Sling IDE tooling Roadmap

{float:right|background: #F0F0F0|border: solid navy} *Table of contents* {toc:indent=10px} {float}

Sling IDE tooling 1.0

The 1.0 release will be a release with minimal release which will allow users to sync content between their IDE and the repository. Also see the Sling IDE tooling Use Cases page for a list of use cases we want to support, either in the initial or in the subsequent releases.

See also the Moving forward with IDE tooling discussion on the Sling dev mailing list for a more in-depth discussion on some of the technical issues.

Naming

Our IDE tooling should not focus on a single IDE and the naming should reflect that. Possible names

- Sling Developer Tools
- Sling IDE

Platform support

We will aim to support Eclipse and IntelliJ with a first 1.0 release.

High-level architecture

Core services

The core services will be IDE-agnostic and aim to support all platforms. As such, they will be constrained to not use specific APIs. Eclipse mandates that all I/O operations be done using its resource layer.

Server control

The server control service will handle communication with a Sling Launchpad instance, including

- connecting (validating credentials)
- starting and stopping (only possible for JAR-based launchpad)

Resource serialization format

The way resources are serialized to disk is outside the scope of the Transport API. As such, we need to define a standard serialization format.

One serialization format is defined by VLT

1. files or all file-like nodes
2. directories for all directory-like folders
3. `.content.xml` for all nodes which have attributes which are not representable with files and directories

This may or may not be the ideal format for our IDE tooling. One serious drawback is that 'dot-files' are hidden by default in IDEs and the command-line. Auto-sync is also not possible for .content.xml files for now , possibly for performance reasons

We can (and probably should) define our own resource serialization format. One proposal is

1. if a resource is a file, it is represented as a file with the same name
2. if a resource is not a file, it is represented as a directory
3. properties of a non-file resource, and all additional metadata of a file is stored in a [content].xml (or json)

Transport API

Contains the APIs needed to connect to Sling launchpad and import/export content.

Transport implementations

VLT transport

File vault is in process of being donated to to ASF and is a good candidate for a transport implementation.

Pros

1. Mature application and library
2. Already used to import, export and sync content

Cons

1. Works at JCR, not at resource level

Lightweight HTTP-based transport

The current implementation is based on the Sling DefaultGetServlet and DefaultPostServlet and is another candidate for a transport implementation.

Pros

1. Works at resource level

Cons

1. Does not work reliably if the DefaultGetServlet is not active for a certain resource

Resource editor

Include a basic resource editor which works with the files defined by our serialization format.

Eclipse implementation

High-level pieces

- WST Server - we will use the Eclipse web tools platform to provide UI elements (server definition, start/stop actions, module deployments)
- Content module - we will define a content module which uses the transport API to sync content from the workspace into the repository. The mechanism is controlled by the WST server definitions
- Transport API bridge - links between the core transport APIs and the Eclipse build/resource APIs
- Manual Import/Export process, integrated with the IDE process. This would allow, for example export of sample or other project code or importing of non-synced content/code

We will inherit some code from the current implementation, and need to work out a way to export content from the repository into the workspace with the proposed UI flow.

Intellij Implementation

High-level pieces

- Application server - uses the IntelliJ application server framework
- Transport API bridge - links between the core transport APIs and the IntelliJ build/resource APIs
- Manual Import/Export process

Sling IDE tooling 1.x

Eclipse implementation

Bundle module

Once we have the content module nailed down we can focus on deploying Java code changes quickly into a Sling launchpad, using a bundle module.

Libra implementation

One possibility is reusing the Eclipse Libra stuff. However, AFAIK Libra works with Apache Felix, not over JCR, and this can be problematic.

Sling-based implementation

Another possibility is to mount a virtual FS provider at `/system/ide/install` and mount each target directory at /system/ide/install/(bundle-symbolicname)-(version).jar using a custom resource provider . The jar can be refreshed on each incremental change.

Bonus points for (somehow?) refreshing only java classes or only affected SCR components. An idea is to use a special /META-INF/last-session-changes.xml file which records what files have changed ; this file is only available when generated by the IDE (maven plugin?) and the JCR installer (or someone else) is smart enough to take it into account.

CLI implementation

If there is interest, we can also build a CLI-only implementation, although for the initial release we can defer to the vlt command-line tool.

Sling IDE 2.x

Instant feedback on changes

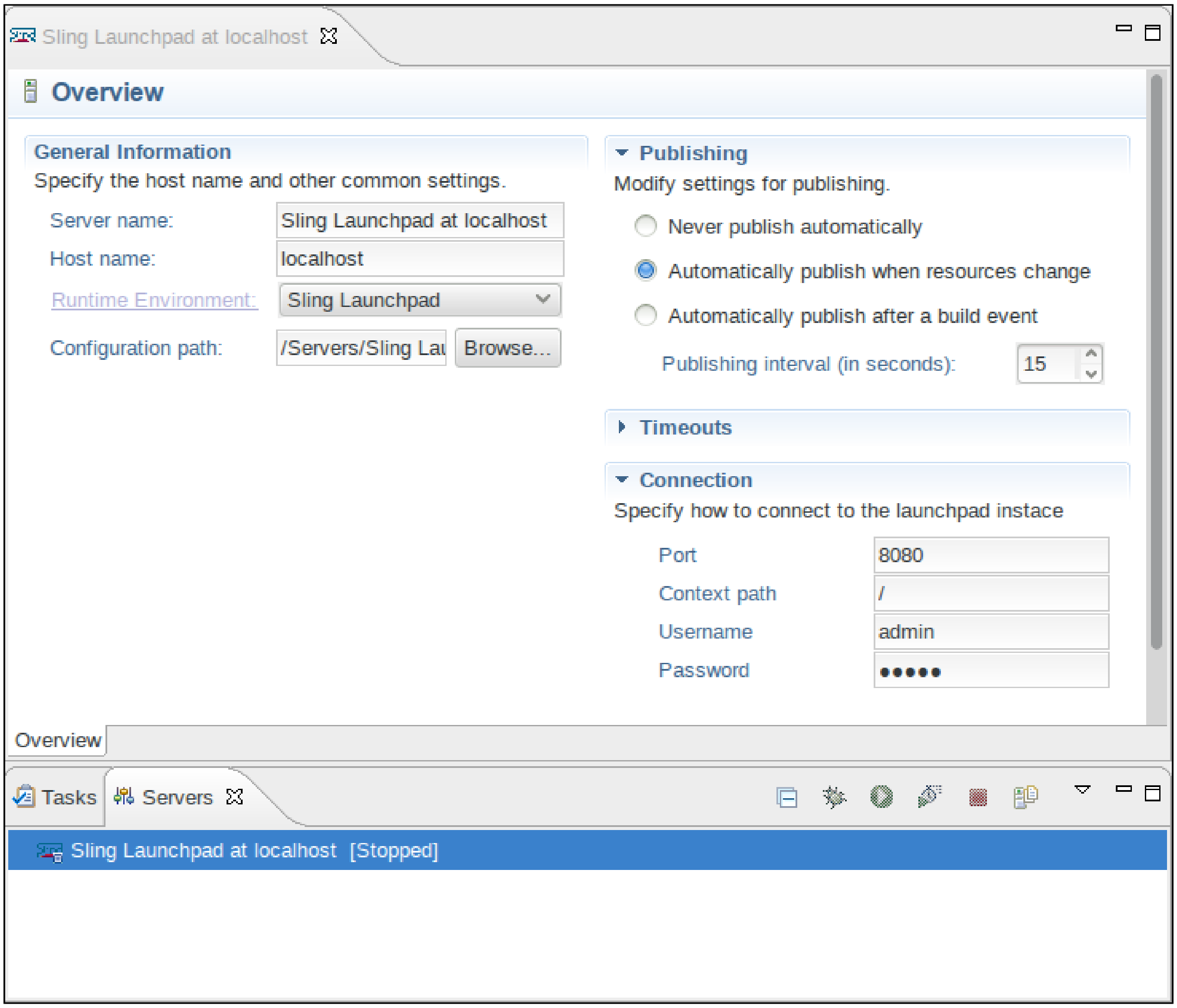
- look into http://livereload.com/ for frontend-related modifications
- look into https://github.com/SpringSource/spring-loaded for Java-related modifications

UI concepts

Eclipse

These mockups are based on a work-in-progress version of Slingclipse. Since it doesn't actually work I haven't pushed it to SVN yet.

Server definition



Sling Launchpad at localhost

Overview

General Information

Specify the host name and other common settings.

Server name:	Sling Launchpad at localhost
Host name:	localhost
Runtime Environment:	Sling Launchpad
Configuration path:	/Servers/Sling Lau

Publishing

Modify settings for publishing.

( ) Never publish automatically
(●) Automatically publish when resources change
( ) Automatically publish after a build event

Publishing interval (in seconds): 15

Timeouts

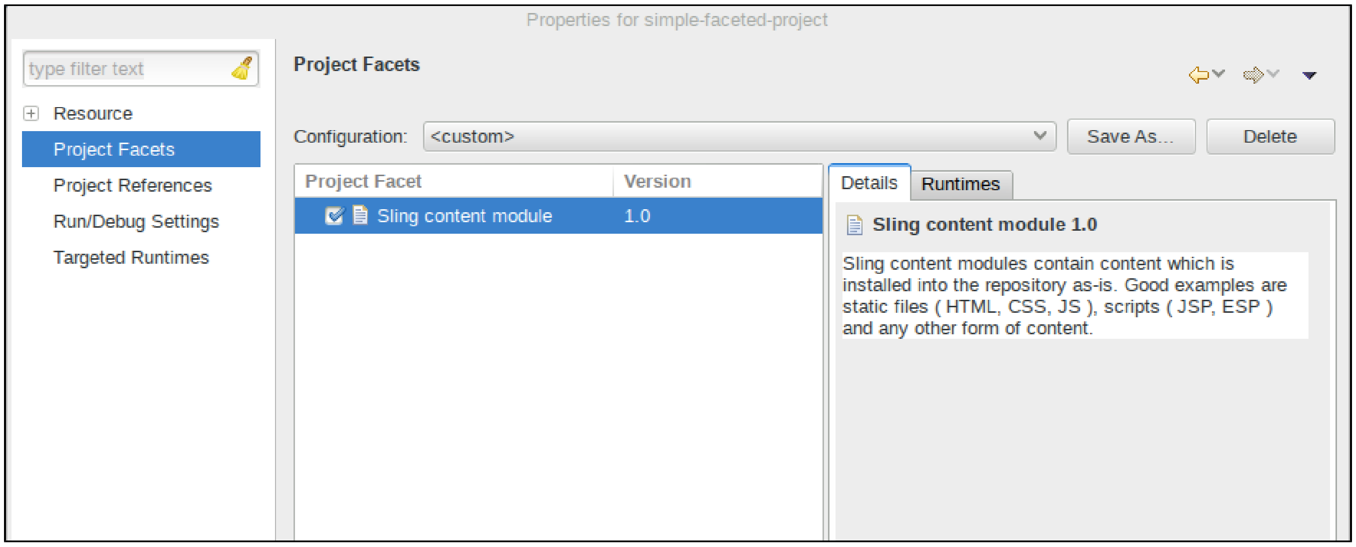
Connection

Specify how to connect to the launchpad instace

Port	8080
Context path	/
Username	admin
Password	●●●●●

Sling Launchpad at localhost [Stopped]

Content module definition



Project Facets

Configuration: <custom>

Project Facet	Version
✓ Sling content module	1.0

Sling content module 1.0

Sling content modules contain content which is installed into the repository as-is. Good examples are static files (HTML, CSS, JS), scripts (JSP, ESP) and any other form of content.