

# XOOPS Cube 2.1 カストマイズ ケーススタディー

ソースから読み解くXOOPS Cubeの  
No Hackカストマイズとは？

--- AutoLoginを題材として---

2006年 3月17日  
NobuNobu



## 本資料について

本資料は、XOOPS Cube2.1 Alphaのソースを元にした資料となっています。

今後のXOOPS Cube2.1開発過程における仕様変更等によって、本資料の内容と相違が出てくる可能性がおおいにある事を、あらかじめご了解ください。



# 目次

## ✿ XOOPS2.0.x

- ✿ なぜHackが必要なのか？
- ✿ コアのファイルは触らないけど…

## ✿ XOOPS Cube 2.1

- ✿ 柔軟性の強化
- ✿ DelegateとEvent
- ✿ Preload
- ✿ AutoLogin実装例
- ✿ LDAP認証実装例
- ✿ その他カストマイズ事例



# XOOPS2.0.x - なぜHackが必要なのか？

```
include ( “ ../../mainfile.php ” );
```

初期化処理

```
include (XOOPS_ROOT_PATH. ” /header.php” );
```

ブロック描画

モジュールロジック

```
include (XOOPS_ROOT_PATH. ” /footer.php” );
```

テーマ描画

- セッション管理

- 言語管理

- 権限設定

- 使用クラス・関数定義

カスタム  
ブロック

テンプレート

テーマ

Smartyプラグイン

一般的な  
カストマイズ

これらの部分にPHPのロジックを埋め込めば、か  
なりの自由度は増える



# XOOPS2.0.x - なぜHackが必要なのか？

- ✿ ログオン・セッション管理
  - ✿ 自動ログイン
  - ✿ ログイン認証方法の置き換え(LDAP・TypeKey)
- ✿ 言語関連
  - ✿ 多言語対応
- ✿ 描画関連
  - ✿ ファイルベーステンプレート
  - ✿ コンテンツキャッシュ
  - ✿ BBCodeへのタグ追加
- ✿ その他
  - ✿ URL置き換え



# XOOPS2.0.x - AutoLogin Hack

include/common.php

```
:  
:  
:  
:  
:  
:  
:  
セッションにxoopsUserIdが存在すれば、現ユーザ情報読込  
Cookieにログイン情報が保存されていれば、Cookieの情報から  
ユーザ情報読込
```

```
:  
:
```

AutoLoginの仕組みを追加しようとすると、common.phpに対して  
**直接ロジックを追加=Hack**  
の必要があった



# XOOPS2.0.x - [siteimg] タグ追加

```
class/module.textsanitizer.php

:
function &xoopsCodeDecode(&$text, $allowimage = 1)
{
:
:
BB CodeをHTMLに変換
[siteimg] タグに対応した変換ロジック
:
:
```



# XOOPS2.0.x - コアのファイルは触らないけど…

- ✿ コアのファイルの複製を使用  
→ コアの元ファイルにバグがあった場合の対応は、  
Hackと同じ
- ✿ コア開発者が意図していないグローバル変数を  
利用  
→ バージョンアップで使えなくなる危険性あり



# XOOPS Cube 2.1 – 柔軟性の強化

何をする？

分離

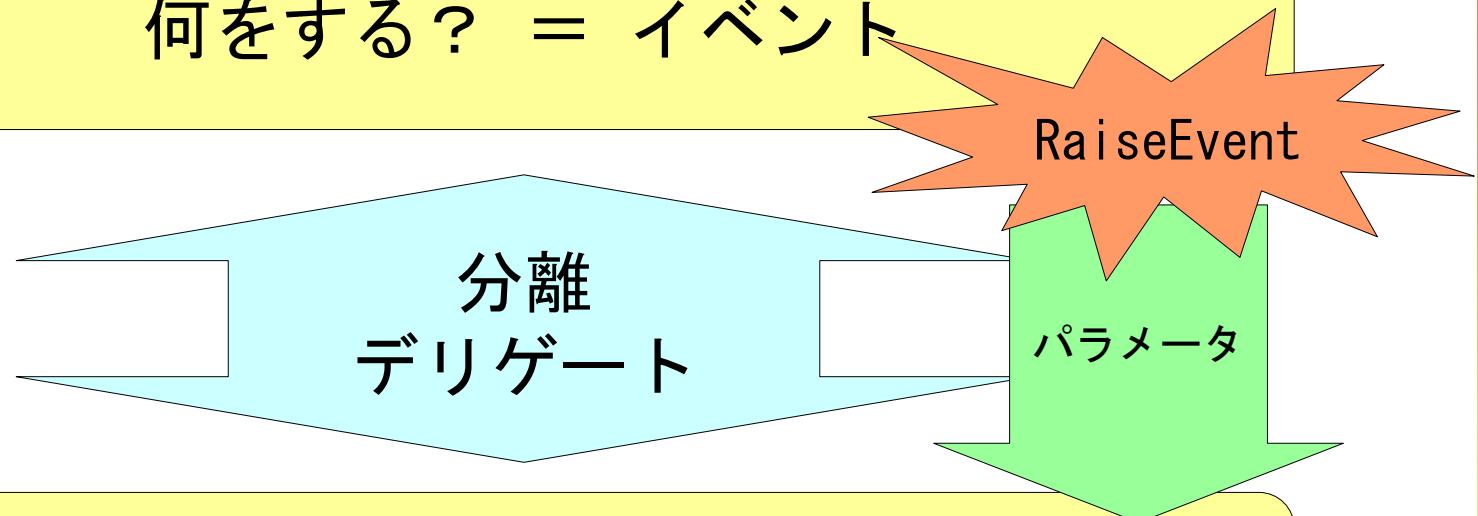
どのようにする？

どのようにする？(追加・代替案)



# XOOPS Cube 2.1 – 柔軟性の強化

何をする？ = イベント



どのようにする？ = 標準コールバック

どのようにする？(追加・代替案)  
= カストマイズコールバック



# XOOPS Cube 2.1 – イベント

- Site.Login
- Site.CheckLogin
- Site.CheckLogin.Success
- Site.Logout
  
- Legacypage.Top.Access
- Legacypage.User.Access
- Legacypage.UserInfo.Access
- Legacypage.Edituser.Access
- Legacypage.Lostpass.Access
- Legacypage.Register.Access
- Legacypage.Pmlite.Access
- Legacypage.Readpmmsg.Access
- Legacypage.Viewpmmsg.Access
- Legacypage.Misc.Access
- Legacypage.Admin.SystemCheck
  
- Legacy.XoopsTpl.New
- Legacy.XoopsTpl.TemplateTouch
  
- Legacy.TextSanitizer.MakeClickablePre
- Legacy.TextSanitizer.MakeClickablePostFilter
- Legacy.TextSanitizer.XoopsCodePre
- Legacy.TextSanitizer.XoopsCodePostFilter
  
- Legacy.Notify.Trigger
  
- Module.Admin.ActionSearch
  
- Module.User.Regist.Success



# XOOPS Cube 2.1

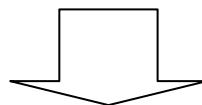
include/common.php

コントローラ初期化

```
$xoopsController->executeCommon();
```

Event	Delegate Function
Site.Login	UserCommonEventFunction::Login
Site.Login	AutoLoginHack::Login

XOOPS Cubeへの  
AutoLoginの仕組追加



Site.Login等の  
イベントに対するDelegate  
の追加

何処で追加？



kernel/XCube\_Controller.class.php

```
Class XCube_Controller {
    function executeCommon() {
        $this->mRoot->setEventManager($this->_createEventManager());
        :
        $this->_setupUser();
        :
    }
}
```

modules/base/kernel/Legacy\_Controller.class.php

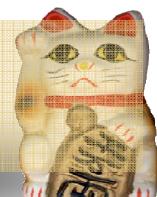
```
Class Legacy_Controller
    extends XCube_Controller {

    function &_createEventManager() {
        :
        $manager->add("Site.Login",
            new XCube_Delegate("UserCommonEventFunction", "Login"));
    }
    function _setupUser() {
        :
        $eventManager->raiseEvent("Site.Login", $this, $EventArgs);
    }
}
```

modules/user/kernel/UserEventProxyRegister.class.php

```
Class class UserCommonEventFunction
    function &Login(&$controller, &$loginEventArgs) {
        セッションにxoopsUserIdが存在すれば、
        現ユーザ情報読込
    }
}
```

Site.Login



# XOOPS Cube 2.1

Include/common.php

コントローラ初期化

```
$xoopsController->executeCommon();
```

kernel/XCube\_Controller.class.php

```
Class XCube_Controller {
    function executeCommon() {
        $this->_setupFilterChain(); //<-- 使用するフィルタのロード
        $this->_processFilter(); //<-- 先頭部分でのProtection等
        :
        $this->__processPreBlockFilter(); //<-- DBを使用してのフィルタ
        :
    }
    function _processFilter() {
        foreach($this->mFilterChain as $filter) {
            $filter->preFilter($this);
        }
    }
    function _processFilter() {
        foreach($this->mFilterChain as $filter) {
            $filter->preBlockFilter($this);
        }
    }
}
```

settings/site\_default.ini.php

[Legacy]

AutoPreload=1

modules/base/kernel/Legacy\_Controller.class.php

```
Class Legacy_Controller
    extends XCube_Controller {
    function &_setupFilterChain() {
        if($this->mRoot->getSiteConfig('Legacy', 'AutoPreload') == 1) {
            XOOPS_ROOT_PATH. "/preload/" 下のActionFilter を登録
        }
    }
}
```

settings/site\_custom.ini.php

[Legacy]

AutoPreload=1

AutoPreload=1になつていれば、preloadフォルダ下に配置された、  
XCube\_ActionFilterクラスの、preFilter()やpreBlockFilter()は、  
自動実行される。



# XOOPS Cube 2.1 - AutoLogin の実装

preload/AutoLoginHack.class.php

```
class AutoLoginHack extends XCube_ActionFilter {
    function preBlockFilter() {
        $root=&XCube_Root::getSingleton();
        $root->mEventManager->add("Site.Login", new XCube_InstanceDelegate($this, "Login"));
        $root->mEventManager->add("LegacyPage.User.Access", new XCube_InstanceDelegate($this, "AccessToUser"));
        $root->mEventManager->add("Site.CheckLogin.Success", new XCube_InstanceDelegate($this, "CheckLoginSuccess"));
        $root->mEventManager->add("Site.Logout", new XCube_InstanceDelegate($this, "Logout"));
    }
    function Login(&$controller, &$EventArgs) {
        Cookieにログイン情報が保存されていれば、Cookieの情報からユーザ情報を読込
        (これにて見なし上セッションが継続する)
    }
    function AccessToUser (&$controller, &$EventArgs) {
        if (user.phpにユーザがログインしていない状態でアクセスしたとき) {
            modules/sysutil/index.php にリダイレクト(RememberMeオプション付きログイン画面表示)
        } else if (user.php?op=login に相当するPOSTがされたとき) {
            RememberMeオプションの状態を取得してインスタンス内に保存
        }
    }
    function CheckLoginSuccess (&$sender, &$EventArgs) {
        ログイン認証画面にてログインが成功したので、RememberMeオプション設定されているときには、
        Cookieに状態保存
    }
    function Logout (&$controller, &$EventArgs) {
        Cookieを破棄
    }
}
```

別途モジュールを用意して、RememberMeオプション付きの  
ログイン画面表示とブロックを用意すれば、実装完了



# XOOPS Cube 2.1 - LDAP認証

XOOPSにユーザー登録しないと、  
XOOPSは使えないのか？

企業内のユーザ認証基盤を  
XOOPSでも活用したい

XOOPSでLDAP認証を可能に！

XOOPS Cube2.1なら、AutoLoginと同様の  
実装方式でLDAP認証が可能

特にLDAPで  
ある必要は無  
い



# XOOPS Cube 2.1 - LDAP認証 しかし……

この方式では、XOOPSのユーザ・グループの前提を超えることは、難しい。

- ・XOOPSのユーザ管理

- ・認証

- ・属性管理

- ・権限管理

The diagram illustrates the integration of Xoops User Management with external authentication modules. A central yellow cloud labeled "User モジュール換装" (User module replacement) contains the text "User モジュール換装". Above it, a blue cloud labeled "LDAP・SSO..." represents an external authentication module. Below the central cloud, an orange cloud contains the question "この部分の拡張性？" (Extensibility of this part?). To the right of the orange cloud, a green cloud contains the text "は無" (None). Ellipses on the left side of the diagram indicate the continuation of the user management process.

userテーブル、groupテーブルの構造に依存



# XOOPS Cube 2.1 - まとめ

## ✿ XOOPS Cubeコア

### ✿ デリゲート・イベント機構によって

- 柔軟性
- 保守性

の両立がねらい

### ✿ さらなるカストマイズ

- Legacy クラスの継承によって、コントローラ、レンダなどの機能拡張
  - settings/ 下のiniファイル編集により換装可能

### ✿ 究極

- Legacy部分の総置き換え

## ✿ 提案 大募集！！

### ✿ こういう目的・用途でイベントを追加したい etc..



# XOOPS Cube 2.1 - おまけ

XOOPS Cube 2.1 カストマイズ情報サイト

<http://www.nobunobu.com/>

ただいま準備構築中・・・

乞う、ご期待！！

