JavaEEの基礎
ドラフト版

目標

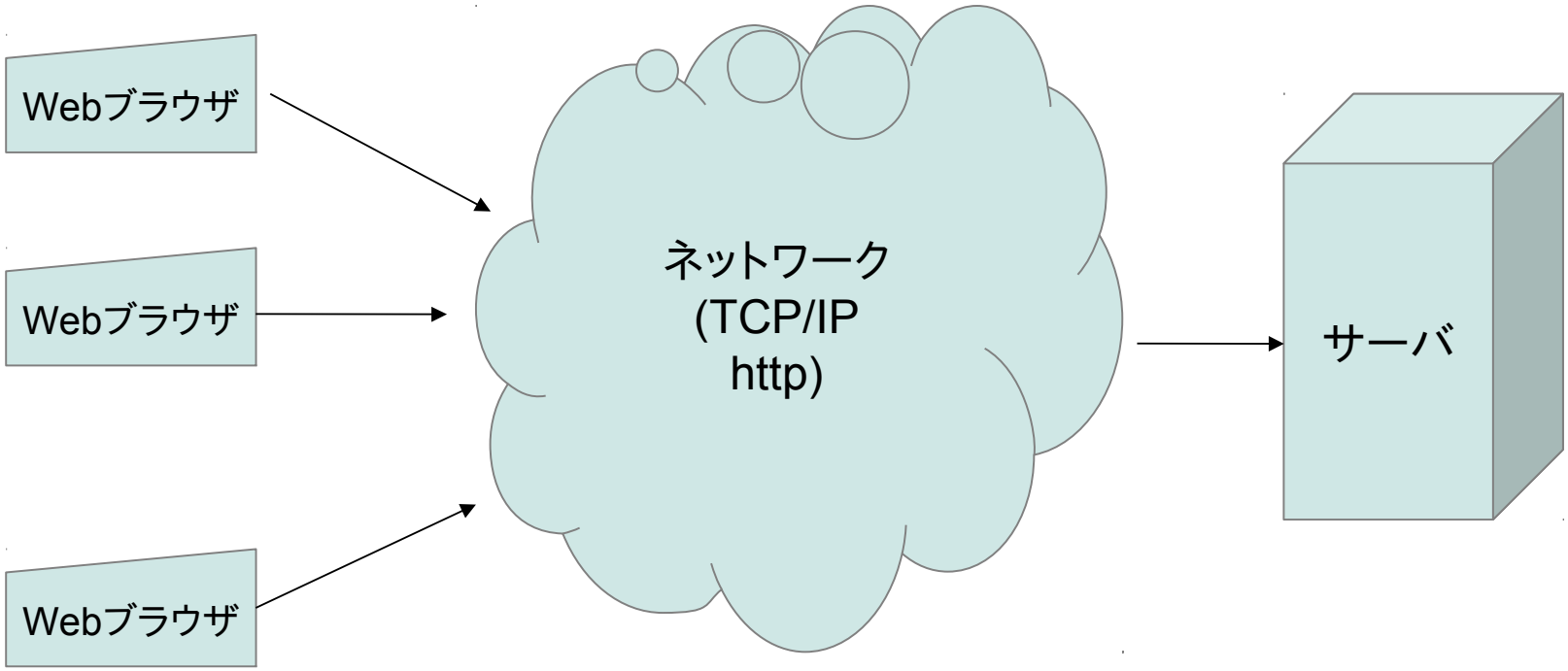
- Webアプリケーションの基本的な仕組みが理解できること。
- JavaEEの構成が理解できること。
- JavaEEの基本的なAPIが理解できること。
- JavaEEで使用される代表的なWebフレームワークについて理解できること。

目次

概要
WebアプリケーションとJavaEE

Webアプリケーションとは

- http/httpsを用いた通信を利用したアプリケーション。
- 主にWebブラウザをクライアントとして利用し、サーバ側で処理を行うことで、動的に結果を得る(単に静的コンテンツを見せるものは「Webアプリケーション」とは呼ばれない)。

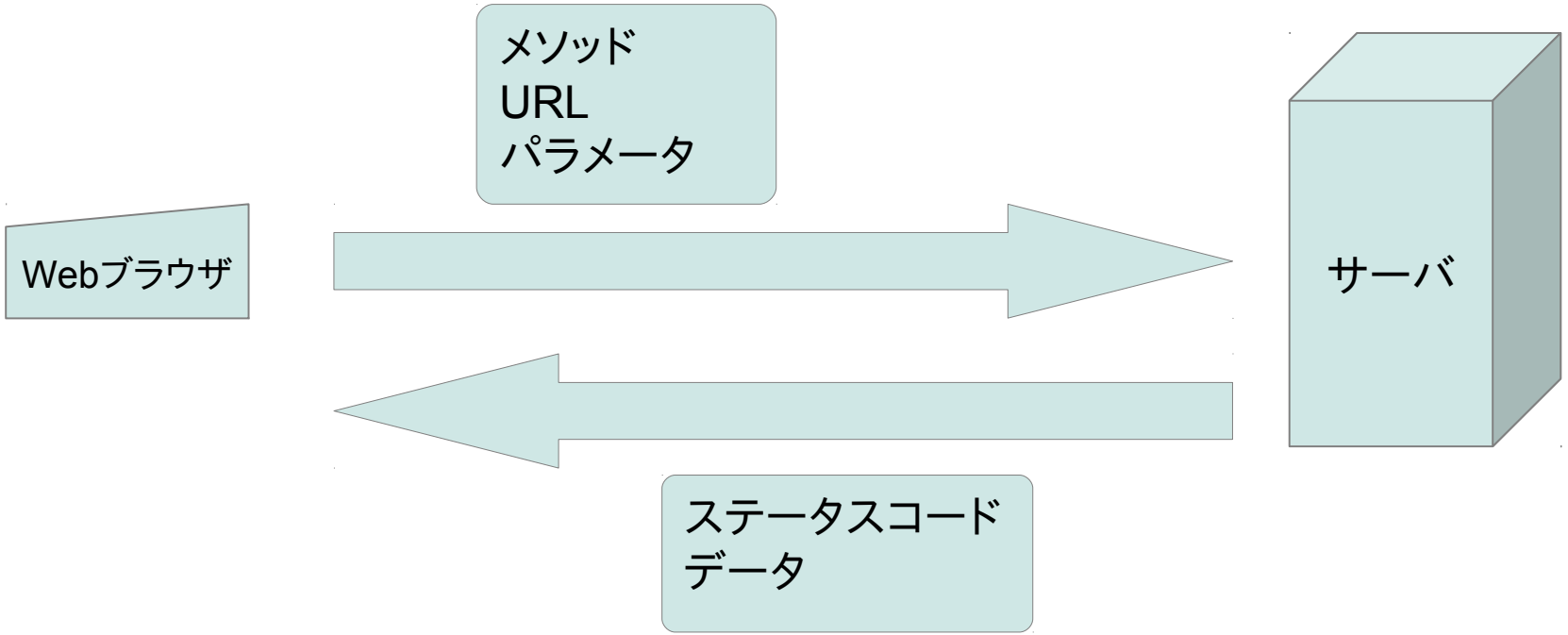


http通信の特徴

- ステートレス
  - サーバとの通信はステートレスなので、1回目のリクエストの状態を2回目リクエスト処理に引き継ぐことができない。
  - つまり買い物かごのように、複数のリクエスト間にまたがって情報を保持することは、本来のhttpの仕組み上はできない。そこでこれを緩和するための方法が幾つか提供されている。
  - ステートレスなので、サーバ側にはクライアントとのやりとりを複数リクエストにまたがって保持しておく必要が無く、少ないサーバで多量のクライアントを処理することができる。
- 複数のリクエストを同時に行うことができる
  例えば1画面の中には文章以外にも、複数の画像が貼られていたりする。Webブラウザは、これらを順番に取り出すのではなく、複数を同時にリクエストすることで、画面全体を描画し終わるまでの時間を短縮することができる。
- 常にクライアント起動
  クライアントからのリクエストにより通信を開始する。クライアント数の増加は、そのままサーバの負荷増大に直結する。
- テキストベース
  ブラウザとサーバとのやりとりの基本はテキストベースなので(バイナリデータも扱える)、電文の内容は理解し易い。

http通信の特徴

- メソッドとステータスコード
  - サーバへの問い合わせの際には、メソッドと呼ばれる文字列のコードを渡すことで、要求する処理内容を大枠を伝える。URLは処理対象となるリソース(データ)の場所を表現する。パラメータを使用して処理内容に付加データを渡すことができる。
  - サーバからの処理結果には、ステータスコードが付加される。ブラウザは、このステータスコードによって処理が正しく処理されたかどうかを判定できる。



メソッド

- 代表的なメソッド
  - GET
    サーバに対してリソースを要求する。
  - POST
    サーバに対してデータを送る。

  - 意味としては上記の通り異なるが、実際にはサーバにパラメータを送信して結果を受け取るという観点では似た用途に使用される。
  - 本来的意味ではGETは「等冪(とうべき)」処理なので、同じパラメータを渡せば、1回目と2回目とで同じものが返ってくるような処理に使用する(キャッシュ可能)。
  - GETリクエストでは、送信可能なデータに上限がある(サーバによって異なるが数kBくらい)。

ステータスコード

- 代表的なステータスコード

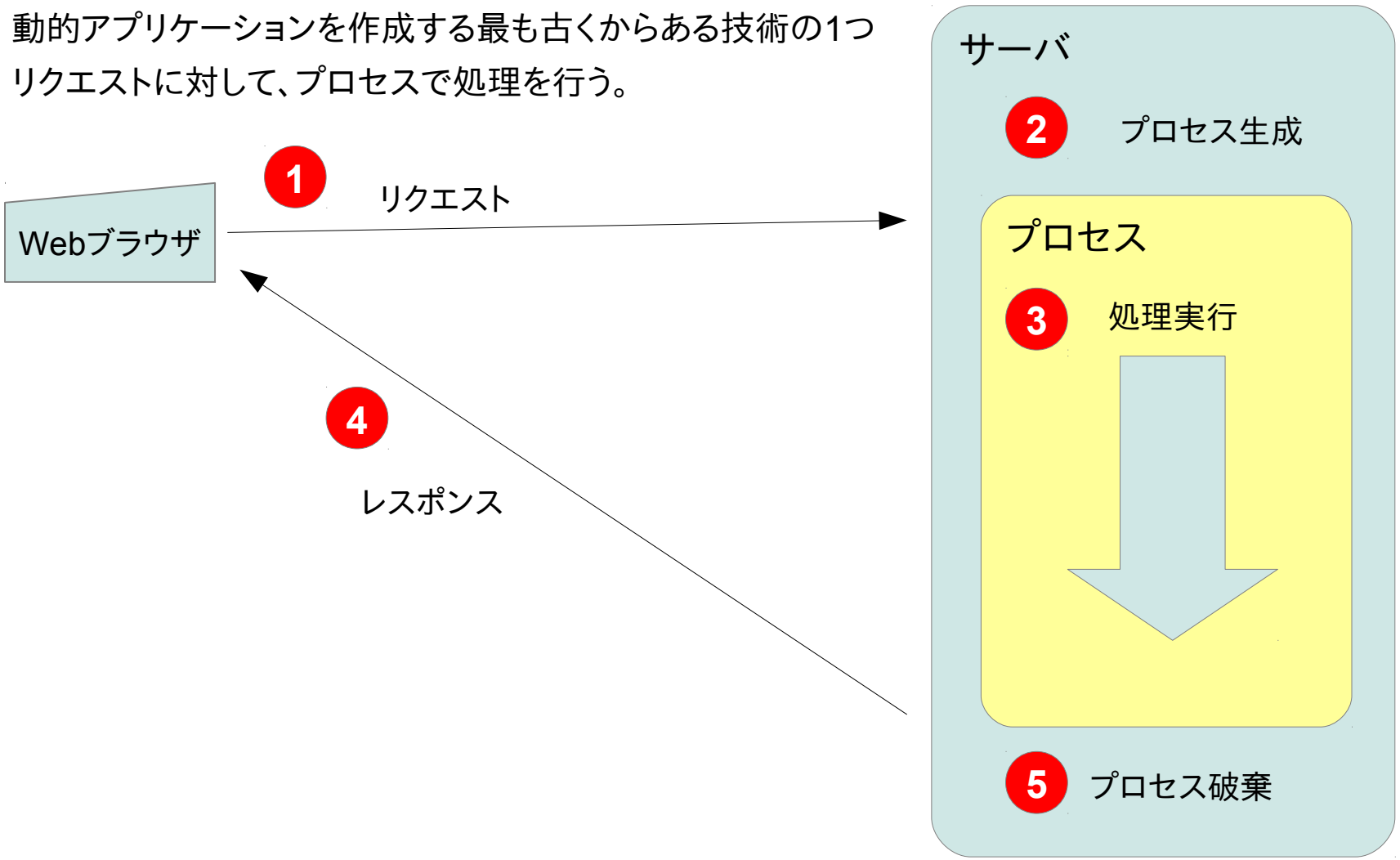
  頭1桁で大まかな意味が規定され、下2桁で細かな意味が付けられている。

  - 200 (成功)
    リクエストが成功した。
  - 301, 302(リダイレクト)
    URLが一時的に移動した。ブラウザはこのステータスコードが返ってくると、自動的に移動先のURLにアクセスする。
  - 401(許可無し)
    パスワードが間違っているなど、認証に失敗した。
  - 404(存在しない)
    指定されたリソースが存在しない。
  - 500(サーバ内部エラー)
    サーバでの処理に失敗した。

Webアプリケーションを実現する仕組み

CGI

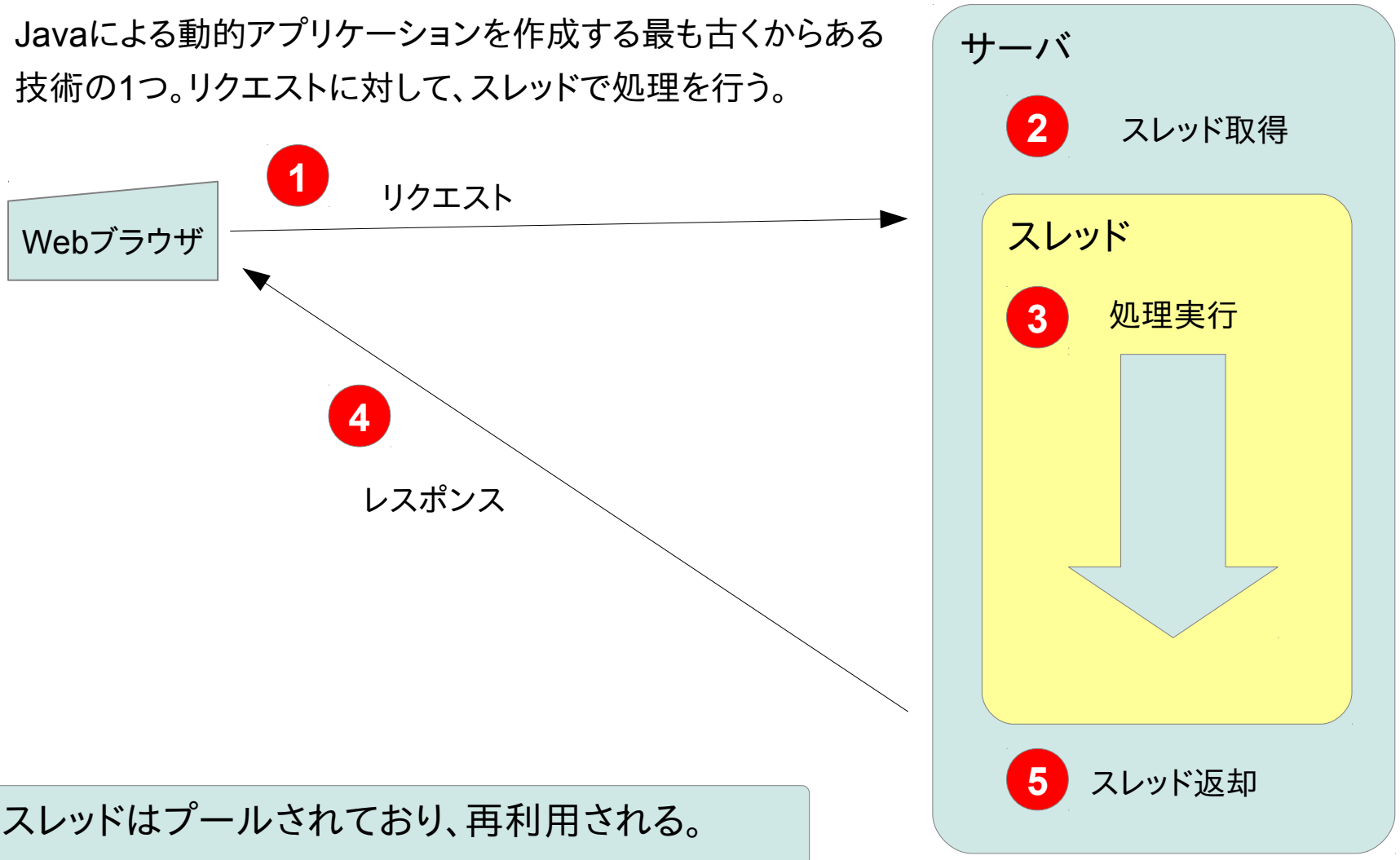
- CGI(Common Gateway Interface)
  動的アプリケーションを作成する最も古くからある技術の1つ
  リクエストに対して、プロセスで処理を行う。



サーブレット

- サーブレット

  Javaによる動的アプリケーションを作成する最も古くからある技術の1つ。リクエストに対して、スレッドで処理を行う。



スレッドはプールされており、再利用される。

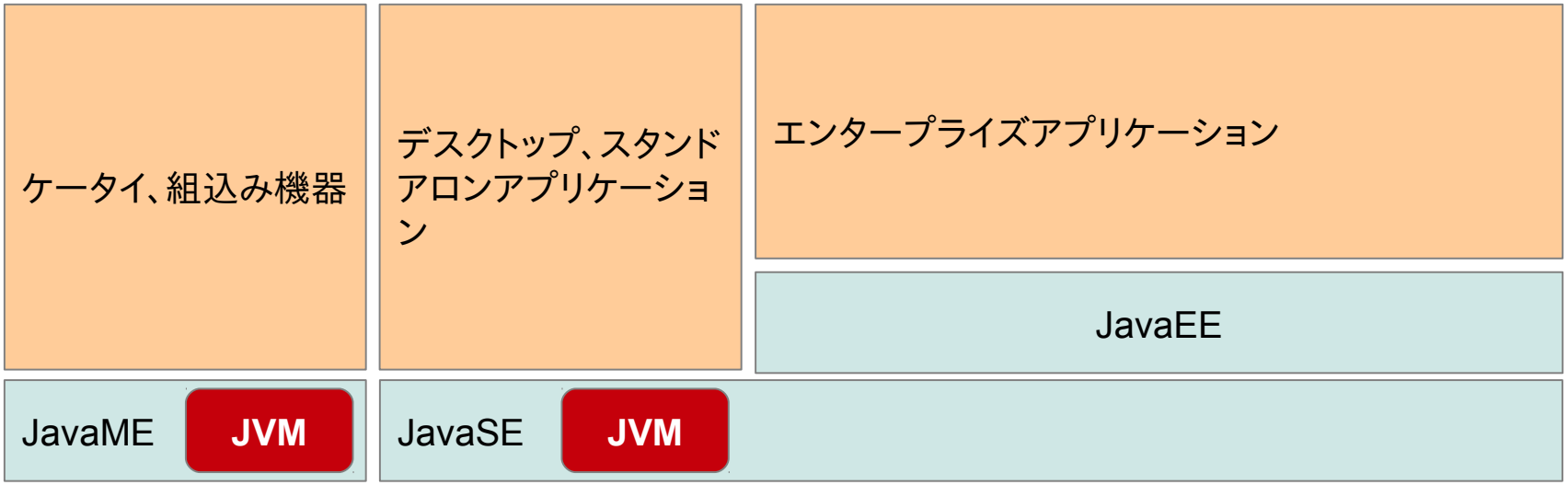
CGIとサーブレット

- CGIは、リクエストごとにプロセスを生成するため、サーブレットよりもサーバの負荷が高い(ただし、CGIにもプロセスをプールする手法がある)。
- サーブレットは、プロセスよりもOSにとって負荷の軽いスレッドを用い、さらにプールを用いてスレッドを再利用するためリクエストあたりの処理に必要となるシステム資源が少なくて済む。
- OSはプロセス単位でリソースの管理を行うため、サーブレットではプログラミングのミスにより、リソース(メモリ、データベース、ファイル)の解放し忘れ(リソースリークと呼ぶ)が起きやすい。
- プロセスはメモリ空間が分かれており、別のプロセスとの分離性が良い。このため、あるプロセスでのプログラミングミスが、他のプロセスに影響を与えにくい。その反面、プロセスの間での情報の共有はスレッドに比べるとやりにくい。
- スレッドはメモリ空間を共有しているため、プログラミングミスで他のスレッドに悪影響を与えやすい。その反面、スレッド間での情報の共有はプロセスよりもやり易い。

JavaEE(サーブレット)アプリケーションは、マルチスレッドで動作するため、プログラミングには十分な注意が必要。

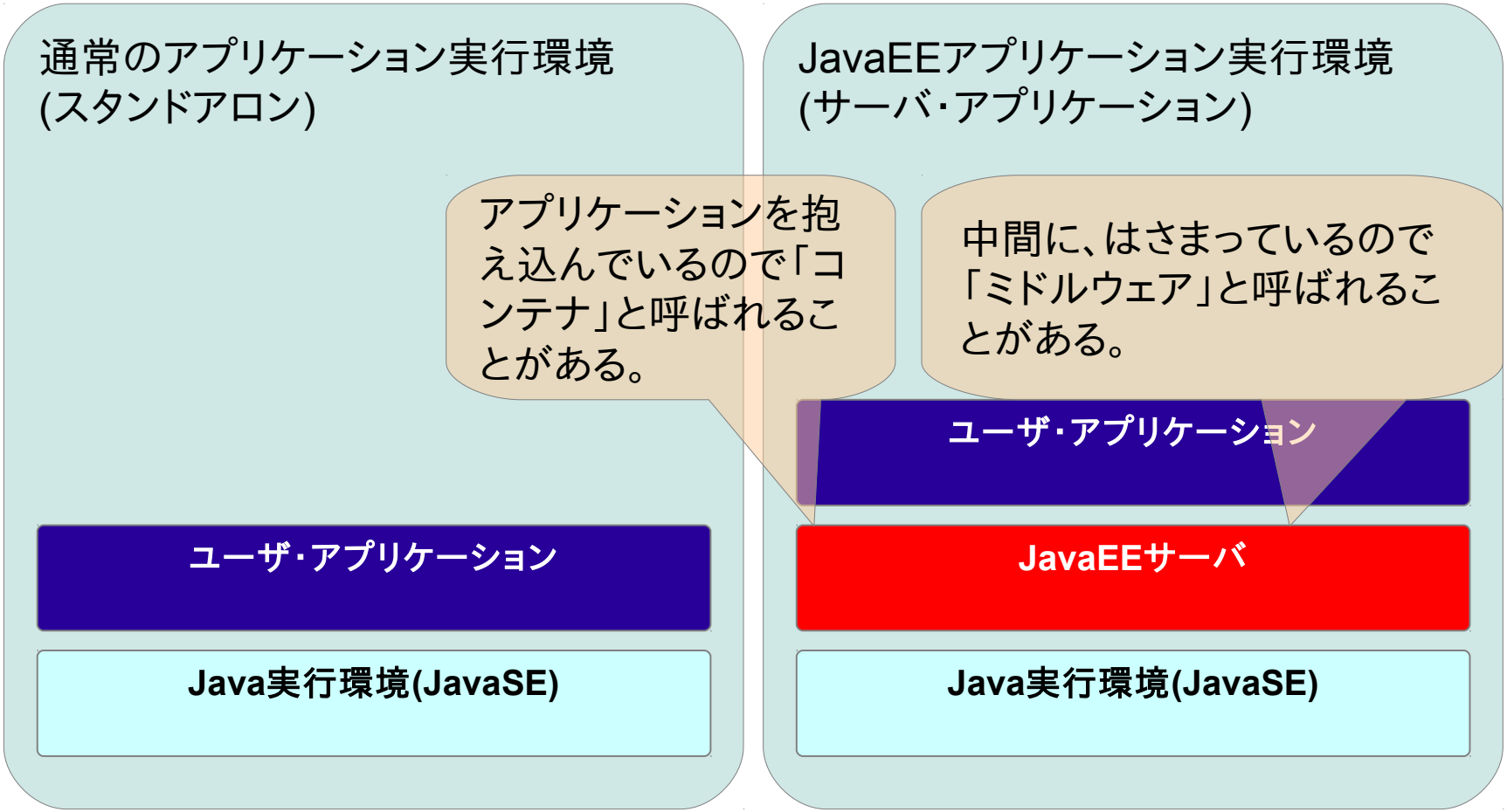
JavaEEとは

- JavaME(Micro Edition)/JavaSE(Standard Edition)/JavaEE(Enterprise Edition)の3つのエディションがある。
- JavaEEは、JavaSEをベースにしてエンタープライズ(主にサーバーサイド)アプリケーションを開発するためのクラスライブラリを提供する。



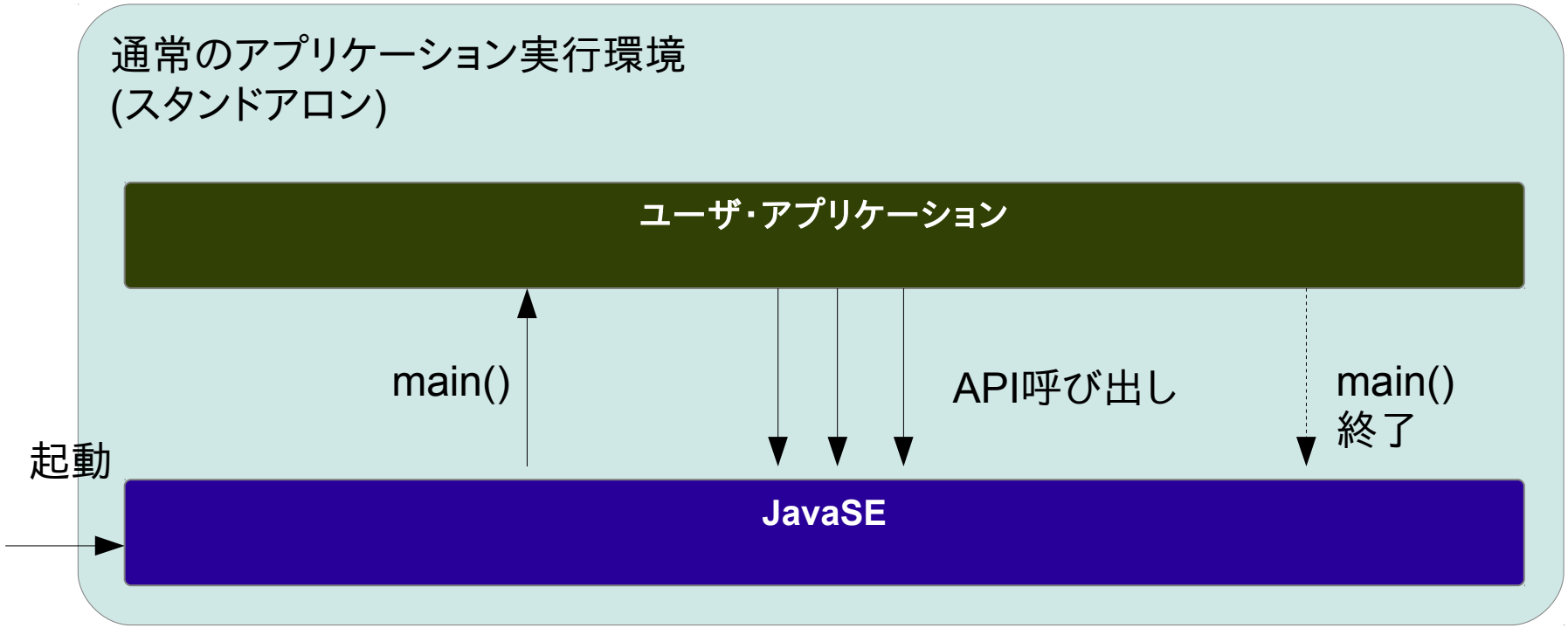
アプリケーション・サーバ

- JavaEEでは、アプリケーション・サーバの上でユーザのアプリケーションを稼動する。このサーバのことを「コンテナ」、「ミドルウェア」と呼ぶ場合がある。



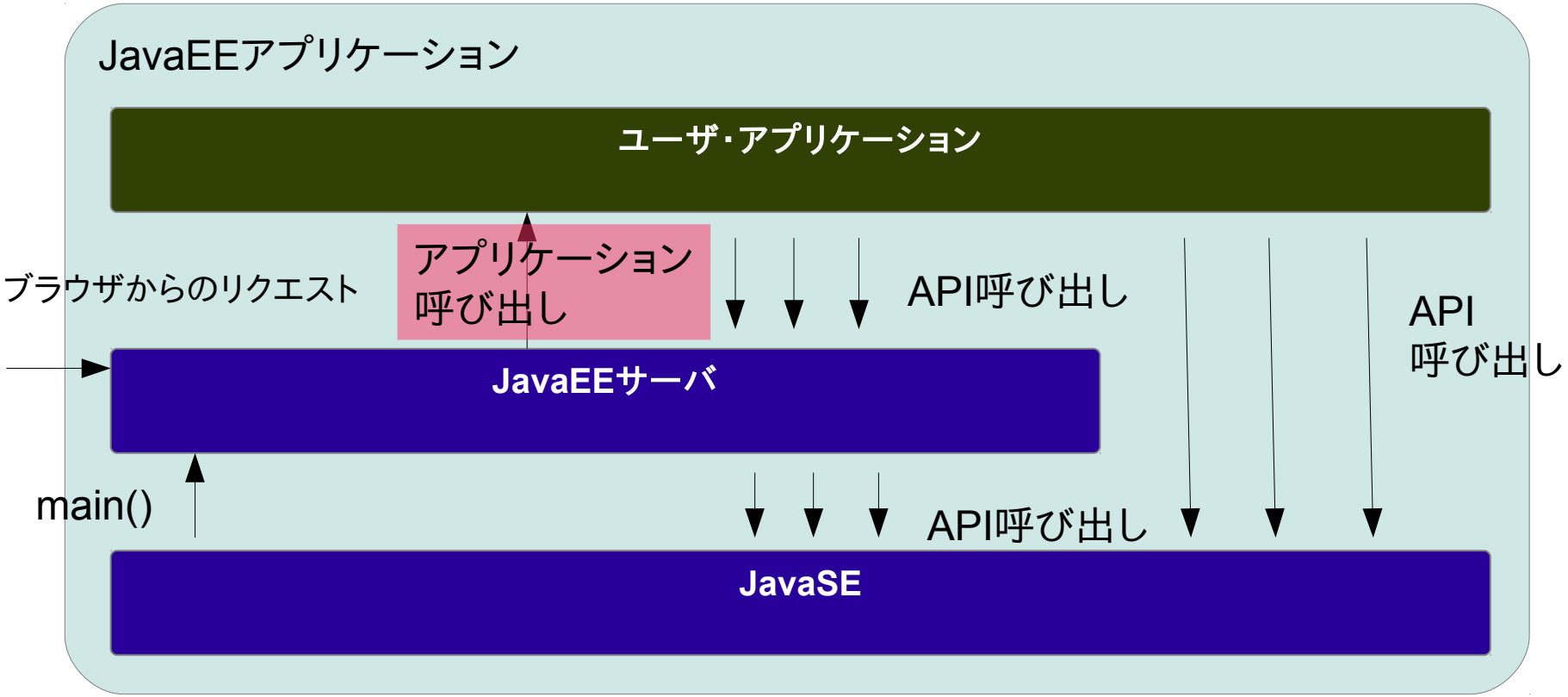
アプリケーション・サーバ

- スタンドアロン・アプリケーションの動作は、以下のようになる。
  - java VM実行開始
  - ユーザアプリケーションのmain()メソッド実行開始
  - ユーザアプリケーションは必要に応じて、Java SEに用意されたAPIを呼び出し。
  - ユーザアプリケーションのmain()メソッド終了
  - java VM実行終了



アプリケーション・サーバ

- これに対し、JavaEEの場合は以下のようになる。
  - アプリケーション・サーバ起動(内部的には、JavaEEサーバ内にあるmainメソッドが呼び出される)
  - 何らかの事象(例:Webブラウザからのリクエスト)発生
  - ユーザアプリケーションの呼び出し



アプリケーション・サーバ

- スタンドアロン・アプリケーション
  - main()の中にアプリケーション開始から終了までの全ての処理を書く。
  - いつ実行を開始して、いつ終了するか(これをライフサイクルと呼ぶ)を自分で管理する。
  - アプリケーションはJavaSEのAPIを利用して必要な機能を実行する(例：画面表示)。
  - 使用するリソース(ファイルやデータベース)などの管理を自分で行う。

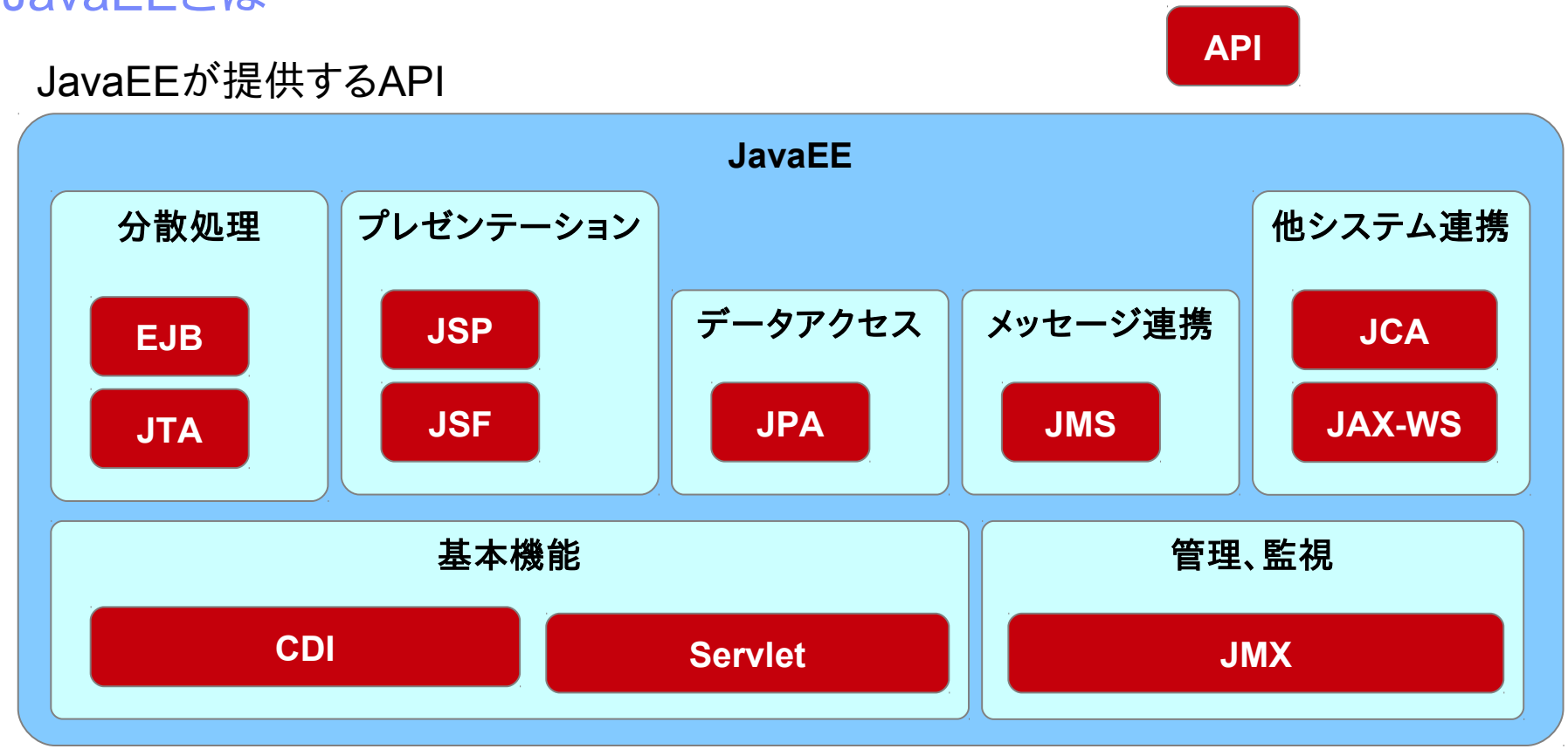
- JavaEEアプリケーション
  - JavaEEサーバから必要な時に呼び出される(いつ呼ばれるかは不明確)。
  - 使用するリソース(データベースなど)の管理の多くが、JavaEEサーバ側で行われ、アプリケーションはそれらを「借りる」イメージ。
  - アプリケーションは必要に応じて、JavaSEのAPI、JavaEEのAPIを利用して必要な機能を実行する。

アプリケーション・サーバとデプロイ

- JavaEEアプリケーションは、JavaEEサーバから呼び出されるので、**予めJavaEEサーバ側に「登録」しておく必要がある**。これを「**デプロイ**」、「配備」と呼ぶ。
- デプロイの際には、アプリケーションのクラスの他に、以下のような情報を指定する。
  - どのような時にアプリケーションを呼び出してもらうか。
  - アプリケーションがどのようなリソースを必要とするか。
  - アプリケーションの初期化の際に渡してもらうパラメータ。
  - …

JavaEEとは

JavaEEが提供するAPI



アプリケーション・サーバは全ての機能を提供しているわけではない。ベンダによって提供する機能が異なる。

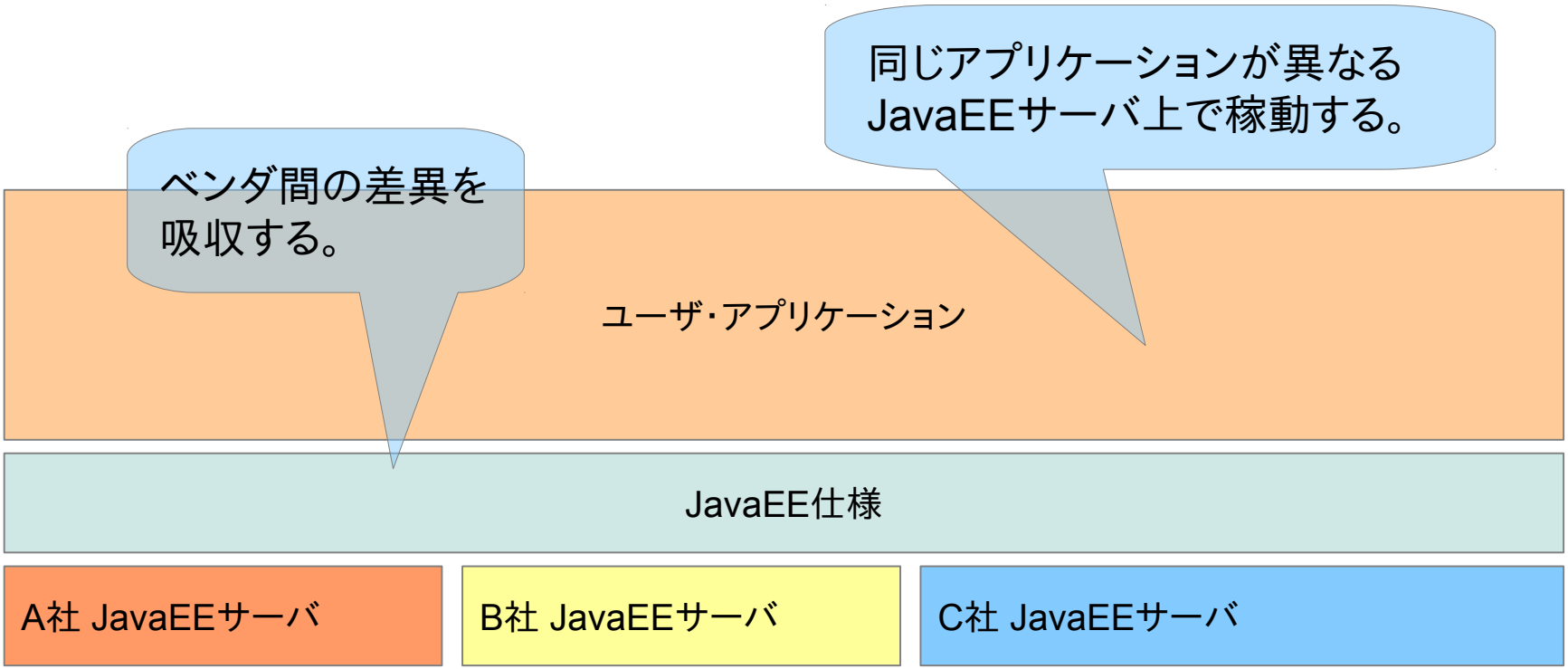
APIとバージョン

- APIにはバージョンがあり、バージョンが上がるごとに機能が増えていく。
- 基本的には、互換性が保たれる(古いAPIで動作するアプリケーションは、新しいAPIでも動作する)が、**非推奨機能が削除されたり、明文化されていなかった動作が変更になったりする場合、アプリケーションに求められる前提事項が増えるケースがある。**
- APIの集合体である、JavaEE自体にもバージョンがあり、JavaEEのバージョンによって、中に含まれるAPIのバージョンが規定される。

参考 JavaEEのバージョンの歴史：
http://en.wikipedia.org/wiki/Java_EE_version_history

JavaEEとは

- JavaEEは、アプリケーションを実行するサーバの仕様も規定する。
- これにより、同じアプリケーションを別のJavaEEサーバ上で動かすこと(サーバの乗り換え)ができる。



まとめ

- JavaEEは、JavaSEをベースにしてエンタープライズ・アプリケーションを構築するための機能を提供する。
- JavaEEは、JavaEE準拠アプリケーションの実行環境を提供するJavaEEサーバが満たすべき仕様を定義する。
- スタンドアロンアプリケーションでは、全ての制御をユーザアプリケーションが自分で行い、リソースの管理も自分で行う。
- JavaEEアプリケーションでは、**必要に応じてサーバからアプリケーションが呼び出される**。リソースの管理も、ほとんどをサーバが行う。
- JavaEEアプリケーションをJavaEEサーバ上で動かすためには、デプロイと呼ばれる登録作業が必要である。

まとめ

- http通信はステートレスである。
- http通信はメソッドによってリクエストの内容を伝え、ステータスコードで処理結果を得る。
- CGIは最も古くからあるWebアプリケーション構築のための仕組みでリクエストをプロセスで処理する。
- サーブレットは、JavaEEの中で最も古くからあWebアプリケーション構築のための仕組みでリクエストをスレッドで処理するので効率が良い。
- サーブレットでは、スレッド間の競合や、リソースの解放忘れなどに気を付けてプログラミングする必要がある。

おまけ(JavaEE、JavaSE、JDK、JREなどの質問と回答集)

- J2EEとJavaEEの違いは何ですか？ J2SEとJavaSEは？

  バージョン1.4まではJ2EE/J2SEと呼ばれていましたが、そもそもJ2の2はバージョン2という意味で意味不明に陥いっていたので、バージョン1.5以降からは、JavaSE/JavaEEと呼ばれるようになりました。

- 現在は、Java7ですが、Java3とかJava4というバージョンはあったのですか？

  Java 1.4までは、0.1ずつバージョンが上がっていましたが、Java5以降は1ずつバージョンが上がるようになりました。このためJava3とかJava4はありません。なおAPI仕様書など一部のドキュメント内では、5ではなく1.5と表記されることがあるので注意してください。

おまけ(JavaEE、JavaSE、JDK、JREなどの質問と回答集)

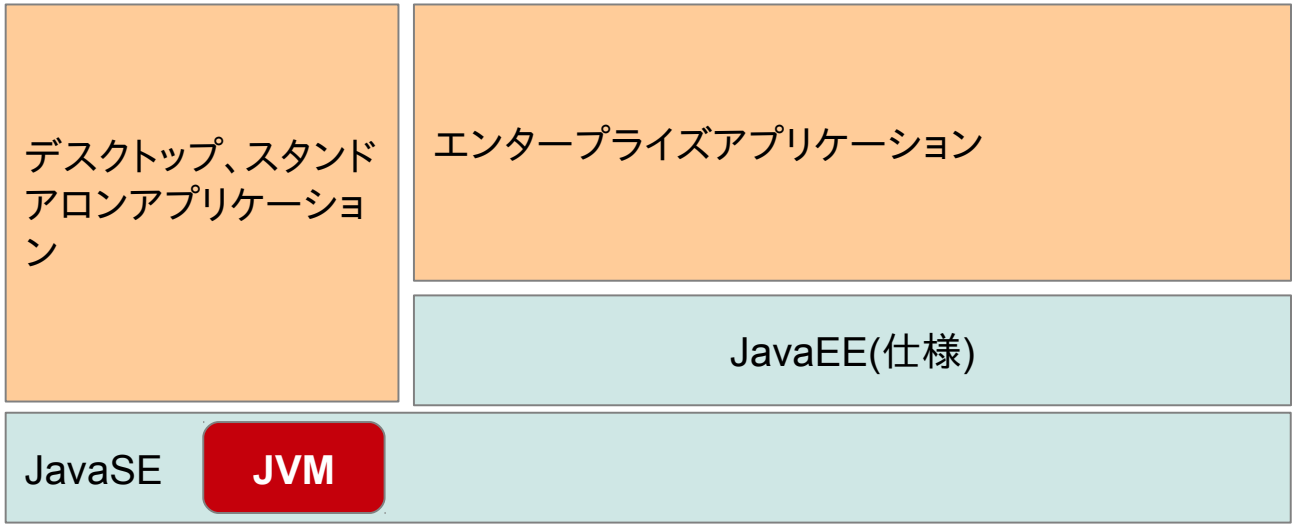
- JDKとJREの違いは何ですか？

JRE(Java Runtime Environment)は、Javaのアプリケーションを動作させるために必要な実行環境を提供します。JDKはこれに加えて、Javaのアプリケーションを開発したり、調査したりするツールが含まれます。例えば、javac(コンパイラ)や、jdb(デバッガ)は、JDKにのみ含まれます。
昔は、JREとJDKとで配布条件が異なりましたが、今は同一です。

おまけ(JavaEE、JavaSE、JDK、JREなどの質問と回答集)

- JavaSEのJDKとJavaEEのJDKは違うものですか？

既に解説した通り、JavaSEとJavaEEはタイプが異なります。JavaSEはJavaのアプリケーションを開発、実行するための環境(具体的にはjava.exeのようなプログラム)を提供します。JavaEEは、**JavaSEの上に**サーバアプリケーションを動作させるためのAPIを提供するものです。つまりJavaEEと言った場合、それは仕様の集合体であって、JDKを含みません。



おまけ(JavaEE、JavaSE、JDK、JREなどの質問と回答集)

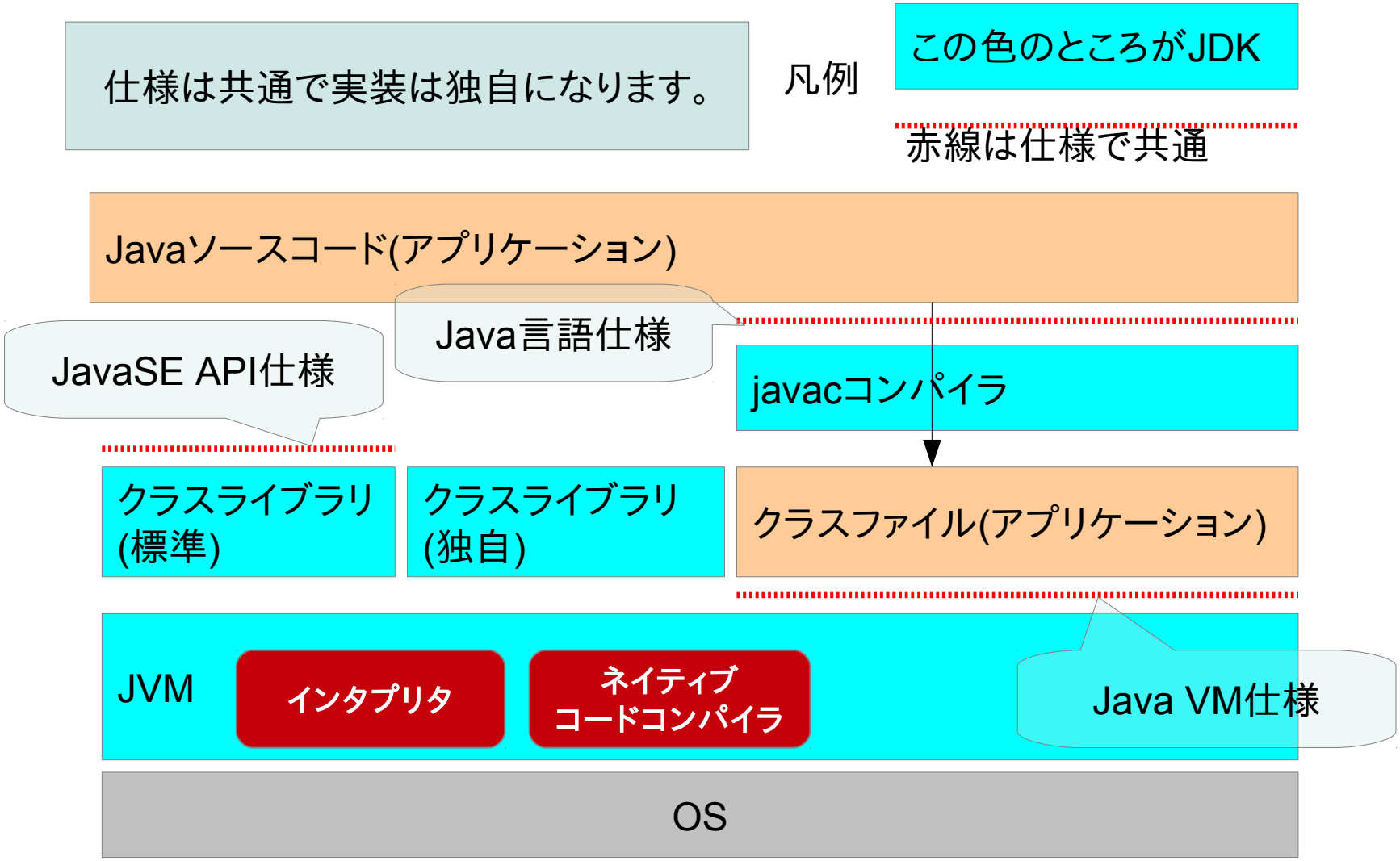
- 良く分からないのですが、JavaSEは無料でダウンロードできて、JavaEEは有料なのですか？

JavaEEは仕様の集合体で、その中に含まれる要素(例えばサーブレット)仕様はJCP(Java Community Process)で策定され、JSR(Java Specification Request)で公開されます。このためJavaEE自体が有料か無料かと言われれば無料です。
しかし、JavaEEは仕様に過ぎないので、それだけでは絵に書いた餅です。このため色々なベンダーがJavaEE仕様に準拠した実装を提供しています。WASは、その1つです。

複数のJavaEE実装があっても、アプリケーションが同じように動くようにするため、OracleはTCK(Technology Compatibility Kit)を提供しています。このTCKは検証プログラムのセットになっていて、これにパスするとJavaEE complientを名乗ることができます。
ですから「JavaEE実装」という意味では、有料のもの(例えばWAS)もあれば、無料のもの(例えばJBoss)もあります。

おまけ(JavaEE、JavaSE、JDK、JREなどの質問と回答集)

- IBMのJDKと、OracleのJDKは何が違うのですか？



おまけ(JavaEE、JavaSE、JDK、JREなどの質問と回答集)

- IBMのJDKは全て独自に実装したものなのですか？

公式にはYesです。

- JDKは無料で使えるのですよね？

JDKによります。OracleのOpenJDKはGPLで配布されており、無料で使えますし、IBMのJDKもWindows用以外は、無料で配布されています。

しかし、例えばOpenJDK 6は、EOL(End of Life)が2012/11で、それ以降はセキュリティバグも含めバグフィックスが提供されなくなります。それで良ければ(つまり頻繁にJDKをアップグレードし続けられる)なら良いですが、そうでなければJava SE for Businessのようなサービス契約が必要になります。これに契約すればEOLになった後までサポートしてもらえます。

IBM版の場合は通常製品に同梱されたものを使用するので、その製品のライフサイクル次第です。

最新情報を常にチェックしてください。

- ここに書いた内容は私個人が調べたものですので、きっと間違いもあれば古くなっているものもあります。
- 必要な場合は常に最新情報をチェックしてください。特にライセンス条件はJavaの歴史の中でも何度も変更を受けています。