APACHE SLING & FRIENDS TECH MEETUP

BERLIN, 23-25 SEPTEMBER 2013

POST Handling in Apache Sling

Dominik Süß, pro!vision GmbH

THE OPTIONS

adaptTo()

The Options for POST Handling

- Custom Servlet
- Sling POST Servlet
  - Default PostOperations
  - Custom PostOperation
  - Custom SlingPostProcessors

Custom Servlet

Custom Servlet

```
@SlingServlet(
    paths={"/apps/adaptto/postsample"}
)
public class CustomServlet extends SlingAllMethodsServlet
{
     @Override
    protected void doPost(SlingHttpServletRequest request,
SlingHttpServletResponse response) throws ServletException,
IOException
    {
        // From now on you're on your own!
    }
}
```

adaptTo()

SlingAllMethodsServlet

- Extends SlingSafeMethodsServlet
- Supports POST, PUT & DELETE
- Just some convenience code
- No further support for building applications

Sling POST Servlet

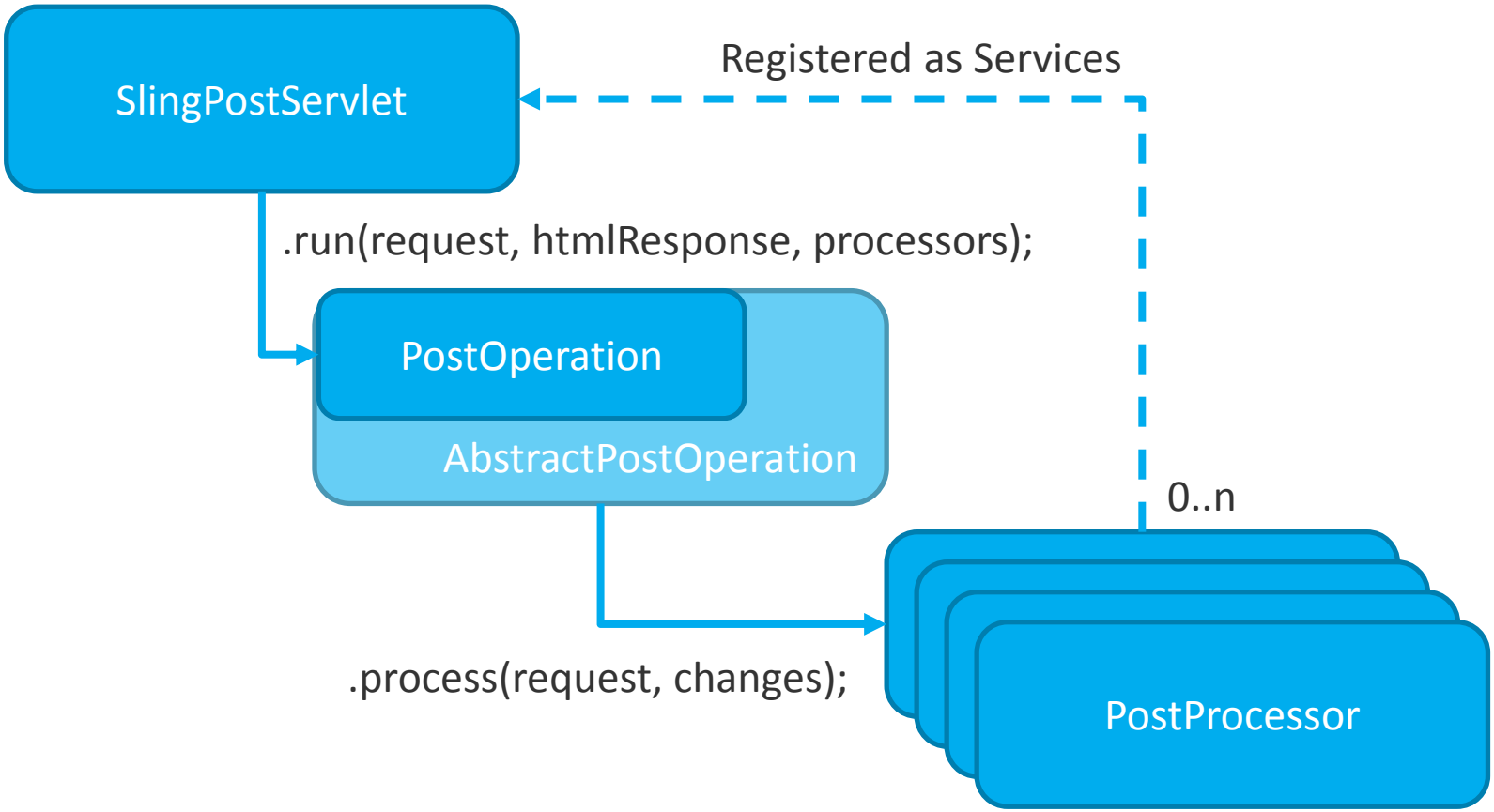
adaptTo()

Sling POST Servlet

- Default Servlet for POST Operations
- Convenient way to …
  - Create
  - Modify
  - Copy
  - Delete
  - Import

  …content in Apache Sling.

Sling POST Servlet Execution Sequence



Custom Post Operations

- Implement Interface PostOperation
- For convenience extend AbstractPostOperation
- Register as Service
- Addressed via „:operation“ Postparameter (e.g. hidden input)

```
@Component
@Service(value = PostOperation.class)
@Property(name="sling.post.operation", value=„myoperation")
public class MyOperation extends AbstractPostOperation {
 @Override
    protected void doRun(SlingHttpServletRequest request,
            PostResponse response, List<Modification> changes)
            throws RepositoryException {
                 // Perform your operation
         }
}
```

Custom Sling Post Processors

- Implement Interface SlingPostProcessor
- Register as Service

```
@Component
@Service(value = SlingPostProcessor.class)
public class MyPostProcessor implements SlingPostProcessor {
        @Override
        public void process(SlingHttpServletRequest request,
        List<Modification> changes) {
                // Perform additional changes
        }
}
```

And where are the problems?

Compability to other Resource Providers

- AbstractOperation is heavily bound to JCR API
  - Internally using Session, Node, Item
- No convenience operations for third ResourceProviders

Transactional Contract

- Although changes are dedicated to just be performed in the **transient space** a session.save() or resolver.commit() is often performed.

- From JavaDoc of SlingPostProcessor

```
After the operation has performed its
changes but before the changes are
persisted, all post processors are called.
```

- This is a contract that is not enforced and is often broken in PostProcessors and custom PostOperations.

Scope of Operations & Processors

- Scope of Operations & Processors is global.
- A developer cannot control the impact of third party code on the „own“ application

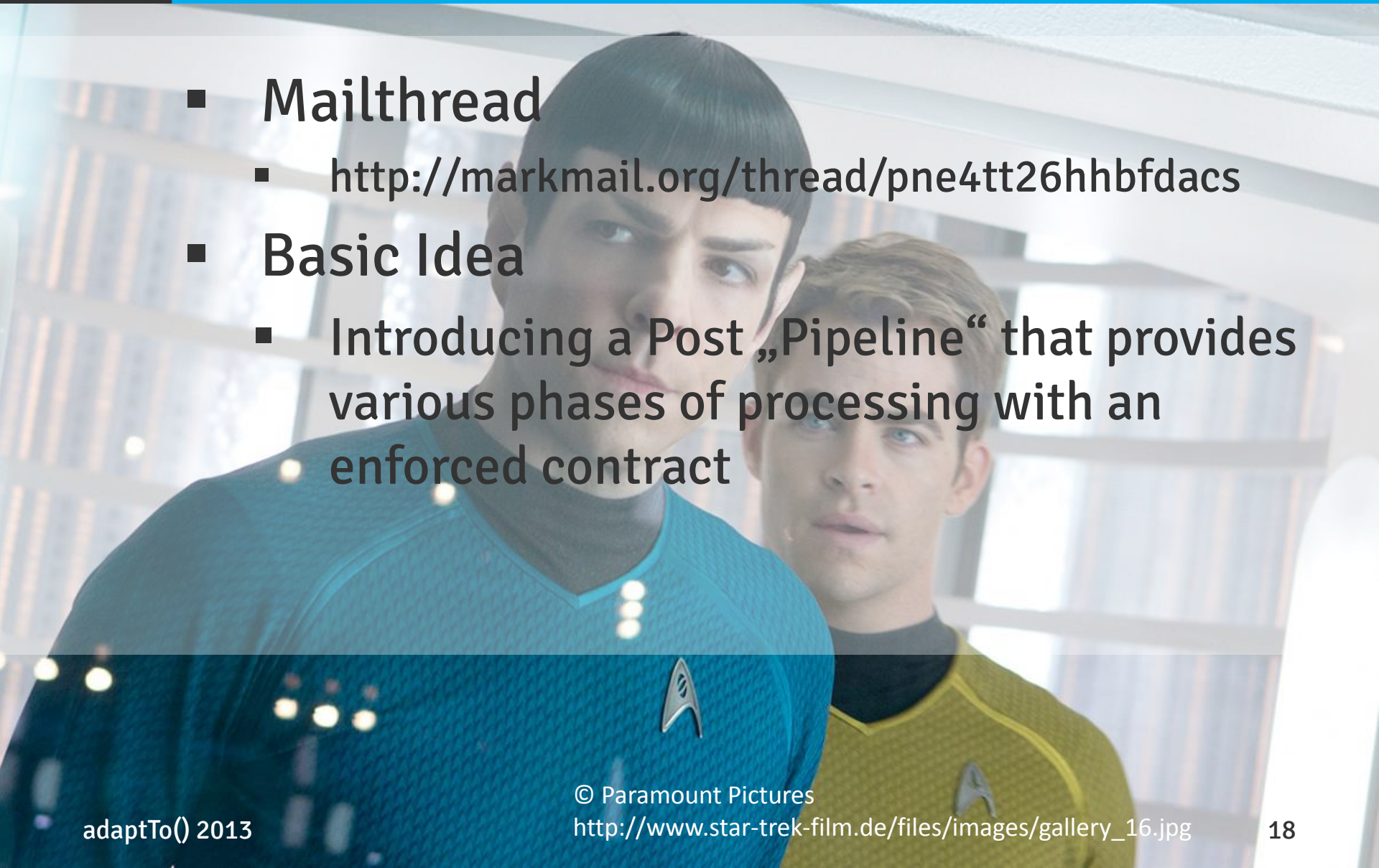
Modifications List not reliable

- The List of Modifications is not reliable *(up to devs to maintain list)*
- There is currently no way to check which modifications the POST handling already has perpared and/or persisted.

THE FUTURE

State of the discussion

- Mailthread
  - http://markmail.org/thread/pne4tt26hhbfdacs
- Basic Idea
  - Introducing a Post „Pipeline“ that provides various phases of processing with an enforced contract



adaptTo()

Potential Phases

- Validation Phase
  - Just Readaccess to Request & Resources
  - Can skip rest when marked invalid
- Preparation Phase
  - Can prepare changes but cannot persist changes
- Postprocessing Phase
  - Performing changes that may perform commit and therefore persist the state.

And what are your requirements?

(Now open for Comments & Questions)

Additional Requirements

- Finegrained options to register „PhaseProcessors“ in a dedicated scope
  - Register with Filterrules
    - (e.g. Selector, Extension, Tenant)
- Add option to restrict executed „Phaseprocessors“
  - Adding (declarative) capibilities to restrict third party code not to be executed