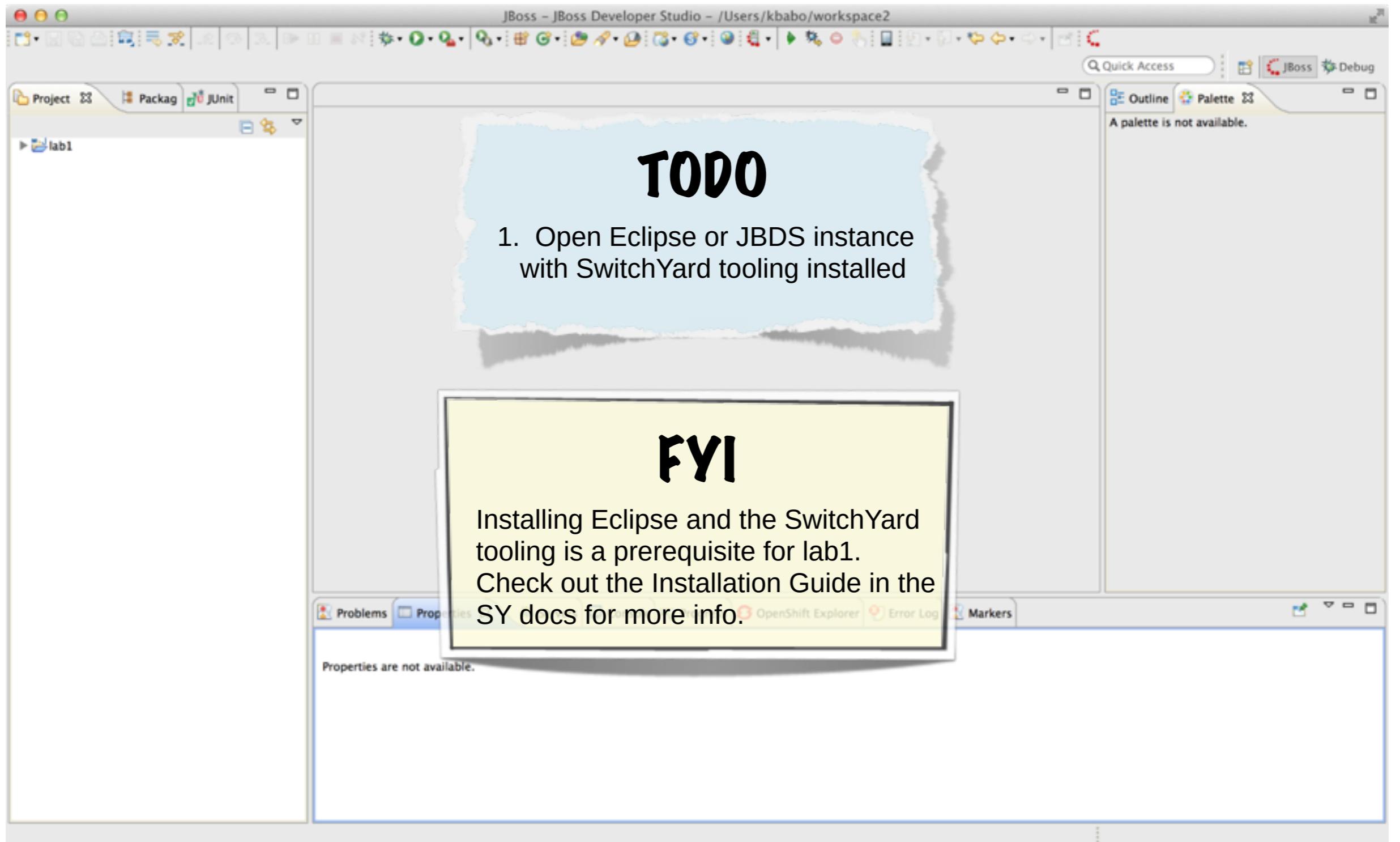


# Lab I

Getting (Getting Started) Started

# Lab Goals

- Walk through an existing SwitchYard application
- Get comfortable with the development environment
- Setup the SOA-P runtime and deploy an application
- Poke around the admin console

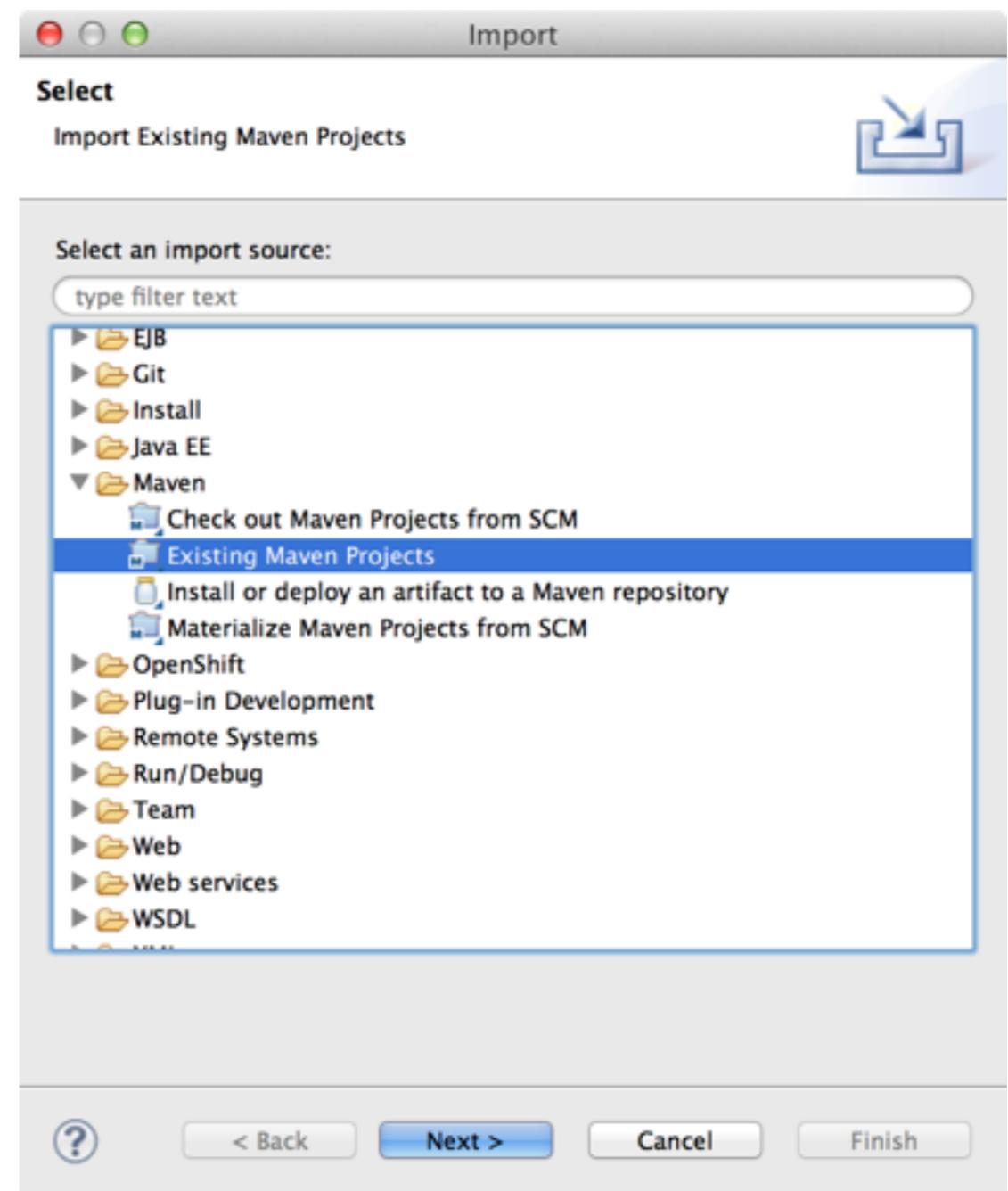


<https://docs.jboss.org/author/display/SWITCHYARD/Installing+Eclipse+Tooling>

# Importing Lab I

## TODO

1. File -> Import ... from the JBDS menu.
2. Select Maven -> Existing Maven Projects
3. Click Next



# Importing Lab 1

## TODO

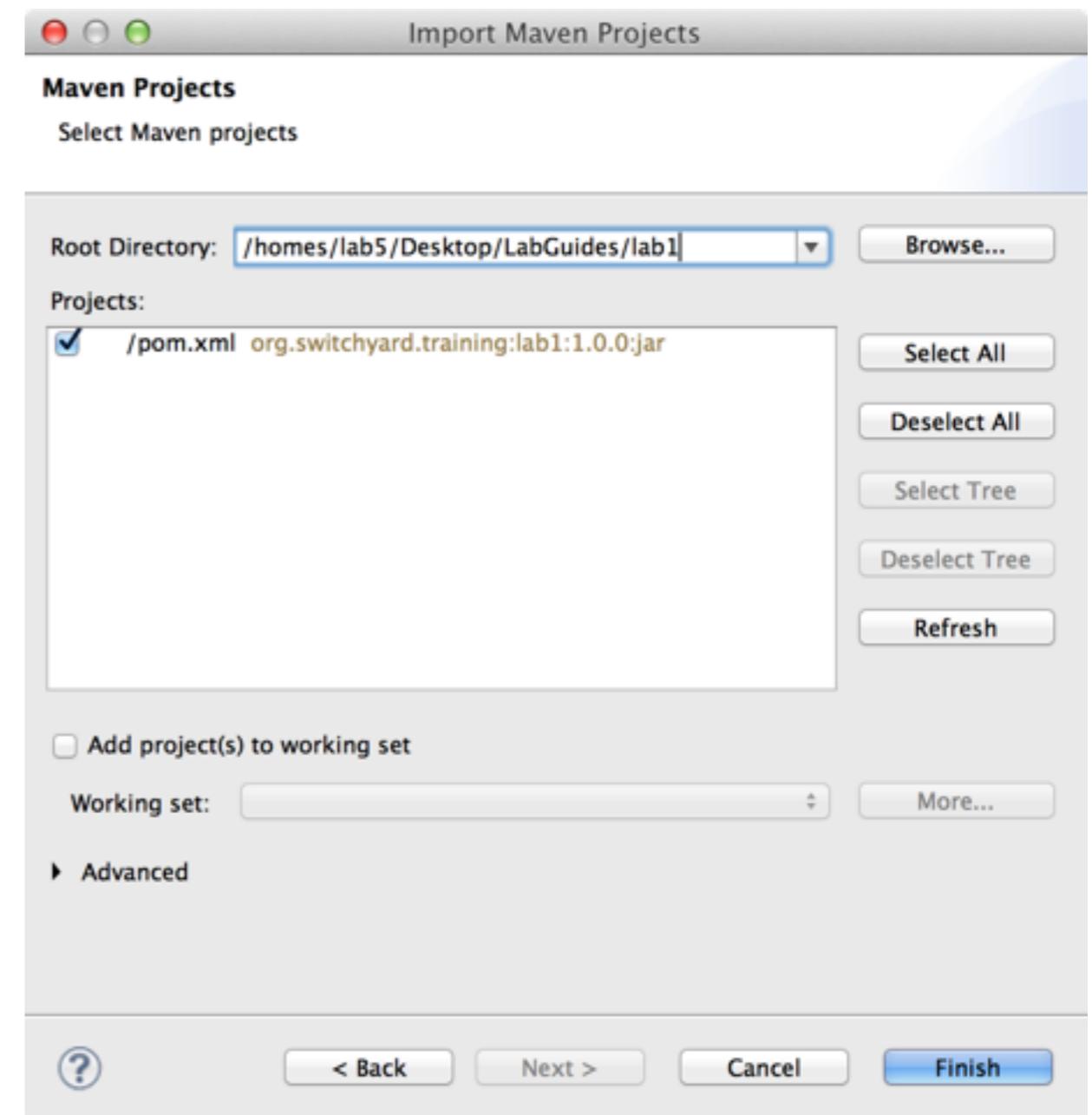
1. Click Browse ... and navigate to the location of lab1. For example:

/home/learning/summit2013/lab1

2. Make sure the pom.xml is checked for:

org.switchyard.training:lab1

3. Click Finish



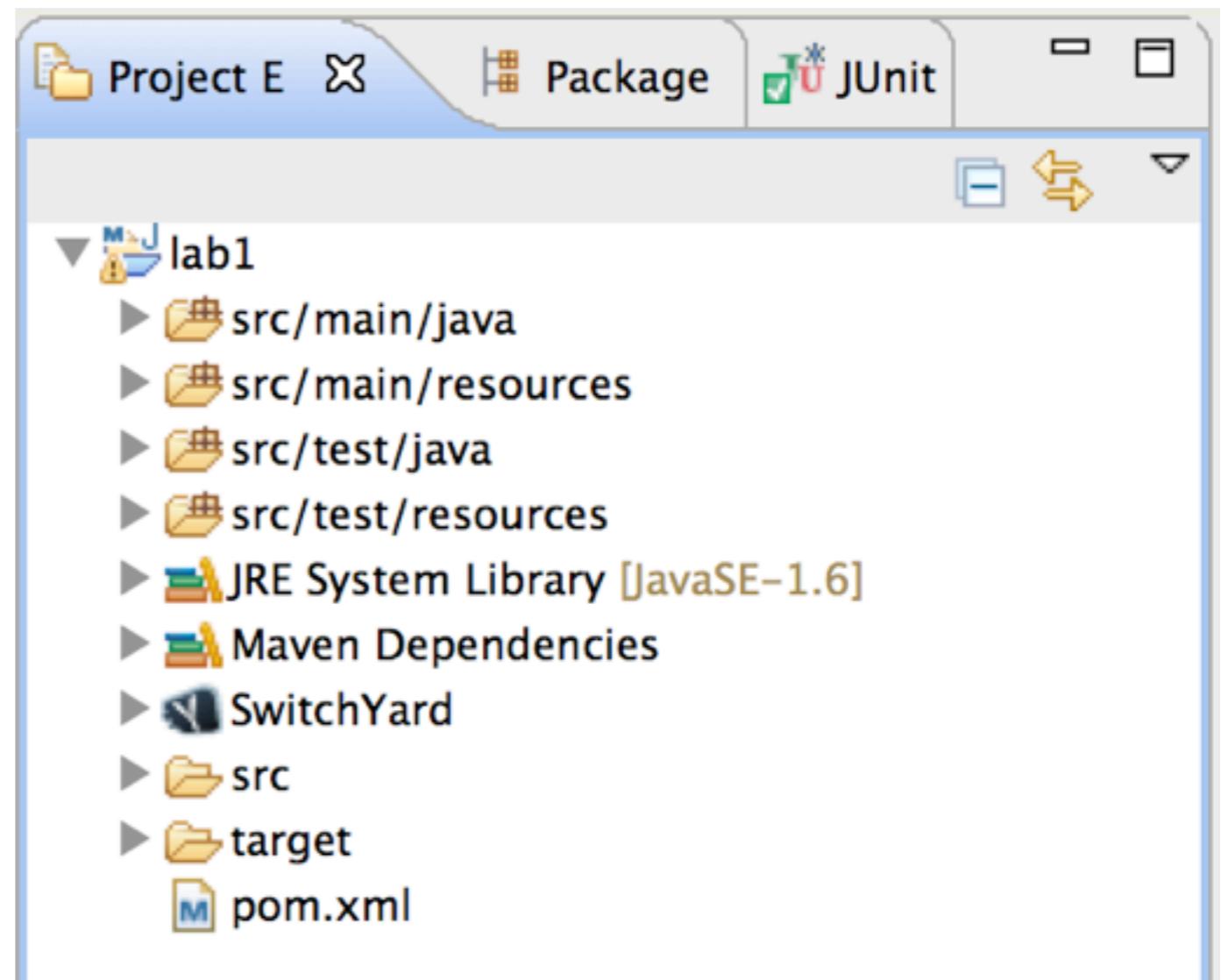
# Application Structure

## TODO

1. Go to the Project Explorer in the left frame and expand the lab1 project node.

## FYI

*The next set of slides will examine each section of the project in detail.*



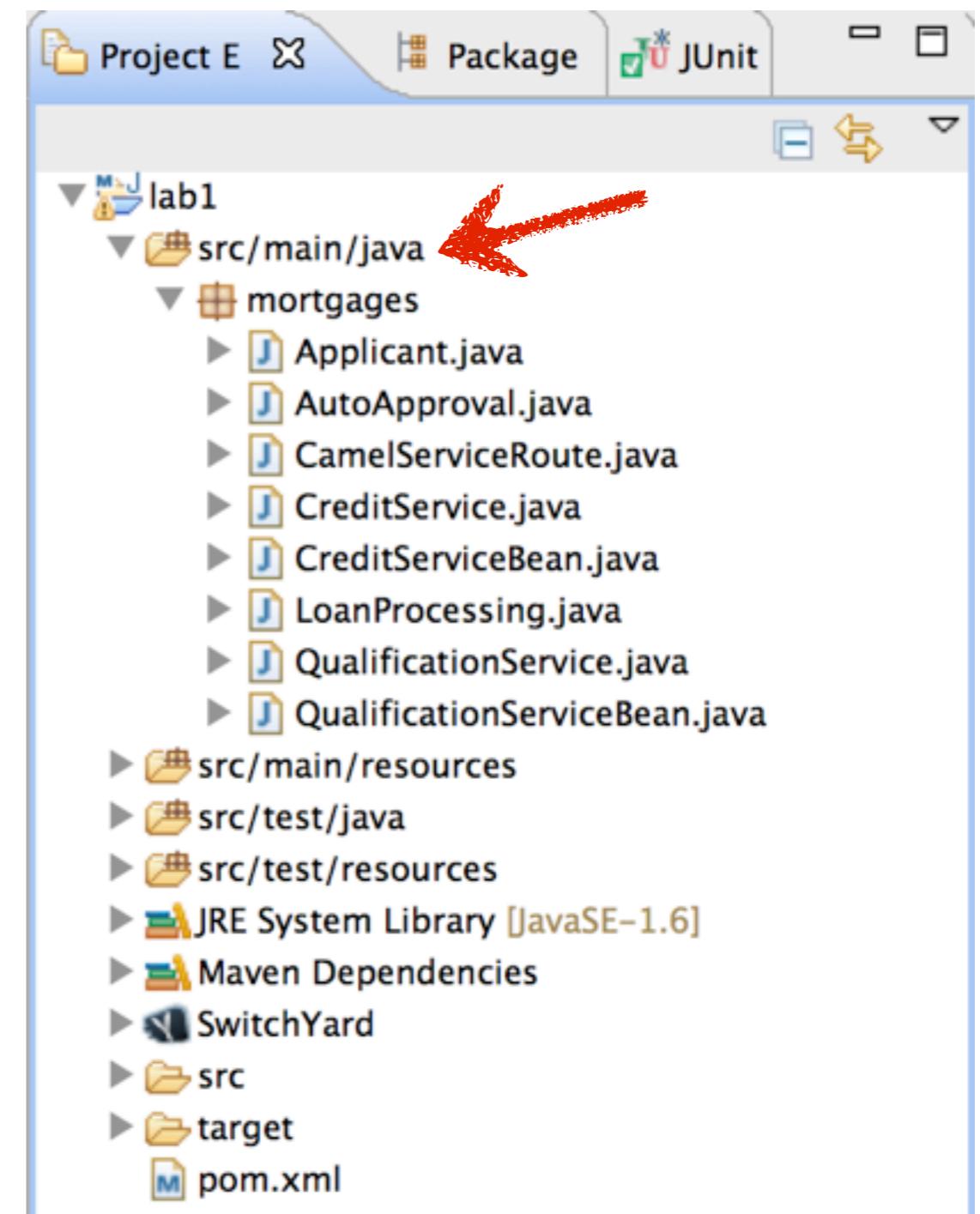
# src/main/java

**FYI**

*This is the home of Java files which provide your application's logic. These files are packaged with your application for deployment.*

*Examples of files you will find here:*

- Java service interfaces
- Camel Java DSL routes
- CDI Beans
- Java Transformers



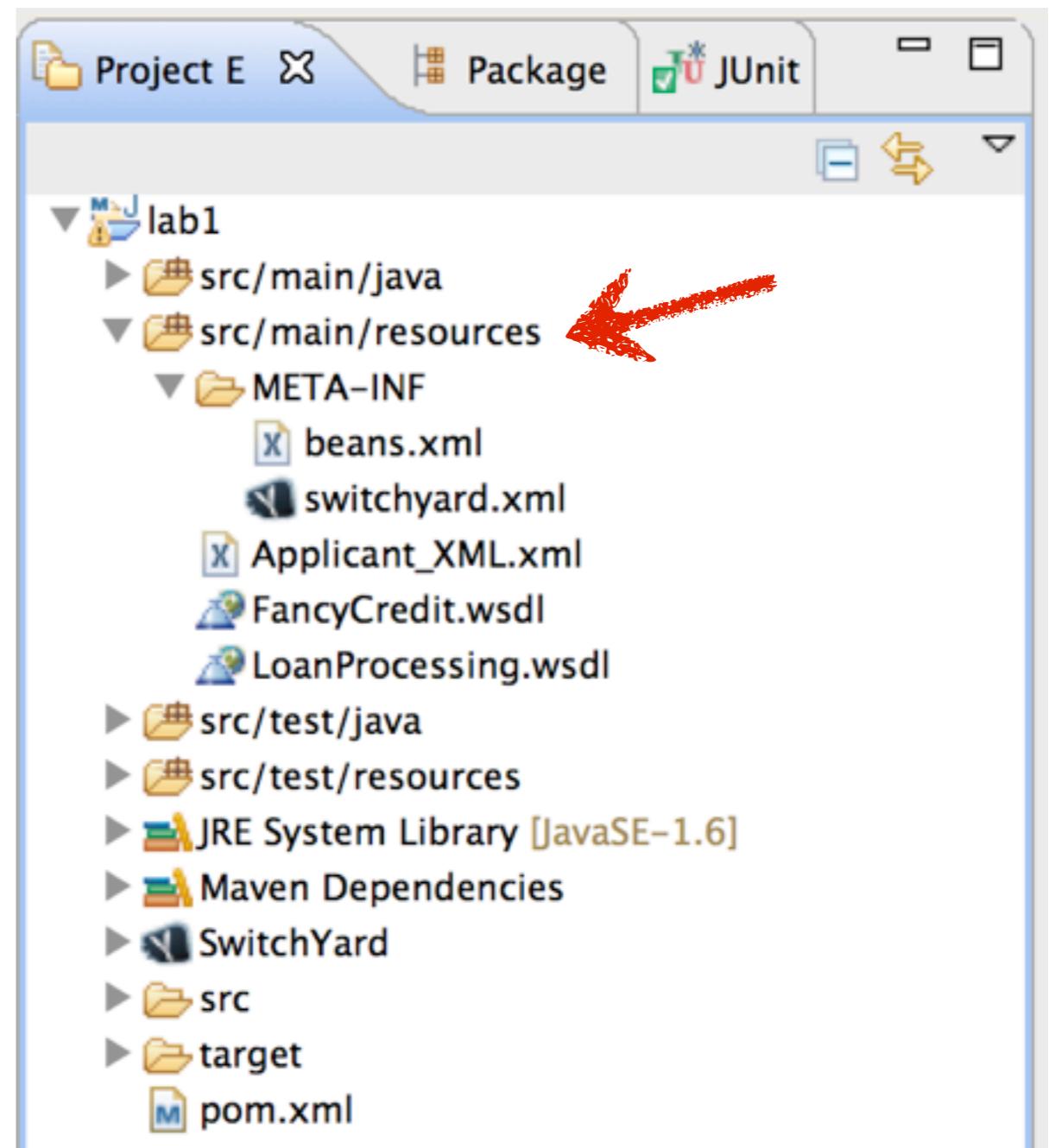
# src/main/resources

**FYI**

*Non-Java resources that will be packaged with your application go here.*

*Examples of files you will find here:*

- *SwitchYard application descriptor (switchyard.xml)*
- *Camel XML route definitions*
- *Transformers (XSLT, Smooks)*
- *Drools rule definitions*
- *Process definitions (BPMN 2, BPEL)*
- *WSDL*
- *Schema*



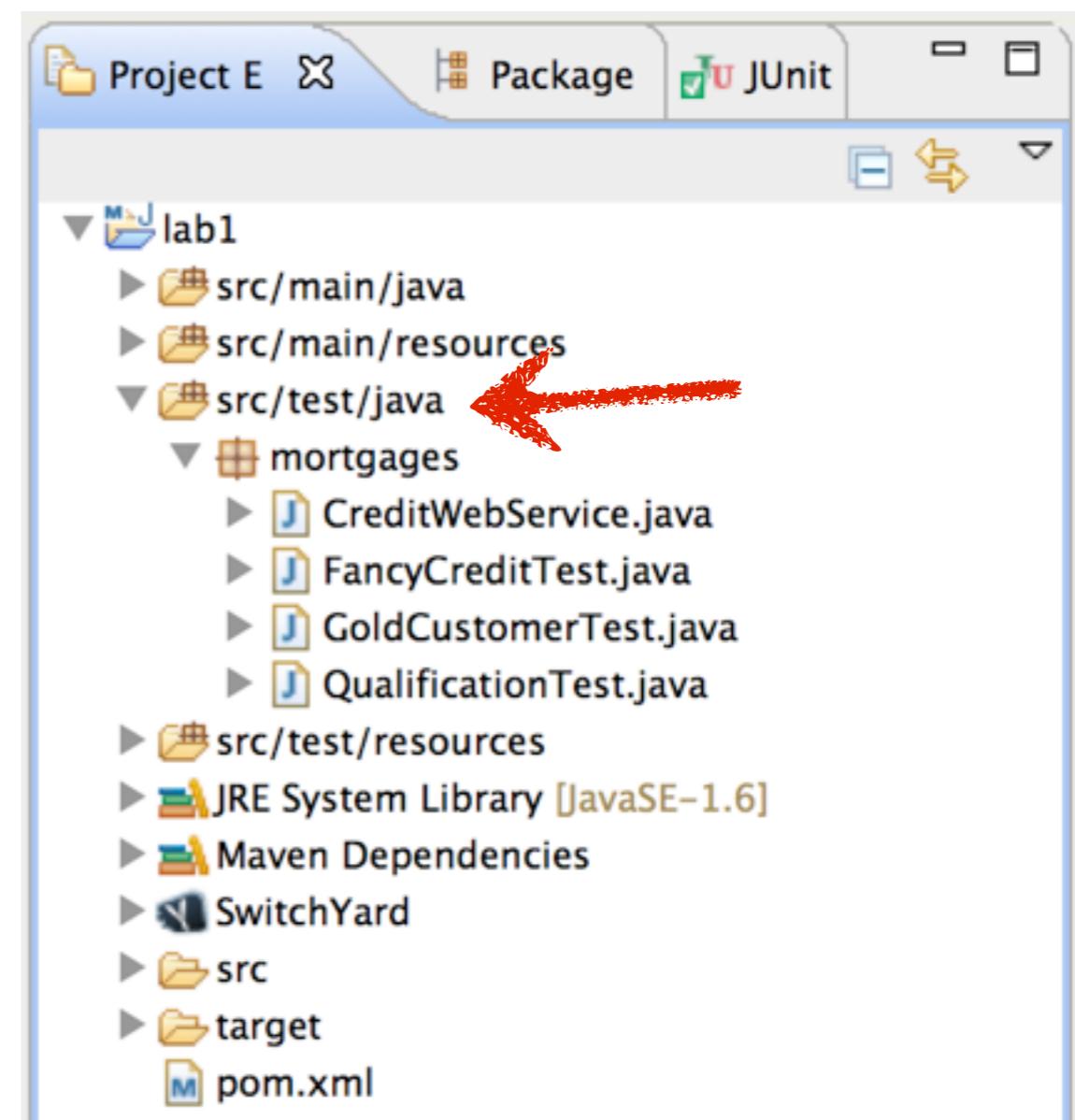
# src/test/java

**FYI**

*All Java classes related to testing your application go in src/test/java.*

*Examples of files you will find here:*

- Service unit tests
- Remote test clients
- Ancillary test classes (e.g. mock web services)



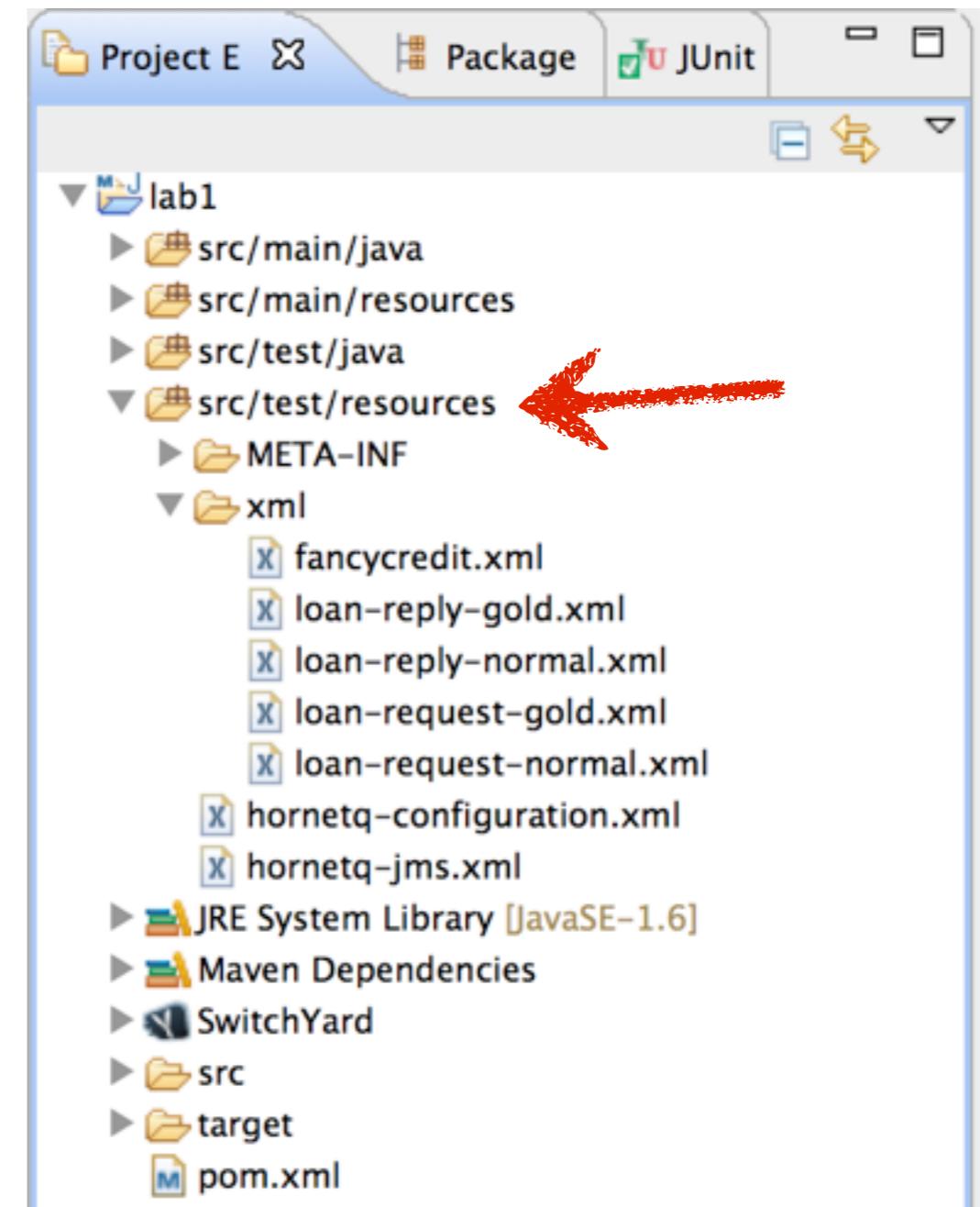
# src/test/resources

**FYI**

*Any resources that are used during test execution, but are not included in the application, belong in src/test/resources.*

*Examples of files you will find here:*

- *Test payloads*
- *Expected results for test assertions*
- *JMS destinations used for testing*
- *log settings used for test execution*



# SwitchYard!!!!

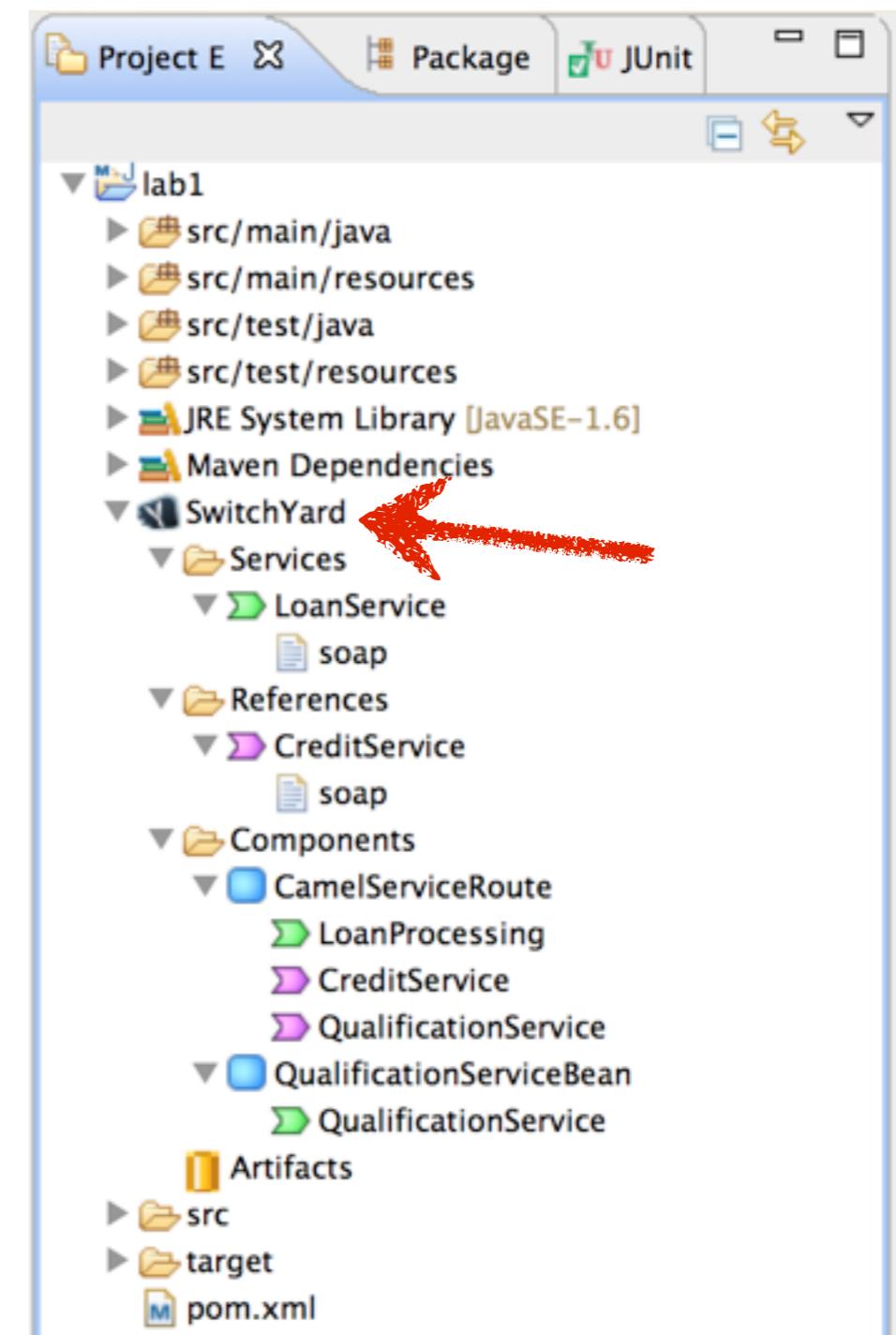
FYI

*The SwitchYard node provides two functions in the Project Explorer:*

- An 'outline' style view of the application configuration (services, references, components, etc.)
- Shortcut to the SwitchYard visual application editor via a double-click.

TODO

1. Double-click the SwitchYard node to begin your journey down the rabbit hole!



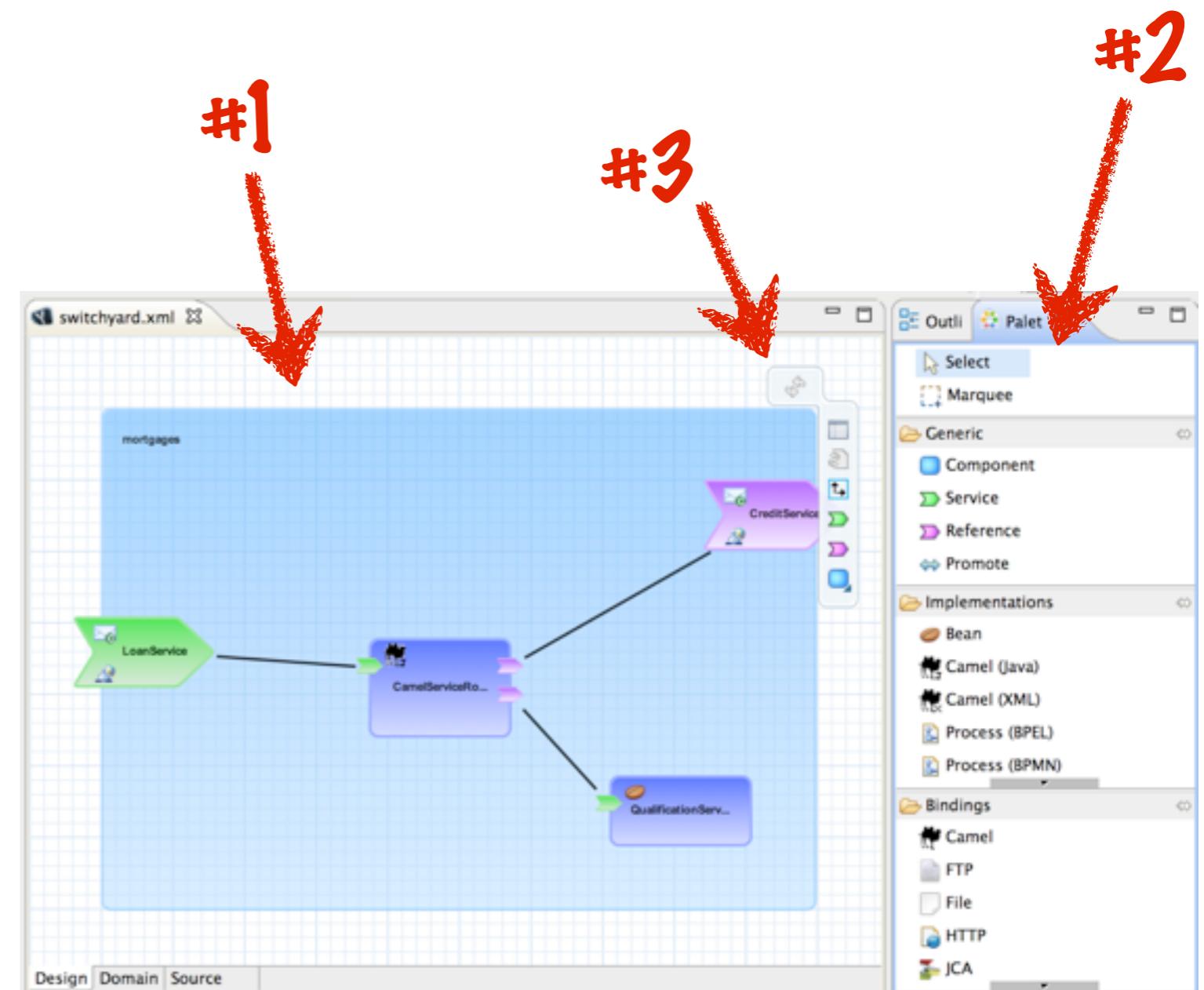
# Visual Editor

FYI

#1 : The **canvas** provides a visual representation of the structure and relationships in a SwitchYard application.

#2 : The **palette** provides a set of application building blocks which can be dragged onto the canvas to create an application.

#3 : The **button bar** appears when you hover over items on the canvas and provides context-sensitive options for each item.



# Living Large with the Button Bar

## FYI

*The button bar is really convenient when interacting with the visual editor. Just hover over the portion of the diagram you want to interact with and choose an option from the button bar.*

*This page provides a key to the more frequently used buttons on the button bar. The properties button is used the most by far, so definitely make note of that one. If the other buttons don't make sense now, don't worry they will soon.*

*NOTE: the focus for the button bar can be kinda touchy at times when there are multiple areas of focus close together. Do or do not. There is no try.*



*Opens properties view*



*Edit the interface for a service or reference.*



*Promotes a component service.*



*Generate WSDL from a Java interface.*



*Adds a reference to a component or composite.*



*Generate Java interface from a WSDL.*



*Adds a service to a component or composite.*



*Create a unit test class for a service.*



*Adds a binding to a composite service or reference.*

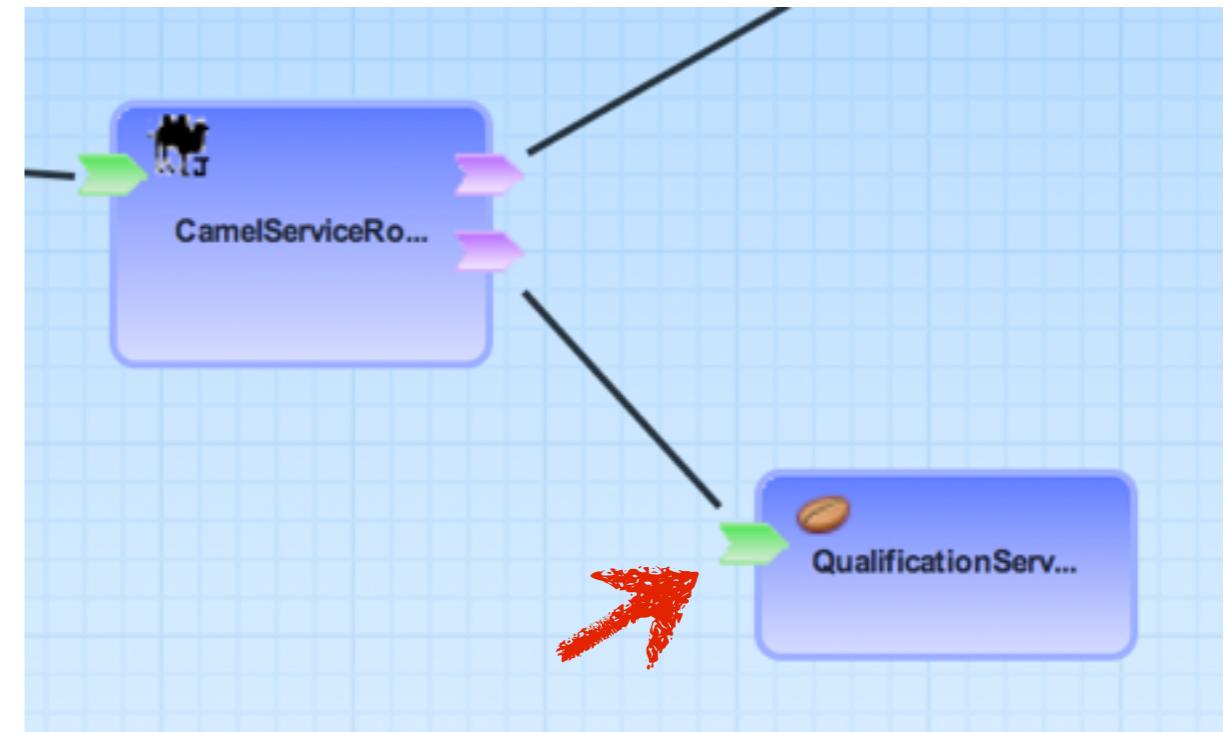
# Component Service Contract

## FYI

All services in a SwitchYard application have a contract. You can view the contract for any service by double-clicking on the green service icon.

## TODO

1. Double-click on the green component service icon (see red arrow) for QualificationService and view the contract.



# QualificationService

## interface.java



```
1 package mortgages;
2
3 public interface QualificationService {
4
5     Applicant qualify(Applicant applicant);
6 }
7
```

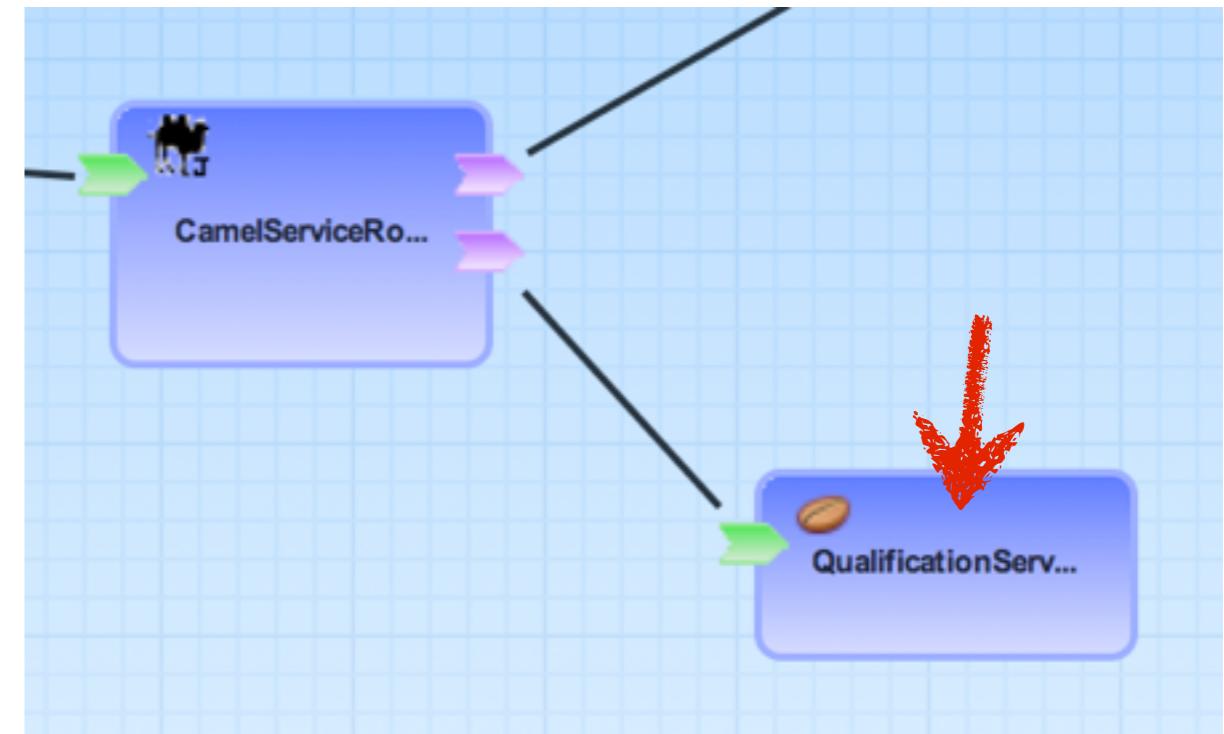
# Component Service Implementation

## FYI

*Service logic is provided in SwitchYard using an 'implementation'.  
Implementation types include: Java/CDI, Camel, BPEL, BPMN 2, and Drools.*

## TODO

1. Double-click on the blue component for QualificationService and view the implementation of the service.



# QualificationService

## implementation.bean



```
J QualificationServiceBean.java X
1 package mortgages;
2
3+ import org.switchyard.component.bean.Property;
4
5
6 @Service(QualificationService.class)
7 public class QualificationServiceBean implements QualificationService {
8
9     private static final String EASY = "easy";
10
11@ Property(name = "creditTerms")
12    private String creditTerms;
13
14@ Override
15    public Applicant qualify(Applicant applicant) {
16        if (EASY.equals(creditTerms)) {
17            applicant.setApproved(true);
18        } else {
19            applicant.setApproved(applicant.getCreditScore() >= 600);
20        }
21
22        return applicant;
23    }
24
25 }
```

# Component Service

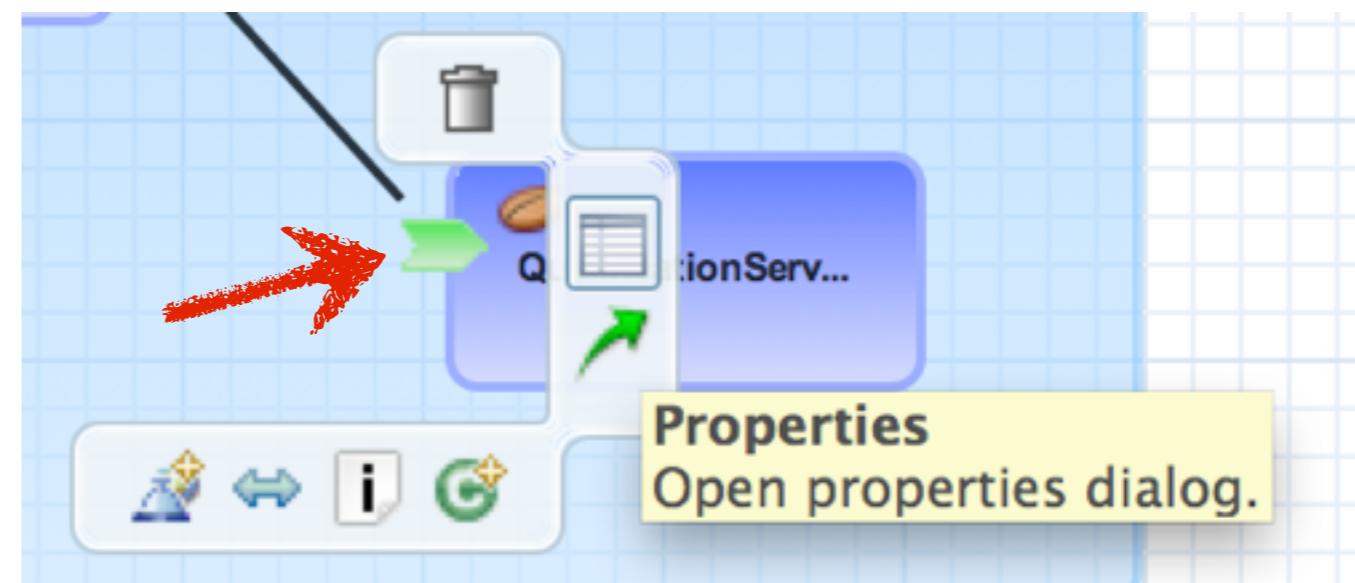
## Properties

### FYI

*The properties view allows you to view the configuration of a service including details on the contract and policy requirements.*

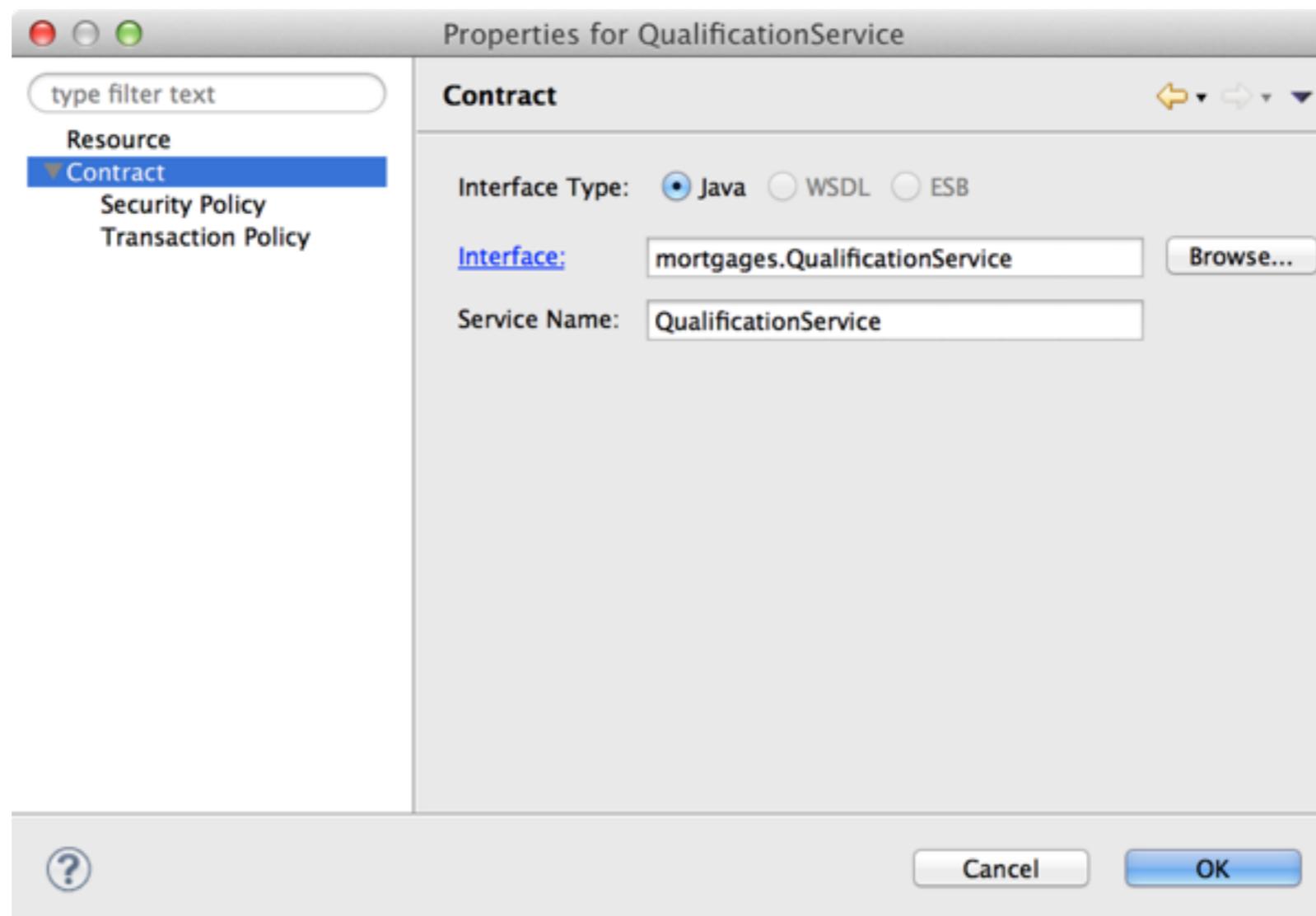
### TODO

1. Hover over the green component service on QualificationService.
2. Select the Properties button on the button bar.
3. In the resulting dialog, select Contract in the left frame.



# QualificationService

## Configuration



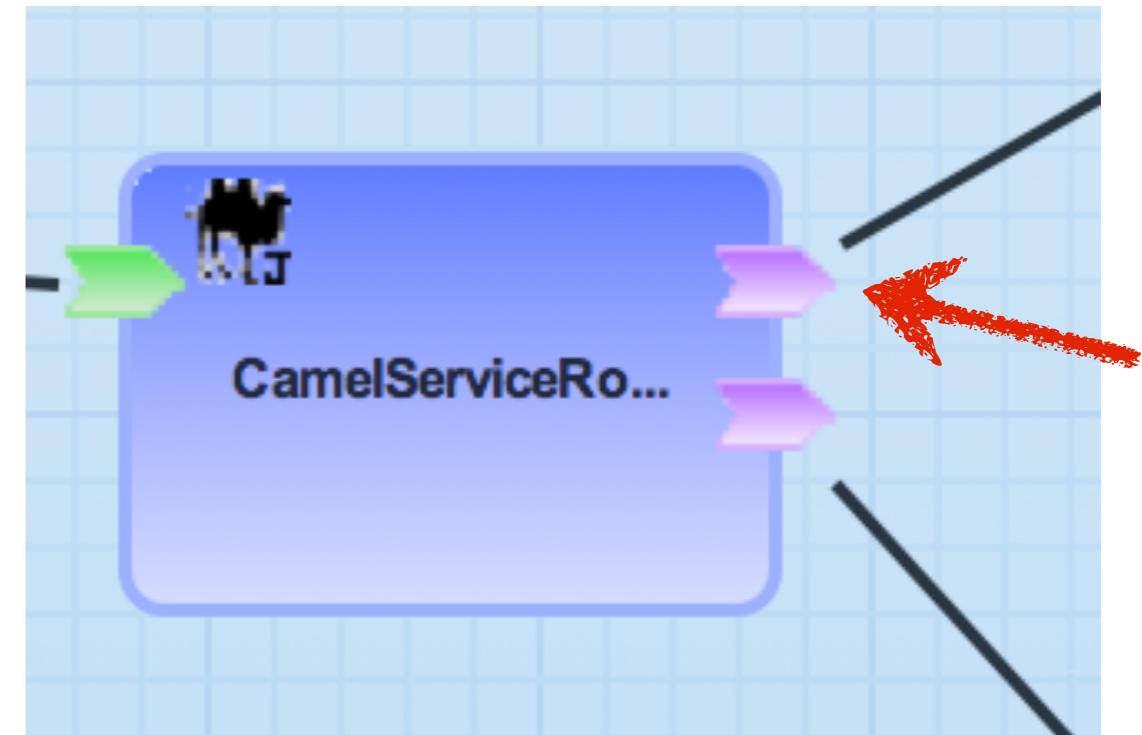
# Component Reference

## FYI

Service implementations can invoke other services through **references**. All references have a contract and can be wired to services inside or outside the application.

## TODO

1. View the contract for a reference by double-clicking on the purple reference icon.
2. View the properties for a reference using its button bar.
3. In the resulting dialog, select Contract in the left frame.



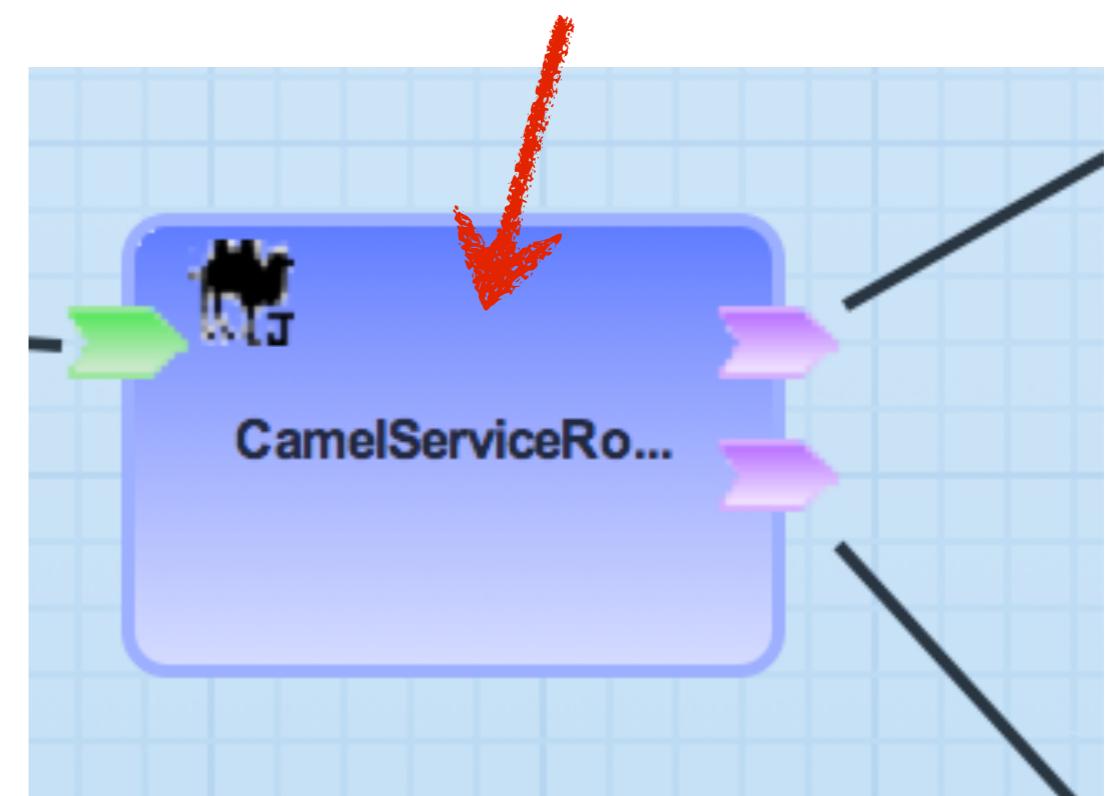
# Camel Implementation

## FYI

*The SwitchYard application model is implementation-agnostic, which means the bits in the diagram look the same no matter what the implementation type is (which is a good thing).*

## TODO

1. Double-click on the CamelServiceRoute component to open the service implementation.



# Camel Route

## implementation.camel

FYI

*There's a lot going on in this route, so if you are not familiar with Camel don't sweat the details. One thing to take note of are the following lines:*

```
.to("switchyard://CreditService")
.to("switchyard://QualificationService")
```

*These map to the service references from the previous slide. All service invocations from an implementation should be done via a reference. SwitchYard uses this information to track service dependencies and decouple services from one another.*

```
1 package mortgages;
2
3 import org.apache.camel.builder.RouteBuilder;
4
5 public class CamelServiceRoute extends RouteBuilder {
6
7     /**
8      * The Camel route is configured via this method. The from:
9      * endpoint is required to be a SwitchYard service.
10     */
11    public void configure() {
12        from("switchyard://LoanProcessing")
13            .log("Request for LoanProcessing : ${body}")
14            .choice()
15                .when(header("{urn:mortgages:1.0}status").isEqualTo("gold"))
16                    .to("bean://AutoApproval")
17                .otherwise()
18                    .filter().simple("${body.creditScore} == 0")
19                        .to("switchyard://CreditService")
20                            .end()
21                            .to("switchyard://QualificationService")
22                                .end()
23                            .end()
24        .log("Result of LoanProcessing: ${body}");
25    }
26
27
28 }
```



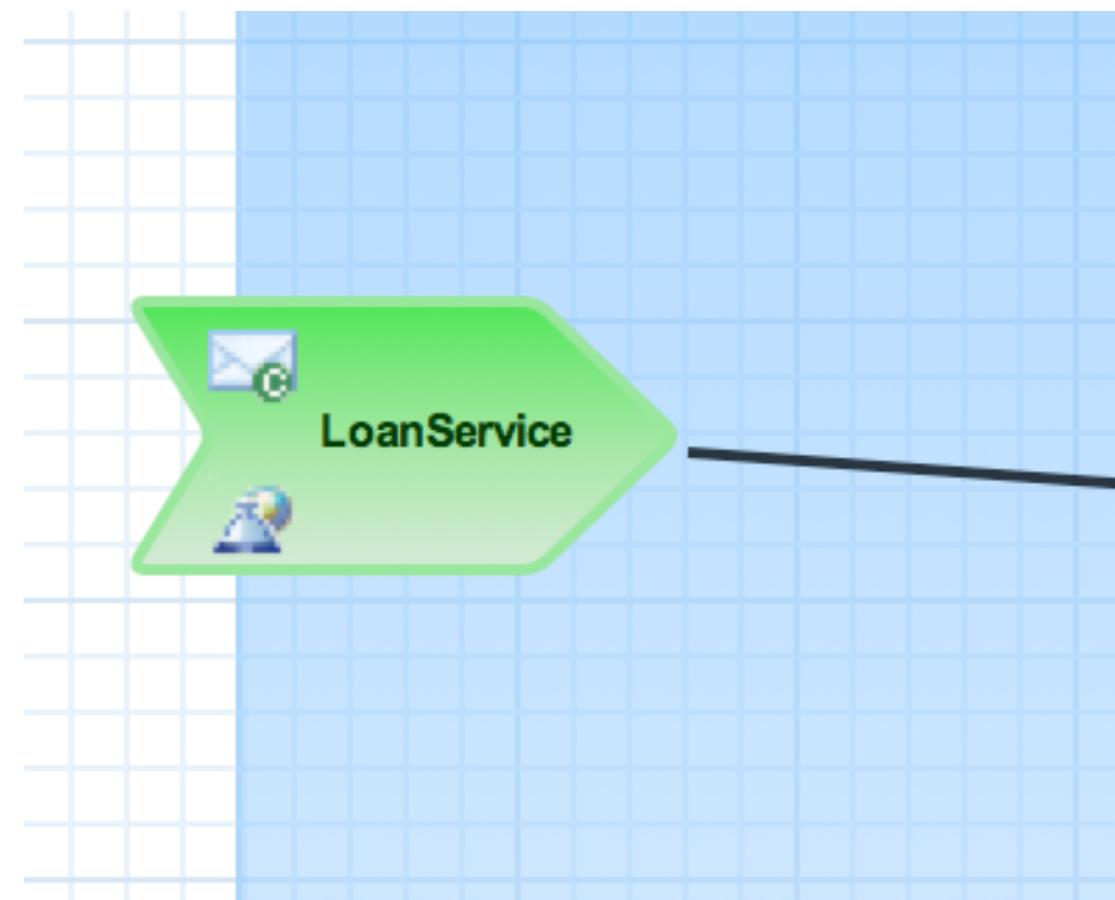
# Composite Service

## FYI

A composite service promotes a service implementation in your application and makes it available to external consumers through one or more service bindings. All composite services have a contract.

## TODO

1. Access the properties view for the composite service using its button bar.

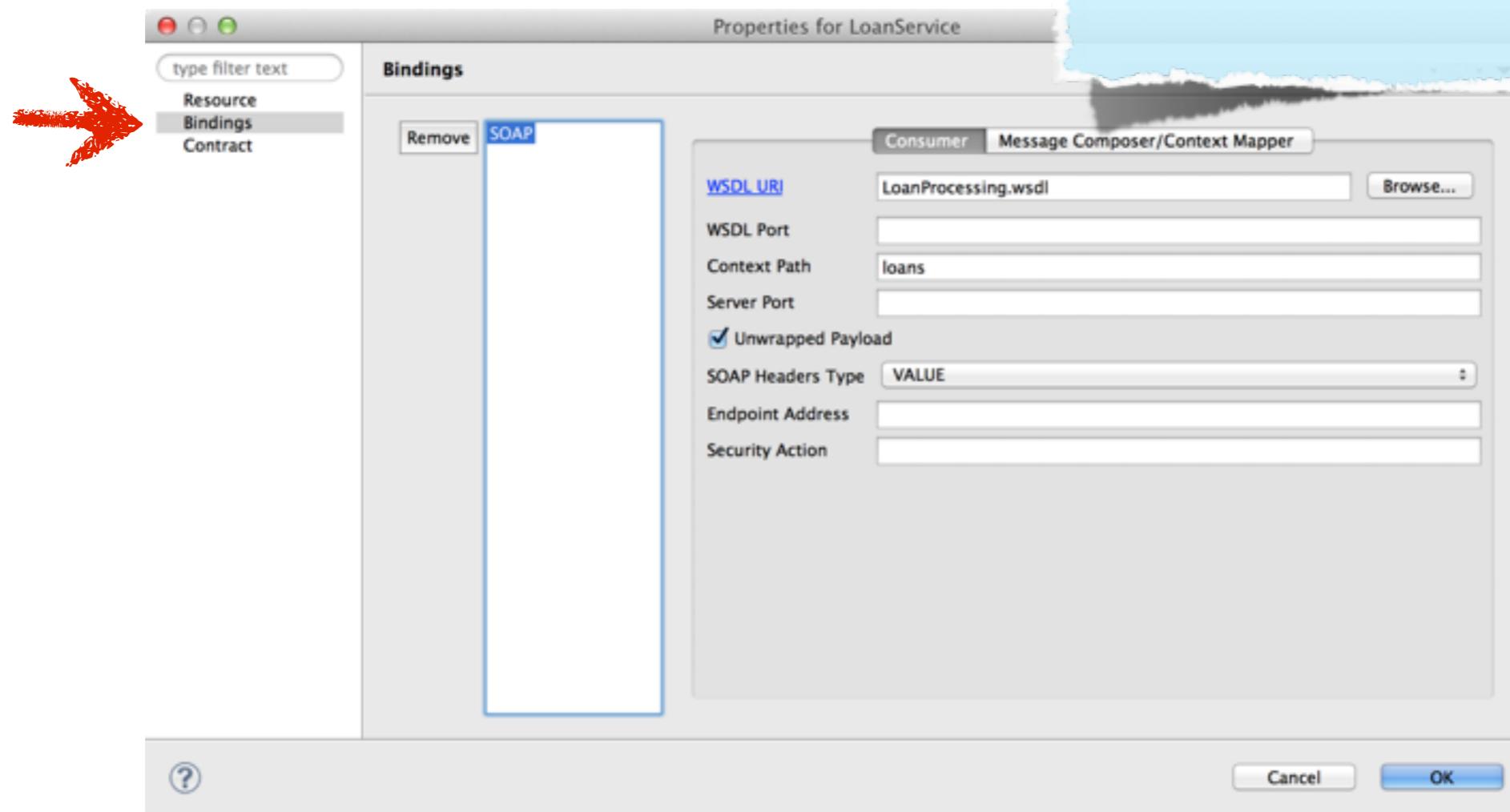


# Composite Service

## bindings

**TODO**

1. Select the bindings view in the left frame to view bindings for this service.



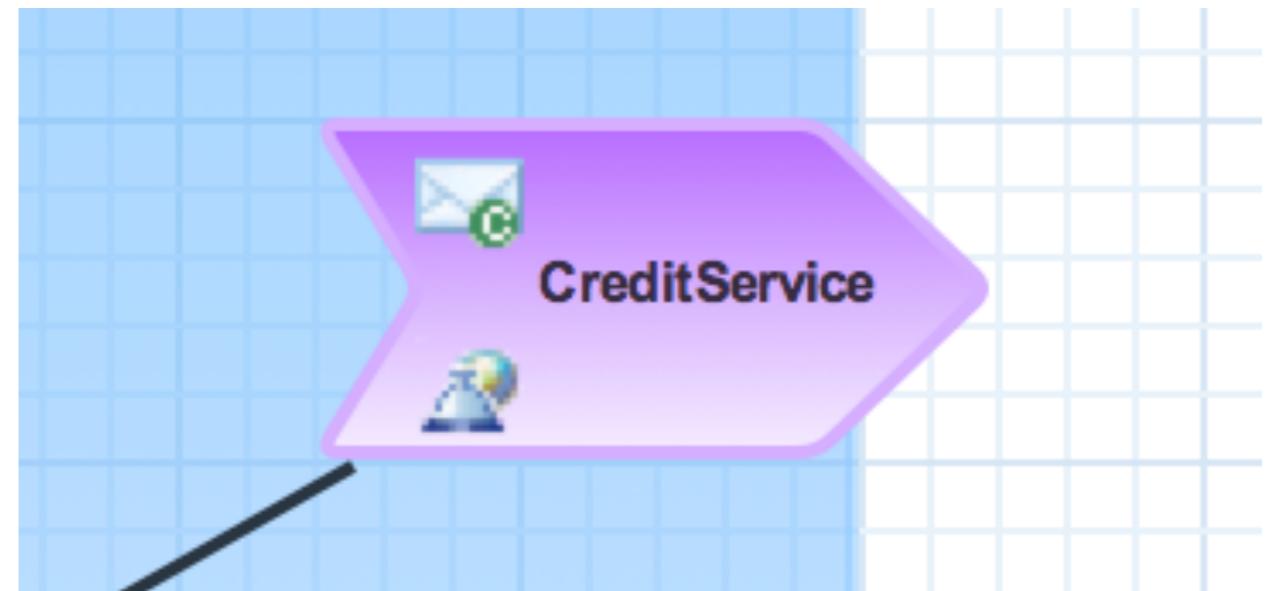
# Composite Reference

## FYI

A composite reference promotes a component reference to allow an implementation to invoke services outside the application through a binding. All composite references have a contract.

## TODO

1. Access the properties view for the composite reference using its button bar.



# Wiring

## FYI

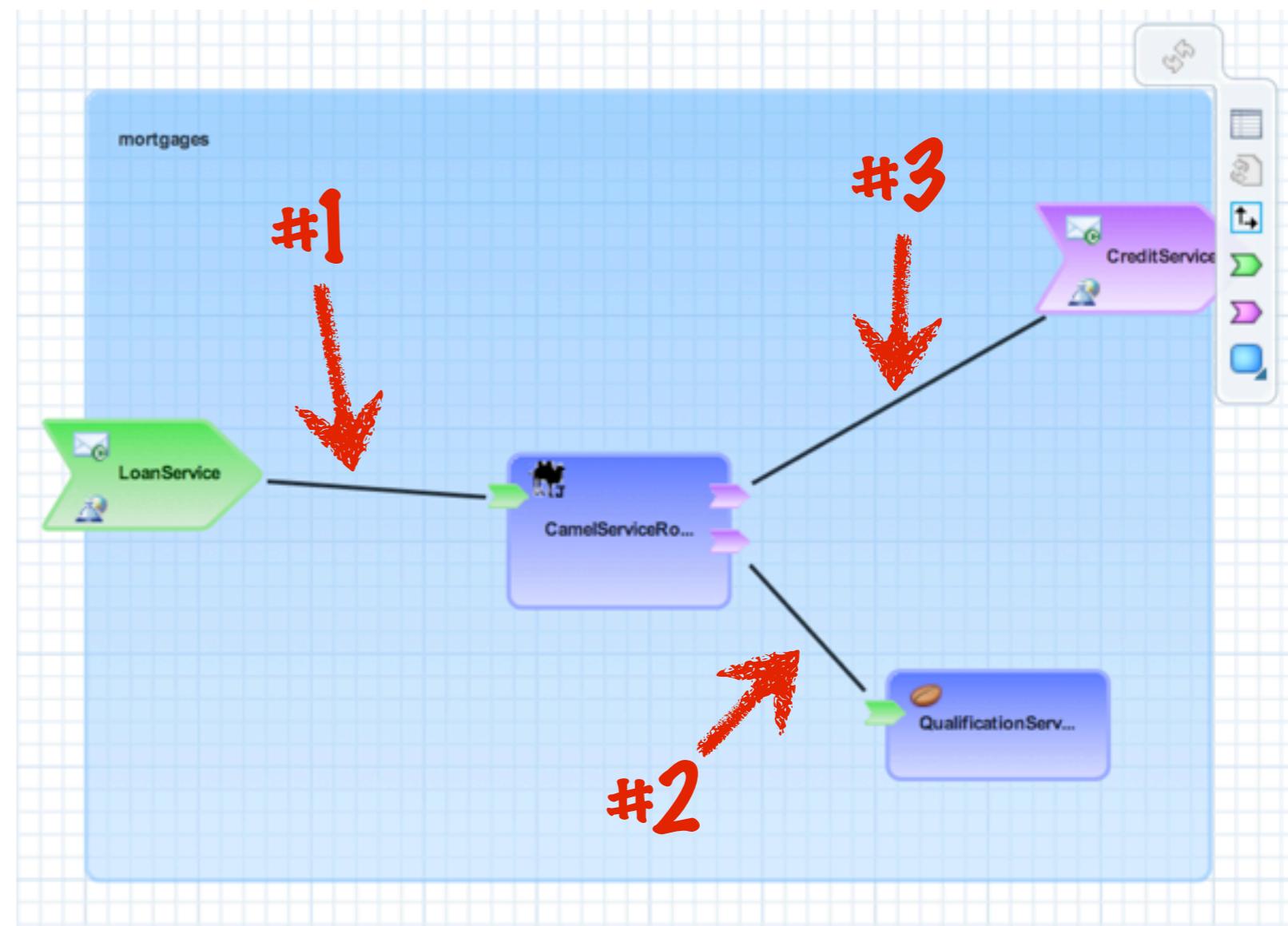
The black connector lines depict wiring which defines the relationship between services and references.

#1 : A component service is wired to a composite service to make it available outside the application through a binding.

#2 : A component reference is wired to a service provided in the local application.

#3 : A component reference is wired to a composite reference to consume an external service through a binding.

Each wire represents a message path where runtime interceptors provide functionality such as transformation, validation, policy enforcement, etc.



# Composite Properties

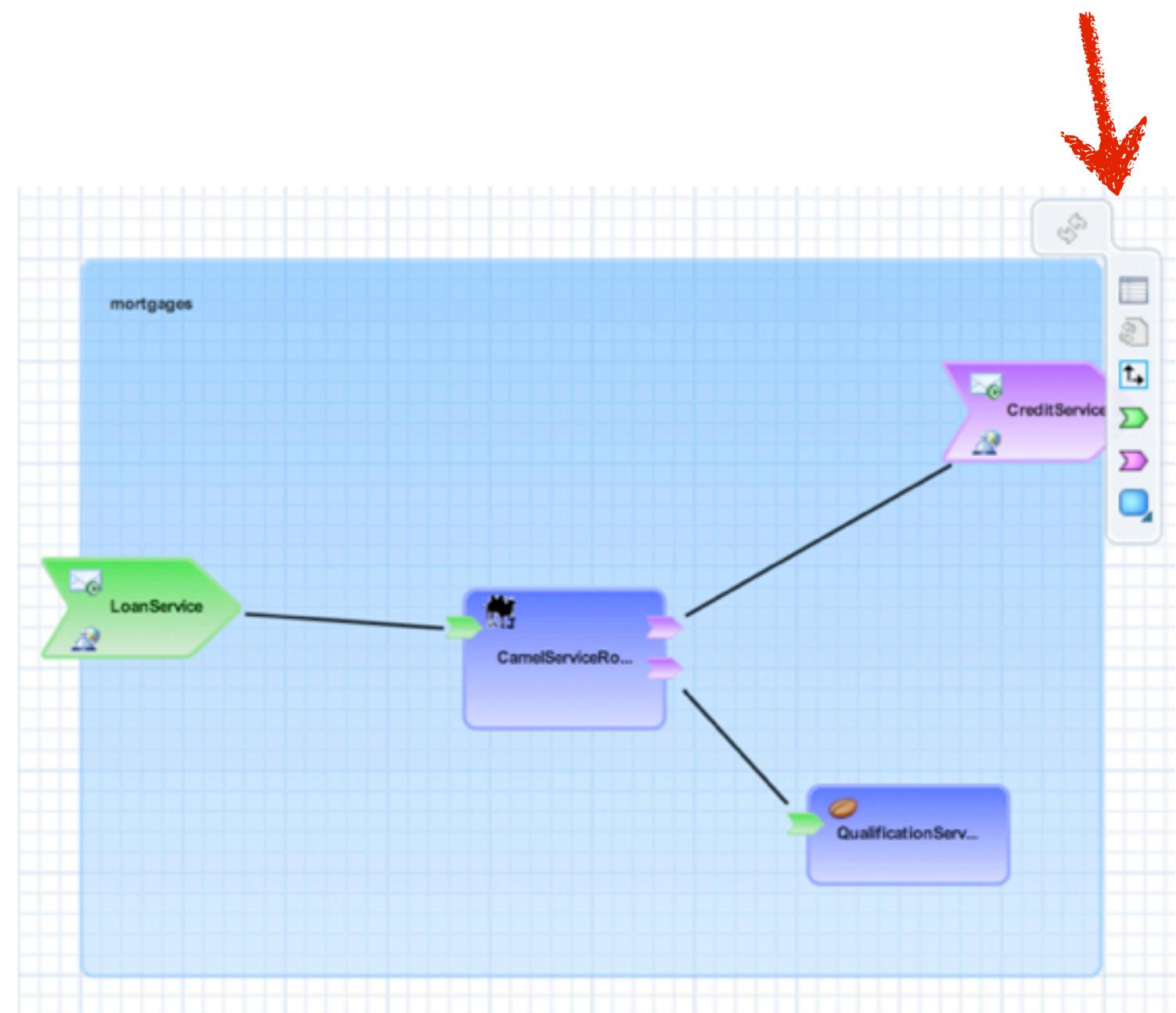
## FYI

The composite properties view provides access to some important aspects of application configuration:

- Transformers
- Validators
- Property Definition
- Name and namespace

## TODO

1. Access the composite button bar on the upper right hand corner of the composite.
2. Select the Properties button.
3. Select Transforms in the left frame to view transformers for the application.



# Composite Properties

## Transforms

Screenshot of a composite properties editor showing the 'Transforms' tab.

The interface includes a toolbar with standard OS X window controls (red, green, blue buttons), a search bar labeled 'type filter text', and a sidebar with navigation links: Resource, Composite, Properties, Git, **Transforms** (which is selected and highlighted in blue), and Validators.

The main area is titled 'Properties for' and contains a table titled 'Transforms'.

From	To	Type	Actions
applicant {http://lab2.mortgages/}	Applicant {mortgages}	Smooks	Add
process {urn:mortgages:1.0}	Applicant {mortgages}	JAXB	Remove
Applicant {mortgages}	applicant {http://lab2.mortgages/}	Smooks	

At the bottom of the dialog are 'Cancel' and 'OK' buttons.

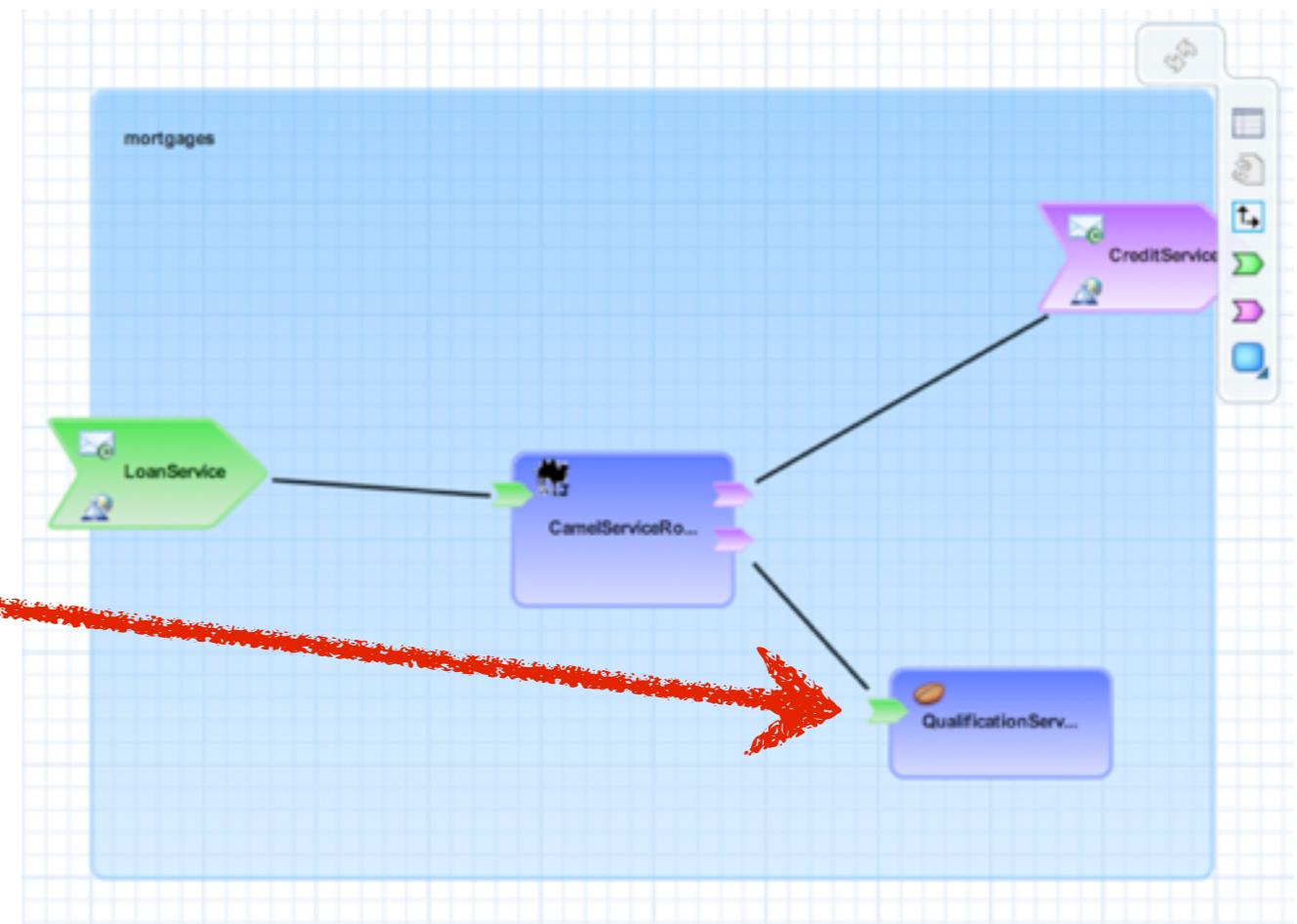
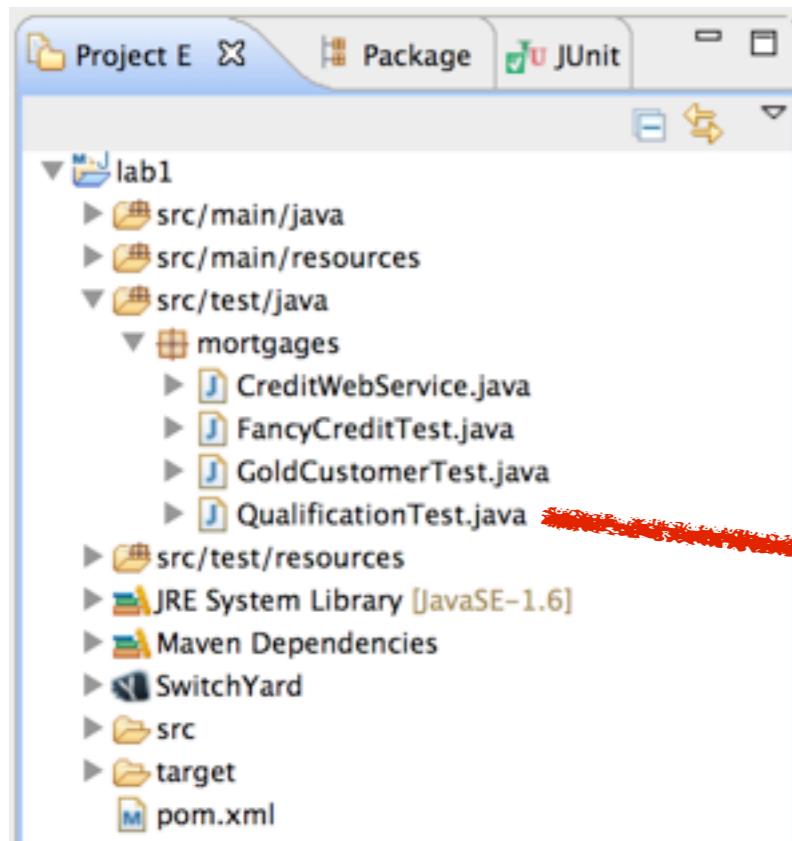
# Unit Testing Services

## FYI

SwitchYard allows you to unit test individual services in your application as you build it. QualificationTest contains test methods which invoke QualificationService directly to verify that its behavior matches requirements.

## TODO

1. Double-click QualificationTest.java in src/test/java to open it.



# QualificationTest

## FYI

The `@ServiceOperation` annotation allows you to inject and invoke component services directly into unit tests.

## TODO

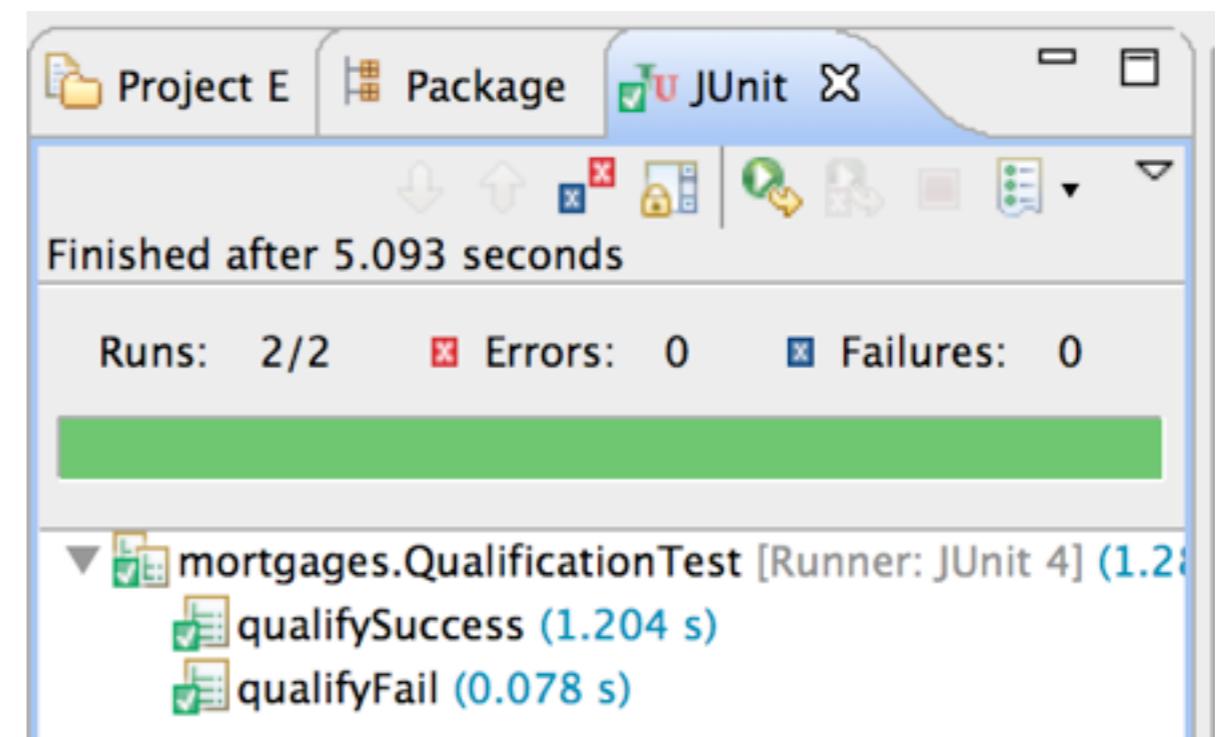
1. Go to the Run menu in the main menu bar and select 'Run As -> JUnit Test' to run the unit test.

```
J QualificationTest.java X
1 package mortgages;
2
3+ import mortgages.Applicant;
4
5 @RunWith(SwitchYardRunner.class)
6 @SwitchYardTestCaseConfig(
7     config = SwitchYardTestCaseConfig.SWITCHYARD_XML,
8     mixins = CDIMixIn.class)
9 public class QualificationTest {
10
11     @ServiceOperation("QualificationService")
12     private Invoker service;
13
14     @Test
15     public void qualifySuccess() {
16         Applicant request = new Applicant();
17         request.setName("Joan Jones");
18         request.setCreditScore(650);
19
20         Applicant reply = service.operation("qualify").sendInOut(request)
21                         .getContent(Applicant.class);
22
23         // validate the results
24         Assert.assertTrue(reply.isApproved());
25     }
26 }
```

# Success?

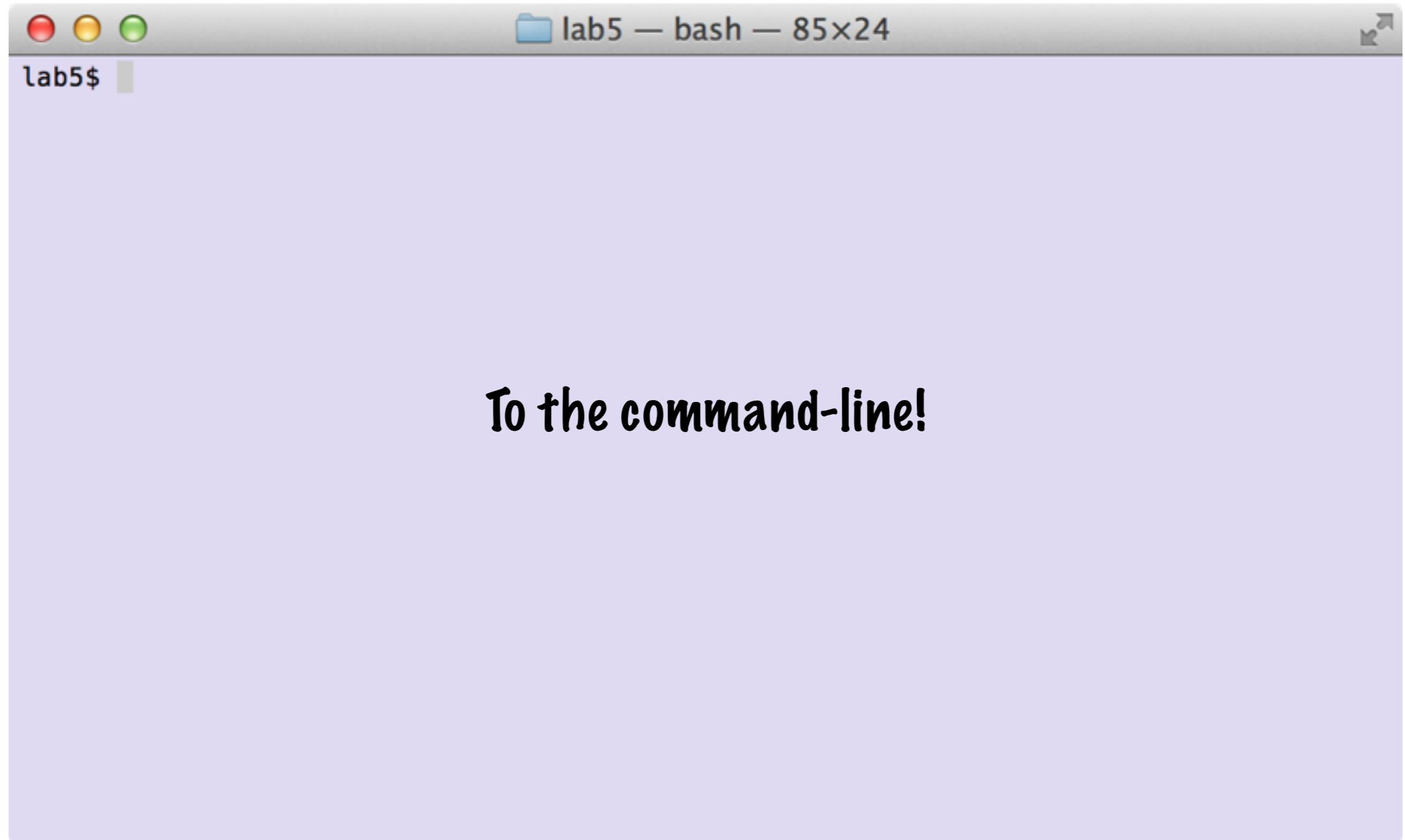
**FYI**

*You should see a nice green bar indicating that the test passed. If you get a red bar indicating test failure, raise your hand and someone will be by to help.*



# SwitchYard Runtime

- It's time to run SwitchYard. In the next part of the lab, we will
  - Install and configure the SwitchYard Server
  - Add the runtime to JBDS
  - Deploy a SwitchYard application



# Installing SwitchYard

- Unzip EAP 6.1 into a directory of your choice
- Run the SwitchYard installer to install SwitchYard into EAP 6.1

## TODO

```
> cd /tmp  
> unzip jboss-eap-6.1.0.zip  
> unzip switchyard-installer-1.0.0.Final.zip  
> cd switchyard-installer-1.0  
> ant -DEAP_PATH=/tmp/jboss-eap-6.1
```

<https://docs.jboss.org/author/display/SWITCHYARD/Installing+SwitchYard>

# Configuring SwitchYard

## Add a management user

### TODO

```
> cd jboss-eap-6.1  
> bin/add-user.sh
```

```
soap-6.0-alpha$ bin/add-user.sh
```

```
What type of user do you wish to add?
```

- a) Management User (`mgmt-users.properties`)
  - b) Application User (`application-users.properties`)
- (a):

```
Enter the details of the new user to add.
```

```
Realm (ManagementRealm) :
```

```
Username : guest
```

```
Password :
```

```
Re-enter Password :
```

```
About to add user 'guest' for realm 'ManagementRealm'
```

```
Is this correct yes/no? y
```

```
Added user 'guest' to file '/homes/lab5/Servers/soap-6.0-alpha/standalone/configuration/mgmt-users.properties'
```

```
Added user 'guest' to file '/homes/lab5/Servers/soap-6.0-alpha/domain/configuration/mgmt-users.properties'
```

```
Is this new user going to be used for one AS process to connect to another AS process?
```

```
e.g. for a slave host controller connecting to the master or for a Remoting connection for server to server EJB calls.  
yes/no? n
```

### Answer Key

- Q1 : <enter>
- Q2 : <enter>
- Q3 : guest
- Q4 : guest-12
- Q5 : guest-12
- Q6 : yes
- Q7 : no

### FYI

*This user will be used to login to the admin console.*

# Configuring SwitchYard

## Add an application user

### TODO

```
> bin/add-user.sh
```

```
soap-6.0-alpha$ bin/add-user.sh
```

```
What type of user do you wish to add?
```

- a) Management User (mgmt-users.properties)
  - b) Application User (application-users.properties)
- (a): b

```
Enter the details of the new user to add.
```

```
Realm (ApplicationRealm) :
```

```
Username : guest
```

```
Password :
```

```
Re-enter Password :
```

```
What roles do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[ ]: guest
```

```
About to add user 'guest' for realm 'ApplicationRealm'
```

```
Is this correct yes/no? y
```

```
Added user 'guest' to file '/homes/lab5/Servers/soap-6.0-alpha/standalone/configuration/application-users.properties'
```

```
Added user 'guest' to file '/homes/lab5/Servers/soap-6.0-alpha/domain/configuration/application-users.properties'
```

```
Added user 'guest' with roles guest to file '/homes/lab5/Servers/soap-6.0-alpha/standalone/configuration/application-roles.properties'
```

```
Added user 'guest' with roles guest to file '/homes/lab5/Servers/soap-6.0-alpha/domain/configuration/application-roles.properties'
```

```
Is this new user going to be used for one AS process to connect to another AS process?
```

```
e.g. for a slave host controller connecting to the master or for a Remoting connection for server to server EJB calls.
```

```
yes/no? n
```

### FYI

*This user will be used to connect to the server from a remote JMS test client.*

### Answer Key

**Q1** : b

**Q2** : <enter>

**Q3** : guest

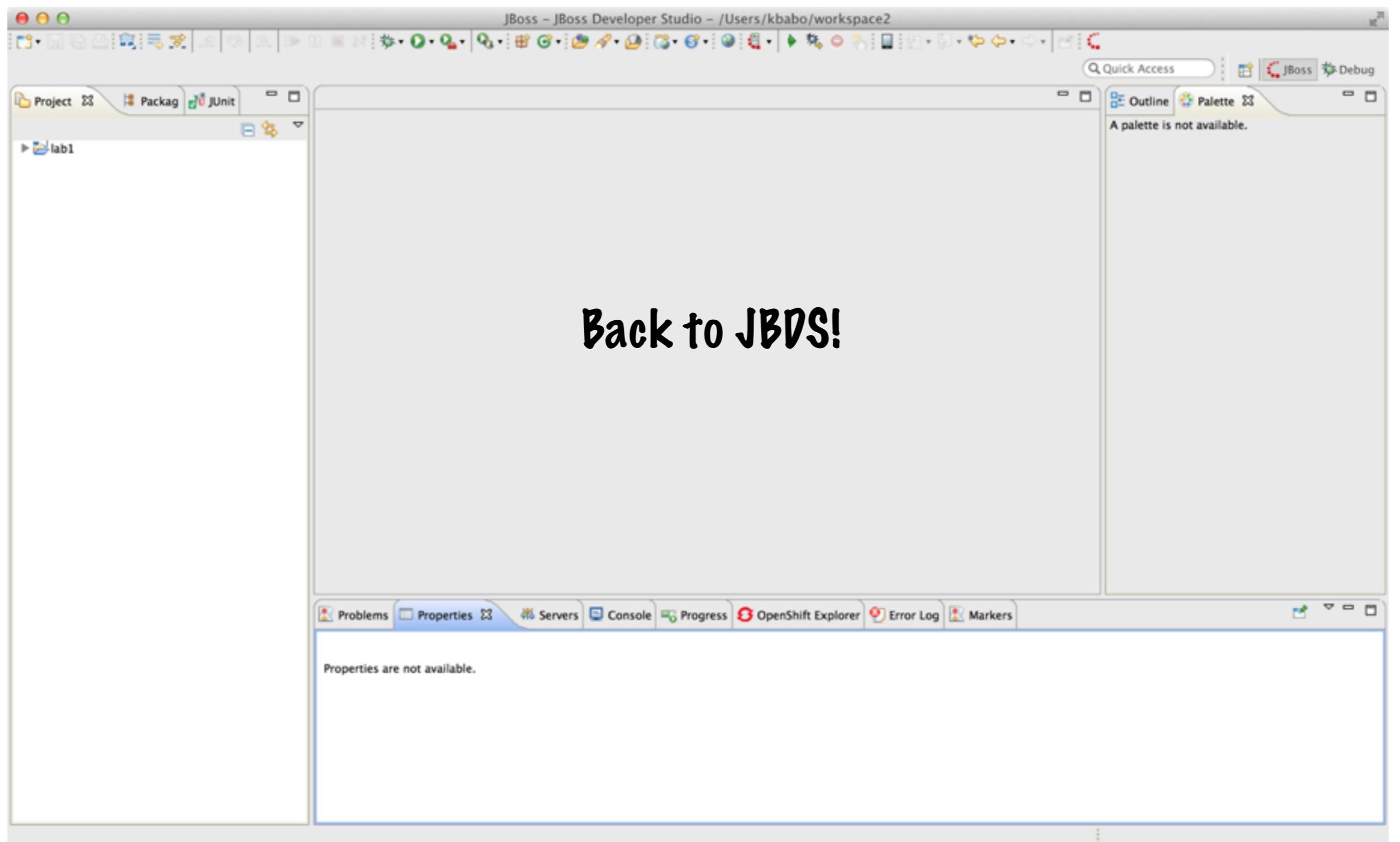
**Q4** : guest-12

**Q5** : guest-12

**Q6** : guest

**Q7** : yes

**Q8** : no



Back to JBDS!

# Adding a SwitchYard Server

## TODO

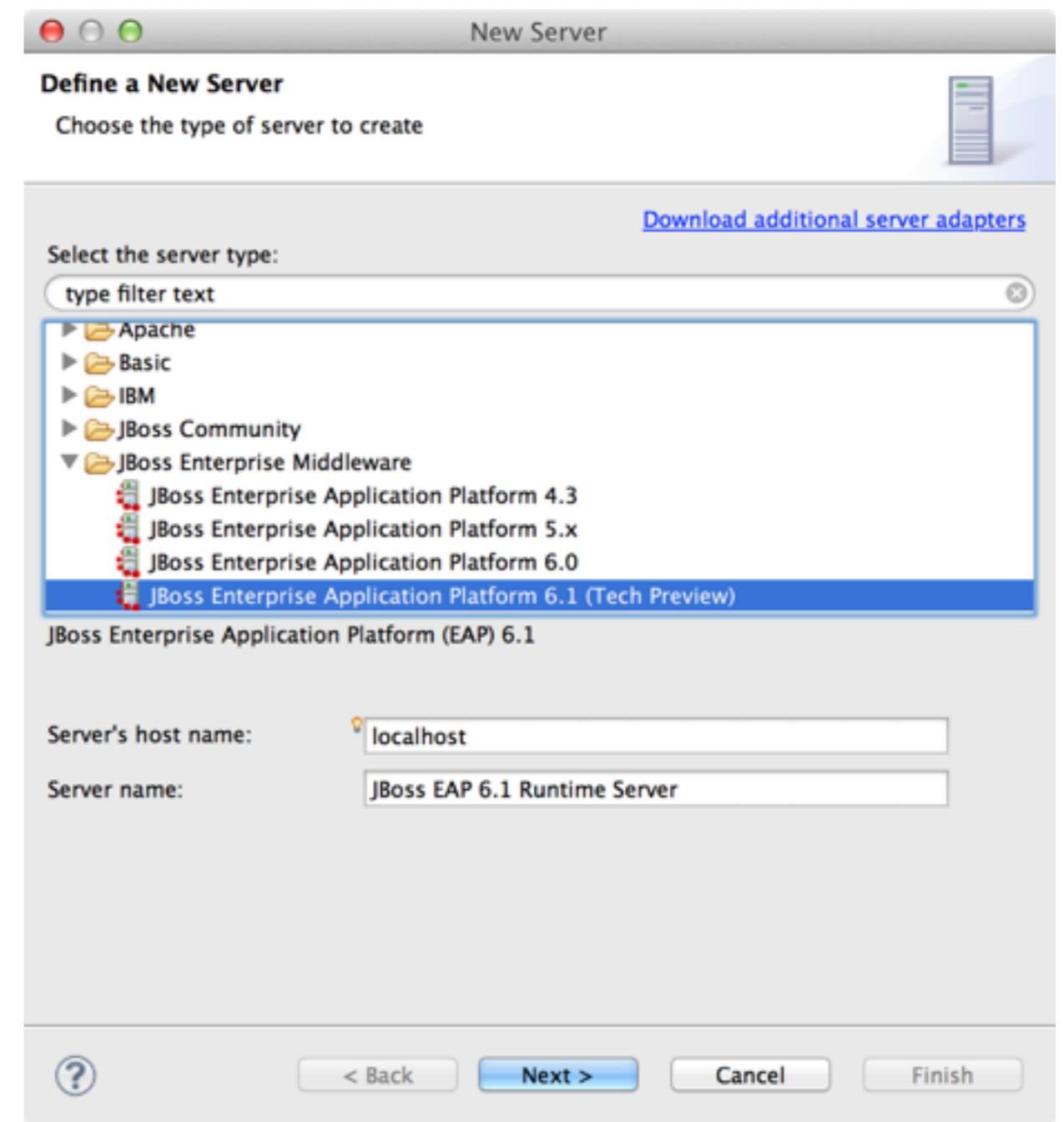
1. Select the 'Servers' tab in the bottom frame of the workspace.
2. Click on 'new server wizard ...'



# New Server

TODO

1. Select JBoss Enterprise Application Platform **6.1** (Tech Preview)
2. Click Next



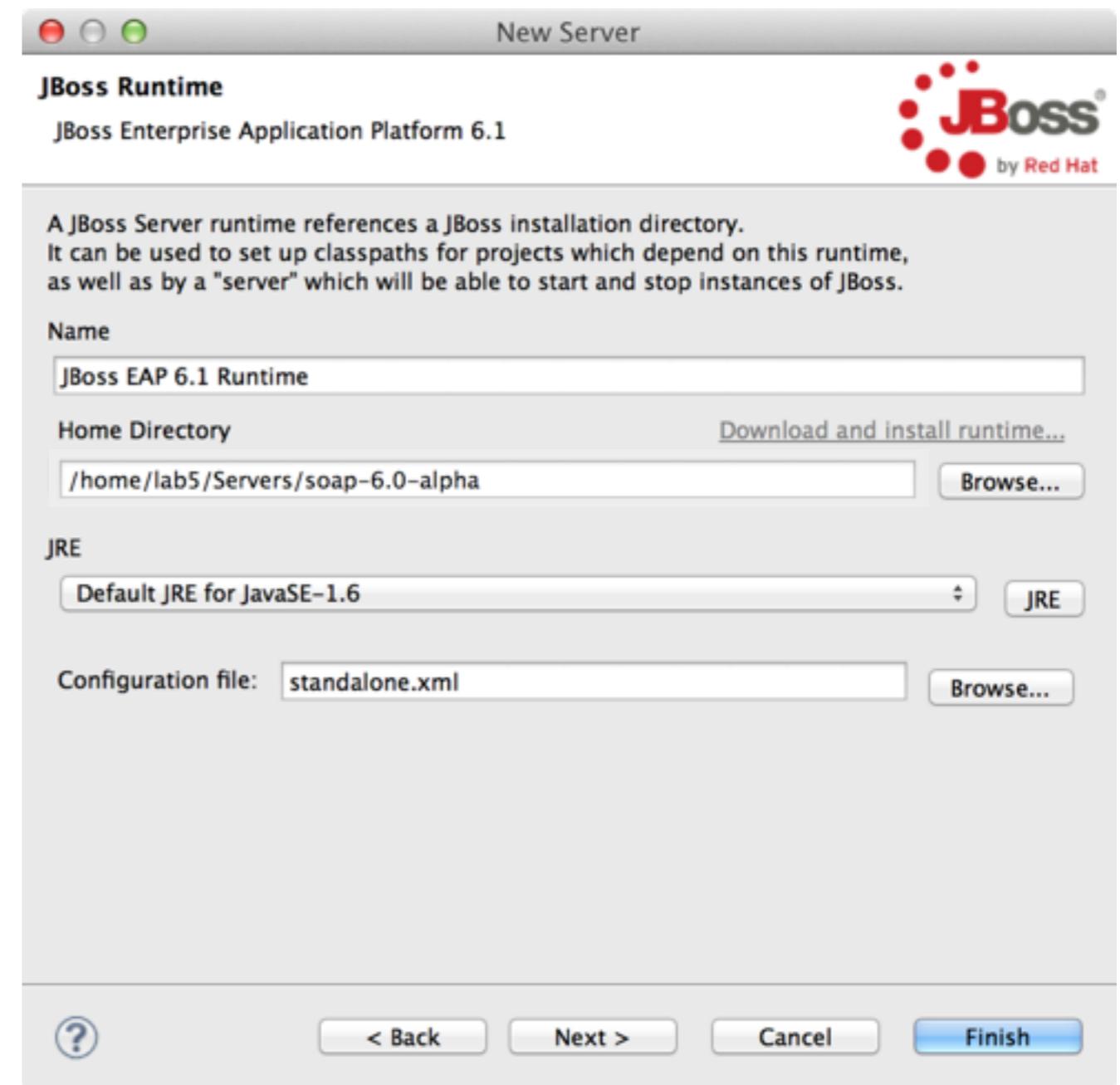
# New Server

**TODO**

1. Enter the directory of your EAP 6.1 install

e.g. /tmp/jboss-eap-6.1

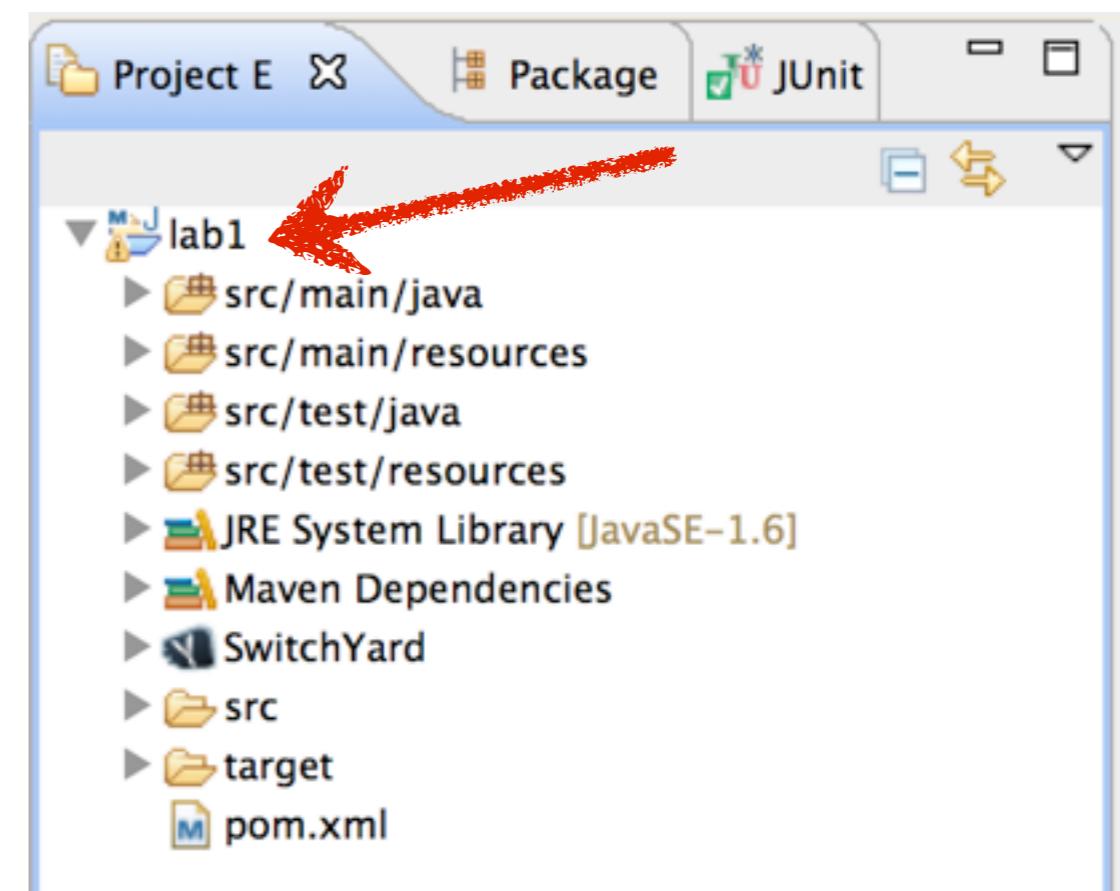
2. Click Finish



# Deploy Application

**TODO**

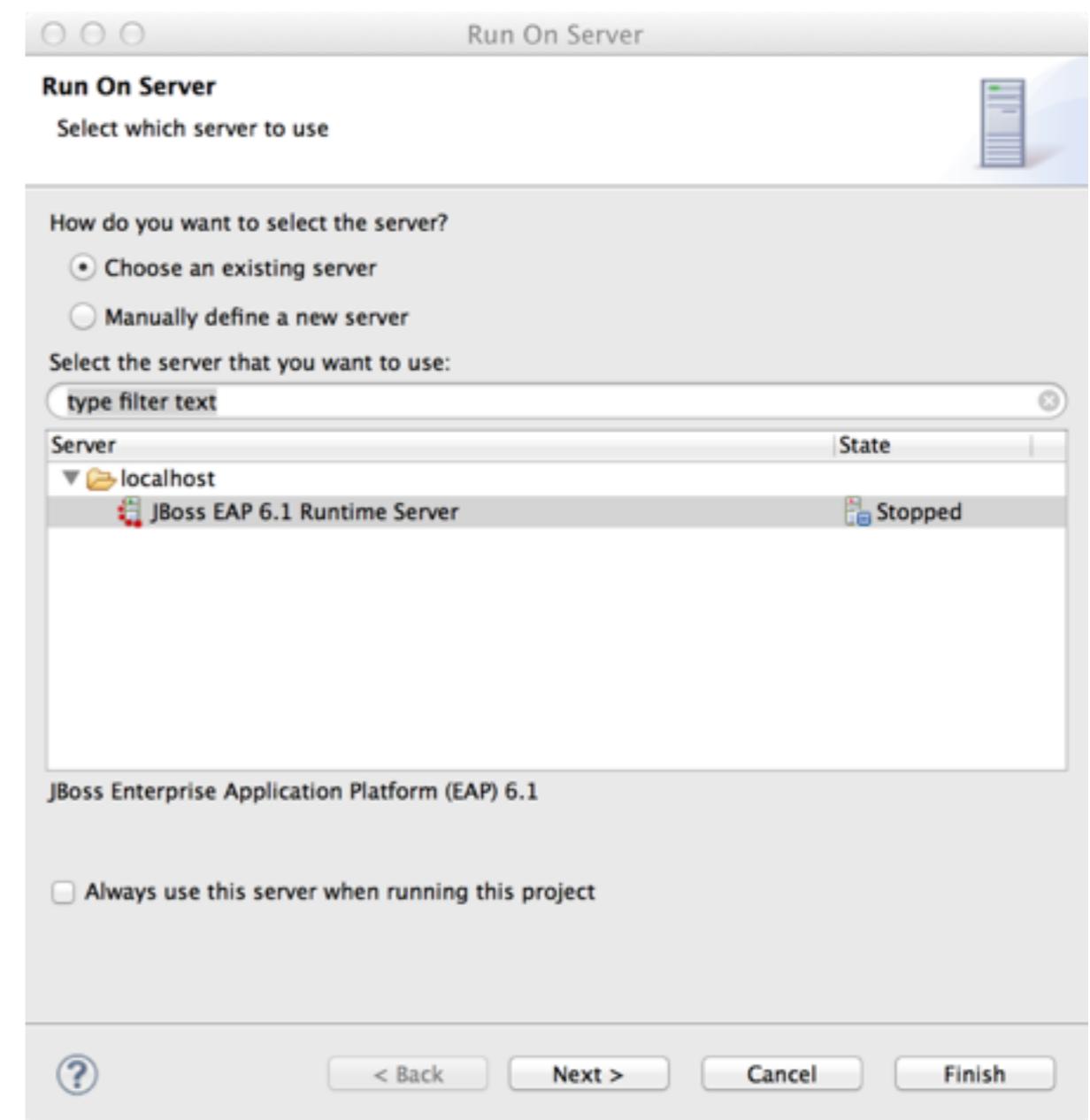
1. Right-click on the lab1 project and select Run As ... Run on Server



# Select Server

**TODO**

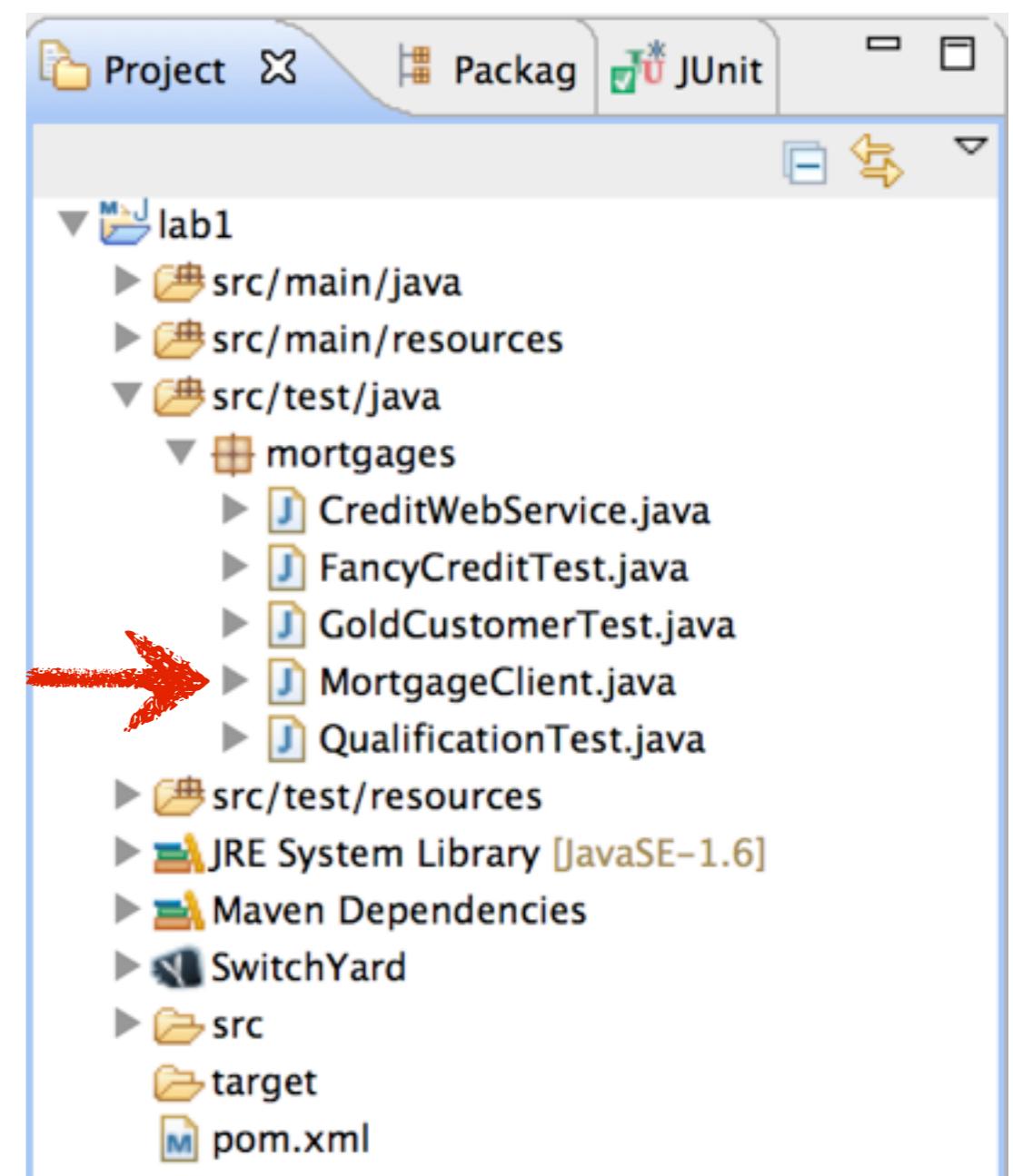
1. Select the JBoss EAP 6.1 Runtime Server
2. Click Finish



# Run Test Client

## TODO

1. Open MortgageClient.java from the Project Explorer view.
2. Go to the Run menu in the main menu bar and select 'Run As -> Java Application'



# Verify Output

TODO

1. Click here to swap between application output and server output.



```
Problems Properties Servers Console Progress OpenShift Explorer Error Log Markers
JBoss EAP 6.1 Runtime Server [JBoss Application Server Startup Configuration] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 6, 2013 9:01:43 PM)
mtomEnabled=false
[0m[0m21:32:08,151 INFO  [org.apache.cxf.service.factory.ReflectionServiceFactoryBean] (MSC service thread 1-16) Creating Service {urn:mortgages:1.0}
[0m[0m21:32:08,158 INFO  [org.apache.cxf.endpoint.ServerImpl] (MSC service thread 1-16) Setting the server's publish address to be http://localhost:8
[0m[0m21:32:08,163 INFO  [org.jboss.ws.cxf.deployment] (MSC service thread 1-16) JBWS024074: WSDL published to: file:/homes/lab5/Servers/soap-6.0-alp
[0m[0m21:32:08,164 INFO  [org.jboss.as.webservices] (MSC service thread 1-2) JBAS015539: Starting service jboss.ws.port-component-link
[0m[0m21:32:08,164 INFO  [org.jboss.as.webservices] (MSC service thread 1-13) JBAS015539: Starting service jboss.ws.endpoint."loans.deployment".LoanP
[0m[0m21:32:08,167 INFO  [org.jboss.ws.common.management] (MSC service thread 1-13) JBWS022050: Endpoint registered: jboss.ws:context=loans,endpoint=
[0m[0m21:32:08,187 INFO  [org.jboss.as.server] (DeploymentScanner threads- 2) JBAS018559: Deployed "lab1.jar" (runtime name : "lab1.jar")
[0m[0m21:33:54,611 INFO  [route2] (default-workqueue-1) Request for LoanProcessing : Applicant: Name=Baby Face Age=4 CreditScore=0 ApplicationDate=Fr
[0m[0m21:33:54,615 INFO  [route2] (default-workqueue-1) Result of LoanProcessing: Applicant: Name=Baby Face Age=4 CreditScore=0 ApplicationDate=Fri A
[0m
```

# **Lab I Complete!**