# Do you speak .. Integration of other scripting languages than JSP

By Sebastian Himberger

# Advantages & Motivation / Why the heck?

- Integration of existing work easier
- Shorter turnaround time
- Change & add classes / scripts without application restart
- Bring in non Java developers
- More fun :)

**It may have dis-advantages too :)**

# Types of integration / How the heck ?

- Writing scripts to create content (like JSPs)
- Writing other things like e.g. scheduled tasks
- Stuff I haven't (yet) thought about (which could include DSLs, validation rules, widget default / select values)

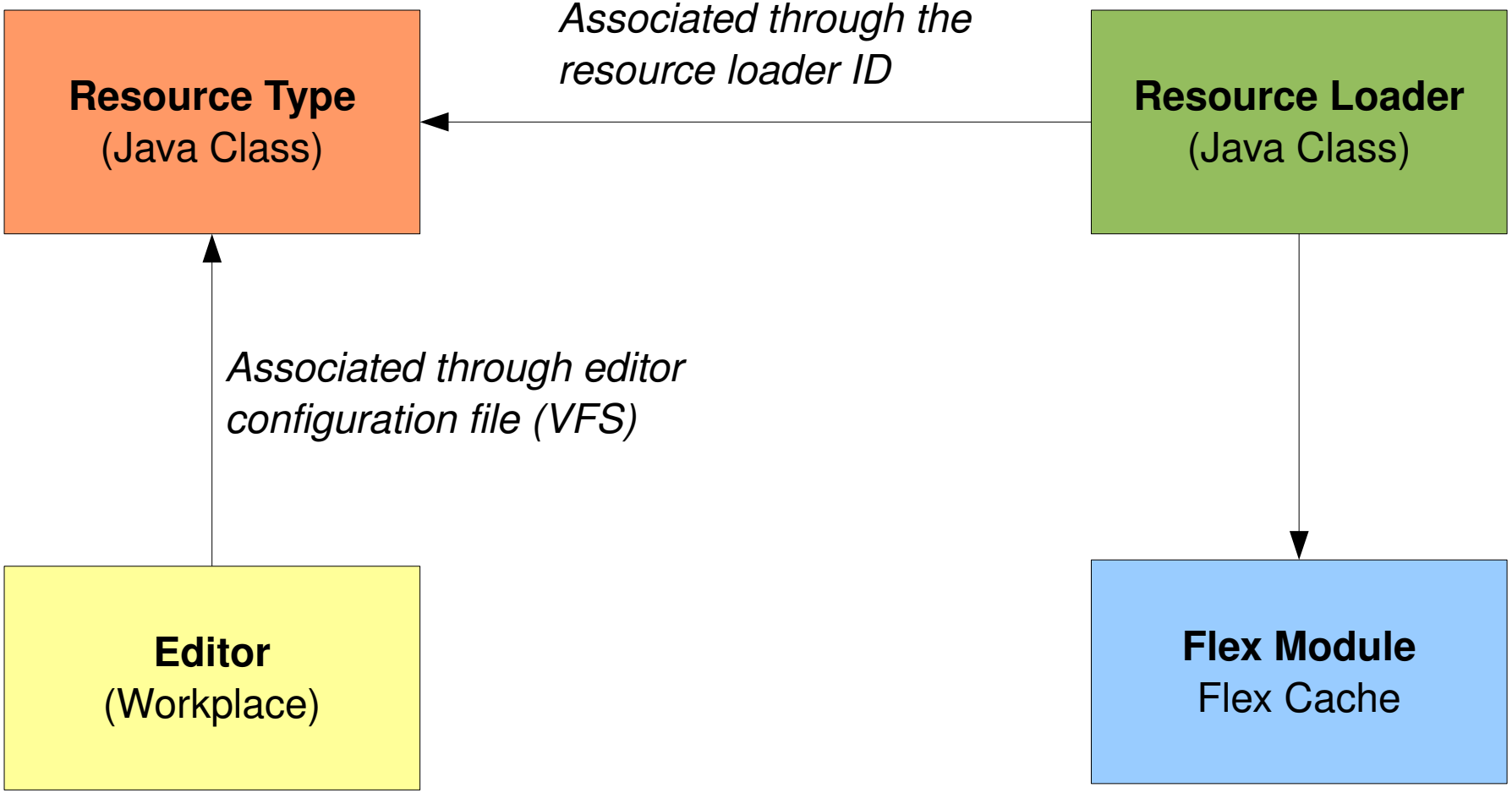
# How to approach this stuff... (we need a plan)

- Pick a language
- Choose how to invoke
  - Servlet dispatch (JSP like)
  - Direct execution
- Implement resource type
- Implement resource loader

# Architectual Overview: OpenCms Resource Types & Loaders

Used for writing and reading the resource

Used for displaying and exporting the resource



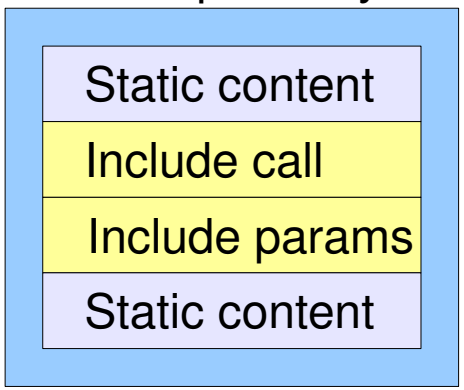

:-)

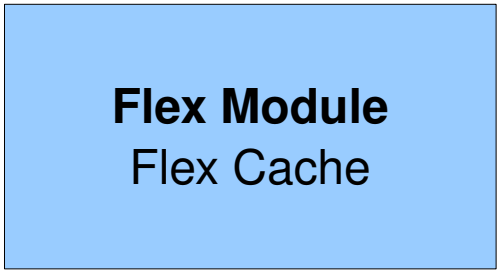
Used for caching and dispatching to JSPs (Servlets)

# The Flex Module

- Consists of wrappers for the Servlet classes (Request , Response, RequestDispatcher) to save generated output & VFS awareness
- A controller which is attached as request attribute to pass through the CmsObject (this is where the context is stored)
- Output is stored in a special Entry-Class
- Include calls are stored special so that you can mix static and dynamic cached content:

An example entry:



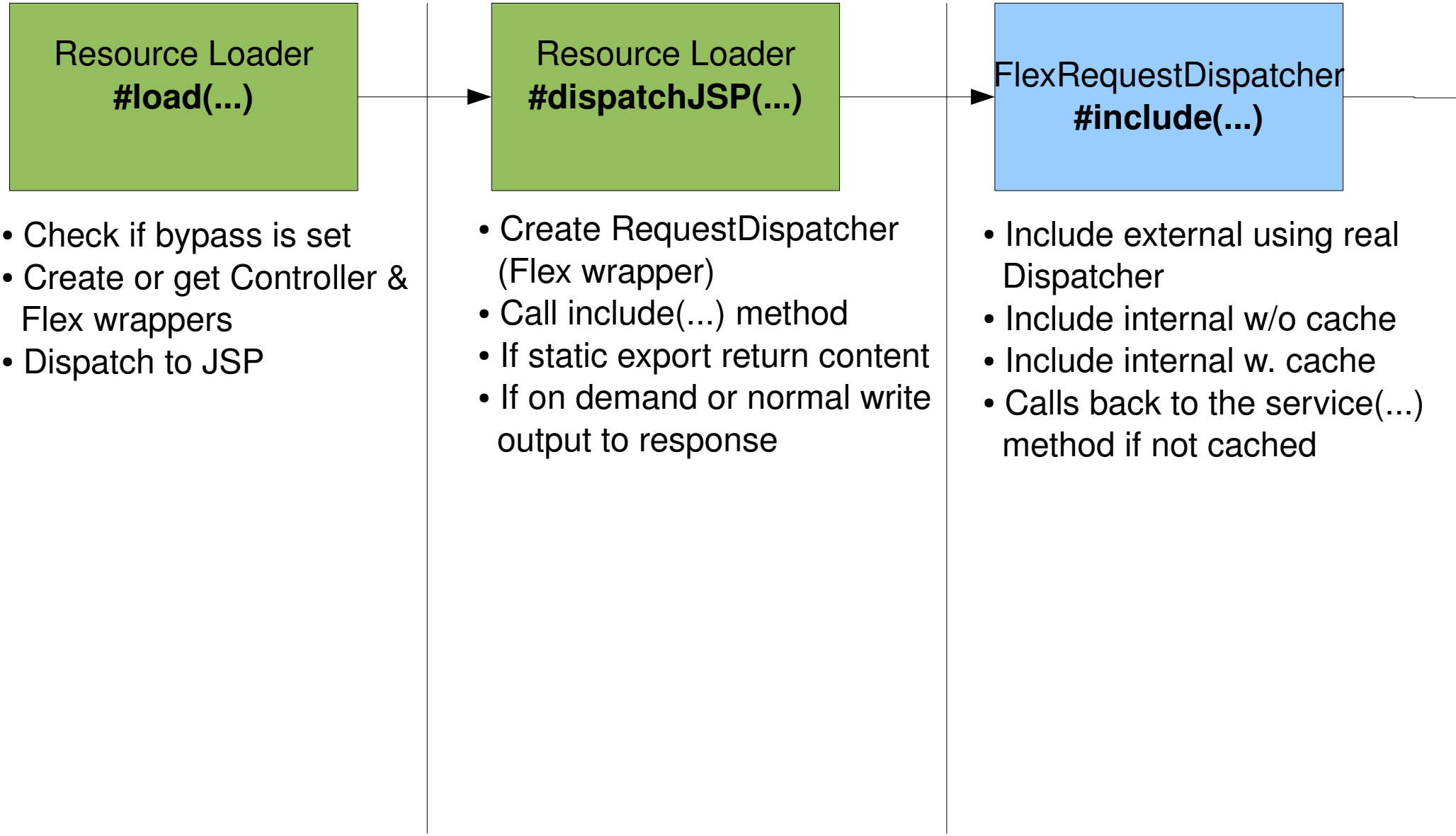



# The Resource Loader (I_CmsResourceLoader)

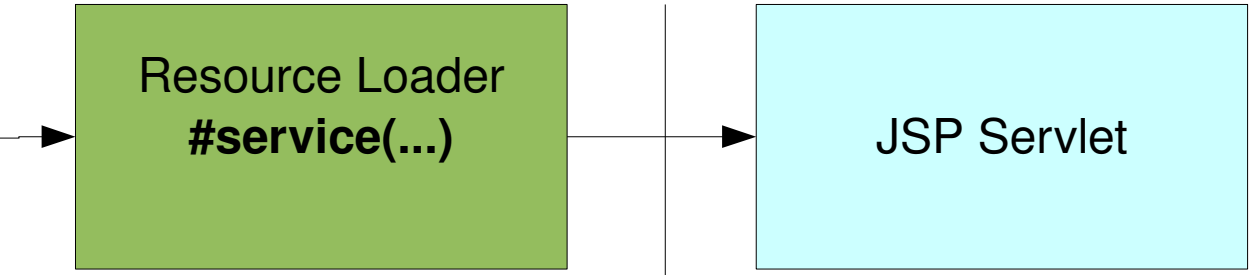
**Resource Loader**
(Java Class)

- byte[] **dump**(cms, resource, elem, locale, req, res)
- byte[] **export**(cms, resource, req, res)
- void **load**(cms,resource,req,res)
- void **service**(cms,resource,req,res)
- .. methods to indicate if usable for export etc. ...

# Request Flow of a JSP (1/2)



# Request Flow of a JSP (2/2)




- Update JSP and dependencies
- Dispatch to external JSP using normal RD
- Ouput is written to the Response wrapper buffer

- Handle the request

# Needed pieces for content generation (dispatch invocation)

- Most of the process is generic
- If repository used the **repository and the file suffix differs**
  - We have to ensure that the repository can be purged
- **Dependency** / **include management** depends on the used language
  - We have to ensure that includes / dependencies are updated correctly
- **Path translation** is dependant on the integrated language
- The **invoked Servlet differs**

# Needed pieces for content generation (dispatch invocation)

**CmsScriptResourceType**

Represents a script file
**Subclasses associate custom loader**

**CmsScriptLoader**

Performs generic loading
**Subclasses provide suffix and ensure dependencies / RFS copies, CRE stuff**

**CmsModuleAction**

Listens to OpenCms Events
**Loops through loaders and urges them to purge repository**

**CmsScriptingUtil**

Generally useful methods
**Path computations, file checking, etc.**

# PHP Integration: An example – What does it look like

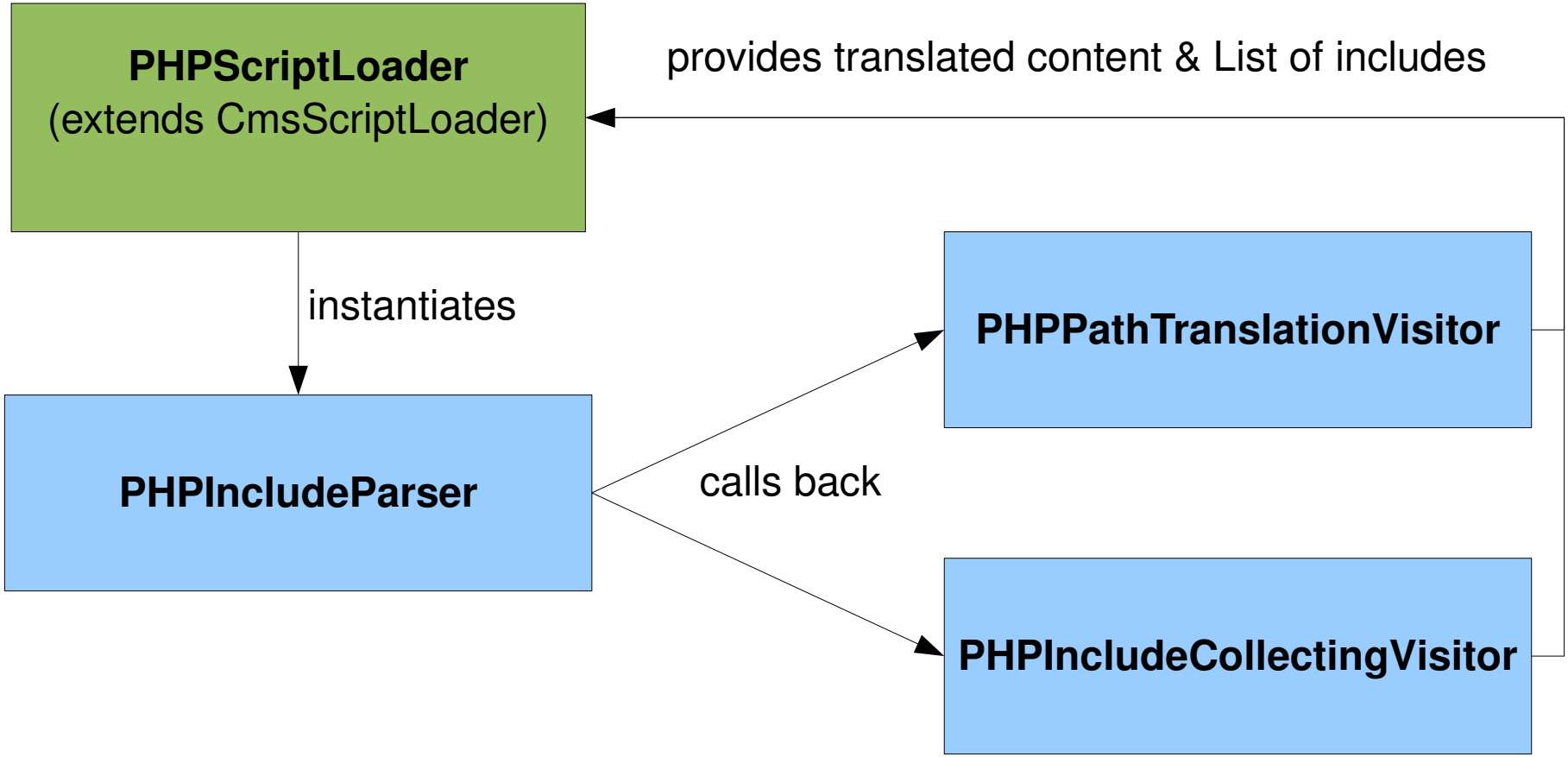
```
<%@ taglib prefix="cms" uri="http://www.opencms.org/taglib/cms" %>

<cms:include property=“template“ element=“head“/>

<cms:include file="productSearch.php" />

<cms:include property=“template“ element=“foot“/>
```

**You now get navigation, templating and caching from OpenCms ... you can even manage your PHP application via OpenCms**

# PHP Integration: An example - overview

- Integration follows the dispatched invocation model
- PHP scripts are stored in a separate RFS repository („WEB-INF/php“)
- PHP is executed by a Servlet (Quercus – it's beta but pretty(!) useable)
- PHP uses functions like „include(...)“ and „require(...)“ for inlcusion
- Not (yet) usable for replacing JSP Templates – not sure if desireable

# PHP Integration: An example – loading & dependencies




Process is recursive (as in JSP integration) to catch transitive includes (includes of includes of includes of includes of includes ... :))

# PHP Integration: An example – deployment

- **Import PHP and Scripting module (brings in resource type)**

- **Register ResourceLoader in *config/opencms-vfs.xml***

```
<resourceloaders>
  ...
  <loader class="net.sf.ocmscript.php.CmsPHPScriptLoader" />
</resourceloaders>
```

- **Assign editor */system/workplace/editors/simple/editor_configuration.xml***

```
<editor>
  <resourcetypes>
    <type>
      <name>php</name>
      <ranking>0</ranking>
      <mapto>php</mapto>
    </type>
  </resourcetypes>...
```

**If you ever need to integrate a custom resource type + editor: These are the hooks!**

## Further examples: Scheduled jobs with Groovy

- A GroovyVFSExecutor is used to execute a script in the VFS
- The CmsObject and the parameters are bound to the script
- Just create a script file in the VFS

```
import org.opencms.main.*

sessionManager = OpenCms.getSessionManager()
sessionManager.sendBroadcast(cms,params['message'])
```

# New Integrations: Steps to perform

- **Write a custom loader (for content generation)**
  - Think of include / dependency management
  - Create a Servlet for executing the script

- **Write a custom resource type (for content generation)**
  - Think of preparsing the code on read / write operations
  - Associate it with an editor

- **Think of script execution and variable binding**
  - Which variables should be available to the scripts, what to use the scripts for anyway

- **Create a module containing your classes**

# New Integrations: Pitfalls

- **Dispatched Servlet does commit the response**
  - For example calls #flushBuffer()
  - Override #isAlwaysCommited() in your loader

- **Dispatched Servlet does not throw a ServletException on error**
  - Rewrite or wrap the Servlet. The OpenCms error handling mechanism has to be invoked to display error screens

- **Most OpenCms Libraries do need a JSP PageContext**
  - Create a PageContext in the executing Servlet if you want to e.g. use the CmsJspActionElement class in your scripts (not done for PHP)

# Status & future outlook

- PHP & Groovy integrations are useable but need some testing
- Refine current integrations
- Integrate with core if needed
- Think of new ways to use scripts in OpenCms
- Develop in the open

# That's it :) - Thanks very much

Visit https://sourceforge.net/projects/ocmscripting/ for source code