redhat.

Working With Large GWT Apps

Harald Pehl
GWTCon 2015

redhat.

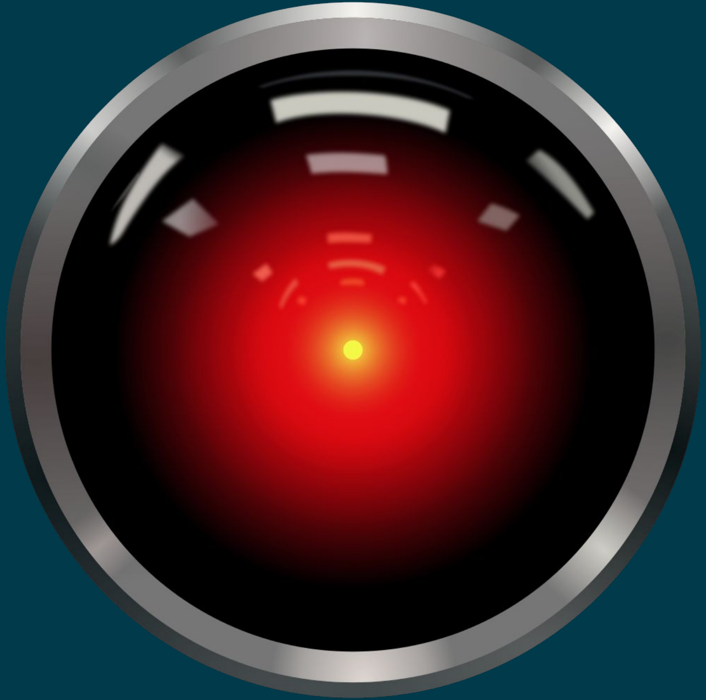
Working With Large GWT Apps

Harald Pehl
GWTCon 2015

About Me

Harald Pehl
Senior Software Engineer - Red Hat
@haraldpehl - http://hpehl.info - http://github.com/hpehl

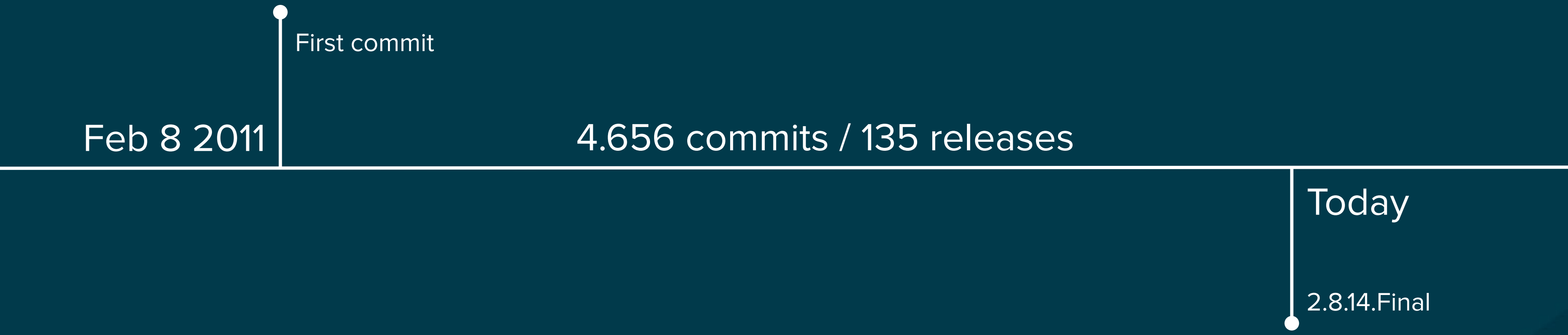
WildFly Management Console



Quick Demo

http://localhost:9990

Timeline



First commit

Feb 8 2011

4.656 commits / 135 releases

Today

2.8.14.Final

Codebase

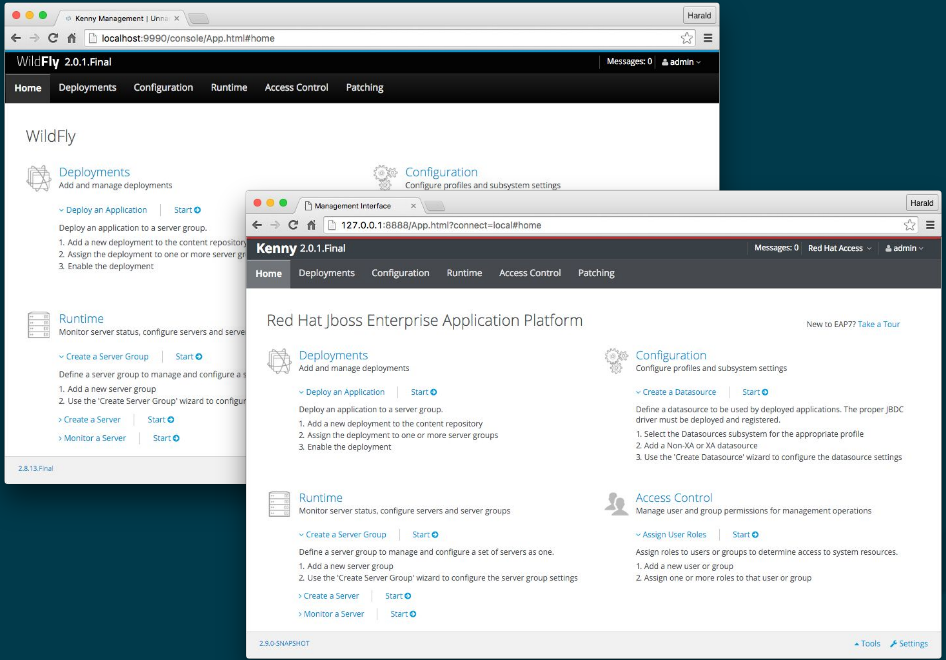
Language	Files	Blank	Comment	Code
Java	1570	32450	29950	124772
CSS	18	1327	291	14804
XML	99	184	179	3963
JavaScript	45	616	787	2905
Less	10	557	95	2522
Maven	15	126	201	1895
HTML	74	76	21	809
Python	1	41	20	204
Shell	7	24	7	45
Sum	**1839**	**35401**	**31551**	**151999**

Community *and* Product Version

Different Look & Feel

Slightly different feature set

Up to ***seven*** years support & maintenance for the product version



WildFly

Deployments

Add and manage deployments

Deploy an application to a server group.

1. Add a new deployment to the content repositor
2. Assign the deployment to one or more server gr
3. Enable the deployment

Configuration

Configure profiles and subsystem settings

Runtime

Monitor server status, configure servers and serve

Define a server group to manage and configure a s

1. Add a new server group
2. Use the 'Create Server Group' wizard to configur

Red Hat Jboss Enterprise Application Platform

Deployments

Add and manage deployments

Deploy an application to a server group.

1. Add a new deployment to the content repository
2. Assign the deployment to one or more server groups
3. Enable the deployment

Configuration

Configure profiles and subsystem settings

Define a datasource to be used by deployed applications. The proper JBDC driver must be deployed and registered.

1. Select the Datasources subsystem for the appropriate profile
2. Add a Non-XA or XA datasource
3. Use the 'Create Datasource' wizard to configure the datasource settings

Runtime

Monitor server status, configure servers and server groups

Define a server group to manage and configure a set of servers as one.

1. Add a new server group
2. Use the 'Create Server Group' wizard to configure the server group settings

Access Control

Manage user and group permissions for management operations

Assign roles to users or groups to determine access to system resources.

1. Add a new user or group
2. Assign one or more roles to that user or group

Dependencies

What works for us

```
com.gwtplatform::gwtp-*
com.google.gwt.inject::gin
com.google.guava::guava-gwt
com.allen-sauer.gwt.log::gwt-log
```

Architecture

Maven Module == GWT Module
Annotation Processors
Code Splitting
Stick to GWT

MVP
Async Flow
Circuit
Composition over Inheritance

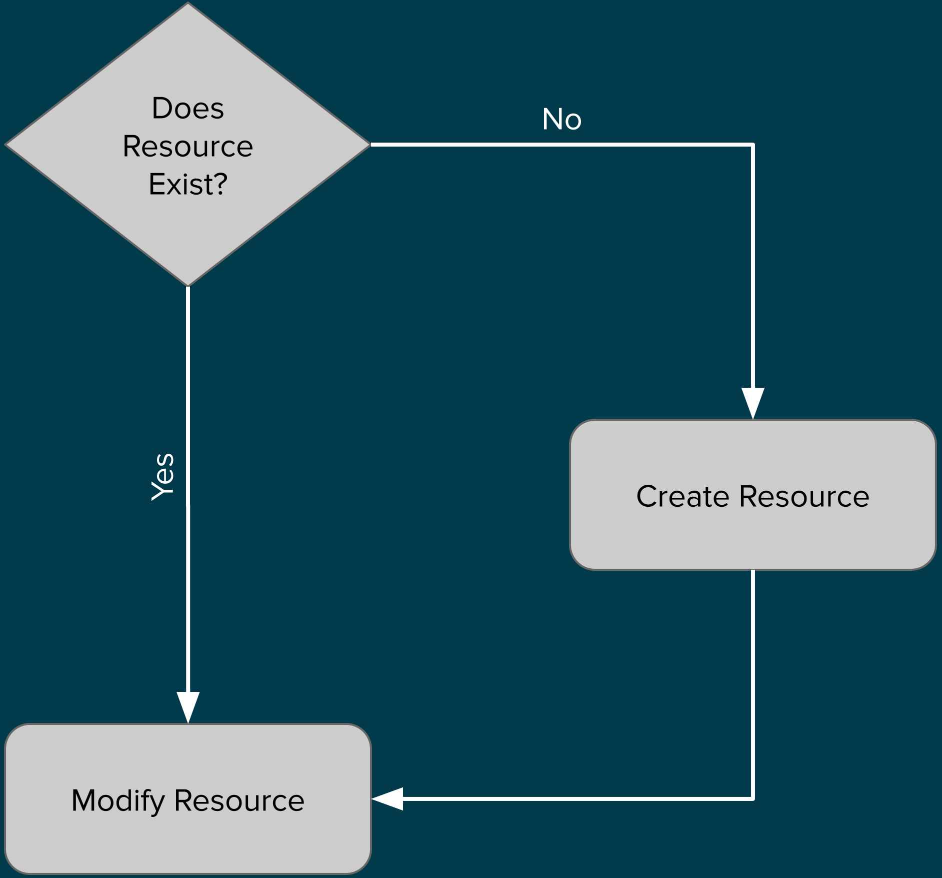
Spin Offs

Async Flow (https://github.com/hal/core/tree/master/flow)
Circuit (https://github.com/hal/circuit)

Async Flow

Escape from “callback hell“

Execute multiple asynchronous calls in a coordinated manner


Does Resource Exist?
No
Yes
Create Resource
Modify Resource

```java
class Lookup implements Function<FunctionContext> {

    public void execute(final Control<FunctionContext> control) {
        Operation op = new Operation.Builder(READ_RESOURCE_OPERATION, address).build();
        dispatcher.execute(new DMRAction(op), new AsyncCallback<DMRResponse>() {

            public void onFailure(final Throwable throwable) {
                control.getContext().push(404);
                control.proceed();
            }

            public void onSuccess(final DMRResponse dmrResponse) {
                control.getContext().push(200);
                control.proceed();
            }
        });
    }
}
```

```java
class Add implements Function<FunctionContext> {

    public void execute(final Control<FunctionContext> control) {
        int status = control.getContext().pop();
        if (status == 404) {
            dispatcher.execute(new DMRAction(new Operation.Builder(ADD, address).build()), new FunctionCallback(control));
        } else {
            control.proceed();
        }
    }
}

class Modify implements Function<FunctionContext> {

    public void execute(final Control<FunctionContext> control) {
        ModelNode op = new ModelNodeAdapter().fromChangeSet(address, changedValues);
        dispatcher.execute(new DMRAction(op), new FunctionCallback(control));
    }
}
```

```
new Async<FunctionContext>(Footer.PROGRESS_ELEMENT).waterfall(
        new FunctionContext(),
        new Outcome<FunctionContext>() {
                public void onFailure(final FunctionContext context) { … }
                public void onSuccess(final FunctionContext context) { … }
        },
        new Lookup(), new Add(), new Modify());
```

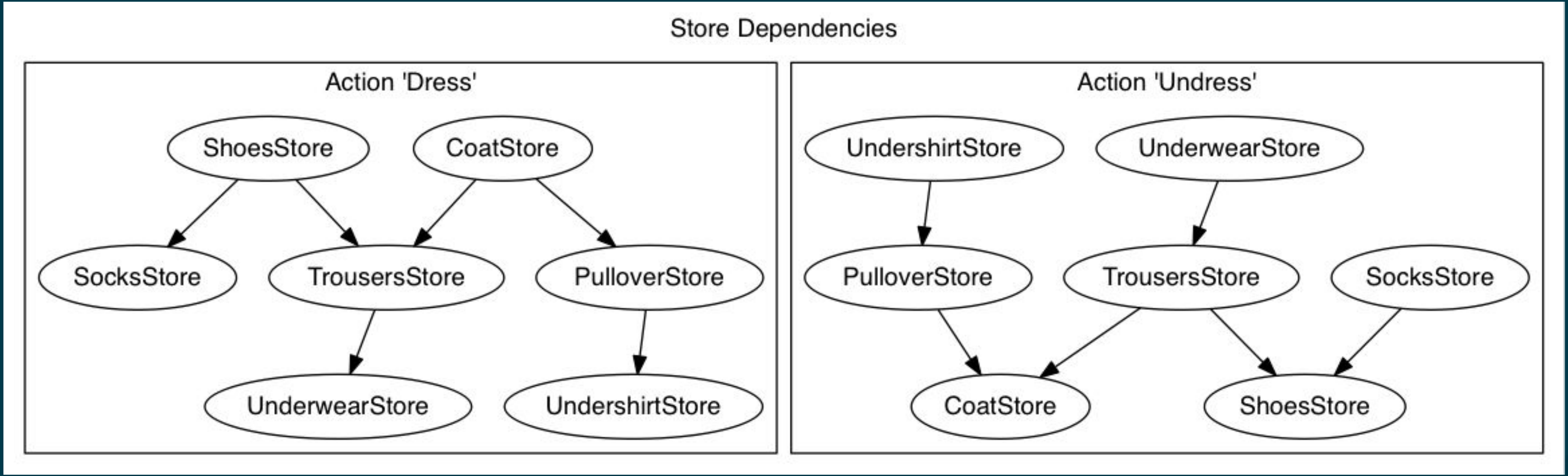
Circuit

Inspired by Flux (part of React.js framework)

Unidirectional data flow

Based on

- a single dispatcher
- stores with optional dependencies
- actions


Store Dependencies
Action 'Dress'
ShoesStore
CoatStore
SocksStore
TrousersStore
PulloverStore
UnderwearStore
UndershirtStore
Action 'Undress'
UndershirtStore
UnderwearStore
PulloverStore
TrousersStore
SocksStore
CoatStore
ShoesStore

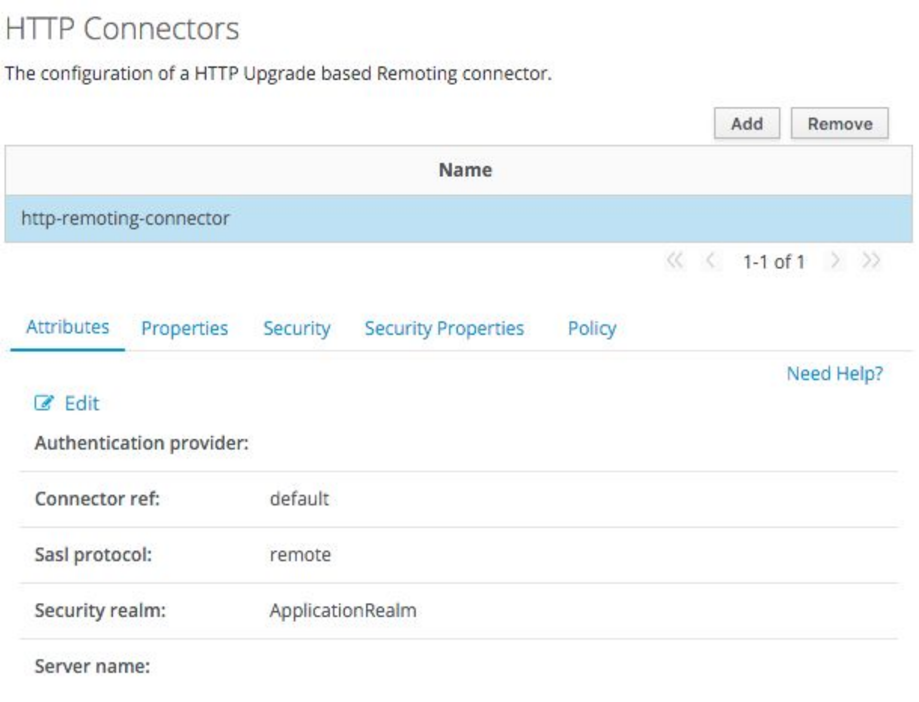
```java
@Store
public class TrousersStore {

   @Process(actionType = Dress.class, dependencies = UnderwearStore.class)
   public void dress(Dispatcher.Channel channel) {
       dispatcher.execute(new DMRAction(op), new AsyncCallback<DMRResponse>() {
           public void onFailure(final Throwable throwable) { channel.ack(throwable); }
           public void onSuccess(final DMRResponse dmrResponse) { channel.ack(); }
       });
   }

   @Process(actionType = Undress.class, dependencies = {CoatStore.class, ShoesStore.class})
   public void undress(Dispatcher.Channel channel) { channel.ack(); }
}
```

```java
public class Dress implements Action {}

public class Undress implements Action {}



Dispatcher dispatcher = new DAGDispatcher();
dispatcher.dispatch(new Dress());
```

UI

No UIBinder
Vanilla GWT Widgets
Cell Widgets
Styled with external CSS (PatternFly)
Builder like APIs

```
Widget layout = new MultipleToOneLayout()
        .setHeadline(title)
        .setDescription(resourceDescription.get(DESCRIPTION)))
        .setMasterTools(tools())
        .setMaster(table())
        .addDetail("Attributes", formPanel())
        .addDetail("Properties", propertyEditor().asWidget())
        .addDetail("Security", saslSecurityEditor.asWidget())
        .addDetail("Security Properties", securityProperties.a
        .addDetail("Policy", saslPolicyEditor.asWidget())
        .build();
```



HTTP Connectors

The configuration of a HTTP Upgrade based Remoting connector.

Name
http-remoting-connector

1-1 of 1

Authentication provider:	
Connector ref:	default
Sasl protocol:	remote
Security realm:	ApplicationRealm
Server name:	

Lessons Learned

KISS

Use what’s there

Composition over inheritance

Do not fear code duplication

Factor out only if necessary

redhat.

THANK YOU

@haraldpehl
http://hpehl.info

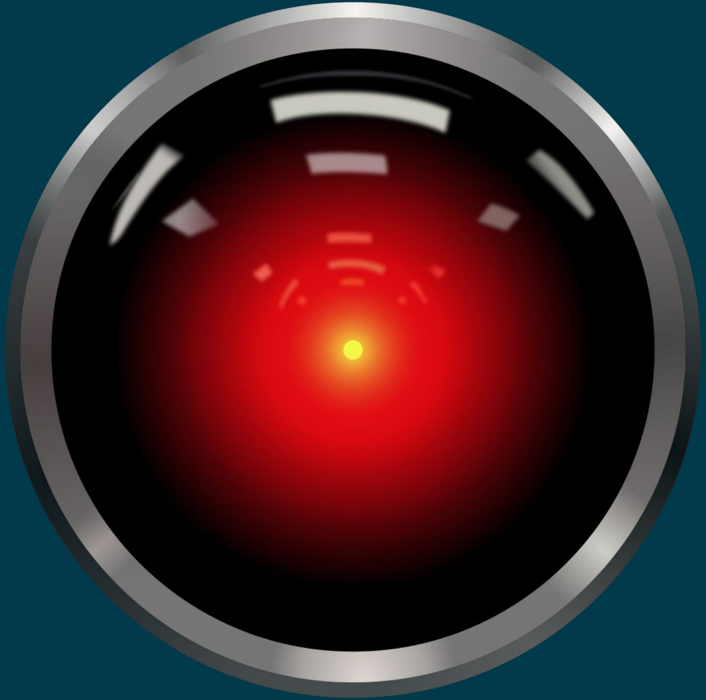
About Me

Harald Pehl

Senior Software Engineer - Red Hat

@haraldpehl - http://hpehl.info - http://github.com/hpehl

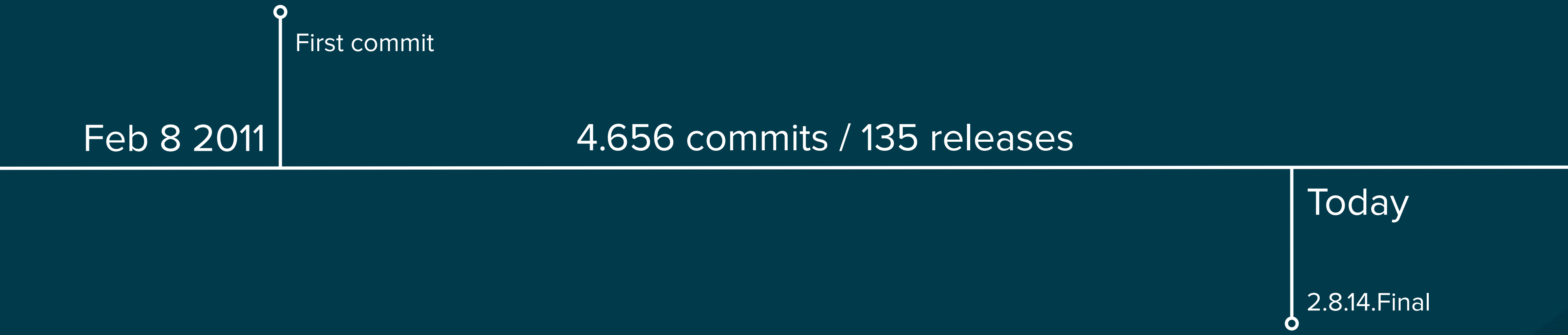
WildFly Management Console



Quick Demo

http://localhost:9990

Timeline



First commit

Feb 8 2011

4.656 commits / 135 releases

Today

2.8.14.Final

Codebase

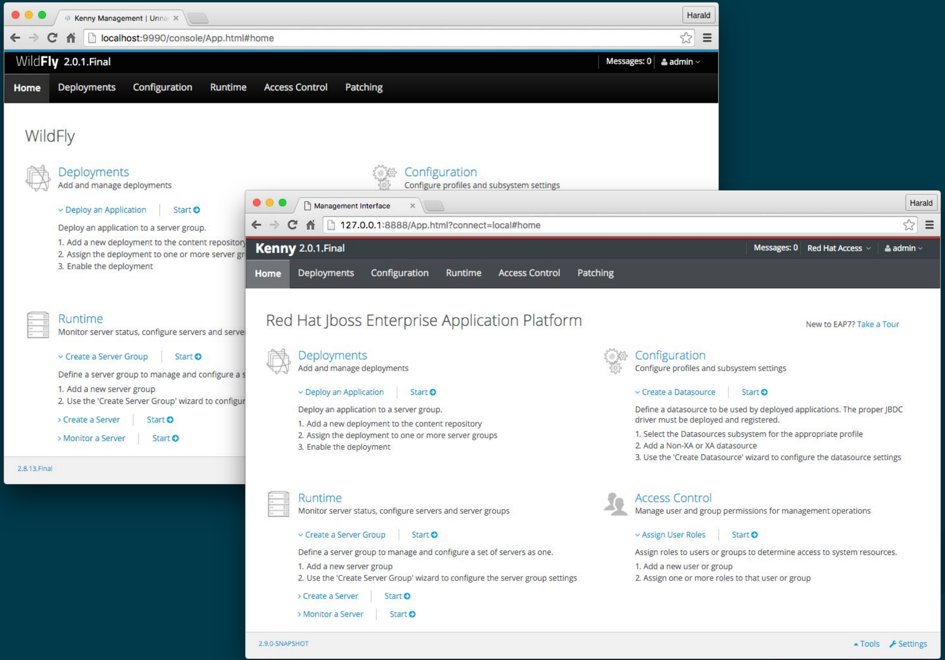
Language	Files	Blank	Comment	Code
Java	1570	32450	29950	124772
CSS	18	1327	291	14804
XML	99	184	179	3963
JavaScript	45	616	787	2905
Less	10	557	95	2522
Maven	15	126	201	1895
HTML	74	76	21	809
Python	1	41	20	204
Shell	7	24	7	45
Sum	**1839**	**35401**	**31551**	**151999**

Community *and* Product Version

Different Look & Feel

Slightly different feature set

Up to ***seven*** years support & maintenance for the product version



WildFly

Deployments
Add and manage deployments

Deploy an Application | Start

Deploy an application to a server group.
1. Add a new deployment to the content repositor
2. Assign the deployment to one or more server gr
3. Enable the deployment

Configuration
Configure profiles and subsystem settings

Runtime
Monitor server status, configure servers and serve

Create a Server Group | Start

Define a server group to manage and configure a s
1. Add a new server group
2. Use the 'Create Server Group' wizard to configur

Create a Server | Start
Monitor a Server | Start

2.8.13.Final

Red Hat Jboss Enterprise Application Platform

New to EAP7? Take a Tour

Deployments
Add and manage deployments

Deploy an Application | Start

Deploy an application to a server group.
1. Add a new deployment to the content repository
2. Assign the deployment to one or more server groups
3. Enable the deployment

Configuration
Configure profiles and subsystem settings

Create a Datasource | Start

Define a datasource to be used by deployed applications. The proper JBDC driver must be deployed and registered.
1. Select the Datasources subsystem for the appropriate profile
2. Add a Non-XA or XA datasource
3. Use the 'Create Datasource' wizard to configure the datasource settings

Runtime
Monitor server status, configure servers and server groups

Create a Server Group | Start

Define a server group to manage and configure a set of servers as one.
1. Add a new server group
2. Use the 'Create Server Group' wizard to configure the server group settings

Create a Server | Start
Monitor a Server | Start

Access Control
Manage user and group permissions for management operations

Assign User Roles | Start

Assign roles to users or groups to determine access to system resources.
1. Add a new user or group
2. Assign one or more roles to that user or group

2.9.0-SNAPSHOT

Dependencies

What works for us

```
com.gwtplatform::gwtp-*
com.google.gwt.inject::gin
com.google.guava::guava-gwt
com.allen-sauer.gwt.log::gwt-log
```

Architecture

Maven Module == GWT Module

Annotation Processors

Code Splitting

Stick to GWT

MVP

Async Flow

Circuit

Composition over Inheritance

Spin Offs

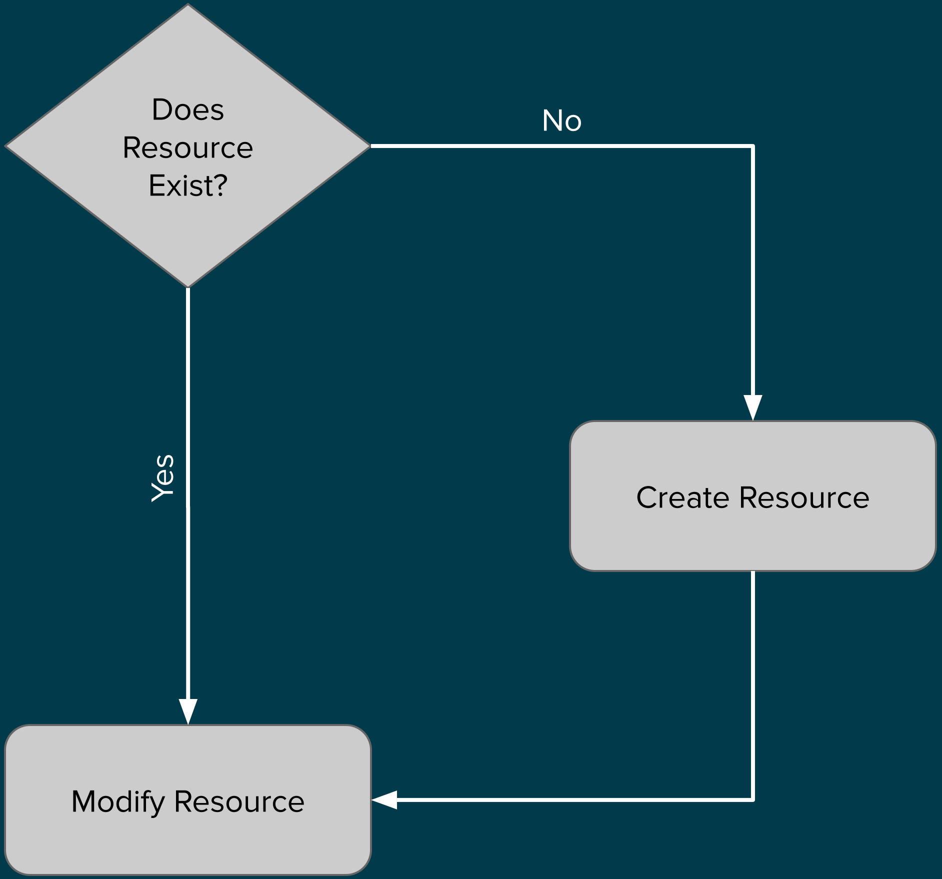
Async Flow (https://github.com/hal/core/tree/master/flow)

Circuit (https://github.com/hal/circuit)

Async Flow

Escape from “callback hell“

Execute multiple asynchronous calls in a coordinated manner



Does Resource Exist?
No
Yes
Create Resource
Modify Resource

```java
class Lookup implements Function<FunctionContext> {

    public void execute(final Control<FunctionContext> control) {
        Operation op = new Operation.Builder(READ_RESOURCE_OPERATION, address).build();
        dispatcher.execute(new DMRAction(op), new AsyncCallback<DMRResponse>() {

            public void onFailure(final Throwable throwable) {
                control.getContext().push(404);
                control.proceed();
            }

            public void onSuccess(final DMRResponse dmrResponse) {
                control.getContext().push(200);
                control.proceed();
            }
        });
    }
}
```

```java
class Add implements Function<FunctionContext> {

    public void execute(final Control<FunctionContext> control) {
        int status = control.getContext().pop();
        if (status == 404) {
            dispatcher.execute(new DMRAction(new Operation.Builder(ADD, address).build()), new FunctionCallback(control));
        } else {
            control.proceed();
        }
    }
}

class Modify implements Function<FunctionContext> {

    public void execute(final Control<FunctionContext> control) {
        ModelNode op = new ModelNodeAdapter().fromChangeSet(address, changedValues);
        dispatcher.execute(new DMRAction(op), new FunctionCallback(control));
    }
}
```

```java
new Async<FunctionContext>(Footer.PROGRESS_ELEMENT).waterfall(
       new FunctionContext(),
       new Outcome<FunctionContext>() {
              public void onFailure(final FunctionContext context) { … }
              public void onSuccess(final FunctionContext context) { … }
       },
       new Lookup(), new Add(), new Modify());
```

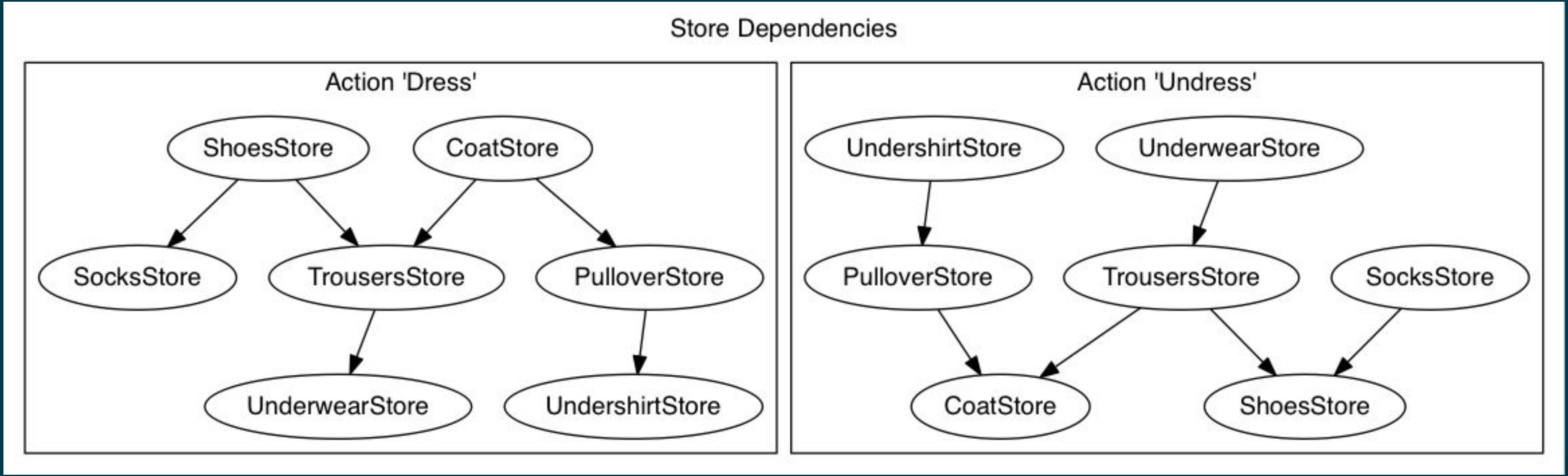
Circuit

Inspired by Flux (part of React.js framework)

Unidirectional data flow

Based on

- a single dispatcher
- stores with optional dependencies
- actions


Store Dependencies
Action 'Dress'
ShoesStore
CoatStore
SocksStore
TrousersStore
PulloverStore
UnderwearStore
UndershirtStore
Action 'Undress'
UndershirtStore
UnderwearStore
PulloverStore
TrousersStore
SocksStore
CoatStore
ShoesStore

```java
@Store
public class TrousersStore {

    @Process(actionType = Dress.class, dependencies = UnderwearStore.class)
    public void dress(Dispatcher.Channel channel) {
        dispatcher.execute(new DMRAction(op), new AsyncCallback<DMRResponse>() {
            public void onFailure(final Throwable throwable) { channel.ack(throwable); }
            public void onSuccess(final DMRResponse dmrResponse) { channel.ack(); }
        });
    }

    @Process(actionType = Undress.class, dependencies = {CoatStore.class, ShoesStore.class})
    public void undress(Dispatcher.Channel channel) { channel.ack(); }
}
```

```java
public class Dress implements Action {}

public class Undress implements Action {}


Dispatcher dispatcher = new DAGDispatcher();
dispatcher.dispatch(new Dress());
```

UI

No UIBinder

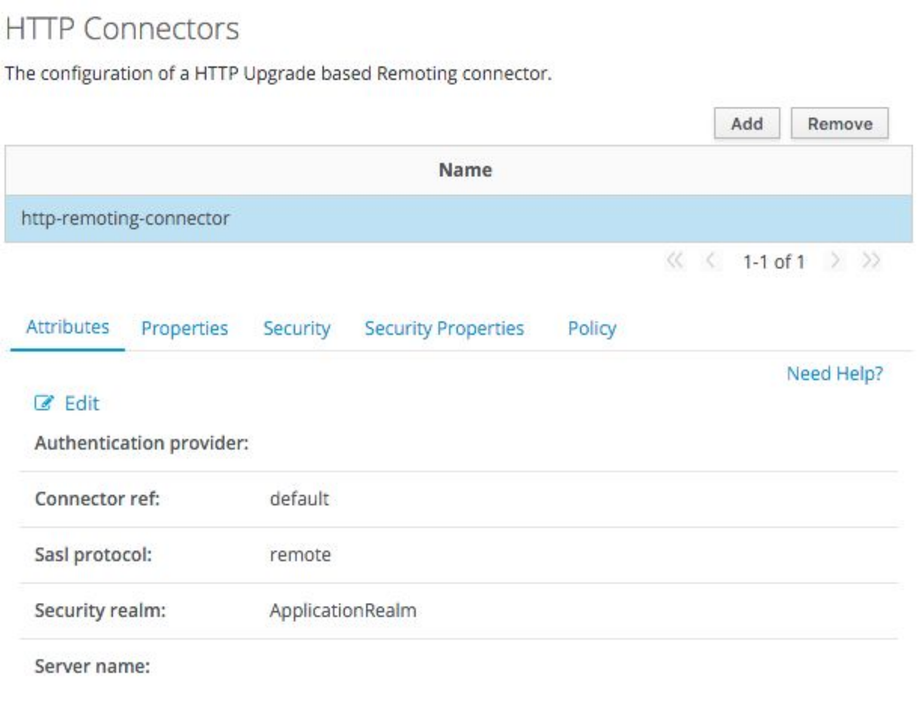
Vanilla GWT Widgets

Cell Widgets

Styled with external CSS (PatternFly)

Builder like APIs

```
Widget layout = new MultipleToOneLayout()
        .setHeadline(title)
        .setDescription(resourceDescription.get(DESCRIPTION)))
        .setMasterTools(tools())
        .setMaster(table())
        .addDetail("Attributes", formPanel())
        .addDetail("Properties", propertyEditor().asWidget())
        .addDetail("Security", saslSecurityEditor.asWidget())
        .addDetail("Security Properties", securityProperties.a
        .addDetail("Policy", saslPolicyEditor.asWidget())
        .build();
```



HTTP Connectors

The configuration of a HTTP Upgrade based Remoting connector.

Name
http-remoting-connector

Authentication provider:	
Connector ref:	default
Sasl protocol:	remote
Security realm:	ApplicationRealm
Server name:	

Lessons Learned

KISS

Use what’s there

Composition over inheritance

Do not fear code duplication

Factor out only if necessary

THANK YOU

@haraldpehl
http://hpehl.info