• RnD Team at Adobe Research Switzerland
• Member of the Apache Software Foundation
  • Apache Felix and Apache Sling (PMC and committer)
  • And other Apache projects
• OSGi Core Platform and Enterprise Expert Groups
• Member of the OSGi Board
• Book / article author, technical reviewer, conference speaker
Web Challenges

- Publish and process huge amount of information
  - Highly dynamic
  - Different types
  - Different output formats
- Collaboration and integration
- Fast changing requirements
  - Rapid prototyping and development
  - Dynamic, extensible but maintainable
Web Challenges – Entering Apache Sling

- Publish and process huge amount of information
  - Highly dynamic
  - Different types
  - Different output formats
- Collaboration and integration
- Fast changing requirements
  - Rapid prototyping and development
  - Dynamic, extensible but maintainable

JCR

REST / ROA

Scripting

OSGI
Apache Sling – The Fun is Back

- Web framework
- Java Content Repository (JCR)
- ROA / REST
- Scripting Inside
- OSGi
- Apache Open Source top level project
  - http://sling.apache.org
- Driving force behind several OSGi related projects at Apache
Key General Takeaways

- Leveraging REST
- Embracing OSG
- Hidden gems in Apache projects
Apache Jackrabbit - A Java Content Repository
Motivation for JCR

- Tried and trusted NoSQL solution
- Standard Java API
  - First spec released in May 2005
  - Various implementations, products, and solutions
  - Open Source implementation since 2006 (Apache Jackrabbit)
- Think about your data use cases / problems
  - JCR might help!
Consider JCR

- Data structure
- Supporting the web
- ACID
- Security
- Additional features
The Structure of Data

- A data storage should be flexible and
- Allow to model app data in the “right” way
  - Optimal way of dealing with the data in the app
The Structure of Data

- A data storage should be flexible and
- Allow to model data in the "right" way
- What is the "right" way?
  - Tables?
  - Key-Value-Pairs?
  - Schema based?
  - Semi structured or even unstructured?
  - Flat, hierarchical or graph?
The Structure of Data

- The right way depends on the application:
  - Tables
  - Key-Value-Pairs
  - Schema based
  - Semi structured and unstructured
  - Flat, hierarchical, and graph
  - ...
- An app might have more than one “right” way
- But: A lot of data can be modeled in a hierarchy
Sample: Product Catalog

- Books
  - English
    - Fiction
      - Douglas Adams
        - A Hitch..
    - IT
      - Databases
    - Apache Jackrabbit
  - SF
    - 2001
    - 2010

- DVDs
  - English
Java Content Repository

- Hierarchical content
  - Nodes with properties
    - (Table is a special tree)
- Structured
  - Nodetypes with typed properties
- And/or semi structured and unstructured
- Fine and coarse-grained
- Single repository for all content!
Sample: Product Catalog

Databases

Apache Jackrabbit

- title
- authors
- ISBN
- cover
Sample: Product Catalog

Databases

Apache Jackrabbit

title authors ISBN cover

Defined by node type
Authentication and Access Control

- Apache Jackrabbit supports JAAS
  - Custom login modules possible
- Deny / Allow of privileges on a node
  - Like read, write, add, delete
  - Inheritance from parent
- Tree allows structuring based on access rights
- Access control is done in the data tier!
Sample Content Structure with ACLs

- **Read for everyone, write for owner**
- **Write for owner**
JSR 170 / JSR 283: Content Repository for Java™ technology API

- (Java) **Standard** – Version 1.0 and 2.0
  - Supported by many vendors
  - Used by many products and projects
  - Several open source solutions
- Data model and features
- Query and observation
- JSR 170/283 reference implementation
- Apache TLP since 2006
- Vital community
- New implementation: OAK (!)

http://jackrabbit.apache.org/
ROA and REST
Data and the Web?

- A website is hierarchical by nature
- Web applications provide data in different ways
  - HTML
  - JSON
- Provide your data in a RESTful way
  - http://…/products/books/english/it/databases/apachejackrabbit.(html|json)
- Avoid mapping/conversion
  - http://…/products.jsp?id=5643564
Resource Oriented Architecture I

- Every piece of information is a resource
  - Descriptive URI
- Stateless web architecture (REST)
  - Request contains all relevant information
  - Targets the resource
- Leverage HTTP
  - GET for rendering, POST/PUT/DELETE for operations
JCR and Apache Jackrabbit are a perfect match for the web
  - Hierarchical
  - From a single piece of information to binaries
- Elegant way to bring data to the web
- Apache Sling is (the|one) web framework
Sample Application: Slingshot

- Digital Asset Management
  - Hierarchical storage of pictures
  - Upload
  - Tagging
  - Searching
  - Automatic thumbnail generation
- Sample application from Apache Sling

Poor man's flickr...
Facts About Slingshot

- Java web application
- Uses Apache Sling as web framework
- Content repository managed by Apache Jackrabbit
- Interaction through Sling’s Resource API
Default behavior for GET

Creating/Updating content through POST
  - Default behavior

Additional operations/methods

Resource-first request processing!
http://localhost/Travel/Europe

Resource: /Travel/Europe
Apache Sling’s abstraction of the *thing* addressed by the request URI

- Usually mapped to a JCR node
- File system, bundle, Cassandra, MongoDB, database...

Attributes of resources

- Path in the resource tree
- Resource type
- Metadata, e.g. last modification date
Resource-first Request Processing

- URI Decomposition
  - Resource and representation
  - /Travel/Europe/Basel.print.a4.html
    - Resource Path Selectors Extension
  - Content retrieved from resource tree
  - Rendering based on resource type, selectors and extension
Basic Request Processing Steps

- Resolve the resource (using URI)
  - Decomposition
- Resolve rendering script
  - Source: resource type, selectors and extension
  - Scripts wrapped by generic servlet
- Create rendering chain
  - Configurable (servlet) filters
  - Rendering servlet
- Invoke rendering chain
Tasks:

- Finding resources
- Getting resources
- Querying resources

Not Thread Safe!

- Includes all objects fetched via resolver
Resource Resolver II

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

```
/  
|___ Travel
    |___ Europe
          
/  
|___ misc
    |___ images
        
/  
|___ workflows
```
Scripting
It’s your choice
- JSP, servlet, ESP, Scala
- `javax.script`
- own script handlers

Scripts stored in OSGi bundles or the resource tree

Scripts are searched at configured locations

Default servlets
- `JSON`, `XML`
- Error Handling
Script Resolving I

- Path to script is build from…
  - Configured search paths ( /apps, /libs )
  - Resource type converted to path (slingshot/Album)
  - Selector string (print/a4)
- Request method & extension
  - GET → Extension (html)
  - Else -> Method ( POST, PUT, DELETE… )
Script Resolving Example

- URI: /Travel/Europe/Basel.print.a4.html
- Resource: /Travel/Europe/Basel
- Resource Type: slingshot:Album
- Script for GET:
  - /apps/slingshot/Album/print/a4/html.*
- Script for POST:
  - /libs/slingshot/Album/print/a4/POST.*
Script Resolving II

- Scripts are searched by best matching
  - /apps/slingshot/Album/print/a4/html.*
  - /libs/slingshot/Album/print/a4/html.*
  - /apps/slingshot/Album/html.*
  - /libs/slingshot/Album/html.*
- Resource has a type and a super type
  - Script inheritance
  - Default script (JSON...)

42
<%@page import="org.apache.sling.api.resource.Resource, org.apache.sling.api.resource.ValueMap" %><%
<%@taglib prefix="sling" uri="http://sling.apache.org/taglibs/sling/1.0" %><%
<sling:defineObjects/>

final ValueMap attributes = resource.getValueMap();
final String albumName = attributes.get("title",Resource.getName());

<html>
<head>
<title>Album <%= albumName %></title>
</head>
<body>...
<h2>Contained Albums</h2>
<%
for ( final Resource current : resource.getChildren() ) {
    if ( current.isResourceType( Constants.RESOURCE_TYPE_ALBUM ) ) {
        %><sling:include resource="<%= current %>">%/</sling:include>%
    }
}<%>
Resource-first Request Processing

- ROA
- URI decomposition
- Resource resolving
- Script resolving
  - Recursion
- Flexible script search algorithm
OSGi
Runtime Requirements

- Modularization – Modularity is key
  - Manage growing complexity
  - Support (dynamic) extensibility
- Lifecycle management
- Configuration management
- Modules, services
- Different distributions/feature sets
- Dynamic system changes
OSGi in 5..ehm..1 Minute

- Specification of a framework
- Module concept (bundles) with lifecycle
- Simple but powerful component model
  - Lifecycle management
  - Publish/Find/Bind service registration
- Dynamic!
- Uses the concept of bundles
An OSGi Bundle

- Leverages the Java packaging mechanism: JAR files
- Contains Java classes and resources
- Additional meta-data
- Implicit dependencies to other bundles
- Package imports.exports
- Semantic versioning of API
OSGi offers an API to register services
- Service is registered by its interface name(s)
- Implementation is bundle private
- Several components for same service possible (from different bundles)
- Bundles can find and use services
  - By interface names
  - With additional filters
The OSGi Core

- Minimal but sufficient API for services
- Minimal overhead: Good for simple bundles
- No support for component management
- No support for configuration management
- Requires sometimes a lot of Java coding
- Additional (optional) OSGi extensions
  - Declarative Service Specification
  - Configuration Admin Service Specification
Component model
Component lifecycle management
Publishing services
Consuming services
Default configuration
Support for Config Admin
Config Admin and Metatype

- OSGi Config Admin
  - Configuration Manager
  - Persistence storage
  - API to retrieve/update/remove configs
  - Works with Declarative Services
- OSGi Metatype Service
  - Description of bundle metadata
  - Description of service configurations
Apache Felix

- Top-level project (March 2007)
- Healthy and diverse community
- OSGi R5 implementation
- Framework (frequent releases)
- Various interesting subprojects
- Tools
  - Maven Plugins, Web Console
Contributions to Apache Felix

- Declarative service implementation
- Config admin implementation
- Metatype implementation
- Preferences implementation
- Web console (!)
- Maven SCR Plugin (!) and SCR tooling
Apache Sling Runtime

- Uses Apache Felix
- Runtime: Apache Sling Launchpad
- Two flavors
  - Standalone Java Application
  - Web application
- But Sling can be deployed in any OSGi framework!
Standalone Java Application

- One single executable JAR file
- Small Launcher
- Starts OSGi Framework (Apache Felix)
- Uses Jetty in an OSGi Bundle
Web Application

- Extends Standalone Application
- Replaces Main with a Servlet
- Uses a bridge to connect Sling to the Servlet Container
Facts About Apache Sling

- Sling API
- Uses resource abstraction
  - Use JCR, MongoDB, Cassandra...
- Highly modular and runtime configurable
- Everything is a OSGi bundle
  - Deploy what you need!
- Commons Bundles (Threads, Scheduling...)
- OSGi Provisioning
- Cloud discovery
- Distributed eventing
Apache Sling – The Fun is Back

- Web Framework
- Java Content Repository
- REST
- Scripting inside
- OSGi
- Apache Open Source project
- Check it out today!