HeartCore
チューニング設定マニュアル（JSP 版）

September 2017　Ver1.3

改訂履歴

	改訂日	改訂内容
1.0	2011 年 1 月	新規作成
1.2	2013 年 10 月	フォーマット改訂
1.3	2017 年 9 月	4.(5)コネクションプールのオプションを tomcat8 に準拠

目次

1. 本文書の目的と対象ライセンス

1.1. 目的

本書では、HeartCore をより快適にご利用いただくための、パフォーマンス向上を目的としたチューニングポイントを説明いたします。これらのチューニングポイントは、パフォーマンス向上を目的とした内容となりますが、動作環境によっては効果が表れない場合もございますのでご了承ください。

2. 概要

2.1. チューニングポイント概要

本書のチューニング設定内容は、大きく分けて下記の２つに分類されます。

【HeartCore におけるチューニング】

HeartCore の提供機能に関するチューニングポイントを説明いたします。

（１）アクセス統計値の分離
（２）スケジューラー機能の停止
（３）コンテンツエディタの速度向上

【システム環境におけるチューニング】

オペレーティング・システム、および、アプリケーションサーバに対するチューニングポイントを説明いたします。HeartCore は様々なプラットフォームでの動作をサポートしておりますが、本書では利用実績の多いプラットフォームを中心に説明しております。その他のプラットフォームに関するチューニング内容につきましては、各環境に応じたシステムチューニングを実施してください。

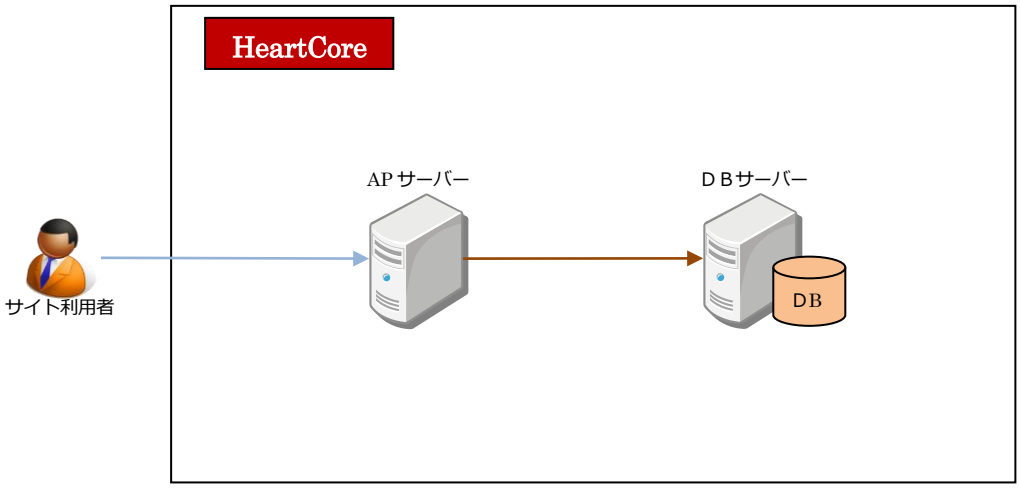
（４）ファイルオープン数の設定（Linux OS）
（５）コネクションプールの設定（Apache Tomcat）
（６）Apache、Tomcat の AJP 連携部のチューニング設定 （Apache Tomcat）

2.2. 各チューニングポイント別概要

（１） アクセス統計値の分離

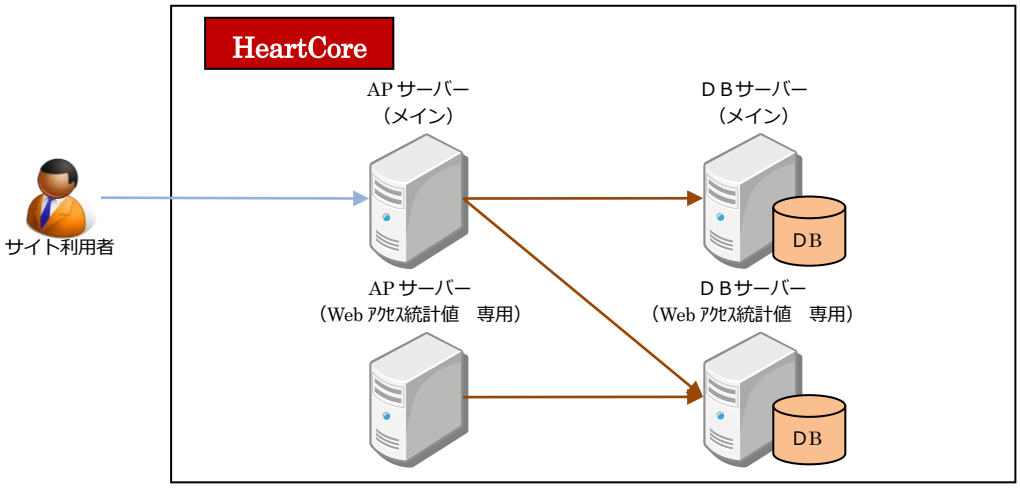
アクセス統計値の分離のチューニング概要のイメージを下図に示します。

【通常構成】（基本的な HeartCore 構成例）



通常の構成です。全てのデータを同一データベースで管理しています。

【Web アクセス統計値を分離した構成】



Web アクセス統計値を分離した構成です。Web アクセス統計値データだけは別のデータベースサーバで管理しています。また Web アクセス統計値を参照する際の専用のアプリケーションサーバを用意することで、統計値出力処理時のメイン HeartCore へのパフォーマンス劣化の影響を防ぐことができます。

（２） スケジューラー機能の停止

HeartCore ではタイマーカウント機能を保持していません。そのために、ページがアクセスされる度に、現在の時間を DB に書き込み、該当時間が来ればコンテンツを公開処理する方式をとっています。

そのため、ページが閲覧される度に DB に書き込みを行います。これによってスケジューラーを動作させていますが、この書き込みを止めることでパフォーマンスを上げることができます。

ページの閲覧時の DB 書き込みを停止させる代わりに、Cron などのスケジュール機能を利用することで「開始時間」「終了時間」の指定を利用することができます。

（３） コンテンツエディタの速度向上

HeartCore のコンテンツエディタは、キーボードの操作、マウスのクリック等履歴を全て取得しております。そのため「Ctrl-Z」で取り消しを行ったり、「リドゥ・アンドゥ」機能で、元に戻したり再度処理をしたりすることが可能です。オフィス製品感覚での作業ができますが、JavaScript で制御をしているため、非力なマシンでやると時間がかかります。

本設定により、エディタの体感速度を大きく向上させる事が設定で可能です。

（４） ファイルオープン数の設定（Linux OS）

HeartCore ではイメージやコンテンツを出力する際にファイルを開いてハンドリングをします。オペレーティング・システムでファイル記述子が不足している場合、ファイルが開けなくなり、HeartCore の出力ができなくなります。ファイル記述子は、様々な種類の開いているファイル（イメージやコンテンツなど）を識別するために、プロセスによって使用されます。

ファイル記述子の上限を上げて設定する事で表示のエラー現象が回避されるので、/etc/security/limits.conf を変更する事で対応が可能になります。

（５） コネクションプールの設定（Apache Tomcat）

コネクションプールを利用したデータベースへの接続方法について説明いたします。
コネクションプールを利用することで、データベースへのアクセス時、アクセスのたびに接続（コネクション）を確立するのではなく、あらかじめ一定数のコネクション確立しておき、それを利用することでデータベースアクセスの負荷を軽減させることができます。

（６） Apache-Tomcat の AJP 連携部のチューニング設定 （Apache Tomcat）

Apache-Tomcat での AJP 連携部のチューニングポイントについて説明いたします。AJP 連携部のチューニング設定を実施することで、デフォルトの状態より、「Apache」－「Tomcat」間の AJP 連携に関するパフォーマンス向上が期待できます。

3. 構築フロー

以下に機能を構築するフローを記載します。尚、当該機能につきましては「インストール手順」は必要ありませんので、機能設定手順のみ実施する事となります。

機能設定手順

「機能設定手順」

（1）～（6）の各チューニングポイントにおける機能設定手順を記載。

4. 機能設定方法

（1） アクセス統計値の分離

詳細手順につきましては、下記のとおりとなります。

なお、本手順では、下記構成を例として説明しており、データベースは「MySQL」を使用しているものとします。

【構成例】

サーバー役割	設定名	設定値
AP サーバー（メイン）	IP アドレス	192.168.1.10
AP サーバー （Web アクセス統計値 専用）	IP アドレス	192.168.1.20
DB サーバー（メイン）	IP アドレス	192.168.1.15
	データベース名	heartcore
	DB ユーザ名	hcuser
	DB パスワード	hcpass
DB サーバー （Web アクセス統計値 専用）	IP アドレス	192.168.1.25
	データベース名	heartcore2
	DB ユーザ名	hcuser2
	DB パスワード	hcpass2

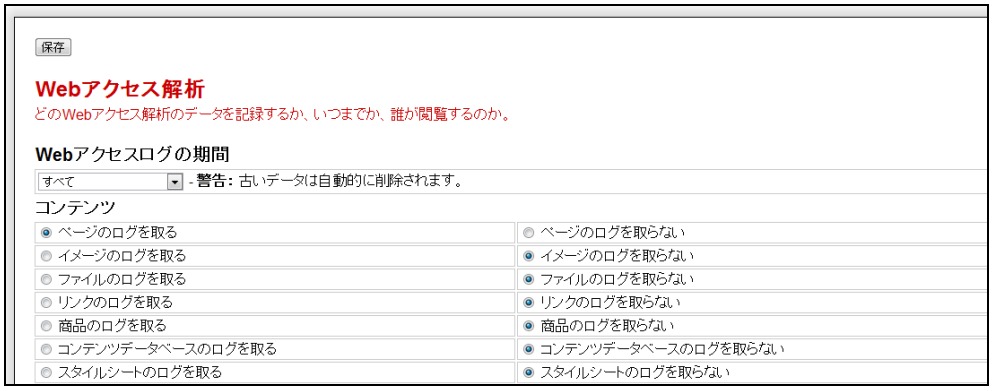
※既にメインの AP サーバー、および DB サーバーは稼働しているものとします。

1．Web アクセス統計値専用の「AP サーバー」および「DB サーバー」を用意します。

2．Web アクセス統計値専用の環境に HeartCore を構築します。メイン側と同じミドルウェアを使用することをお奨め致します。また、Web アクセス統計値専用のデータベースユーザは「DB サーバー（メイン）」からアクセスできるように設定してください。

3．環境の準備ができたら、HeartCore 管理画面にアクセスし、「設定クイックスタート」を実施してください。「ステップ 1」から「ステップ 3」までは、メインで使用している HeartCore と同様の設定を実施します。「ステップ 4：ウェブサイトコンテンツ」では、アクセス統計値専用のデータベースのためコンテンツなどは必要御座いませんので、「empty」をインポートしてください。

４．「設定クイックスタート」完了後、Web アクセス統計値 専用側の HeartCore にログインし、設定 ＞ システム ＞ Web アクセス解析 において、ログの取得期間、および取得するログの種類を設定します。



５．メイン側 HeartCore を停止し、「config.static.jsp」を編集します。（※注）
先ず、「myconfig.setTemp("logdatabase", "");」の行をコメント化（先頭に // を追記）します。
次に、直後にある「//myconfig.setTemp(”logdatabase”, …」のコメントを解除（先頭にある // を削除）し、Web アクセス統計値を管理する対象のデータベース情報を指定し、保存してください。

（編集前）※　実際は２行で記載されております。

```
myconfig.setTemp(″logdatabase″, ″″);
//myconfig.setTemp(″logdatabase″,
″mysql:com.mysql.jdbc.Driver:username:password@jdbc:mysql://localhost/database″);
```

（編集後）※　実際は２行で記載されております。また、赤字部分は環境に応じて適宜置き換えてください。

```
//myconfig.setTemp(″logdatabase″, ″″);
myconfig.setTemp(″logdatabase″,
″mysql:com.mysql.jdbc.Driver:hcuser2:hcpass2@jdbc:mysql://192.168.1.25/heartcore2″);
```

６．メイン側の HeartCore を起動します。

７．メイン側のコンテンツページにアクセスした際に、Web アクセス統計値専用で使用しているデータベースに Web アクセス統計値データが保存されることを確認してください。

※制限事項 1

（２） スケジューラー機能の停止

①スケジューラー機能の停止手順

１． 稼働中のアプリケーションサーバ(Tomcat)を停止します。

２．「config.static.jsp」を編集します。（※注）

「//myconfig.setTemp("scheduled_next", "");」のコメントを解除（先頭にある // を削除）し、保存してください。

（編集前）※ 実際は１行で記載されております。

```
//myconfig.setTemp(″scheduled_next″, ″″); // !!!!! disables scheduled publishing/expiration functionality unless ″/webadmin/publishscheduled.jsp″ is called periodically
```

（編集後）※ 実際１行で記載されております。

```
myconfig.setTemp(″scheduled_next″, ″″); // !!!!! disables scheduled publishing/expiration functionality unless ″/webadmin/publishscheduled.jsp″ is called periodically
```

３．Tomcat を起動します。

②スケジューラー機能の cron 実行手順

上記を実施することで、ページへアクセスする際の DB への書き込みが実施されなくなり、パフォーマンスを上げることができます。

しかし、このままでは「開始時間」「終了時間」の指定が機能しなくなりますので、代わりに、OS のスケジュール機能(Cron など)を利用して「/webadmin/publishscheduled.jsp」を実行させる必要があります。Linux OS の Cron の場合、下記のように数分毎に「/webadmin/publishscheduled.jsp」へアクセスさせることで、スケジューラー機能を利用することができます。

例）cron で 5 分毎にスケジューラー機能を実行させる場合

（赤字部分は HeartCore の URL に置き換えてください）

```
*/5 * * * * wget -O publishscheduled.jsp -P /tmp wget http://xxx.xxx.xxx.xxx/webadmin/publishscheduled.jsp
```

※制限事項２

（3）コンテンツエディタの速度向上

1． 設定 ＞ 機能設定 ＞コンテンツエディタ画面を開きます。

2．「HeartCore Web エディター」タブを開きます。

3．「カスタマイズした Java スクリプト機能」欄に下記内容を追記します。

```
webeditor.undo = false;
```

4．「保存」ボタンをクリックし、設定内容を保存してください。

これで Redo/Undo の機能が無効になり、速度が大きく向上します。

（４）ファイルオープン数の設定（Linux OS）

１． vi エディタなどで「/etc/security/limits.conf」を開きます。

２． （最終行に）下記内容を追記します。
例）「root」で実行されるプロセス単位の最大ファイルオープン数を 10000 に設定。

```
root soft nofile 10000
root hard nofile 10000
```

（変更前） ※ ulimit -a で調査

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
pending signals         (-i) 1024
max locked memory       (kbytes, -l) 32
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024  ←ファイル記述子の設定がデフォルトの 1024 です。
pipe size               (512 bytes, -p) 8
POSIX message queues    (bytes, -q) 819200
stack size              (kbytes, -s) 10240
cpu time                (seconds, -t) unlimited
max user processes      (-u) 105471
virtual memory          (kbytes, -v) unlimited
file locks              (-x) unlimited
```

（変更後）※ ulimit -a で調査

core file size	(blocks, -c) 0
data seg size	(kbytes, -d) unlimited
file size	(blocks, -f) unlimited
pending signals	(-i) 1024
max locked memory	(kbytes, -l) 32
max memory size	(kbytes, -m) unlimited
open files	(-n) 10000 ←上限を 1024 から 10000 に変更
pipe size	(512 bytes, -p) 8
POSIX message queues	(bytes, -q) 819200
stack size	(kbytes, -s) 10240
cpu time	(seconds, -t) unlimited
max user processes	(-u) 105471
virtual memory	(kbytes, -v) unlimited
file locks	(-x) unlimited

【機能の確認方法】

「ps -ef | grep java」 コマンドで Java のプロセス番号を調べます。

例えば、ID が 12345 だとします。

「lsof -p12345 | wc -l　(12345 はＪＶＭの pid)　)で確認をします。

HeartCore のページにアクセスをすると、数値が増えていきます。ただし、上限値があり無限に増えると言うことではありません。サーバーの環境によりますが 1000 から 3000 までの数値で推移します。

デフォルトの 1024 個では足りなくなります。

（５） コネクションプールの設定（Apache Tomcat）

１．Tomcat を停止します。

２．Tomcat インストールディレクトリ/conf 内にある「server.xml」を開きます。

３．<Host> ～ </Host> 内の最終行に下記内容を追記します。
※ 下記内容は MySQL 設定例となります。赤字部分につきましては、環境に応じて設定してください。

```
<Context
  path=""  ← ①  オプションパラメータ
  docBase="ROOT" ← ②  オプションパラメータ
  debug="0" ← ③  オプションパラメータ
  reloadable="false" ← ④  オプションパラメータ
  crossContext="true">← ⑤  オプションパラメータ
  <Resource
        allowLinking="true"      ← ⑥  オプションパラメータ
        name="jdbc/heartcore"  ← ⑦  オプションパラメータ
        auth="Container"  ← ⑧  オプションパラメータ
        type="javax.sql.DataSource"  ← ⑨  オプションパラメータ
        maxTotal="512"  ← ⑩  オプションパラメータ
        maxIdle="512"  ← ⑪  オプションパラメータ
        maxWaitMillis="10000" ← ⑫  オプションパラメータ
        username="hcuser"  ← ⑬  オプションパラメータ
        password="hcpass"  ← ⑭  オプションパラメータ
        driverClassName="com.mysql.jdbc.Driver"  ← ⑮  オプションパラメータ      ↓ ⑯  オプションパラメータ
        url="jdbc:mysql://localhost:3306/heartcore?useUnicode=true&amp;characterEncoding=UTF-8"
        removeAbandonedOnMaintenance="true"  ← ⑰  オプションパラメータ
        removeAbandonedOnBorrow="true"  ← ⑱  オプションパラメータ
        removeAbandonedTimeout="300"  ← ⑲  オプションパラメータ
        logAbandoned="true"      ← ⑳  オプションパラメータ
        validationQuery="select 1"㉑  オプションパラメータ
        testOnBorrow="true"  ← ㉒  オプションパラメータ
        testOnReturn="true"← ㉓  オプションパラメータ
        testWhileIdle="true"  ← ㉔  オプションパラメータ
        minEvictableIdleTimeMillis="60000"  ← ㉕  オプションパラメータ
        />
</Context>
```

以下、上記フォーム中で使用されているパラメータの説明を記載します。

① Web アプリケーションコンテキストパスを指定。”” だと全てが対象。
② コンテキストのルートディレクトリを指定します。HeartCore ファイルを「ROOT」として配置している場合は、「ROOT」のままで問題ありません。
③ デバッグ出力情報レベル。
④ 「true」の場合、プログラムに変更が発生した際に自動的に再読み込みを行いますが、「false」に比べて動作が遅くなります。「false」に設定した場合は、プログラムに変更があった際は、Tomcat の再起動が必要です。
⑤ 「true」にすると他のコンテキストにアクセスできる。
⑥ 「true」の場合、シンボリックリンクを利用できる。
⑦ Java データソース名。任意ですが、ここで指定したデータソース名を HeartCore の「ini.jsp」ファイルに定義する必要があります。
⑧ リソースマネージャに接続する主体を指定
⑨ JNDI リソースのクラス型、インターフェース型を指定
⑩ データベース最大接続数を指定。0 の場合は無制限。
⑪ アイドル状態の最大接続数を指定。0 の場合は無制限。
⑫ 接続しているコネクションの再度利用されるまでの待ち時間を指定。指定した時間を過ぎると例外が発生する。-1 を指定すると無制限。
⑬ データベースへの接続ユーザー名。
⑭ データベースへの接続パスワード。
⑮ JDBC ドライバの Java クラスを指定。
⑯ データベース接続 URL
⑰ 有効にすると Evictor スレッドによってクローズ漏れとなったコネクションを回収する。
⑱ 有効にするとコネクションを取得する際にクローズ漏れとなったコネクションを回収する。
⑲ 利用不可能とみなすまでの時間を秒単位で指定する。
⑳ 接続の close に失敗した場所をログに出力する。
㉑ データベースによっては、接続を長時間維持していると、自動的に接続を切断する場合がある。
コネクションプーリングでは、このようにして無効となった接続をアプリケーションに引き渡すのを防ぐために、あらかじめ検証クエリで接続が維持されているかを確認する。
※DBMS によってクエリ文が異なりますのでご注意ください。
㉒ プールから取得する前に、接続の有効性を確認する。
㉓ 使用済接続をプールに返却する前に、接続の有効性を確認する。
有効性が確認できない場合接続はプールに保管されず破棄される。
㉔ 監視処理時、アイドル接続の有効性を確認する。
有効性が確認できない場合、その接続はプールから削除される。
㉕ 監視処理時、アイドル接続の生存期間をチェックする。
アイドル接続が一定の期間使われなかった場合その接続をプールから削除する

４．HeartCore のデータベース接続定義ファイルを下記のように編集します。

【ROOT/ini.jsp】
　例）ＤＢ接続ユーザ名「hcuser」、ＤＢパスワード「hcpass」、データソース名「jdbc/heartcore」の場合

```
<%
ini_cache_database.put("default","mysql:hcuser:hcpass@jdbc/heartcore");
%>
```

【ROOT/ini.webadmin.jsp】（HeartCore V7.2 以前の場合は「ROOT/webadmin/ini.jsp」）
　例）ＤＢ接続ユーザ名「hcuser」、ＤＢパスワード「hcpass」、データソース名「jdbc/heartcore」の場合

```
<%
[default]database=mysql:hcuser:hcpass@jdbc/heartcore
%>
```

５．Tomcat を起動し、HeartCore へアクセスしてデータベースへ接続できていることを確認してください。

（６） コネクションプールの設定（Apache Tomcat）

１．Tomcat を停止します。

２．Tomcat インストールディレクトリ/conf 内にある「server.xml」を開きます。

３．AJP プロトコルに関する定義部分に下記項目を設定します。

※ 下記内容は設定例となります。赤字部分につきましては、環境に応じて設定してください。

```
<Connector port="8009"
           maxHttpHeaderSize="8192" ← ① オプションパラメータ
           maxThreads="300" ← ② オプションパラメータ
           minSpareThreads="25" ← ③ オプションパラメータ
           maxSpareThreads="100" ← ④ オプションパラメータ
           enableLookups="false" ← ⑤ オプションパラメータ
           redirectPort="8443"
           acceptCount="300" ← ⑥ オプションパラメータ
           connectionTimeout="20000" ← ⑦ オプションパラメータ
           disableUploadTimeout="true" ← ⑧ オプションパラメータ
protocol="AJP/1.3" />
```

以下、上記フォーム中で使用されているパラメータの説明を記載します。

① リクエストとレスポンスの HTTP ヘッダの最大サイズ（バイト）。
② リクエスト処理スレッド最大数。
③ 最小プールサイズ。
④ 最大プールサイズ。
⑤ 「true」をセットすると DNS lookup が実行され、リモート・クライアントの実際のホスト名が返される。
「false」をセットすると、DNS lookup が省略され、IP アドレスが返される（よって性能が向上する）
⑥ すべての可能なリクエスト処理スレッドが使用中のとき、到来する接続要求に対するキューの最大長。
⑦ 接続を accept した後、リクエストの URI が現れるのを待つ（ミリ数秒）。デフォルト値は「60000」。
⑧ 「true」をセットするとコネクターの接続をクローズさせずにサーブレットを続行する。
「false」をセットすると別オプション connectionUploadTimeout の値を使用する。

４．Tomcat を起動すると、設定内容が反映されます。

※制限事項 3

5. 制限事項

(1) HeartCore のライセンス体型としては、CPU 単位となりますので、DB サーバ（本番環境の AP サーバと別筐体）に HeartCore をインストールした場合は、追加ライセンスが必要になります。
アクセス統計を確認するのみでご利用の場合は、開発ライセンスが必要となる為、価格等に関しましては弊社営業までご連絡をお願い致します。

(2) 設定例は取得ファイルの保存先ディレクトリを『/tmp』で指定した場合の例となります。設定時には、既存の『/webadmin/publishscheduled.jsp』ファイルへ上書きしないようご注意下さい。

(3) Linux OS の「vi エディタ」でファイルを編集すると、保存した際にファイルの最後尾に空行が追加されます。「config.static.jsp」を「vi エディタ」で編集すると、この空行の影響で『ウェブサイトコンテンツ』のコンテンツ一覧などで「リスト表示」で一覧を表示しようとしても「Waiting for data...」⇒「Request for data timed out!」となり、コンテンツリストが表示できなくなる現象が発生いたします。
「config.static.jsp」を編集する際は、Windows OS のテキストエディタなどを使用して編集していただくことを推奨としております。編集後、同環境に配置し、変更内容を反映させてください。