Zend_Controller_Plugin_SessionExpire

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Controller_Plugin_SessionExpire Component Proposal

Proposed Component Name	Zend_Controller_Plugin_SessionExpire
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Controller_Plugin_SessionExpire
Proposers	Justin Hendrickson
Zend Liaison	TBD
Revision	1.0 - 17 January 2011: Initial Draft. (wiki revision: 8)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Zend_Controller_Plugin_SessionExpire is a Zend_Controller plugin that provides userland control of session expiration by tracking a users last access time and redirecting to a custom module/controller/action if a timeout is exceeded.

2. References

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** forward requests to a specified action if the session has expired.
- This component **will not** modify the session.gc_maxlifetime or session.cookie_lifetime.
- This component **will** include a Zend_Application resource class.
- This component **will not** destroy the session (the session expire handler will be responsible for that decision).

4. Dependencies on Other Framework Components

- Zend_Application_Resource_ResourceAbstract
- Zend_Controller_Plugin_Abstract
- Zend_Date
- Zend_Session_Namespace

5. Theory of Operation

This component takes a Zend_Config or an array for configuration. The expired session module/controller/action and expiration timeout can be configured.

A Zend_Application resource class is also available which includes the ability to set the stack index to register the plugin at.

6. Milestones / Tasks

- Milestone 1: Working prototype checked into the incubator supporting all use cases.
- Milestone 2: Unit tests exist, work, and are checked into SVN.
- Milestone 3: Initial documentation exists.

7. Class Index

- Zend_Application_Resource_SessionExpire
- Zend_Controller_Plugin_SessionExpire

8. Use Cases

UC-01

Default settings

```
Zend_Controller_Front::getInstance()->registerPlugin(new Zend_Controller_Plugin_SessionExpire());
```

UC-02

Customized expired session module/controller/action and timeout.

```
$plugin = new Zend_Controller_Plugin_SessionExpire(array(
    "expiredSessionHandlerAction" => "expired",
    "expiredSessionHandlerController" => "expired",
    "expiredSessionHandlerModule" => "expired",
    "timeout" => 12345
));

Zend_Controller_Front::getInstance()->registerPlugin($plugin);
```

UC-03

Custom session namespace.

```
$plugin = new Zend_Controller_Plugin_SessionExpire(array(
    "session" => "mynamespace"
));

Zend_Controller_Front::getInstance()->registerPlugin($plugin);
```

UC-04

Custom session namespace object.

```
$plugin = new Zend_Controller_Plugin_SessionExpire(array(
    "session" => new Zend_Session_Namespace("mynamespace")
));

Zend_Controller_Front::getInstance()->registerPlugin($plugin);
```

9. Class Skeletons

```


class Zend_Controller_Plugin_SessionExpire extends Zend_Controller_Plugin_Abstract
{

    /**
     * Expired session handler action
     * @var string
     */
    private $_expiredSessionHandlerAction = "session_expired";

    /**
     * Expired session handler controller
     * @var string
     */
    private $_expiredSessionHandlerController = "error";

    /**
     * Expired session handler module
     * @var string
     */
    private $_expiredSessionHandlerModule;

    /**
     * Session namespace
```

```
     * @var Zend_Session_Namespace
     */
    private $_session;

    /**
     * Session timeout
     * @var integer
     */
    private $_timeout = 1440;

    /**
     * Current timestamp
     * @var Zend_Date
     */
    private $_timestamp;

    /**
     * Constructor
     *
     * @param array|Zend_Config $options
     */
    public function __construct($options = null)
    {
        if ($options instanceof Zend_Config) {
            $this->setConfig($options);
        } elseif (is_array($options)) {
            $this->setOptions($options);
        }
    }

    /**
     * Get the session namespace
     *
     * @return Zend_Session_Namespace
     */
    public function getSession()
    {
        if (!$this->_session) {
            $this->_session = new Zend_Session_Namespace("session-expire-namespace");
        }
        return $this->_session;
    }

    /**
     * Get the expired session handler action
     *
     * @return string
     */
    public function getExpiredSessionHandlerAction()
    {
        return $this->_expiredSessionHandlerAction;
    }

    /**
     * Get the expired session handler controller
     *
     * @return string
     */
    public function getExpiredSessionHandlerController()
    {
        return $this->_expiredSessionHandlerController;
    }

    /**
     * Get the expired session handler module
     *
     * @return string
     */
    public function getExpiredSessionHandlerModule()
    {
```

```
        if (!$this->_expiredSessionHandlerModule) {
            $this->_expiredSessionHandlerModule = Zend_Controller_Front::getInstance()
                                                                    ->getDispatcher()
                                                                    ->getDefaultModule();
        }
        return $this->_expiredSessionHandlerModule;
    }

    /**
     * Get the session timeout
     *
     * @param integer $timeout
     * @return Zend_Controller_Plugin_SessionExpire
     */
    public function getTimeout()
    {
        return $this->_timeout;
    }

    /**
     * Called before Zend_Controller_Front begins evaluating the
     * request against its routes.
     *
     * @param Zend_Controller_Request_Abstract $request
     * @throws UnexpectedValueException
     */
    public function routeStartup(Zend_Controller_Request_Abstract $request)
    {
        /** Zend_Date */
        require_once "Zend/Date.php";

        $now = new Zend_Date(
            $request->REQUEST_TIME,
            Zend_Date::TIMESTAMP
        );

        if (!isset($this->getSession()->lastAccess)) {
            $this->getSession()->lastAccess = $now->getTimestamp();
            return;
        }

        $expirationTimestamp = new Zend_Date(
            $this->getSession()->lastAccess + $this->getTimeout(),
            Zend_Date::TIMESTAMP
        );

        if ($now->isLater($expirationTimestamp)) {
            $request->setModuleName($this->getExpiredSessionHandlerModule())
                    ->setControllerName($this->getExpiredSessionHandlerController())
                    ->setActionName($this->getExpiredSessionHandlerAction())
                    ->setDispatched(false);
        } else {
            $this->getSession()->lastAccess = $now->getTimestamp();
        }
    }

    /**
     * Set options via a Zend_Config object
     *
     * @param Zend_Config $config
     * @return Zend_Controller_Plugin_SessionExpire
     */
    public function setConfig(Zend_Config $config)
    {
        return $this->setOptions($config->toArray());
    }

    /**
     * Set the expired session handler action
     *
```

```
     * @param string $action
     * @return Zend_Controller_Plugin_SessionExpire
     */
    public function setExpiredSessionHandlerAction($action)
    {
        $this->_expiredSessionHandlerAction = $action;
        return $this;
    }

    /**
     * Set the expired session handler controller
     *
     * @param string $controller
     * @return Zend_Controller_Plugin_SessionExpire
     */
    public function setExpiredSessionHandlerController($controller)
    {
        $this->_expiredSessionHandlerController = $controller;
        return $this;
    }

    /**
     * Set the expired session handler module
     *
     * @param string $module
     * @return Zend_Controller_Plugin_SessionExpire
     */
    public function setExpiredSessionHandlerModule($module)
    {
        $this->_expiredSessionHandlerModule = $module;
        return $this;
    }

    /**
     * Set the options via an array
     *
     * @param array $options
     * @return Zend_Controller_Plugin_SessionExpire
     */
    public function setOptions(array $options)
    {
        foreach ($options as $key => $value) {
            $method = "set" . ucfirst($key);

            $this->$method($value);
        }
        return $this;
    }

    /**
     * Set the session namespace
     *
     * @param string|Zend_Session_Namespace $session
     * @return Zend_Controller_Plugin_SessionExpire
     * @throws UnexpectedValueException
     */
    public function setSession($session)
    {
        if (is_string($session)) {
            /** Zend_Session_Namespace */
            require_once "Zend/Session/Namespace.php";

            $this->_session = new Zend_Session_Namespace($session);
        } elseif ($session instanceof Zend_Session_Namespace) {
            $this->_session = $session;
        } else {
            throw new UnexpectedValueException("Invalid session");
        }
        return $this;
    }
```

```
    /**
     * Set the session timeout
     *
     * @param integer $timeout
     * @return Zend_Controller_Plugin_SessionExpire
     */
    public function setTimeout($timeout)
    {
        $this->_timeout = $timeout;
        return $this;
```

```
    }

}
```

```
class Zend_Application_Resource_SessionExpire extends Zend_Application_Resource_ResourceAbstract
{

    /**
     * Stack index for the session expire plugin
     * @var integer
     */
    private $_stackIndex;

    /**
     * Initialize resource
     *
     * @return Zend_Application_Resource_SessionExpire
     * @throws Zend_Application_Resource_Exception
     */
    public function init()
    {
        /** Zend_Application_Resource_SessionExpire */
        require_once "Zend/Application/Resource/SessionExpire.php";

        $plugin = new Zend_Controller_Plugin_SessionExpire(
            $this->getOptions()
        );

        $this->getBootstrap()
             ->bootstrap("FrontController")
             ->getResource("FrontController")
             ->registerPlugin($plugin, $this->getStackIndex());

        return $plugin;
    }

    /**
     * Get the stack index to register the session expire plugin at
     *
     * @return integer
     */
    public function getStackIndex()
    {
        return $this->_stackIndex;
    }

    /**
     * Set the stack index for the session expire plugin
     *
     * @param integer $stackIndex
     * @return Zend_Application_Resource_SessionExpire
     */
    public function setStackIndex($stackIndex)
    {
        $this->_stackIndex = $stackIndex;
        return $this;
    }

}
```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>