

The Power of the Apache Sling Resource API

Carsten Ziegeler
chiegeler@apache.org

adaptTo() 2012 – Berlin

- Member of the ASF
 - Current PMC Chair of Apache Sling
 - Apache Sling, Felix, Portals, Incubator
- RnD Team at Adobe Research Switzerland
- Article/Book Author, Conference Speaker
- Technical Reviewer
- JSR 286 Spec Group (Portlet API 2.0)



CRUD with Apache Sling

- Resource API for reading
- Workaround resource API for updating
- No resource API for create, delete (update)
 - Fallback to storage API (JCR)
- High demand for full CRUD support

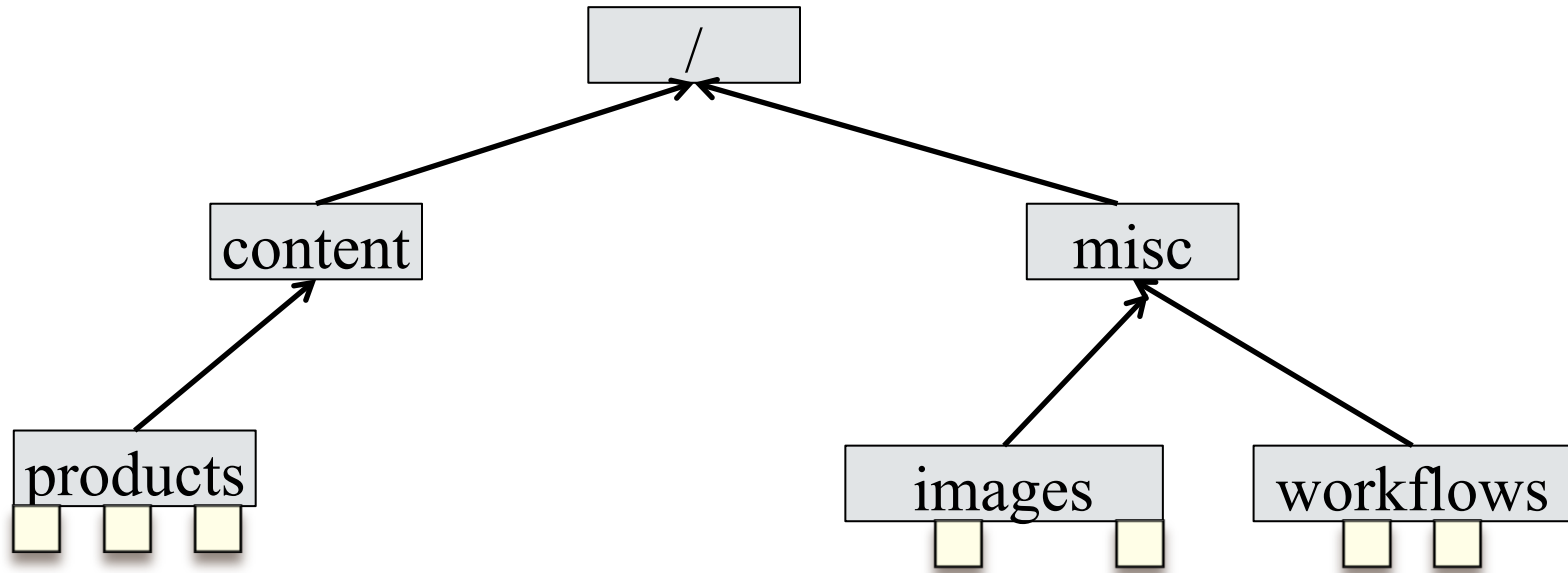
1 ROA and REST

Resource Oriented Architecture

- Every piece of information is a resource
 - News entry, product, image
 - Addressable by a (descriptive) URI
- Leverages HTTP
 - Stateless
 - Request contains all relevant information
 - Methods (GET, POST, ...) for operations
- Different Representations
 - Request contains representation wish

- Default behaviour for GET
- Creating/Updating content through POST
 - Default behaviour
- Additional operations/methods
- Resource-first request processing!

Resource Tree



<http://localhost/content/products>

Resource: /content/products

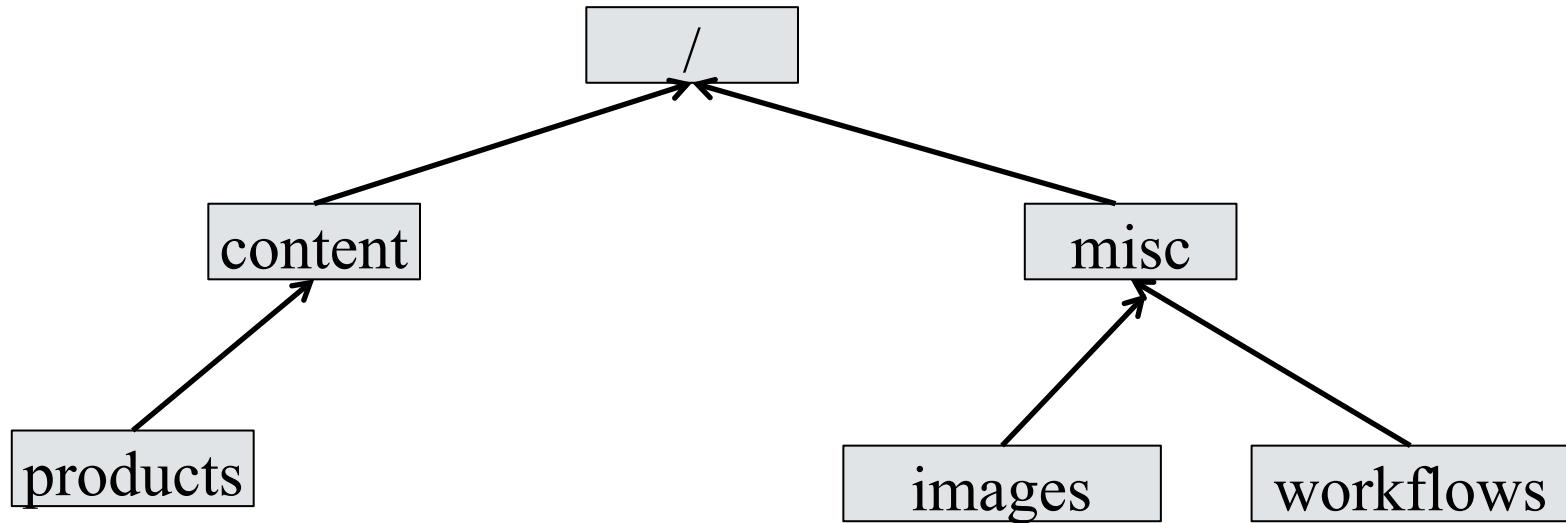
- Apache Sling's abstraction of the *thing* addressed by the request URI
 - Usually mapped to a JCR node
 - File system, bundle resource, database..
- Attributes of resources
 - Path in the resource tree
 - Resource type
 - Metadata, e.g. last modification date

2 Resource Resolver

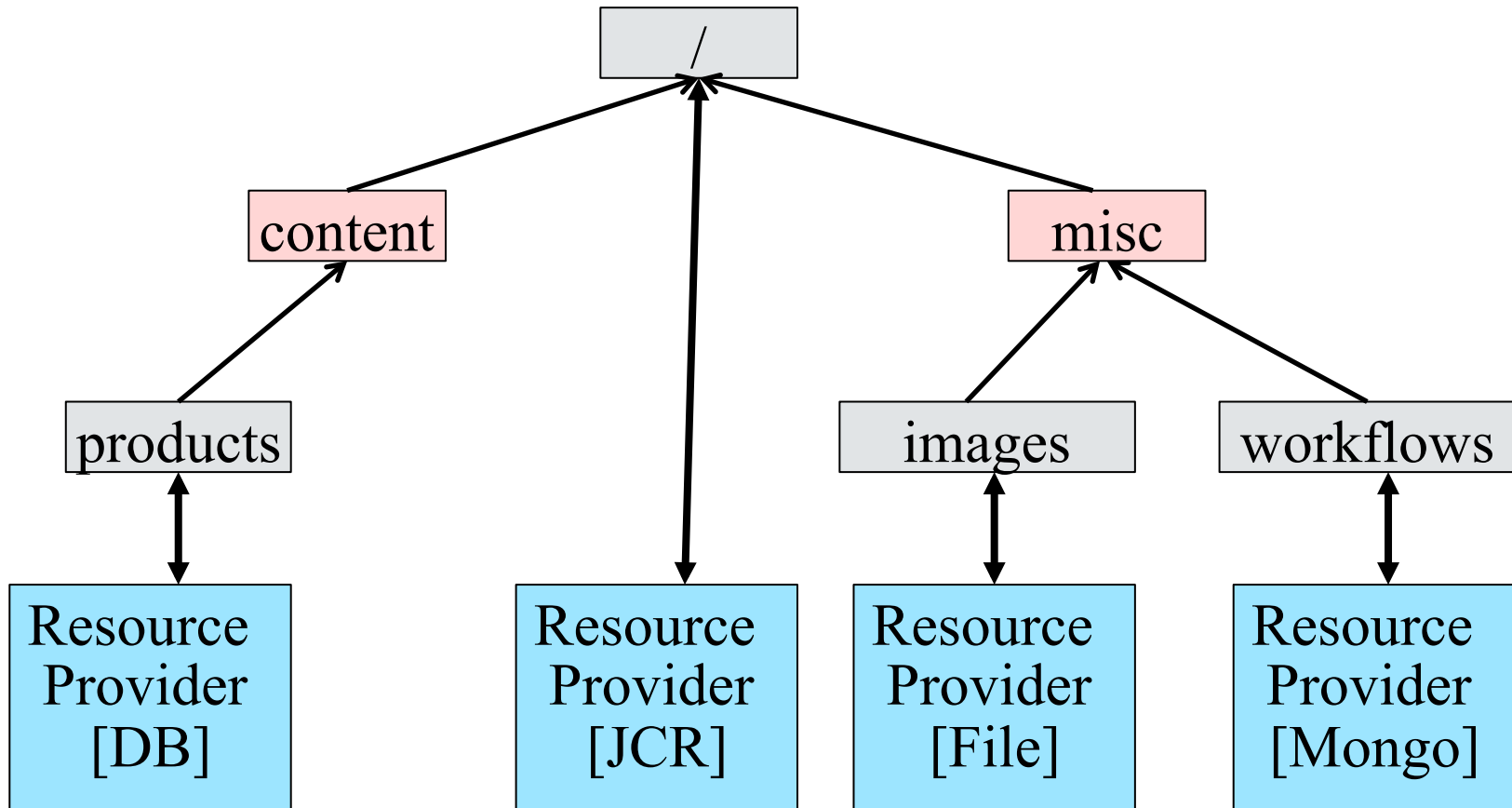
- Tasks:
 - Finding resources
 - Getting resources
 - Querying resources
- Not Thread Safe!
 - Includes all objects fetched via resolver

- Central gateway for resource handling
- Abstracts path resolution
- Abstracts access to the persistence layer(s)
- Configurable
 - Mappings (Multi site mgmt, beautify paths)

Resource Tree



Mounting Resource Providers



Resource Resolver API

```
public interface ResourceResolver extends Adaptable {  
    Resource getResource(String path);  
}
```

```
Resource r1 = resolver.getResource("/content/products/apacheSling");  
Resource r2 = resolver.getResource("/misc/images/apache/committers/cziegeler");  
Resource r3 = resolver.getResource("/misc/workflows/instances/first");  
Resource r4 = resolver.getResource("/libs/myapp/components/products/product.jsp");  
Resource r5 = resolver.getResource("/somewhere/over/the/rainbow");
```

Resource API

```
public interface Resource extends Adaptable {  
  
    String getPath();  
    String getName();  
    Resource getParent();  
    Iterator<Resource> listChildren();  
    Resource getChild(String relPath);  
    String getResourceType();  
    String getResourceSuperType();  
    boolean isResourceType(String resourceType);  
    ResourceMetadata getResourceMetadata();  
    ResourceResolver getResourceResolver();  
}
```

3 **CRUD**

The Adapter Pattern

- Resource implements Adaptable

```
<AdapterType> AdapterType adaptTo(Class<AdapterType> type);
```

- Pluggable mechanism
- Each resource *should* be adaptable to ValueMap
 - Storage independent access
- Resources might be adaptable to DSL objects
- Resources might be adaptable to persistent specific objects
 - JCR Node

Reading Resources - The Value Map

```
public interface ValueMap extends Map<String, Object> {  
    <T> T get(String name, Class<T> type);  
    <T> T get(String name, T defaultValue);  
}
```

```
String title = vm.get("title", "<no title>");
```

```
int length = vm.get("length", 80);
```

```
String[] names = vm.get("names", String[].class);
```

Traversing Resources

```
public interface Resource extends Adaptable {  
    Resource getParent();  
    Iterator<Resource> listChildren();  
    Resource getChild(String relPath);  
    ResourceResolver getResourceResolver();  
}
```

4 CRUD

CRUD Support

- Latest development
 - Check out from svn
 - MongoDB provider
- Basic API is stable
- Transient storage
 - Changes are not persisted until commit

Create and Delete

```
public interface ResourceResolver {  
    Resource create(Resource parent, String name, Map<String, Object> properties)  
    throws PersistenceException;  
  
    void delete(Resource resource)  
    throws PersistenceException;  
  
    void revert();  
  
    void commit() throws PersistenceException;  
  
    boolean hasChanges();  
}
```

```
Resource products = resolver.getResource("/content/products");  
Resource a = resolver.create(products, "a", null);  
  
Resource instances = resolver.getResource("/misc/workflow/instances");  
Resource b = resolver.create(instances, "xyz", null);  
  
resolver.delete(resolver.getResource("/somewhere/over/the/rainbow"));  
  
resolver.commit();
```

Update

```
public interface ModifiableValueMap extends ValueMap {  
}
```

```
ValueMap vm = rsrc.adaptTo(ModifiableValueMap.class);  
if ( vm != null ) {  
    vm.put("title", "Apache Sling Rocks");  
    vm.put("votes", 500);  
  
    rsrc.getResourceResolver().commit();  
} else {  
    // not modifiable!  
}
```

5 Factories

How to Get a Resource Resolver?

- SlingHttpServletRequest

```
ResourceResolver getResourceResolver();
```

- ResourceResolverFactory

```
public interface ResourceResolverFactory {
```

```
String USER = "user.name";  
String PASSWORD = "user.password";
```

```
ResourceResolver getResourceResolver(Map<String, Object> authenticationInfo)  
throws LoginException;
```

```
ResourceResolver getAdministrativeResourceResolver(  
    Map<String, Object> authenticationInfo)
```

```
throws LoginException;
```

```
}
```

Resource Provider Factory

```
public interface ResourceProviderFactory {  
    String ROOTS = "provider.roots";  
    String PROPERTY_REQUIRED = "required";  
    ResourceProvider getResourceProvider(Map<String, Object> authenticationInfo)  
    throws LoginException;  
    ResourceProvider getAdministrativeResourceProvider(  
        Map<String, Object> authenticationInfo) throws LoginException;  
}
```

Resource Provider

```
public interface ResourceProvider {  
    Resource getResource(ResourceResolver resourceResolver, String path);  
    Iterator<Resource> listChildren(Resource parent);  
}  
  
public interface DynamicResourceProvider extends ResourceProvider {  
    boolean isLive();  
    void close();  
}
```

Resource Provider - CRUD

```
public interface ModifyingResourceProvider extends ResourceProvider {  
    Resource create(ResourceResolver resolver,  
                   String path, Map<String, Object> properties)  
    throws PersistenceException;  
  
    void delete(ResourceResolver resolver, String path)  
    throws PersistenceException;  
  
    void revert(ResourceResolver resolver);  
  
    void commit(ResourceResolver resolver)  
    throws PersistenceException;  
  
    boolean hasChanges(ResourceResolver resolver);  
}
```

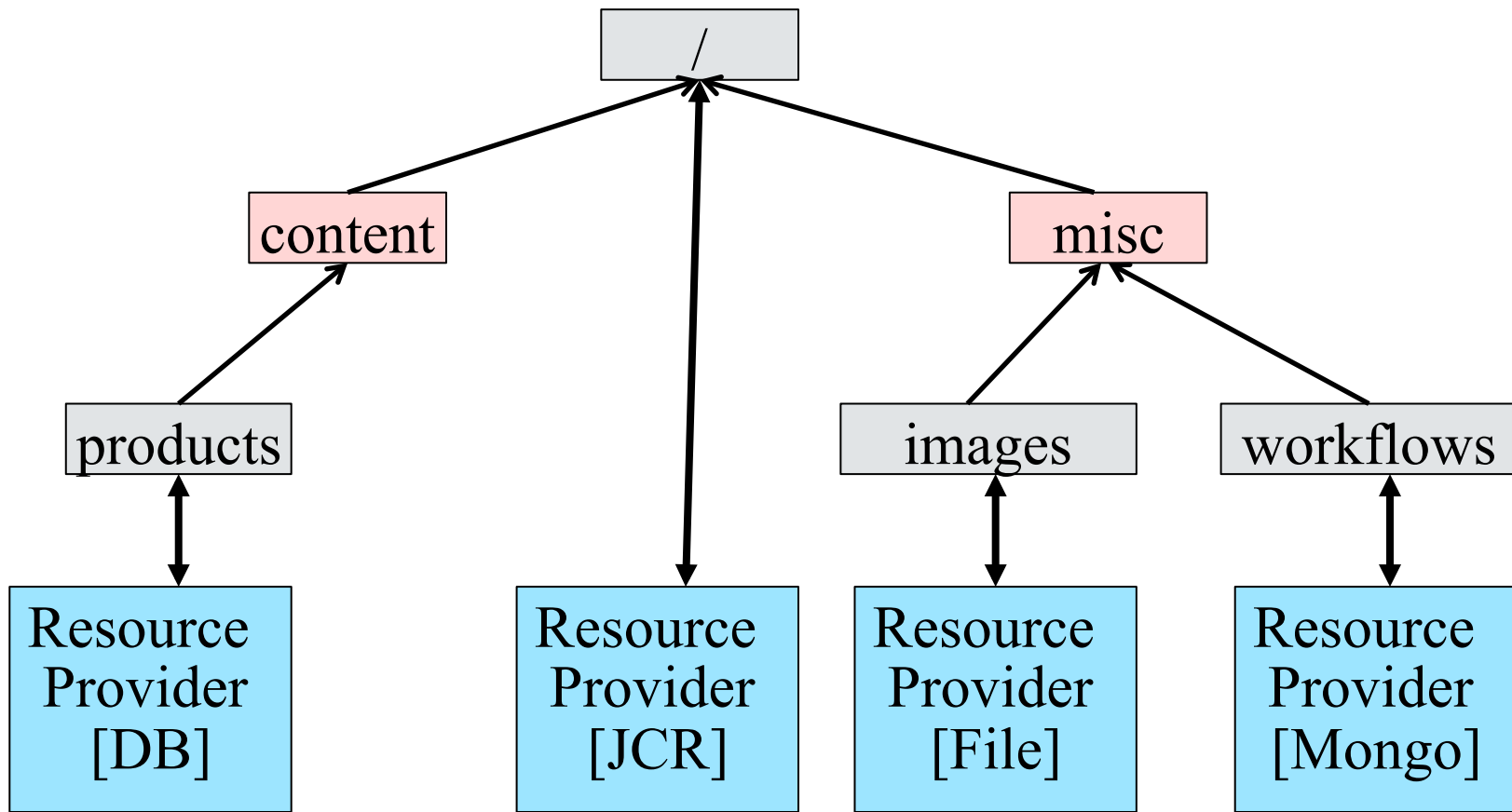
Resource Provider - Querying

```
public interface QueriableResourceProvider extends ResourceProvider {  
    String LANGUAGES = "provider.query.languages";  
    Iterator<Resource> findResources(ResourceResolver resolver,  
                                    String query, String language);  
    Iterator<Map<String, Object>> queryResources(ResourceResolver resolver,  
                                                String query, String language);  
}
```

Resource Provider – Resource Resolver Attributes

```
public interface AttributableResourceProvider extends ResourceProvider {  
    Collection<String> getAttributeNames(ResourceResolver resolver);  
    Object getAttribute(ResourceResolver resolver, String name);  
}
```

Resource Tree with Providers



Resource Tree without Providers



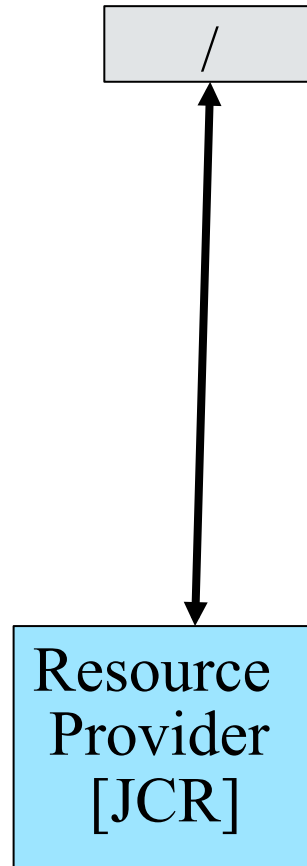
- Valid OSGi setup
- Resource Resolver (Factory) are available
- Nothing is working

Resource Tree without Providers



- Resource Provider Factory configuration:
- `resource.resolver.required.providers`
 - String array of PIDs and service filters
- Default value:
`org.apache.sling.jcr.resource.internal.helper.jcr.JcrResourceProviderFactory`

Resource Tree with Required Providers



Latest Resource API

- Full CRUD support
- Query support
- Simple, clean and sufficient API
- Transparent storage access
- Pluggable through resource provider (factories)
 - Separation of concerns

6 Outlook

Apache Sling Resource API

- CRUD Resource API defined and implemented
 - JCR Resource Provider as root provider
 - MongoDB prototype
- TODO
 - Change POST servlet to use new API
 - Test "Look and Feel"
 - Feedback
 - Release 😊
- Wish
 - Additional resource providers
 - Cassandra, CouchDB
 - File system

The End...