Webアプリケーション

2019年1月7日 月曜日　　12:05

サーバサイドアプリケーション

APサーバに配置され、リクエストを受け取った際にサーバ上でWebページを作成する。リアルタイム性の高いAPを作成可能

サーブレット：Java言語を用いたWebアプリケーション。特定のOSやハードウェアに依存することなく、サーブレットAPIを実装したあらゆるWebサーバーで稼働させることができる

ASP：ASP.NETは、Microsoftが開発したWebアプリケーションフレームワークで、動的なWebサイトやWebアプリケーション開発が可能

CGI：動的なWebサイトを構築する技術。CGIは様々な言語で記述できるが、主にParlやC言語などが用いられる。小〜中規模開発で用いる

クライアントサイドアプリケーション

Webページの内容に含まれてクライアントにダウンロードされ、Webブラウザ内で動作する。ユーザビリティの優れたAPを開発可能

JavaScript：Webブラウザにダウンロードされ、Webブラウザ内に動的な動きを持たせることが可能

Applet：webブラウザにダウンロードされ、Webブラウザ内で動作するJavaプログラム

データベース

Oracle、DB Ⅱ、MySQL、PostgreSQL、SQLServer、Accessなど

JSP・サーブレット

2019年1月7日 月曜日　　12:37

サーブレット

通常のJavaプログラムと同様に、メソッドを使ってHTMLコードを書き出す

リクエストを受け取り、JSPやJavaBeansなど他のリソースに処理を遷移させるといった制御の役割を担当することが一般的

JSP

HTMLコードを直接記述した中に、Java言語のコードを埋め込む

表示部分に使用することが一般的

MVCアーキテクチャ

2019年1月7日 月曜日　　12:42

MVCアーキテクチャ(APを次の3つの要素に分けて構築する考え方)

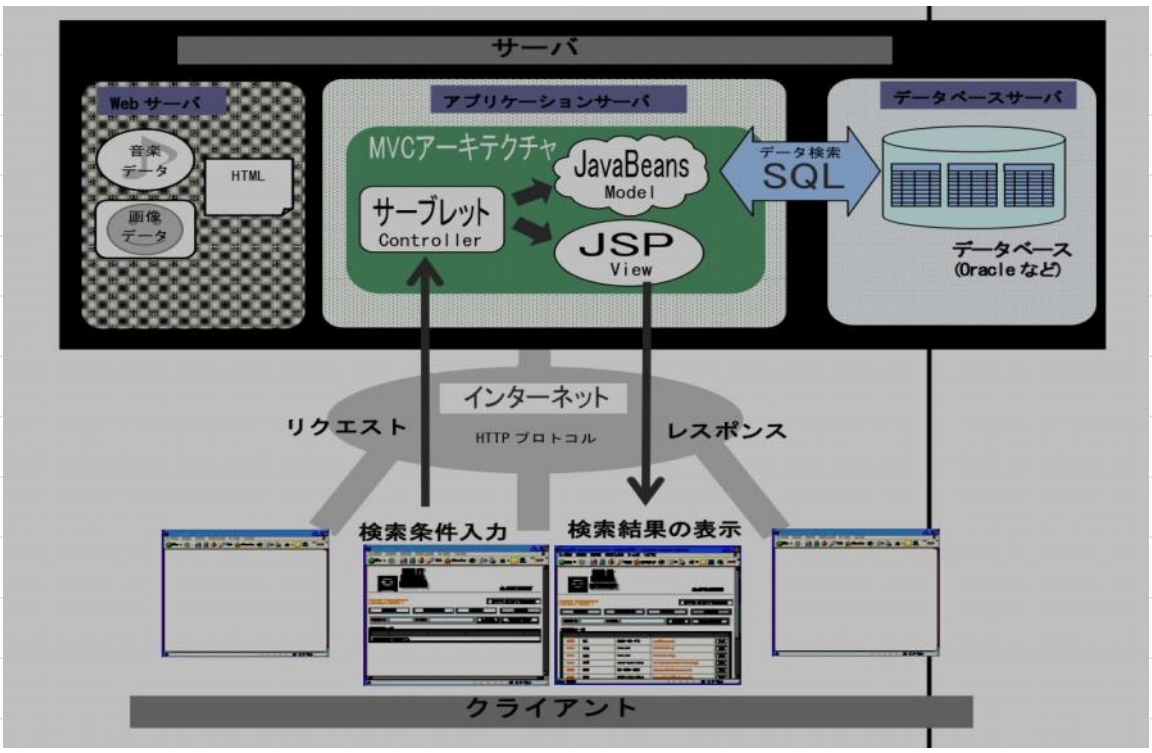
Model：処理の中核を担う、機能ごとの処理　　JavaBeans(通常のJava)

View：表示・出力を担う　　JSP

Controller：ViewとModelを制御、処理の振り分け　　サーブレット

3つに役割分担を行うことで、APの拡張、修正が非常にしやすくなる利点がある

MVCを用いて作成するWebアプリケーションは一般的に以下の構成になる



Servletの実行環境

2019年1月7日 月曜日　　12:58

J2EE

JavaによるWebアプリケーションの仕様のことをいう

サーブレットの開発は、このJ2EEの仕様に準拠している

J2EE APIはWebアプリケーションに使用されるクラス群でサーブレットやJSPの開発をするためには必須

Javaの基本機能のセットであるJ2SEにサーバ用のAPIや諸機能を付加したもの

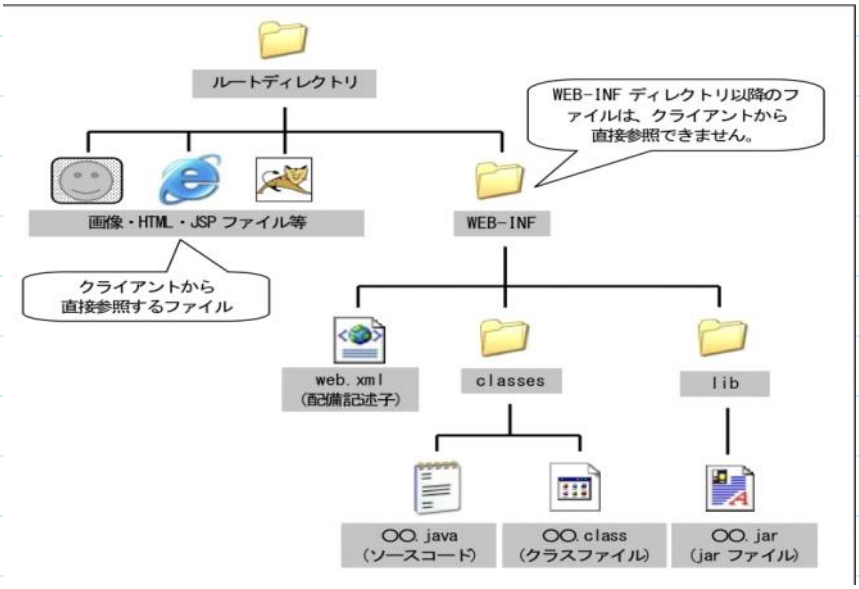
サーブレットコンテナ

サーブレットを実行するための環境

Webサーバーと連携してクライアントからのリクエストに従いサーブレットを実行させることができる

Tomcatが主流でインストロール時にJ2EE APIも一緒にインストールされる

ディレクトリ階層は以下を参照



配備記述子(Web.xml)

2019年1月7日 月曜日　　15:30

配備記述子

デプロイメントディスクリプタとも呼ばれ、Webアプリケーションの動作を規定する設定ファイル

サーブレット定義、セッション定義、セキュリティ定義、ウェルカムページ・エラーページ設定などの設定を行うことができる

J2EEの仕様では、配備記述子として、web.xmlというファイルをWEB-INFフォルダの直下に設置する必要がある

web.xml

```
1.  <?xml version="1.0" encoding="ISO-8859-1"?>
2.
3.  <web-app xmlns="http://java.sun.com/xml/ns/j2ee"
4.      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5.      xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
6.  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">
7.
8.    <servlet>
9.
10.     <servlet-name>sampleservlet</servlet-name>
12.     <servlet-class>SampleServlet</servlet-class>
13.
14.   </servlet>
15.
16.   <servlet-mapping>
17.
18.     <servlet-name>sampleservlet</servlet-name>
19.
20.     <url-pattern>/sample</url-pattern>
21.
22.   </servlet-mapping>
23.
24. </web-app>
```

サーブレットに付ける名前

同名

サーブレットクラス名

URL

formのactionに指定する遷移先とurl-patternは一致する必要がある

サーブレットクラス

2019年1月7日 月曜日　　16:22

javax.servlet.http.Http.Servlet

サーブレットクラスのスーパークラスとなるクラス

サーブレットを作成するには、各メソッドを適宜オーバーライドする

サーブレットの代表的なメソッドは以下を参照

javax.servlet.http.HttpServlet	
戻り値	説明
void	init(ServletConfig config) Servlet コンテナが実行するメソッドです。サーブレットの起動時に初期化処理をします。
void	doGet(HttpServletRequest req, HttpServletResponse res) HTTP の GET リクエストに応じて呼び出されます。
void	doPost(HttpServletRequest req, HttpServletResponse res) HTTP の POST リクエストに応じて呼び出されます。
void	service(HttpServletRequest req, HttpServletResponse res) public で定義された service メソッド経由で標準 HTTP リクエストを受け取り、それらをこのクラスで定義された doXXX メソッドにディスパッチします。
void	service(ServletRequest req, ServletResponse res) HTTP リクエストを受け取り、それらをこのクラスで定義された doXXX メソッドにディスパッチします。
void	destroy() Servlet コンテナが実行するメソッドです。サーブレットがサービス提供を 停止するときに実行され、終了処理をします。

リクエストインタフェース

2019年1月7日 月曜日　　16:27

javax.servlet.http.HttpServletRequest

クライアントから送信されたリクエスト情報をラップするインタフェース

HttpServletRequestインタフェース型のオブジェクトをリクエストオブジェクトと呼ぶ場合もある

クライアントから送信された、入力パラメータ、ブラウザ情報、リクエストヘッダ情報などを取得することができる

HttpServletRequestインタフェースの代表的なメソッドは以下を参照

javax.servlet.http.HttpServletRequest	
戻り値	説明
Cookie[]	getCookies() このリクエストと一緒にクライアントから送られてきた全てのCookie オブジェクトの配列を返します。
String	getHeader(String name) 指定されたリクエストヘッダの値を String として返します。
String	getParameter(String name) リクエストパラメータの値を String 型のオブジェクトで返しますが、パラメータが存在しない場合は null を返します
HttpSession	getSession() このリクエストに関連づけられている現在のセッションを返します。
void	setCharacterEncoding(String env) このリクエストのメッセージボディで使われている文字エンコーディング名を上書きします。

レスポンスインタフェース

2019年1月7日 月曜日　　16:33

javax.servlet.http.HttpServletResponse

クライアントに返すレスポンス情報をラップするインタフェース

HttpServletResponseインタフェースのオブジェクトをレスポンスオブジェクトと呼ぶ場合もある。クライアントに返すWebページ情報、コンテンツタイプ、レスポンスヘッダ情報などを設定可能

HttServletResponseインタフェースの代表的なメソッドは以下参照

javax.servlet.http.HttpServletResponse	
戻り値	説明
void	addCookie(Cookie cookie) 指定された Cookie をレスポンスに追加します。
void	setHeader(String name, String value) 指定された名称で指定された値を持つレスポンスヘッダを設定します。
void	setStatus(int sc) このレスポンスのステータスコードを設定します。
void	setContentType(String type) クライアントに送り返されるレスポンスのコンテントタイプをセットします。

リクエストパラメータ

2019年1月7日 月曜日　　16:39

リクエストパラメータ

クライアントからサーバに送信される値にリクエストパラメータがあり、formから送信されるデータもリクエストパラメータと呼ぶ

GETメソッド

ブラウザに直接URをL指定してのリクエストや、フォームからのリクエストのデフォルトでの送信方法はGETが用いられる
リクエストパラメータはURLの後ろにクエリ文字となって付加される

POSTメソッド

リクエストパラメータはURLに付加されず、内部で送信される。
個人情報やパスワードなど、セキュリティ上URLに付加するのが望ましくないデータや、長い文字列はPOSTメソッドで送信する

サーブレットで、formから入力されたデータを取得するには
HtttServletRequestインタフェースのgetParameter()メソッドを使用
リクエストパラメータの値をString型のオブジェクトで返すが、
パラメータが存在しない場合はnullを返す

文字エンコーディング

2019年1月8日 火曜日　　9:46

リクエストパラメータに対する文字エンコーディング

リクエストパラメータをどのような文字エンコードで解釈するかを指定

setCharacterEncoding(String env)メソッドを使用

引数に文字コードを指定し、リクエスト情報をエンコーディングする

レスポンスに対する文字エンコーディング

Webアプリケーションからクライアントに送られるデータを、

クライアント側でどの文字コードで解釈するかを指定

setContentType(String type)メソッドを使用

引数にクライアントに送信するデータの種類を指定可能

e.g. response.setContentType(“text/html;charset=“Shift_JIS”);

リクエストとレスポンスのどちらのエンコーディングの設定が抜けていても文字化けを起こす原因になるので必ず両方設定する。

MIMEタイプ：ファイルに含まれているデータのタイプを指定する

ファイル形式	一般的な拡張子	MIME タイプ
テキスト	.txt	text/plain
HTML 文書	.htm.html	text/html
XML 文書	.xml	text/xml
JavaScript	.js	text/javascript
CSS	.css	text/css
GIF 画像	.gif	image/gif
JPEG 画像	.jpg.jpeg	image/jpeg
PNG 画像	.png	image/png
Word 文書	.doc	application/msword

HTTP

2019年1月8日 火曜日　　10:00

HTTPにおけるWebサーバとWebクライアントの対話は要求と返答の2つで成り立っている。このやり取りに欠かせないのがヘッダ情報

リクエストヘッダ

HTTPリクエストに付加されるWebブラウザの種類やバージョン、使用可能な言語、対応できるMIMEタイプなどのクライアント側の情報
リクエスト行、リクエストヘッダ、空白行、ドキュメントの本体から成る

レスポンスヘッダ

HTTPレスポンスに付加されるドキュメントの記述に使われている言語、ページの送信時のファイル圧縮形式、送信するドキュメントのMIMEタイプなどのサーバの情報

HTTP リクエストメッセージの構造

リクエストライン	POST / servlet / Register HTTP / 1.1
ヘッダ	
一般ヘッダ	Connection : Keep-Alive Cache - Control: no-cache
リクエストヘッダ	Accept : image / gif, image / x-xbitmap, image / jpeg , */* Accept-Language : ja Accept-Encoding : gzip, deflate User-Agent : Mozilla / 4.0 (Compatible ; MSIE 6.0 ; Windows NT 5.0) Host : localhost
エンティティヘッダ	Content-Type : application/x-www-form-urlencoded
空白行	
メッセージボディ	login=Delon&password=kenit&password=kenit (POST メソッドでデータが送られる場合ココに入れられる)

主なリクエストヘッダ

ヘッダ名	説明
Accept	利用可能なメディアの種類(複数指定可能)
Accept-Language	利用可能な言語の種類(複数指定可能)
Accept-Encoding	利用可能なエンコーディング形式(複数指定可能)
User-Agent	ブラウザのシグネチャ情報(ブラウザや OS の情報が含まれているもの)
Host	リクエスト先のサーバ名(HTTP1.1 では必須)

HTTPリクエストヘッダの利用

2019年1月8日 火曜日　　9:56

サーブレットで、HTTPリクエストヘッダの情報を取得するには
リクエスト情報を保持するリクエストオブジェクトのメソッドを利用
使用する主なメソッドを以下に示す

javax.servlet.http.HttpServletRequest インタフェース	
戻り値	説明
String	getHeader(String name) 指定されたリクエストヘッダの値を String オブジェクトとして返します。
Enumeration	getHeaderNames() このリクエストに含まれる全てのヘッダ名の Enumeration オブジェクトを返します。
String	getMethod() GET や POST などの HTTP リクエストメソッドの名前を所得する。
StringBuffer	getRequestURL() クライアントがこのリクエストを生成するのに使った URL を再構築します。
String	getProtocol() リクエストのプロトコル名とバージョンを HTTP/1.1 のように プロトコル名/メジャーバージョン番号.マイナーバージョン番号 の形式で返します。

ステータスコード

2019年1月8日 火曜日　　17:44

ステータスコード

レスポンスに含まれる情報で、サーバがクライアントのリクエストを正確に処理したかどうかなどを数字を使って表す。

このステータスコードは必ず整数3桁で表現され、100の位の数字から判断し、大きく分けて5種類の意味を持つ

ステータスコード	意味	例
100 ～ 199 ・1xx	HTTP サーバからクライアント(Web ブラウザ)への問い合わせや連絡などを表します。	100:Continue 101:Switching Protocols
200 ～ 299 ・2xx	クライアントが送信した HTTP リクエストが成功したことを表します。	200:OK 201:Created 202:Accepted 203:Non-Authoritative Information
300 ～ 399 ・3xx	要求されたファイルが移動したことを表します。 ※主にリダイレクトに利用されます。	300:Multiple Choises 301:Moved Parmanentry 302:Moved Temporarily
400 ～ 499 ・4xx	クライアント側が原因によるエラー、あるいはサーバからの拒否を表します。※ファイルが存在しない(NotFound)など	400:Bad Request 401:Unauthorized 403:Forbidden 404:Not Found
500 ～ 599 ・5xx	リクエストの処理中にサーバが原因によるエラーが発生したことを表します。	500:Internal Server Error 501:Not Implemented 502:Bad Gateway

正規表現

2019年1月8日 火曜日　　17:48

正規表現

文字列のパターンを表現する表記法

文字列パターンを表すには、メタ文字と呼ばれる特殊文字を組み合わせて使用する。

特殊文字	用法
.	任意の一文字にマッチする。 （改行コード”¥n”にはマッチしない）
*	直前の表現０回以上の繰り返しにマッチする。
+	直前の表現１回以上の繰り返しにマッチする。
?	直前の表現０回か１回にマッチする。
^	通常は行頭にマッチする。 [ ]の中で使用した場合には、「以外」という意味になる。
$	行末にマッチする。
|	Or の意味になる。
{ }	直前の表現の繰り返し回数を表す。{n}は n 回、{n,}は n 回以上、{n,m}は n 回以上 m 回以下の繰り返しにマッチする。
[ ]	[ ]内の１文字にマッチする。
( )	表現内でのグループを作る。また、マッチした値を順番に取り出すことができる。
¥	この記号を前に付けることで、表示不可能な文字や特定の意味を持った文字の集合にマッチする。上記の特殊文字にマッチさせる場合は”¥”記号を用いて、”¥.”や”¥*”などのように表現する。

特殊記号	用法
¥c{x}	CTRL-“x”に該当するコントロール文字にマッチします。 ｘは大文字のＸでも構いません。
¥f	フォームフィードにマッチします。
¥n	ラインフィード（改行コード）にマッチします。
¥r	キャリッジリターン（改行コード）にマッチします。
¥s	ホワイトスペース（空白・ラインフィード・キャリッジリターン・タブ・垂直タブなど）にマッチします。(”[¥f¥n¥r¥t¥v]”とした場合と同じです。)
¥S	ホワイトスペース以外の文字にマッチします。(”[^¥f¥n¥r¥t¥v]”とした場合と同じです)
¥t	タブにマッチします。
¥v	垂直タブにマッチします。
¥x{n}	文字コード”n”で表される文字１文字にマッチします。(”¥x20”はスペースになります。)

サーブレットで正規表現を用いた文字列検索を行うには、次のクラスを使用する

java.util.regex.Pattern	
戻り値	説明
static Pattern	compile(String regex) 引数には正規表現を表す文字列を指定します。 指定された文字列を正規表現パターンにコンパイルします。
Matcher	matcher(CharSequence input) 引数には検索対象となる文字列を指定します。 指定された検索対象文字列と、正規表現パターンにマッチした部分を抜き出して Matcher オブジェクトを生成します。

java.util.regex.Matcher	
戻り値	説明
boolean	find() このパターンとマッチする次の文字列を検索します。マッチする文字列があれば true、なければ false を返します。
String	group() 前回のマッチで一致した文字列を返します。

セッション

2019年1月9日 水曜日　　10:01

HTTPでのセッション

セッションとは、通信の単位のことをいう

HTTPプロトコルでの通信の場合は、クライアントからのリクエストに応答して、サーバがレスポンスを返すまでが1セッションになる

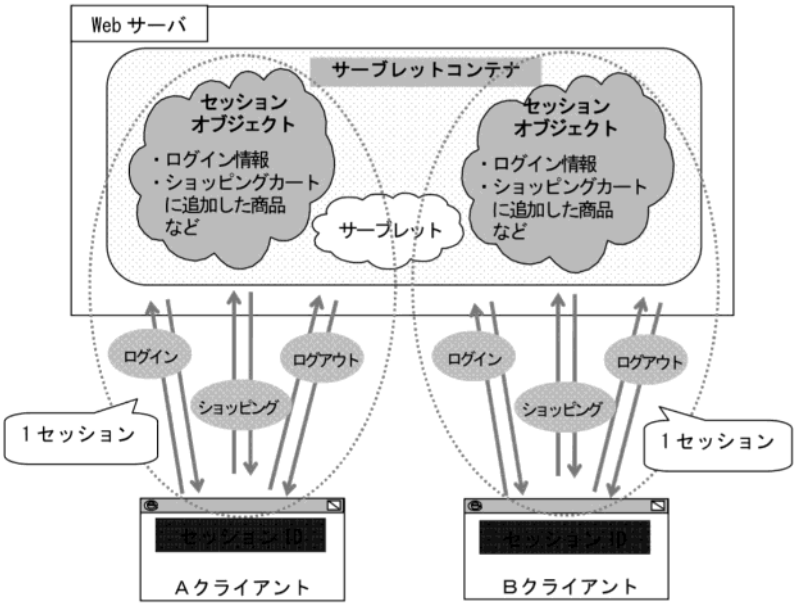
同一のクライアントが再び同一のWebサーバーにリクエストを行うと、別セッションとなり、サーバは同一のクライアントからのリクエストだという判断はできない

このため、サーブレットでは、複数のリクエスト間で同一のクライアントから利用できるセッションオブジェクトが用意されている

セッショントラッキング

複数のセッションを管理すること

サーブレットでは、セッションオブジェクトを用いることで、セッショントラッキングを行うことができる。このセッションオブジェクトはクライアントごとに生成され、セッションIDという番号でクライアントを判別する



セッション管理-1

2019年1月9日 水曜日　　10:17

セッショントラッキングを行なうためのセッションオブジェクトは、
Javax.servlet.http.HttpSessionインタフェースのオブジェクト

javax.servlet.http.HttpSession	
戻り値	説明
void	setAttribute(String name, Object value) 指定された名前でこのセッションにオブジェクトを結びつけます。
Object	getAttribute(String name) 指定された名前でこのセッションに結びつけられているオブジェクトを返します。
void	invalidate() セッションを無効にし、結びつけられている全てのオブジェクトを解放します。
void	setMaxInactiveInterval(int interval) Servlet コンテナがクライアントから最後にリクエストを受けてからこのセッションを無効化するまでの最大の秒数を指定します。

サーブレットでセッショントラッキングを利用する手順

1. セッションオブジェクトの取得
2. セッションオブジェクトからの情報の取り出しと記録
3. セッションの破棄

1.セッションオブジェクトの取得

セッションオブジェクトを取得するにはHttpServletRequest
インタフェースのgetSession()メソッドを使用。

javax.servlet.http.HttpServletRequest	
戻り値	説明
HttpSession	getSession(boolean create) 指定された名前でこのセッションに結びつけられているオブジェクトを返します。 引数が true の場合 2 回目以降のリクエストで、すでにセッションオブジェクトが存在している場合は、それを返し、初めてのリクエスト時など、**まだセッションオブジェクトが生成されていない場合は、新規にセッションオブジェクトを生成して返します。** 引数が false の場合 2 回目以降のリクエストで、すでにセッションオブジェクトが存在している場合は、それを返し、初めてのリクエスト時など、**まだセッションオブジェクトが生成されていない場合は、セッションオブジェクトを生成せずに、null を返します。**
HttpSession	getSession() 引数が true の場合の getSession()メソッドと同じ働きをします。

クッキー-1

2019年1月9日 水曜日　　11:08

クッキー

クライアント(ブラウザ)に保存される情報

ブラウザを閉じる、PCの電源を切るなどをしても再度利用できる

サーブレットでのクッキーの利用手順

1. クッキーオブジェクトの生成
2. クッキーの発行
3. クッキーの取得

1.クッキーオブジェクトの生成

javax.servlet.http.Cookieクラスのオブジェクトとして生成する

javax.servlet.http.Cookie	
コンストラクタ	
Cookie(String name, String value) 指定された名前と値で Cookie を生成します。	
メソッド	
戻り値	**説明**
String	getName() Cookie の名前を返します。
int	getMaxAge() Cookie の最長存続期間の値を秒単位の数値で返します。
void	setMaxAge(int expiry) Cookie の最長存続期間を秒単位で設定します。
String	getValue() Cookie の値を返します。
void	setValue(String newValue) Cookie が生成された後で、新しい値を設定します。

クッキー-2

2019年1月9日 水曜日　　11:16

2.クッキーの発行

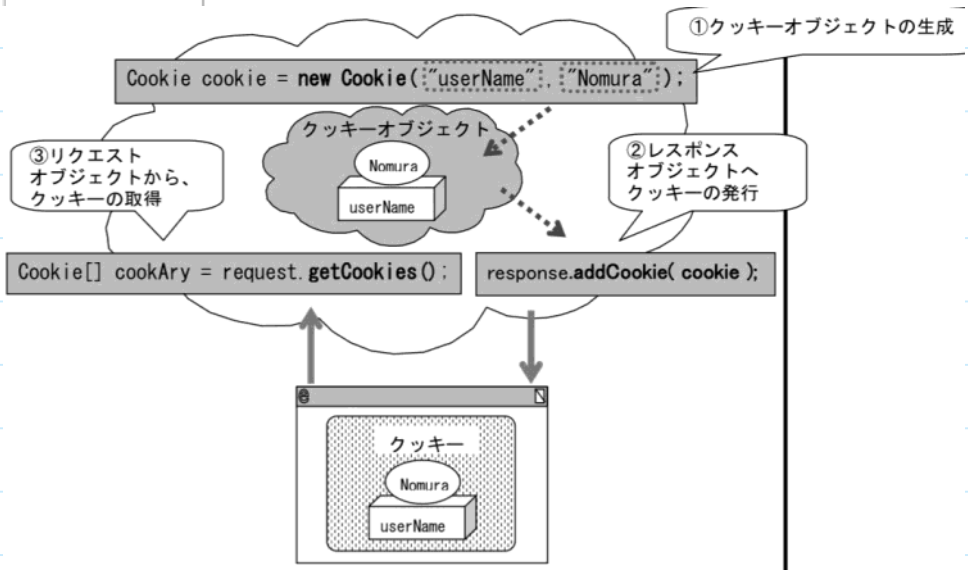
javax.servlet.http.HttpServletResponseインタフェースの下記メソッドを使用

javax.servlet.http.HttpServletResponse	
戻り値	説明
void	addCookie(Cookie cookie) 引数に指定されたCookieをレスポンスに追加します。

3.クッキーの取得

javax.servlet.http.HttpServletResponseインタフェースの下記メソッドを使用

javax.servlet.http.HttpServletRequest	
戻り値	説明
Cookie[]	getCookies() このリクエストと一緒にクライアントから送られてきた全てのCookieオブジェクトの配列を返します。



URLエンコーディング

2019年1月9日 水曜日　　11:20

URLエンコーディング

Web上で送信されるデータをURLで利用可能な文字列に変換するためのエンコーディング

クッキーには用途が決められている記号や使用不可能な文字があるため、URLエンコーディングを行いそれらの文字を別の文字に置き換えることで文字化けや不具合を回避する。

クッキーを設定・取得する際は必ずURLエンコーディングを行う

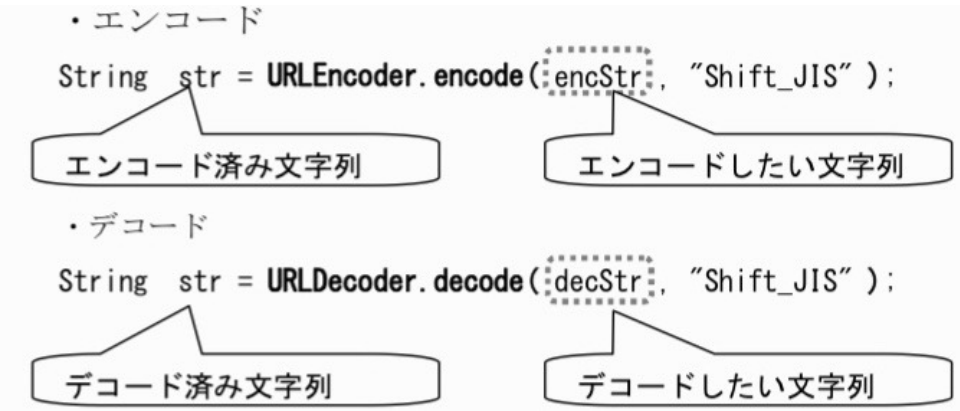
変換前	変換後
A～Z、a～z、0～9（半角英数）	変換しない
* - . @ _	変換しない
半角スペース	＋（半角のプラス）に変換する
その他	「%」＋２桁の１６進数の文字コードに変換する

エンコード：文字列をエンコードに基づいて変換すること

デコード：変換された文字列を元に戻す

URLエンコーディングを行うには、java.net.URIEncoderクラスを使用

デコードを行うには、java.net.URLDecorderクラスを使用



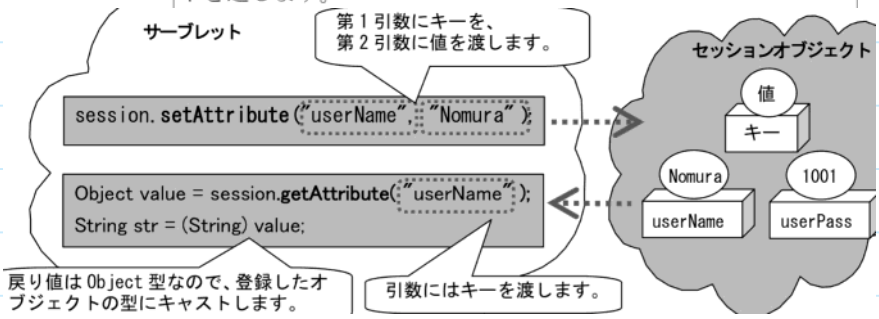
セッション管理-2

2019年1月9日 水曜日　　10:29

2.値の登録と取り出し

セッションオブジェクトに値を登録するには、HttpSessionインタフェースのsetAttribute()メソッドを、取得するには、get Attribute()メソッドを使用し、セッションオブジェクトはキーと値のセットで登録

javax.servlet.http.HttpSession	
戻り値	説明
void	setAttribute(String name, Object value) 指定されたキーでこのセッションにオブジェクトを登録します。
Object	getAttribute(String name) 指定されたキーでこのセッションに登録されているオブジェクトを返します。



3.セッションの破棄

サーブレットとWebブラウザの間のセッションは、一定時間以上アイドル状態が続くと、自動的に無効化される。このアイドル時間はHttpSessionインタフェースのsetMaxInactiveInterval()で指定可能

アイドル時間を待たず明示的に破棄するには、invalidate()を使用

javax.servlet.http.HttpSession	
戻り値	説明
void	invalidate() セッションを無効にし、結びつけられている全てのオブジェクトを解放します。
void	setMaxInactiveInterval(int interval) Servlet コンテナがクライアントから最後にリクエストを受けてからこのセッションを無効化するまでの最大の秒数を指定します。デフォルトは 30 分となっています。

JavaBeans

2019年1月11日 金曜日　　11:27

JavaBeans

Java言語のプログラムを機能ごとにコンポーネント化(部品化)して作成する手法のこと
セットにして利用するデータをまとめて保持するような
コンポーネント化されたオブジェクトのことをBeanという

Beanのルール

1. 引数を持たないpublicなコンストラクタやデフォルトコンストラクタを持つ
2. フィールド全てはprivate
3. フィールドにはアクセス用のメソッド(Setter, Getter)でアクセス

Beanの利点

Beanクラスを利用することで、セットで利用する複数の値をまとめて保持することが可能。違う型のデータをまとめて管理できる。

JSP

2019年1月15日 火曜日　　10:07

JSP

HTMLコードの中にJavaプログラムを記述することで、
Javaプログラムで生成した値をwebページに反映させることが可能
JSPの実行の際は、明示的にコンパイルをする必要はなく、コンテナが内部でコンパイルを行う。

JSPにリクエストが来た際は、コンテナが次のような処理を行う

1. ブラウザからリクエストを受け取る
2. JSPページをJavaコードに変換する
3. 変換されたJavaコードをコンパイルする
4. コンパイルされたクラスファイルをロード(読み込み)する
5. 実行(Webページの作成)

2回目以降のリクエストはロード済みのインスタンスを利用するので処理時間は短くなる

ディレクティブ

2019年1月15日 火曜日　　10:15

JSPでは、Javaの文法以外にJSP固有の記述が存在する
そのひとつがディレクティブである。
コンテナにJSPのページを処理する方法を指示する。

ディレクティブの種類
page：ページの処理方式を設定する
taglib：JSPページが使用するタグライブラリを宣言する
include：JSPページに外部ファイルの内容をJavaコード変換時に挿入

ディレクティブの中でも一番よく使用されるのがpageディレクティブ
pageディレクティブは、属性に値を設定することでそのページが読み込まれるときに自動的に処理が行われる。設定はそのページのみ有効
ページの先頭に記述するのが一般的。複数回記述可能だが、import属性を除いて、同一の属性を複数回指定することはできない。
pageディレクティブの設定は毎回行う。
contentType、importはよく使うので記述できるようにする。

page ディレクティブ要素(一部)

contentType	レスポンスデータの MIME タイプと文字エンコードを指定します。デフォルトは「text/html;charset=8859_1」です。
extends	JSP ページから生成されるサーブレットクラスのスーパークラスを指定します。デフォルトは、コンテナで用意されているクラス(org.apach.jasper.runtime.HttpJspBase)です。他のものを指定する場合は、パッケージ名を含めたクラス名を指定する必要があります。
import	JSP ページでインポートするクラスを指定します。複数インポートする場合には「,」で区切ります。 例) <%@ page import="java.util.*, java.sql.*" %>
info	JSP ページの情報をセットします。 この値は、「javax.servlet.Servlet」の「getServletInfo」で返される値になります。一般的には、プログラムの概要、作者、バージョンなどを記述します。
language	JSP ページで使用する言語を指定します。JSP1.2 の時点で「java」以外ありません。
session	セッションを有効にするか無効にするか指定します。デフォルトは「true」で、セッションが使用可能な状態です。

スクリプティング要素

2019年1月15日 火曜日　　10:27

スクリプティング要素

Javaコードを記述するためのタグ

スクリプトレット、宣言、式の3種類のタグが存在

スクリプトレット

そのページへのリクエストがある度に実行されるJavaプログラム

doGet()やdoPost()に記述したコードと同じ役割となる

e.g. <%　Javaコード　%>

スクリプトレットの中に記述したコードはJSPがJavaコードに変換される際に、_jspService()の中に反映されるコードになる

宣言タグ

宣言タグに記述した処理は、JSPがJavaコードに変換され際に、クラスのフィールドに反映される。変数はインスタンス変数になり、メソッドの宣言をすることも可能。

e.g. <%!　Javaコード　%>

式タグ

式タグは、データを出力するためのスクリプトレット

e.g. <%=値 %>

上記のように記述すると値がWebページに反映される。

スクリプトレットを用いて<% out.print(値); %>と記述可能だが、式タグを用いたほうがより簡潔に出力処理を記述できる

JSPのコメントはクライアントに出力されず、隠しコメントと呼ばれることもある。

<%—　コメント　—%>

HTMLのコメントはクライアントに出力される。

リソースの読み込み

2019年1月15日 火曜日　　14:55

includeディレクティブ

JSPページにテキストやHTML、JSPなどの他のファイルの内容をJavaコード変換時に挿入する。2回目以降の実行はロードされたインスタンスが動作するので、読み込み処理が2回目以降はなくなる。
同じ内容を複数のJSPページに記述する際に、リソースを読み込んで利用することができる。
同じ内容を複数のページに読み込んで利用できるので、リソースの再利用性が高まる。

e.g.　<%@ include file = “hoge” %>
※インクルードするファイルがJSPの場合は、読み込み先の宣言タグに記述した変数やメソッドは、読み込み元で利用可能

includeアクション

読み込み元のJSPの実行時に他のファイルを実行させ、結果を読み込んで表示することができる。
includeアクションは実行時に処理が移り、読み込み先が単独で実行され、処理結果のみ読み込まれる。
したがって、実行中の条件に応じて読み込むファイルが異なるような場合には、includeアクションを利用する。
また、読み込み先のファイルが更新された場合、読み込み先のファイルのみ再ロードし、結果のみ読み込むことができる。

e.g.　<jsp:include page=“include.jsp” />
※インクルードするファイルがJSPの場合は、インクルード先のファイルの宣言タグの内容を読み込み元で利用することはできない。

暗黙オブジェクト

2019年1月15日 火曜日　　10:49

暗黙オブジェクト

JSPでは自動的にオブジェクトを生成してくれて特に宣言しなくても使用できる特別なオブジェクトがあり、これを暗黙的オブジェクトという。スクリプトレット・式で使用する。

out オブジェクト
javax.servlet.jsp.JspWriter 型のオブジェクトです。
out オブジェクトは、クライアントへのレスポンス情報の書き込みに使用されます。

request オブジェクト
HttpServletRequest 型のオブジェクトです。
request オブジェクトはクライアントからのリクエストに関連づけられており、リクエストのパラメータやヘッダ情報の取得を行う事ができます。

response オブジェクト
HttpServletResponse 型のオブジェクトです。
response オブジェクトはクライアントへのレスポンスに関連づけられており、レスポンスヘッダを操作する場合などに使用します。

session オブジェクト
session オブジェクトは HttpSession 型のオブジェクトです。
page ディレクティブの session 属性を false に設定すると使用できなくなります。
（session 属性のデフォルト値は true）

application オブジェクト
ServletContext 型のオブジェクトです。
application オブジェクトは、コンテナとやり取りするための一連のメソッドを提供します。application オブジェクトはアプリケーション全体からアクセス出来るため、getAttribute()メソッドを使用して、アプリケーション全体でデータを共有する事が可能です。

config オブジェクト
ServletConfig 型のオブジェクトです。
config オブジェクトはサーブレットの初期化パラメータにアクセスする為に使用されます。

pageContext オブジェクト
現在の JSP ページに関する情報を提供します。これは「ページ属性(page attribute)とも呼ばれたりします。これには、session オブジェクト、request オブジェクト、response オブジェクト、out オブジェクトへの情報が含まれます。また、フォワード、インクルード、エラー処理のためのメソッドも用意されています。

page オブジェクト
java の this キーワードと同じものを示します。
このオブジェクトは、JSP ページが現在のリクエストの為に生成したサーブレットのインスタンスを表します。あまり使用する事は無いでしょう。

exception オブジェクト
java.lang.Throwable 型のオブジェクトです。
このオブジェクトは、エラーページの呼び出しに繋がるキャッチされない Throwable オブジェクトを表します。page ディレクティブの errorPage 属性を使用する場合、エラーページで exception オブジェクトにアクセスする事により、エラーの内容を突き止めたりする事が出来ます。

使用頻度の高いものは、out, request, response, sessionの4つ

リクエストの転送

2019年1月15日 火曜日　　15:40

forwardアクション

リクエストを他のページに転送することができる。

リクエストディスパッチともいう。

他のページに転送した場合は、同じリクエストオブジェクトを転送先でも使用可能。

e.g. <jsp:forward page=“forward.jsp” />

includeとforwardの違い

includeは読み込み元が読み込み先の内容を取り込んで、読み込み元からレスポンスを返すが、forwardの場合は、処理の流れ自体を転送先に移す。したがって、レスポンスは転送先から返されるところが大きな違いになる。

JSPでのBeanの利用-1

2019年1月16日 水曜日　　14:30

JSPにはBeanを利用するアクションタグが用意されている。
Beanのインスタンス化、プロパティへの値の設定、取得などを行うことができる。アクションタグを用いることで、JSPの特徴でもある、HTMLに近い記述でBeanクラスを利用することが可能になる。
Beanを扱うタグには以下のものがある。

<jsp:useBean>　　Beanのインスタンス化をする
<jsp:setProperty>　　Beanのプロパティ値を設定する
<jsp:getProperty>　　Beanのプロパティから値を取得する

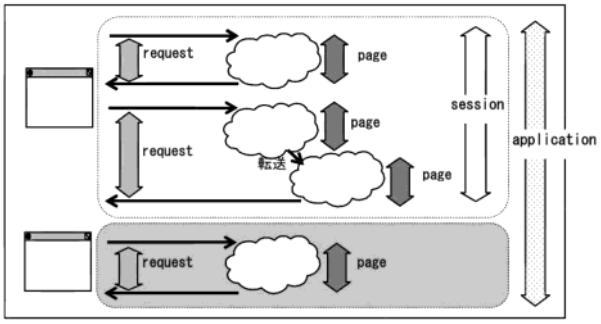
<jsp:useBean>アクションタグ
e.g. <jsp:useBean id="変数名" class="クラス名" scope="scope"/>

<jsp:useBean>アクションタグの属性

id	Bean オブジェクトを使用するときの変数名。Bean に定義された変数やメソッドを利用するにはここに指定した変数名を使用。
scope	Bean オブジェクトの有効範囲。4 つのスコープからいずれかを指定する。デフォルトでは「page」スコープ。 Bean オブジェクトは、この属性で指定した範囲で共通で使用されるので、id 属性で指定した変数名は、「scope」属性で指定したスコープ内では一意でなくてはならない。
下の属性は、次の組み合わせのいずれかで指定します。 **class のみ** **type のみ** **class と type** **beanName と type**	
class	Bean オブジェクトのクラス名を完全修飾名で記述する。
type	Bean オブジェクトを格納する変数の型。 オブジェクトと同じか、スーパークラス型を指定する。
beanName	java.beans.Beans.instantiate()メソッドで要求される Bean の名称を記述する。

scope 属性に指定する値

page	1 つの JSP やサーブレット内のみ利用できる。(デフォルト)
request	リクエストを受け取ってから、レスポンスを返すまでの間の JSP やサーブレットから利用できる。 途中転送をしても同一リクエスト間であれば利用できる。
session	同一セッション内の JSP やサーブレットから利用できる。
application	同一 Web アプリケーション内の JSP やサーブレットから利用できる。



JSPでのBeanの利用-2

2019年1月16日 水曜日　14:41

<jsp:setProperty>アクションタグ

<jsp:setProperty>を記述した箇所は、BeanのsetXXX()メソッドを呼び出した結果で置き換えられる。

e.g. <jsp:setProperty name="変数名" property="プロパティ名" value="値"/>

name属性には、捜査対象の変数名を指定(<jsp:useBean>アクションタグのid属性で指定した名称)

<jsp:getProperty>アクションタグ

<jsp:getProperty>を記述した箇所は、BeangetXXX()メソッドを呼び出した結果で置き換えられる。

e.g. <jsp:getProperty name="変数名" property="プロパティ名" />

name属性には、操作対象の変数名を指定(<jsp:useBean>アクションタグのid属性で指定した名称)

setProperty, getPropertyタグはString型のプロパティの設定、取得しかできない

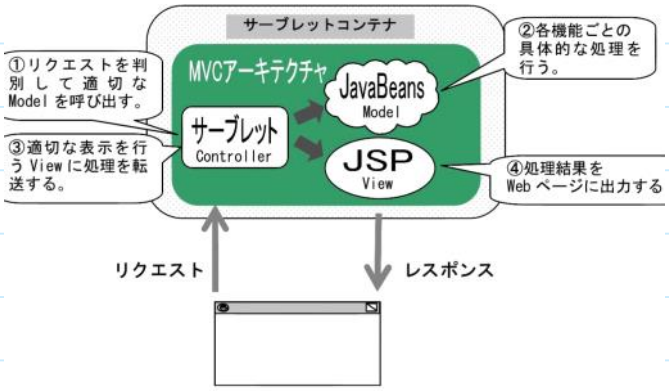
Beanのプロパティがstring型以外の場合には、これまでどおり、セッター、ゲッターを用いて値の設定、取得を行う。

サーブレットとJSPの連携

2019年1月17日 木曜日　　9:47

Webアプリケーションの処理の流れ

1. リクエストをサーブレットが受け取り、処理内容ごとにJavaBeansを呼び出す
2. JavaBeansがリクエストに合った具体的な処理をし、表示に必要なデータを作成する
3. サーブレットが処理結果に合ったJSPに処理を転送する
4. JSPが処理結果をWebページに出力し、レスポンスを返す



サーブレットではリクエストを受け取った後に処理を途中でJSPに転送必要がある。サーブレット・JSPで、処理を途中で他のリソースに転送するには次のオブジェクトメソッドを使用する

javax.servlet.http.HttpServlet	
戻り値	説明
ServletContext	getServletContext() 呼び出したクラスが実行されている ServletContext への参照が返されます。

javax.servlet.RequestDispatcher	
戻り値	説明
void	forward(ServletRequest request, ServletResponse response) Servlet からサーバ上の他のリソース(Servlet、 JSP ファイル、HTML ファイル)へとリクエストを転送します。

使用例)

```
ServletContext context = getServletContext();
RequestDispatcher rd = context.getRequestDispatcher("/転送先の URL");
rd.forward(request,response);
```

※ServletContext オブジェクトとは、Web アプリケーション全体で利用する情報を管理するオブジェクトです。