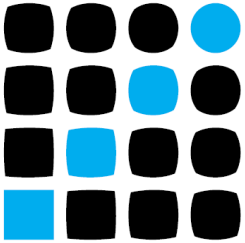


adaptTo()

APACHE SLING & FRIENDS TECH MEETUP

BERLIN, 26-28 SEPTEMBER 2016

From 0 to O(ak) in 30

Davide Giannella – Apache / Adobe

About me

About me

- Who cares? :)
- Use Google.

THE APACHE® SOFTWARE FOUNDATION

Apache Jackrabbit™

Presentation Goals

Presentation Goals

- Showing how easy it is
- Prototypes
- Only a POC
- Loads more tunings/settings

Presentation Goals

!!! !!! !!!

IT'S

NOT

PRODUCTION

READY

!!! !!! !!!

What is Oak ?

What is Oak?

Jackrabbit Oak

the next generation content repository

Jackrabbit Oak is an effort to implement a scalable and performant hierarchical content repository for use as the foundation of modern world-class web sites and other demanding content applications.

The Oak effort is a part of the Apache Jackrabbit project. Jackrabbit is a project of the Apache Software Foundation.

http://jackrabbit.apache.org/oak/

What is Oak?

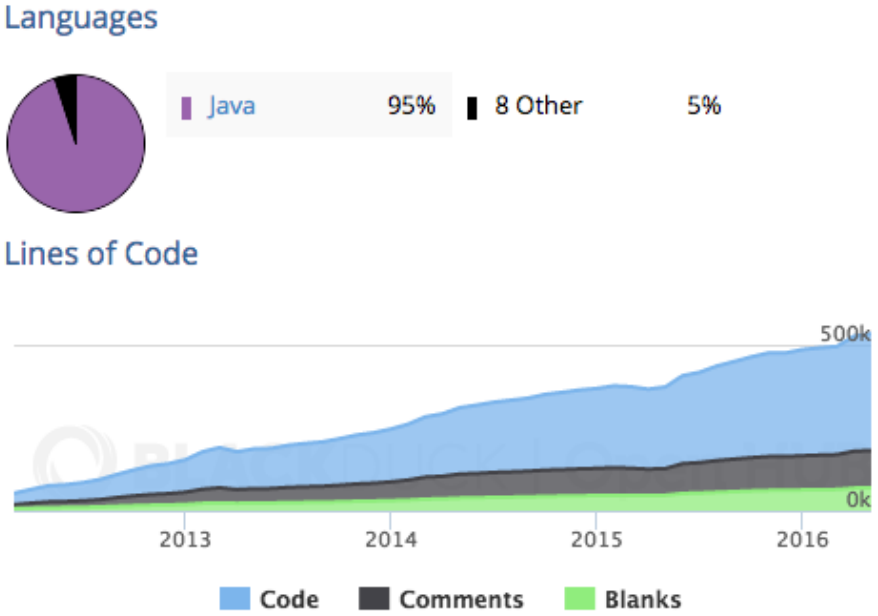
In a Nutshell, Apache Jackrabbit Oak...

... has had 10,794 commits made by 26 contributors
representing 351,343 lines of code

... is mostly written in Java
with an average number of source code comments

... has a codebase with a long source history
maintained by a large development team
with stable Y-O-Y commits

... took an estimated 94 years of effort (COCOMO model)
starting with its first commit in March, 2012
ending with its most recent commit 4 months ago



As of May 2016

https://www.openhub.net/p/jackrabbit-oak

Demo

https://github.com/davidegiannella/adaptTo16

Code Extracts

Coarse Steps

- Initialise the persistence
- Initialise JCR Content repository
- Initialise/Define indexes
- Default content
- Create the Content Repository

Segment Store initialisation (Persistence)

```
private NodeStore initialiseSegmentStore() throws IOException {
    // initialising repo root on FS
    File f = new File(REPO_PATH);
    if (!f.exists()) {
        f.mkdir();
    }

    // initialising datastore on FS
    BlobStore blob;
    File blobDir = new File(DS_PATH);
    if (!blobDir.exists()) {
        blobDir.mkdir();
    }
    FileDataStore fileDataStore = new FileDataStore();
    fileDataStore.setPath(blobDir.getAbsolutePath());
    fileDataStore.init(null);
    blob = new DataStoreBlobStore(fileDataStore);

    LOG.debug("Initalsing the NodeStore");
    FileStoreBuilder fileBuilder = FileStoreBuilder.fileStoreBuilder(new File(SEGMENT_PATH)).withBlobStore(blob);
    try {
        fileStore = fileBuilder.build();
    } catch (InvalidFileStoreVersionException e) {
        LOG.error("Error initialising the repository", e);
        throw new IOException(e);
    }
    return SegmentNodeStoreBuilders.builder(fileStore).build();
}
```

Segment Store initialisation (Persistence)

```
private NodeStore initialiseSegmentStore() throws IOException {
    // initialising repo root on FS
    File f = new File(REPO_PATH);
    if (!f.exists()) {
        f.mkdir();
    }

    // initialising datastore on FS
    BlobStore blob;
    File blobDir = new File(DS_PATH);
    if (!blobDir.exists()) {
        blobDir.mkdir();
    }
    FileDataStore fileDataStore = new FileDataStore();
    fileDataStore.setPath(blobDir.getAbsolutePath());
    fileDataStore.init(null);
    blob = new DataStoreBlobStore(fileDataStore);

    LOG.debug("Initalsing the NodeStore");
    FileStoreBuilder fileBuilder = FileStoreBuilder.fileStoreBuilder(new File(SEGMENT_PATH)).withBlobStore(blob);
    try {
        fileStore = fileBuilder.build();
    } catch (InvalidFileStoreVersionException e) {
        LOG.error("Error initialising the repository", e);
        throw new IOException(e);
    }
    return SegmentNodeStoreBuilders.builder(fileStore).build();
}
```

Segment Store initialisation (Persistence)

```
private NodeStore initialiseSegmentStore() throws IOException {
    // initialising repo root on FS
    File f = new File(REPO_PATH);
    if (!f.exists()) {
        f.mkdir();
    }

    // initialising datastore on FS
    BlobStore blob;
    File blobDir = new File(DS_PATH);
    if (!blobDir.exists()) {
        blobDir.mkdir();
    }
    FileDataStore fileDataStore = new FileDataStore();
    fileDataStore.setPath(blobDir.getAbsolutePath());
    fileDataStore.init(null);
    blob = new DataStoreBlobStore(fileDataStore);

    LOG.debug("Initalsing the NodeStore");
    FileStoreBuilder fileBuilder = FileStoreBuilder.fileStoreBuilder(new File(SEGMENT_PATH)).withBlobStore(blob);
    try {
        fileStore = fileBuilder.build();
    } catch (InvalidFileStoreVersionException e) {
        LOG.error("Error initialising the repository", e);
        throw new IOException(e);
    }
    return SegmentNodeStoreBuilders.builder(fileStore).build();
}
```

Segment Store initialisation (Persistence)

```
private NodeStore initialiseSegmentStore() throws IOException {
    // initialising repo root on FS
    File f = new File(REPO_PATH);
    if (!f.exists()) {
        f.mkdir();
    }

    // initialising datastore on FS
    BlobStore blob;
    File blobDir = new File(DS_PATH);
    if (!blobDir.exists()) {
        blobDir.mkdir();
    }
    FileDataStore fileDataStore = new FileDataStore();
    fileDataStore.setPath(blobDir.getAbsolutePath());
    fileDataStore.init(null);
    blob = new DataStoreBlobStore(fileDataStore);

    LOG.debug("Initalsing the NodeStore");
    FileStoreBuilder fileBuilder = FileStoreBuilder.fileStoreBuilder(new File(SEGMENT_PATH)).withBlobStore(blob);
    try {
        fileStore = fileBuilder.build();
    } catch (InvalidFileStoreVersionException e) {
        LOG.error("Error initialising the repository", e);
        throw new IOException(e);
    }
    return SegmentNodeStoreBuilders.builder(fileStore).build();
}
```

Segment Store initialisation (Persistence)

```
private NodeStore initialiseSegmentStore() throws IOException {
    // initialising repo root on FS
    File f = new File(REPO_PATH);
    if (!f.exists()) {
        f.mkdir();
    }

    // initialising datastore on FS
    BlobStore blob;
    File blobDir = new File(DS_PATH);
    if (!blobDir.exists()) {
        blobDir.mkdir();
    }
    FileDataStore fileDataStore = new FileDataStore();
    fileDataStore.setPath(blobDir.getAbsolutePath());
    fileDataStore.init(null);
    blob = new DataStoreBlobStore(fileDataStore);

    LOG.debug("Initalsing the NodeStore");
    FileStoreBuilder fileBuilder = FileStoreBuilder.fileStoreBuilder(new File(SEGMENT_PATH)).withBlobStore(blob);
    try {
        fileStore = fileBuilder.build();
    } catch (InvalidFileStoreVersionException e) {
        LOG.error("Error initialising the repository", e);
        throw new IOException(e);
    }
    return SegmentNodeStoreBuilders.builder(fileStore).build();
}
```

Initialise Content Repository

```
LOG.debug("Initialising Jcr Content Repository");
Jcr jcr = new Jcr(store);
```

Initialise Indexes (Query)

```
// initialising property index - http://jackrabbit.apache.org/oak/docs/query/property-index.html
jcr.with(new PropertyIndexEditorProvider());
jcr.with(new RepositoryInitializer() {
    @Override
    public void initialize(@Nonnull NodeBuilder builder) {
        NodeBuilder index = builder.getChildNode("oak:index").getChildNode("colour");
        if (index.exists()) {
            return;
        }

        LOG.debug("Property Index not found defining `colour`");

        index = IndexUtils.getOrCreateOakIndex(builder);
        IndexUtils.createIndexDefinition(index, "colour", true, false, of("colour"), null);
    }
});
```

Initialise Indexes (Query)

```
// initialising property index - http://jackrabbit.apache.org/oak/docs/query/property-index.html
jcr.with(new PropertyIndexEditorProvider());
jcr.with(new RepositoryInitializer() {
    @Override
    public void initialize(@Nonnull NodeBuilder builder) {
        NodeBuilder index = builder.getChildNode("oak:index").getChildNode("colour");
        if (index.exists()) {
            return;
        }

        LOG.debug("Property Index not found defining `colour`");

        index = IndexUtils.getOrCreateOakIndex(builder);
        IndexUtils.createIndexDefinition(index, "colour", true, false, of("colour"), null);
    }
});
```

Initialise Indexes (Query)

```
// initialising property index - http://jackrabbit.apache.org/oak/docs/query/property-index.html
jcr.with(new PropertyIndexEditorProvider());
jcr.with(new RepositoryInitializer() {
    @Override
    public void initialize(@Nonnull NodeBuilder builder) {
        NodeBuilder index = builder.getChildNode("oak:index").getChildNode("colour");
        if (index.exists()) {
            return;
        }

        LOG.debug("Property Index not found defining `colour`");

        index = IndexUtils.getOrCreateOakIndex(builder);
        IndexUtils.createIndexDefinition(index, "colour", true, false, of("colour"), null);
    }
});
```

Default Content (RepositoryInitializer)

```
// initialising a bunch of nodes
jcr.with(new BunchOfColours());
```

```
@Override
public void initialize(@Nonnull NodeBuilder builder) {
    if (builder.getChildNode(UNITED_COLOURS).exists()) {
        // nodes are already there.
        return;
    }
    NodeBuilder unitedColours = builder.child(UNITED_COLOURS);
    for (int i = 0; i<100; i++) {
        unitedColours.child(String.format("n%03d", i))
                .setProperty(JCR_PRIMARYTYPE, NT_OAK_UNSTRUCTURED)
                .setProperty("colour", COLOURS.getRandomColour());
    }
}
```

Content Repository Creation

```
jcrRepo = jcr.createRepository();
LOG.debug("Jcr Content Repository initialised. {}", jcrRepo);
```

Unit Testing ?

```
public class TestRepository extends Repository {
    public TestRepository() throws IOException {
        super(new MemoryNodeStore());
        setRepo(this);
    }
}
```

```
public class ListCommandTest {
    private final Command CMD = new ListCommand();

    @Test
    public void missingMandatoryPath() throws IOExce
        // initialising a test repository
        new TestRepository();

        StringWriter writer = null;
        PrintWriter pw = null;
```

Oh Yeah! Easy Peasy!

Q&A

@flyingedivad