Amazon S3

- 1.
- 2.
- 3.
- 4.
- 5./ API
- 6.
- 7.
  - a.
  - b.
  - c.
  - d
  - e. Felix
  - f.
- 8.

Amazon S3 Datalist Binder Bean Shell

1.

Amazon S3

2.

Datalist Binder Datalist Binder

> Datalist BinderAmazon S3API

3.

Amazon S3 Datalist Binder

1. Amazon S3 API
2. Amazon S3 API
3. S3
4. Amazon S3 Bucket
5. /

Amazon S3

4.

Amazon S3

5./ API

Java AWS

6.

Joget

Macbook ProJoget5.0.1

```
- Home
  - joget
    - plugins
    - jw-community
      -5.0.1
```

“”“jw-community”Joget Workflow

“plugins”maven

```
cd joget/plugins/
~/joget/jw-community/5.0.1/wflow-plugin-archetype/create-plugin.sh org.joget amazon_s3_datalist_binder 5.0.1
```

shellmaven

```
Define value for property 'version':  1.0-SNAPSHOT: : 5.0.0
[INFO] Using property: package = org.joget
Confirm properties configuration:
groupId: org.joget
artifactId: amazon_s3_datalist_binder
version: 5.0.0
package: org.joget
Y: : y
```

“BUILD SUCCESS”“plugins”“amazon_s3_datalist_binder”

IDEmaven NetBeans

7.

a.

“org.joget”“AmazonS3DatalistBinder”**org.joget.apps.datalist.model.DataListBinderDefault** Datalist **org.joget.plugin.base.PluginWebSupport** Ajax Web

b.

AppPluginUtil.getMessagei18nMESSAGE_PATH

Implementation of all basic abstract methods

```
package org.joget;

import org.joget.apps.app.service.AppPluginUtil;
import org.joget.apps.app.service.AppUtil;
import org.joget.apps.datalist.model.DataListBinderDefault;
import org.joget.plugin.base.PluginWebSupport;

public class AmazonS3DatalistBinder extends DataListBinderDefault implements PluginWebSupport {
    private final static String MESSAGE_PATH = "message/AmazonS3DatalistBinder";

    @Override
    public String getName() {
        return "Amazon S3 Datalist Binder";
    }
    @Override
    public String getVersion() {
        return "5.0.0";
    }

    @Override
    public String getClassName() {
        return getClass().getName();
    }

    @Override
    public String getLabel() {
        //support i18n
        return AppPluginUtil.getMessage("org.joget.AmazonS3DatalistBinder.pluginLabel", getClassName(),
MESSAGE_PATH);
    }
    @Override
    public String getDescription() {
        //support i18n
        return AppPluginUtil.getMessage("org.joget.AmazonS3DatalistBinder.pluginDesc", getClassName(),
MESSAGE_PATH);
    }

    @Override
    public String getPropertyOptions() {
        return AppUtil.readPluginResource(getClass().getName(), "/properties/amazonS3DatalistBinder.json",
null, true, MESSAGE_PATH);
    }

}
```

UIgetPropertyOptions "/properties/amazonS3DatalistBinder.json" "amazon_s3_datalist_binder / src / main" "resources / properties" "properties" "amazonS3DatalistBinder.json"

"@@ message.key @@" i18n AJAXAmazon S3

```
[{
    title : '@@AmazonS3DatalistBinder.config@@',
    properties : [{
        name : 'folder',
        label : '@@AmazonS3DatalistBinder.folder@@',
        type : 'textfield'
    }],
    validators : [{
        type : 'AJAX',
        url : '[CONTEXT_PATH]/web/json/app[APP_PATH]/plugin/org.joget.AmazonS3DatalistBinder/service?
action=validate'
    }]
}]
```

awsS3.properties wflow

getColumnsgetPrimaryKeyColumnNamegetDatagetDataTotalRowCount

```
protected static AmazonS3 s3;
protected static Properties properties;
protected static DataListColumn[] columns;
protected List<Map<String, Object>> cacheData;

protected static String getPropertiesPath() {
    return SetupManager.getBaseSharedDirectory() + "awsS3.properties";
}

public static AmazonS3 getClient() throws Exception {
    if (s3 == null) {
        FileInputStream fis = null;
        try {
            properties = new Properties();
            fis = new FileInputStream(new File(getPropertiesPath()));
            properties.load(fis);

            BasicAWSCredentials awsCreds = new BasicAWSCredentials(properties.getProperty("access_key_id"),
properties.getProperty("secret_access_key"));
            s3 = new AmazonS3Client(awsCreds);


            Region region = Region.getRegion(Regions.fromName(properties.getProperty("region")));
            s3.setRegion(region);

            if (!s3.doesBucketExist(properties.getProperty("bucket"))) {
                Bucket bucket = s3.createBucket(properties.getProperty("bucket"));
                if (bucket == null) {
                    throw new RuntimeException(AppPluginUtil.getMessage("AmazonS3DatalistBinder.
bucketFilToCreate", AmazonS3DatalistBinder.class.getName(), MESSAGE_PATH));
                }
            }
        } catch (Exception e) {
            LogUtil.error(AmazonS3DatalistBinder.class.getName(), e, "");
            s3 = null;

            if (e instanceof FileNotFoundException) {
                throw new RuntimeException(AppPluginUtil.getMessage("AmazonS3DatalistBinder.
configureationFileIsMissing", AmazonS3DatalistBinder.class.getName(), MESSAGE_PATH));
            } else {
                throw e;
            }
        } finally {
            try {
                if (fis != null) {
                    fis.close();
                }
            } catch (Exception e) {}
        }
    }
    return s3;
}

protected List<Map<String, Object>> getCacheData() {
    if (cacheData == null) {
        cacheData = new ArrayList<Map<String, Object>>();
        try {
            AmazonS3 client = getClient();
            String prefix = getPropertyString("folder");
            if (prefix.isEmpty()) {
                prefix = null;
            }
            ObjectListing listing = client.listObjects(properties.getProperty("bucket"), prefix);
            boolean cont;
            do {
                cont = false;

                List<S3ObjectSummary> summaries = listing.getObjectSummaries();
```

```
                for (S3ObjectSummary s : summaries) {
                    Map<String, Object> obj = new HashMap<String, Object>();
                    String key = s.getKey();
                    int pos = key.lastIndexOf("/");
                    obj.put("key", key);
                    obj.put("path", (pos > 0)?(key.substring(0, pos-1)):"");
                    obj.put("filename", (pos > 0)?(key.substring(pos+1)):key);
                    obj.put("owner", s.getOwner().getDisplayName());
                    obj.put("md5", s.getETag());
                    obj.put("size", s.getSize());
                    obj.put("storageClass", s.getStorageClass());
                    obj.put("lastModified", s.getLastModified());

                    cacheData.add(obj);
                }

                if (listing.isTruncated()) {
                    cont = true;
                    listing = client.listNextBatchOfObjects(listing);
                }
            } while (cont);
        } catch (Exception e) {}
    }
    return cacheData;
}

public DataListColumn[] getColumns() {
    if (columns == null) {
        Collection<DataListColumn> list = new ArrayList<DataListColumn>();
        list.add(new DataListColumn("key", AppPluginUtil.getMessage("AmazonS3DatalistBinder.key",
getClassName(), MESSAGE_PATH), true));
        list.add(new DataListColumn("path", AppPluginUtil.getMessage("AmazonS3DatalistBinder.path",
getClassName(), MESSAGE_PATH), true));
        list.add(new DataListColumn("filename", AppPluginUtil.getMessage("AmazonS3DatalistBinder.filename",
getClassName(), MESSAGE_PATH), true));
        list.add(new DataListColumn("owner", AppPluginUtil.getMessage("AmazonS3DatalistBinder.owner",
getClassName(), MESSAGE_PATH), true));
        list.add(new DataListColumn("md5", AppPluginUtil.getMessage("AmazonS3DatalistBinder.md5",
getClassName(), MESSAGE_PATH), false));
        list.add(new DataListColumn("size", AppPluginUtil.getMessage("AmazonS3DatalistBinder.size",
getClassName(), MESSAGE_PATH), true));
        list.add(new DataListColumn("storageClass", AppPluginUtil.getMessage("AmazonS3DatalistBinder.
storageClass", getClassName(), MESSAGE_PATH), true));
        list.add(new DataListColumn("lastModified", AppPluginUtil.getMessage("AmazonS3DatalistBinder.
lastModified", getClassName(), MESSAGE_PATH), true));

        columns = list.toArray(new DataListColumn[]{});
    }
    return columns;
}

public String getPrimaryKeyColumnName() {
    return "key";
}

public DataListCollection getData(DataList dataList, Map properties, DataListFilterQueryObject[]
filterQueryObjects, String sort, Boolean desc, Integer start, Integer rows) {
    //TODO: handle filterQueryObjects

    List list = PagingUtils.sortAndPage(getCacheData(), sort, desc, start, rows);
    DataListCollection data = new DataListCollection();
    data.addAll(list);

    return data;
}
public int getDataTotalRowCount(DataList dataList, Map properties, DataListFilterQueryObject[]
filterQueryObjects) {
    //TODO: handle filterQueryObjects

    return getCacheData().size();
}
```

AJAXAmazon S3webServiceAPI

```
 public void webService(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
 boolean isAdmin = WorkflowUtil.isCurrentUserInRole(WorkflowUserManager.ROLE_ADMIN);
 if (!isAdmin) {
 response.sendError(HttpServletResponse.SC_UNAUTHORIZED);
 return;
 }

 String action = request.getParameter("action");
 if ("validate".equals(action)) {
 String message = "";
 boolean success = true;
 try {
 AmazonS3DatalistBinder.getClient();
 } catch (Exception e) {
 LogUtil.error(this.getClassName(), e, "");
 success = false;
 message = StringUtil.escapeString(e.getMessage(), StringUtil.TYPE_JAVASCIPT, null);
 }
 try {
 JSONObject jsonObject = new JSONObject();
 jsonObject.accumulate("status", (success?"success":"fail"));
 JSONArray messageArr = new JSONArray();
 messageArr.put(message);
 jsonObject.put("message", messageArr);
 jsonObject.write(response.getWriter());
 } catch (Exception e) {
 //ignore
 }
 } else {
 response.setStatus(HttpServletResponse.SC_NO_CONTENT);
 }
}
```

c.

POM“jsp-api”“aws-java-sdk-s3”

```
<!-- Change plugin specific dependencies here -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.0</version>
</dependency>
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-s3</artifactId>
    <version>1.10.56</version>
</dependency>
<!-- End change plugin specific dependencies here -->
```

d

getLabelgetDescriptioni18ni18n

“amazon_s3_datalist_binder / src / main”“resources / message”“AmazonS3DatalistBinder.properties”

```
org.joget.AmazonS3DatalistBinder.pluginLabel=Amazon S3 Datalist Binder
org.joget.AmazonS3DatalistBinder.pluginDesc=Used to retrieve the available files in Amazon S3.
AmazonS3DatalistBinder.config=Configure Amazon S3 Datalist Binder
AmazonS3DatalistBinder.configureationFileIsMissing=AWS S3 configuration file is missing.
AmazonS3DatalistBinder.bucketFilToCreate=AWS Bucket fail to create.
AmazonS3DatalistBinder.key=Key
AmazonS3DatalistBinder.path=Path
AmazonS3DatalistBinder.filename=File Name
AmazonS3DatalistBinder.owner=Owner
AmazonS3DatalistBinder.md5=MD5 Hash
AmazonS3DatalistBinder.size=Size
AmazonS3DatalistBinder.storageClass=Storage Class
AmazonS3DatalistBinder.lastModified=Last Modified
AmazonS3DatalistBinder.folder=Folder
```

e. Felix

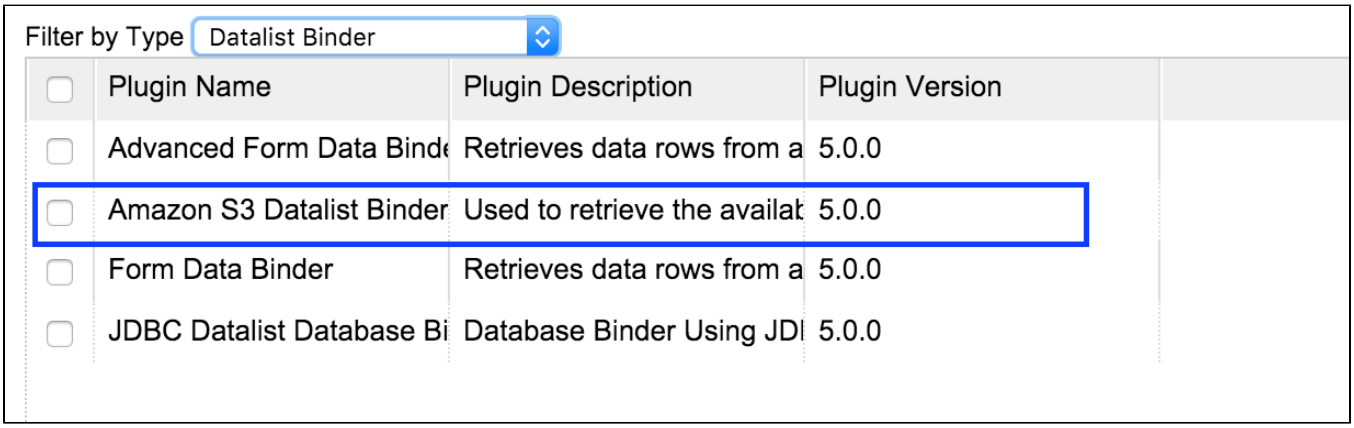
ActivatorFelix

```
 public void start(BundleContext context) {
 registrationList = new ArrayList<ServiceRegistration>();
 //Register plugin here
 registrationList.add(context.registerService(AmazonS3DatalistBinder.class.getName(), new
AmazonS3DatalistBinder(), null));
}
```

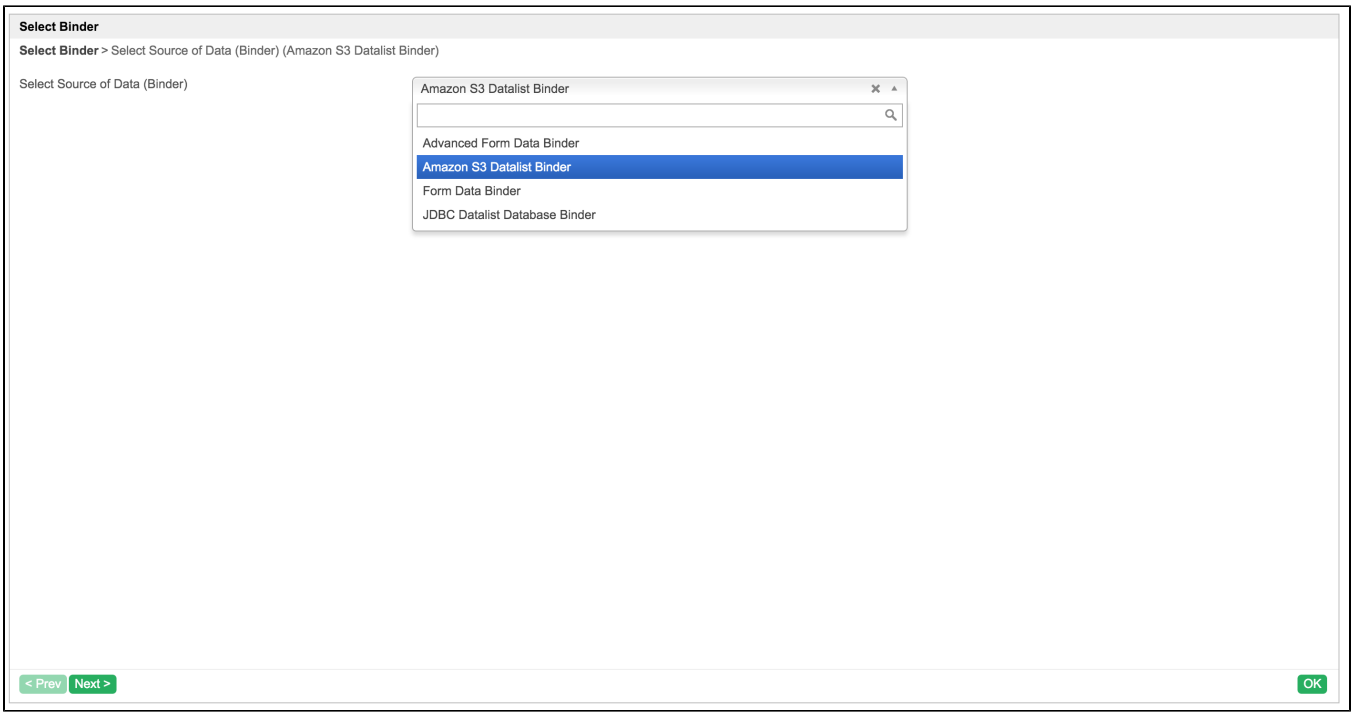
f.

"amazon_s3_datalist_binder / target" "amazon_s3_datalist_binder-5.0.0.jar"

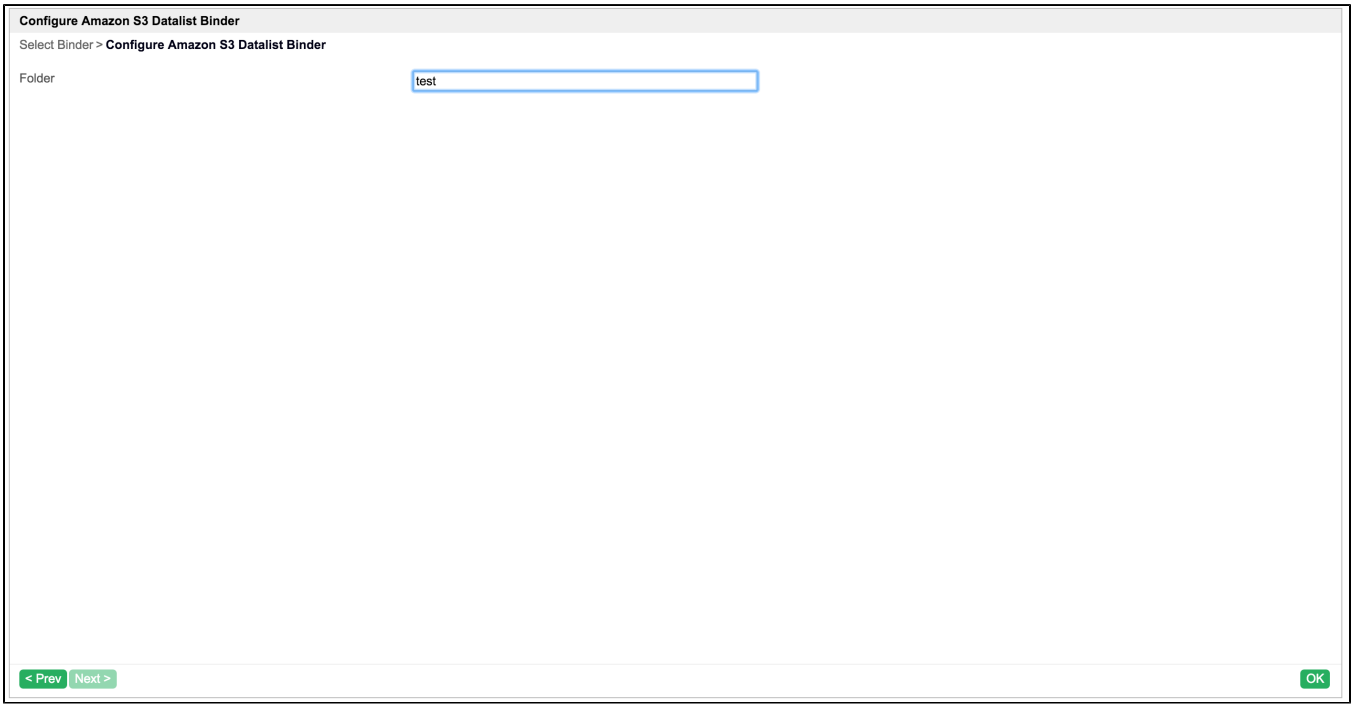
jar jar



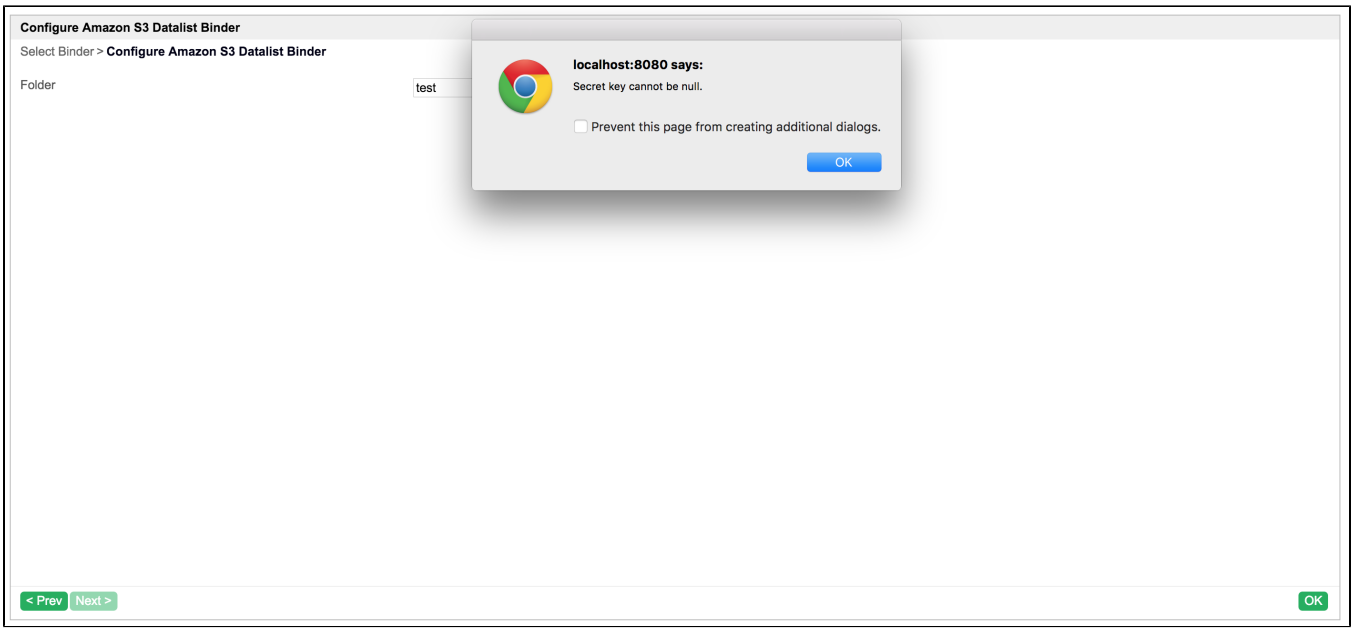
Amazon S3 Datalist Binder

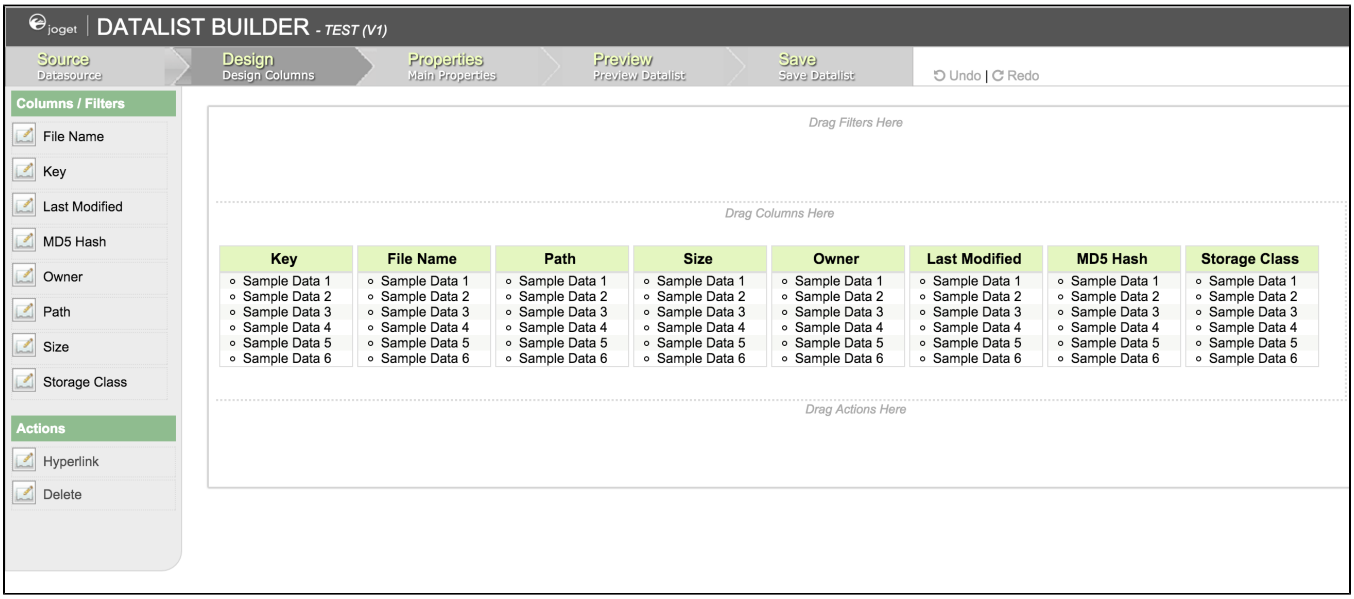


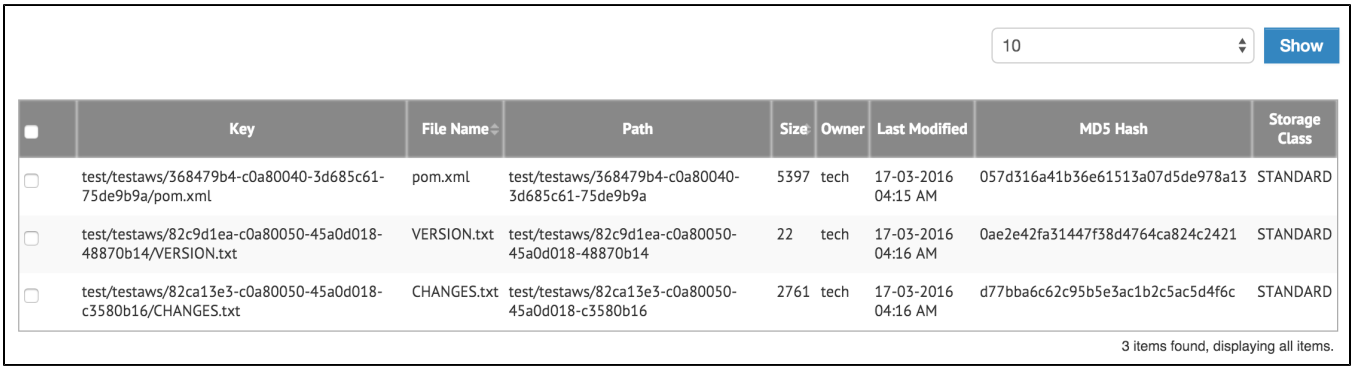
Amazon S3 Datalist



awsS3.properties







	Key	File Name	Path	Size	Owner	Last Modified	MD5 Hash	Storage Class
	test/testaws/368479b4-c0a80040-3d685c61-75de9b9a/pom.xml	pom.xml	test/testaws/368479b4-c0a80040-3d685c61-75de9b9a	5397	tech	17-03-2016 04:15 AM	057d316a41b36e61513a07d5de978a13	STANDARD
	test/testaws/82c9d1ea-c0a80050-45a0d018-48870b14/VERSION.txt	VERSION.txt	test/testaws/82c9d1ea-c0a80050-45a0d018-48870b14	22	tech	17-03-2016 04:16 AM	0ae2e42fa31447f38d4764ca824c2421	STANDARD
	test/testaws/82ca13e3-c0a80050-45a0d018-c3580b16/CHANGES.txt	CHANGES.txt	test/testaws/82ca13e3-c0a80050-45a0d018-c3580b16	2761	tech	17-03-2016 04:16 AM	d77bba6c62c95b5e3ac1b2c5ac5d4f6c	STANDARD

3 items found, displaying all items.

8.

amazon_s3_datalist_binder_src.zip