



## タイガースアプライアンスを つくる Open Qube Appliance カスタマイズ入門

2003年9月11日

Masahiro Watanabe

## おさらい | Sausalitoとはなにか

Sausalitoとは、Cobalt Serverのソフトウェアに一貫した操作性を与えるために開発された、アーキテクチャである。

	従来のCobalt	Sausalito
管理画面の構成	画面ごとにPerlのCGIで作成	XMLを使って自動生成されたメニューと、PHPのUIFCライブラリを利用して作成した画面から構成
管理画面の認証方法	HTTPのBASIC認証を利用	Sausalito内の独自認証を利用(ユーザー権限ごとにメニュー構成を変更することが可能)
国際版対応	各国版毎にPerlのソースを書き換えてリリース	表示されるメッセージをStringファイルとして独立させ、Stringファイルを言語毎に用意することにより対応(ソースは各国語版同一)
システムリソースへのアクセス	Perlにてアクセス	サーバ管理用デーモン(CCE)経由でアクセス

## Sausalitoの特長

- 柔軟な機能拡張が可能
  - XMLを用いたメニュー自動生成処理 (Navigation Manager)により、メニューツリーへのメニュー追加が他の既存のメニューに影響を与えずに可能
- 各国語版パッチリリースに要する期間の短縮および信頼性の向上
  - 各国語版間でソースが共通であるため、シンプルなバージョン管理が可能
- ユーザインターフェースのLook & Feelの統一性の保持が可能 => **今回のテーマ UIカスタマイズ**
  - ユーザインターフェース用API (UIFC) が公開されているため、サードパーティのアプリケーションにおいてもCobaltオリジナル機能と同一のLook & Feelを実現可能

## Sausalitoの特長 (つづき)

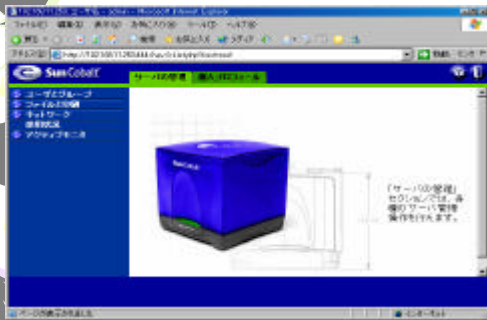
- サードパーティのアプリケーションにおいてもSausalitoのユーザ認証の利用が可能
  - サードパーティのアプリケーションにおいてもNavigation ManagerおよびUIFCを用いることにより、Sausalitoのユーザ認証を利用可能 (独自の認証機構を準備する必要がない)
- Sausalito対応機種間でのアプリケーションの移植が容易
  - ハードウェアやシステム情報へのアクセス用のAPI (CCE) が提供されていることにより、機種間のアプリケーションの移植が従来に比べ容易であると思われる。

## Sausalitoの画面構成 (ログイン画面)



この画面はQube3(OSS1.4)のものである。

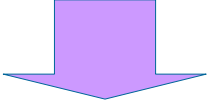
## Sausalitoの画面構成 (ログイン後)



この画面はQube3(OSS1.4)のものである。

### SausalitoがOpen化！

- 2003年7月、Cobaltのサーバ管理環境である、SausalitoがOpen化された。



- せっかくOpen化されたのだから、Sausalitoをつかって自分だけのオリジナルのQubeを作ってみよう！

### Sausalito対応アプリケーション作成の準備

用意するもの

- Sausalitoアーキテクチャのサーバ管理者権限が必要
- IE5以上が動作するPC
- Telnetクライアントソフト
  - TeraTerm(フリーウェア)がおすすめ
- FTPクライアントソフト
  - FFFTP(フリーウェア)がおすすめ
- tar, gzip, rpmに対応した圧縮・解凍ソフト
  - Exptzh(シェアウェア)+tar32.dllがおすすめ
- テキストエディタ
  - 秀丸(シェアウェア)がおすすめ

### Sausalito対応アプリケーション作成の準備

- あるとうれしい予備知識 (なくても可?)
  - TelnetおよびShellを使ったことがある
  - htmlのソースを見たり書いたことがある
  - xmlについて知っている
  - rpmのSPECファイルを書いたことがある
  - Perl, PHP, Awkのソースを見たことがある
  - EUC, SJISコードとは何のことか知っている
  - vi, moreを使ったことがある
  - ‘/’と書いてrootと読める

### Sausalitoのディレクトリ構成

```

/usr/
├── share/locale/
│   ├── ja/LC_MESSAGES/ --- *.mo
│   ├── ...
│   └── en/LC_MESSAGES/ --- *.mo
├── sausalito/
│   ├── constructor/ --- *.pl
│   ├── destructor/ --- *.pl
│   ├── handler/ --- *.pl
│   └── schemas/ --- *.schema
├── ui/
│   ├── menu/ --- *.xml
│   ├── web/ --- *.phphtml.pl
│   ├── style/ --- *.xml
│   └── conf/
│       └── libPpp/
├── bin/
├── ...
└── lib/
  
```

Stringファイル (ja/LC\_MESSAGES, en/LC\_MESSAGES)

CCE関連ファイル (constructor, destructor, handler, schemas)

menu定義ファイル (menu)

CGIファイル (web)

Styleファイル (style)

Sausalitoシステムディレクトリ (ユーザは直接使用しない)

### 本日のお題

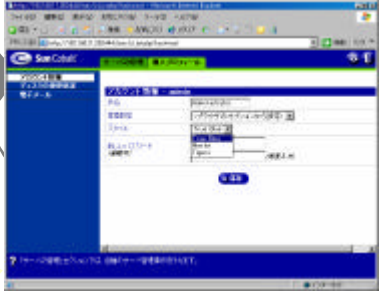
課題1) Sausalitoの画面を、タイガースバージョンに変えてみる

課題2) Sausalitoを使って、タイガース選手データ管理ツールを作ってみる。(画面イメージまで)

おまけ) タイガースアプライアンス専用筐体について

### 課題1 画面スタイルの変更

Sausalitoのメニュー画面をタイガースバージョンにしてみよう



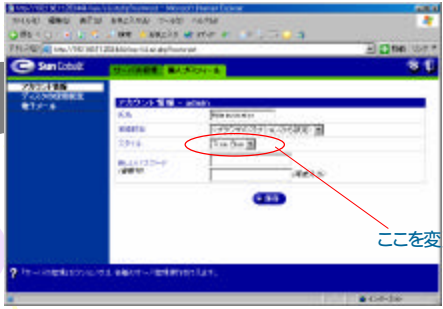
### 課題 1 画面スタイルの変更

Sausalitoのメニュー画面をタイガースバージョンにしてみよう



### STEP 1 :スタイル変更機能を使ってみる

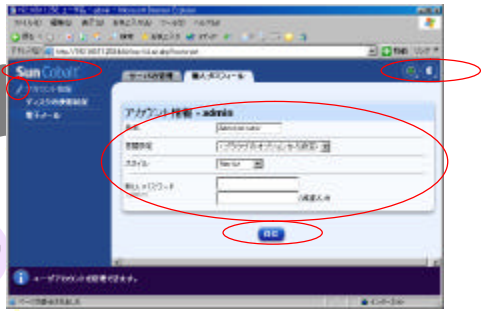
個人プロフィールのアカウント情報のなかの「スタイル」を「Merlot」に変更してみる。



ここを変更

### STEP 1 続き

ロゴマークや色調、ボタンの形などが変化した。



### STEP2 :スタイルファイルの構造を見てみる

/usr/sausalito/ui/styleの下を覗く

```
[root@localhost root]# cd /usr/sausalito/ui/style/
[root@localhost style]# ls
Merlot.xml
Merlot.xml.compiled.96c9bad7913ffe46497ce19cceb5a19
trueBlue.xml
trueBlue.xml.compiled.96c9bad7913ffe46497ce19cceb5a19
trueBlue.xml.ja
trueBlue.xml.ja.compiled.d40a99326f8ba4faadd9e85dc16bfa48
[root@localhost style]#
```

xmlファイルはスタイルを定義するファイルであり、xml.compiled.\*\*\* というファイルは、xmlファイルから自動生成されたファイルである。

### STEP 2 続き

trueBlue.xml ファイルを覗いてみる。

```
<styleResource name="[[trueBlue.trueBlue]]">
<style id="Login">
<property name="fontFamily" target="medium" value="sans-serif" />
</style>
<property name="backgroundColor" target="button" value="#990000" />
</style>
<style id="Page">
<property name="aLinkColor" value="#0033CC"/>
</style>
<style id="SimpleBlock">
</style>
</styleResource>
```

### STEP 2 続き

xmlファイルの構造を推測する。

エレメント	属性	意味
styleResource	name	スタイル名を定義する。[[ ]]でくくられている文字列は、ストリングファイル内の文字で置換される。
style	id	スタイルを適用するパーツを指定する。(Login, Tab, Page など)
property	name	スタイルのプロパティ名 (fontFamily, color, fontSize, imageなど)
property	target	(optional) パーツ内のスタイルの適用箇所を指定する。
property	Value	プロパティに対する値

```
<styleResource name="[[trueBlue.trueBlue]]">
<style id="Login">
<property name="fontFamily" target="medium" value="sans-serif" />
</style>
</styleResource>
```

### STEP 3 :タイガーススタイルを作る

Merlot.xmlをコピーして、Tigers.xmlを作成する

```
[root@localhost style]# cp Merlot.xml Tigers.xml
[root@localhost style]#
root@localhost style]# ls
Merlot.xml
Merlot.xml.compiled.96cdbad7913ffe46497ce19ccebf5a19
Tigers.xml
trueBlue.xml
trueBlue.xml.compiled.96cdbad7913ffe46497ce19ccebf5a19
trueBlue.xml.ja
trueBlue.xml.ja.compiled.d40a99326f8ba4faadd9e85dc16bf48
[root@localhost style]#
```

### STEP 3 続き

Tigers.xmlのstyleResource名を書き換える

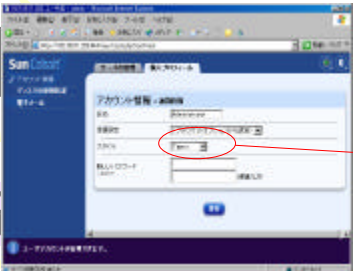
```
<styleResource name="Tigers">
.....
```



Tigersスタイルが追加された。

### STEP 3 続き

Tigersスタイルに切り替える。



Tigersスタイルに切り替わった。(今はMerlotスタイルのコピーであるため、Merlotスタイルと同一のスタイルとなっている。)

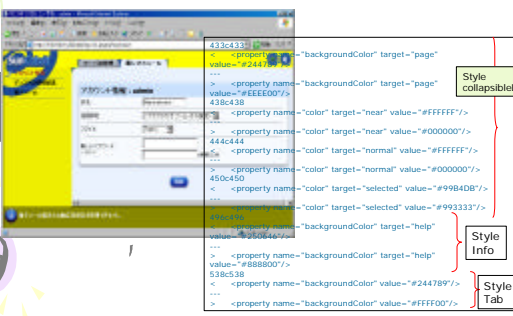
### タイガーススタイルのイメージを作る



どうやったら、タイガースっぽくなるか？

### STEP 4 :タイガーススタイルを編集する

Tigers.xmlの背景色を変更してみる



```
<property name="backgroundColor" target="page" value="#888800"/>
<property name="color" target="near" value="#FFFFFF"/>
<property name="color" target="normal" value="#FFFFFF"/>
<property name="color" target="normal" value="#000000"/>
<property name="color" target="selected" value="#9984D8"/>
<property name="color" target="selected" value="#993233"/>
<property name="backgroundColor" target="help" value="#888800"/>
<property name="background" value="#244789"/>
<property name="backgroundColor" value="#FFD900"/>
```

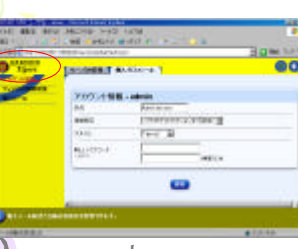
Style collapsibleList

Style Info

Style Tab

### STEP 4: 続き

左上のロゴをタイガースロゴに入れ替える。



HANSHIN Tigers

TigersLogo.gifを /usr/sausalito/ui/web/libImagesにコピー

Tigers.xmlのlogoのvalue書き換え。

```
539c539
< <property name="logo" value="/libImage/SunCobaltLogo.gif"/>
...
> <property name="logo" value="/libImage/TigersLogo.gif"/>
```

Style Tab

### STEP4: 続き

メニューの裏のビットマップを消す。



Property nameを存在しないものに変えることにより、プロパティを消す。

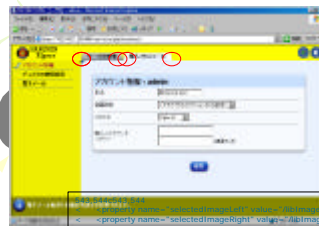
```

434c434
< <property name="backgroundImage" target="page"
value="/libImage/NavListCorner.gif"/>
---
> <property name="@backgroundImage" target="page"
value="/libImage/NavListCorner.gif"/>
  
```

Style collapsibleList

### STEP4: 続き

タブの横の、青色が残っている部分を消す。



タブの左右は画像になっているので、画像を保存し編集をおこなって、編集後の画像をTigers.xmlに設定する。

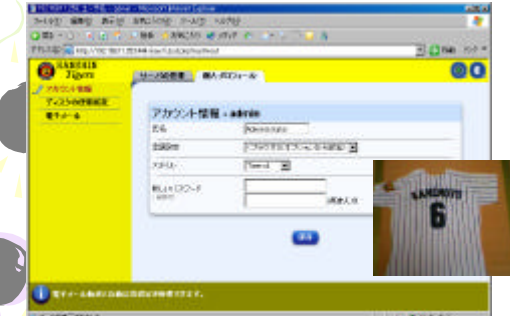
```

<property name="selectedImageLeft" value="/libImage/MerlotTabLeftSelected.gif"/>
<property name="selectedImageRight" value="/libImage/MerlotTabRightSelected.gif"/>
---
> <property name="selectedImageLeft" value="/libImage/TigerTabLeftSelected.gif"/>
> <property name="selectedImageRight" value="/libImage/TigerTabRightSelected.gif"/>
547.548:547.548
< <property name="unselectedImageLeft" value="/libImage/MerlotTabLeftUnselected.gif"/>
< <property name="unselectedImageRight" value="/libImage/MerlotTabRightUnselected.gif"/>
---
> <property name="unselectedImageLeft" value="/libImage/TigerTabLeftUnselected.gif"/>
> <property name="unselectedImageRight" value="/libImage/TigerTabRightUnselected.gif"/>
  
```

Style Tab

### STEP4: 続き

だいぶタイガースっぽくなったが、もう少し。



### STEP4: 続き

ページの背景に縦じまを入れる。



Stripe-back.gifを /usr/sausalito/ui/web/libImagesにコピー

Tigers.xmlを書き換え。

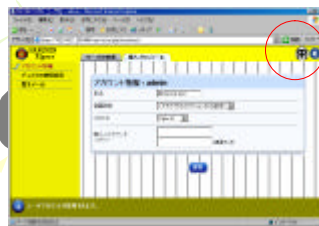
```

<property name="backgroundImage" value="/libImage/BackgroundCorner.gif"/>
<property name="backgroundRepeat" value="no-repeat"/>
---
> <property name="backgroundImage" value="/libImage/Stripe-back.gif"/>
> <property name="backgroundRepeat" value="repeat"/>
  
```

Style Page

### STEP4: 続き

アクティブモニタのアイコンをタイガースマークにする。



Stripe-back.gifを /usr/sausalito/ui/web/libImagesにコピー

Tigers.xmlを書き換え。

```

1c541
<property name="monitorOffImage" value="/libImage/MerlotStatusOkay.gif"/>
---
> <property name="monitorOffImage" value="/libImage/TigersHTbutton.gif"/>
  
```

Style Tab

### STEP4: 続き

どくなってきたが、もうひとつだけ。





### STEP4: 続き

フォームのタイトバックを赤に、文字を黄色にする。

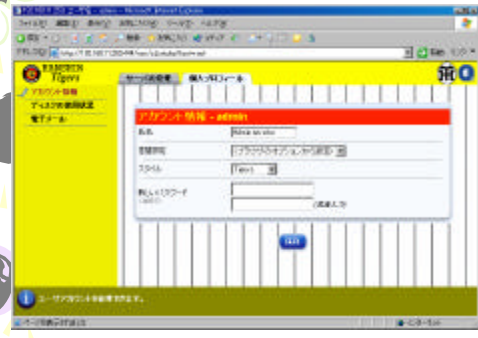


```

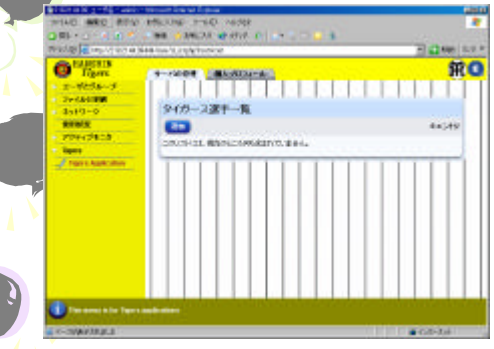
409c108
< <property name="backgroundColor" target="titleLabel" value="#C9E0FD"/>
< <property name="backgroundColor" target="titleLabel" value="#FF2200"/>
144c144
< <property name="color" target="titleLabel" value="#000000"/>
> <property name="color" target="titleLabel" value="#FFFF00"/>
  
```

Style Tab

### 課題 1 完成



### 課題 2 選手データ管理ツールを作ってみる。



### STEP1 :タイガースメニューを追加する

メニュー追加のためのディレクトリを /usr/sausalito/ui/menu の下に追加する。

```

[root@localhost root]# cd /usr/sausalito/ui/menu
[root@localhost menu]# mkdir tigers
[root@localhost menu]# cd tigers
[root@localhost tigers]#
  
```

### STEP1 続き

メニュー追加のための xm ファイルを /usr/sausalito/ui/menu/tigers の下に作成する。

**tigersmenu.xml**

```

<item
  id="base_tigers"
  label="Tigers"
  description="This menu is for Tigers applications" >
  <parent id="base_administration" order="110"/>
</item>
  
```

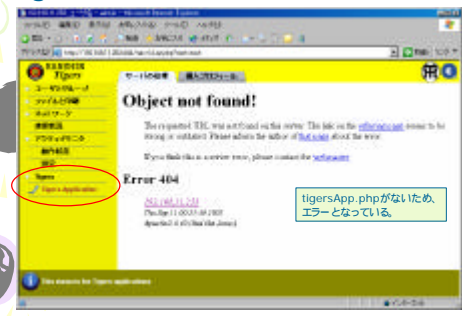
**tigersApp.xml**

```

<item
  id="base_tigersApp" label="Tigers Application"
  description="Tigers Application"
  url="/tigers/tigersApp.php" >
  <parent id="base_tigers" order="10">
</parent>
</item>
  
```

### STEP1 続き

Tigersメニューが追加された！



Object not found!  
The requested URL was not found on this server. This likely means the requested resource is not in the directory. Please check the path of the request and the error.

Error 404  
tigersApp.php がないため、エラーとなっている。

## STEP2: PHPファイルの骨組みを作る

/usr/sausalito/ui/web/tigersの下に、  
tigersApp.phpファイルを作成する。

```
[root@localhost root]# cd /usr/sausalito/ui/web
[root@localhost web]# mkdir tigers
[root@localhost web]# cd tigers
[root@localhost tigers]# touch tigersApp.php
[root@localhost tigers]#
```

## STEP2: 続き

tigersApp.phpで、UIFCを試してみる。  
tigersApp.php

```
<?php
include("ServerScriptHelper.php");
$serverScriptHelper = new ServerScriptHelper();
$cocClient = $serverScriptHelper->getCocClient();
$factory = $serverScriptHelper->getHtmlComponentFactory("tigers");
$page = $serverScriptHelper->getHtml("tigers");
$page = $factory->getPage();

// build the scroll list
$scrollList = $factory->getScrollList("playerListTitle", array("uniformNumber", "playerName",
"role", "playerDesc", "listAction"), array(1));
$scrollList->setAlignments(array("right", "left", "left", "left", "center"));
$scrollList->setColumnWidths(array("10%", "30%", "30%", "10%"));
$scrollList->addButton($factory->getAddButton(""));

$serverScriptHelper->destructor();
print($page->toHeaderHtml());
print($scrollList->toHtml());
print($page->toFooterHtml());
?>
```

## STEP2: 続き

メッセージソースファイルtigers.po

```
# --- Menu
msgid "tigersMenuTitle"
msgstr "タイガース"

msgid "tigersAppMenuTitle"
msgstr "タイガース選手情報"

# --- Common
msgid "playerListTitle"
msgstr "タイガース選手一覧"

msgid "playerName"
msgstr "名前"

msgid "uniformNumber"
msgstr "背番号"

msgid "role"
msgstr "守備位置"

msgid "playerDesc"
msgstr "詳細"

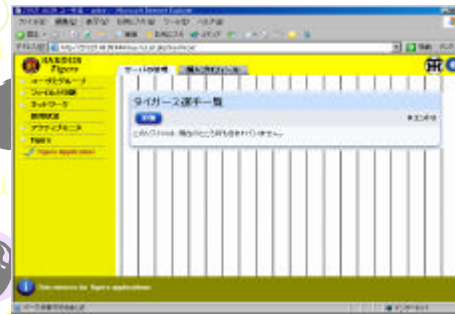
msgid "listAction"
msgstr "操作"

msgid "AddButton"
msgstr "追加"

msgid "CancelButton"
msgstr "キャンセル"
```

## 課題2 画面イメージ完成

表示結果



## 課題2 :このあとに行うこと

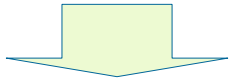
- CCEのデータベース (CoDB)へのテーブル構造 (schema)定義
- tigersApp.phpへのコード追加 (DB内容のリストの表示)
- tigersDataEntry.phpの作成 (データ入力用フォームの作成)
- tigersDataWrite.phpの作成 (データ書き込み用の作成)

## まとめ

本日は、Sausalitoを使って、オリジナルのアプリケーションを作る方法について知った。

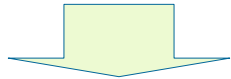
アプリケーション (専用サーバ)にはやっぱり専用の筐体が必要!

おまけ 筐体のタイガースアプライアンス化  
UIがタイガースになったら、筐体デザインもタイガースにしたい。



タイガースアプライアンスに適したPCはないか？

おまけ 筐体のタイガースアプライアンス化  
UIがタイガースになったら、筐体デザインもタイガースにしたい。



タイガースアプライアンスに適した筐体を発見した！

おまけ 筐体のタイガースアプライアンス化

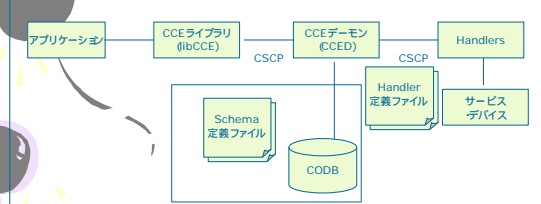


補足: CCEでのデータ管理

CCE (Cobalt Configuration Engine) とは

- CCEは、アプリケーションから、サーバ上の各機能 (ユーザ管理、デバイス、各種サービス) を管理する、Sausalito アーキテクチャーの頭脳である。

< CCEのブロックダイアグラム >



補足 :スキーマ定義ファイル

テーブル構造を定義する: テーブル: playerList

フィールド名	タイプ	入力規則 (正規表現)
uniformNumber	Scalar	
playerName	re	^.*\$
PlayerRole	re	^.*\$
PlayerDesc	re	^.*\$

スキーマ定義ファイル tigersApp.php

```

<!-- Tigers Schema -->
<typedef name="role" type="re" data="^.*$"/>
<typedef name="name" type="re" data="^.*$"/>
<typedef name="desc" type="re" data="^.*$"/>

<class name="playerList" namespace="tigers" version="1.0">
  <property name="uniformNumber" type="scalar"/>
  <property name="playerName" type="name"/>
  <property name="playerRole" type="role" default="Pitcher"/>
  <property name="playerDesc" type="desc" default=""/>
</class>

```

補足その2 :メッセージファイルについて

昨年の合宿では、メッセージファイルの日本語文字コードはSJISと説明していたが、RaQ 550及びQube3Plus、OSSでは、メッセージファイルの日本語文字コードはEUCに変わっているので、注意が必要である。