

❓ Q2. セッションタイムアウトの設定方法は？

web.xmlの<session-config><session-timeout>30</session-timeout></session-config>で単位で指定できます。

❓ Q3. JSPとサーブレットの使い分けは？

サーブレットはビジネスロジックと制御処理、JSPは表示処理という役割分担が基本です。

❓ Q4. includeとforwardの違いは何ですか？

includeは他のページを現在のページに組み込み、forwardは制御を他のページに移します。

❓ Q5. デプロイメントエラーの対処法は？

Tomcatのlogsフォルダ内のエラーログを確認し、クラスパスやweb.xmlの設定をチェックしてください。

❓ Q6. EL式が動作しない場合は？

web.xmlのServletバージョンを確認し、2.4以上でELが有効になります。isELIgnored="false"で明示的に有効化も可能です。

❓ Q7. JSPのパフォーマンス最適化のコツは？

プリコンパイル、コネクションプール、セッション使用の最小化、適切なキャッシュ設定が効果的です。

サポートリソース

- Oracle JSPドキュメント
- Tomcat公式サイト
- Stack Overflow
- Java開発コミュニティ

参考リンク・資料

📖 公式ドキュメント

Oracle JSP Documentation

JSP仕様と公式チュートリアル

Apache Tomcat Documentation

Tomcatサーバーの設定と管理

JSP 2.3 Specification

最新のJSP仕様書

🏠 オンライン学習プラットフォーム

Oracle Java Tutorials

体系的なJava Web開発学習

Codecademy / Udemy

動画・インタラクティブ学習コース

👥 コミュニティ・フォーラム

Stack Overflow / Qiita

技術質問・記事・サンプルコード

GitHub

JSPサンプルプロジェクト

→ 次のステップ

サーブレット基礎

コントローラ層の理解を深める

Spring MVC

モダンなWebフレームワーク

RESTful API設計

API開発のベストプラクティス

Thymeleaf

次世代テンプレートエンジン

♥ 学習完了おめでとうございます！

このJSP基礎スライドで学んだ知識を活かして、実践的なWebアプリケーション開発に挑戦してください。