Qi4j - Composites

Rickard Öberg

Agenda

Agenda

:: Composites
:: Fragments
:: Mixins
:: Modifiers
:: Constraints
:: Concerns
:: SideEffects
:: Generic Fragments
:: Abstract Fragments
:: Instantiating Composites

Composites

Composites

:: The most basic element in Qi4j is the Composite

Composites

:: The most basic element in Qi4j is the Composite
:: A Composite is created by composing a number of Fragments.

Composites

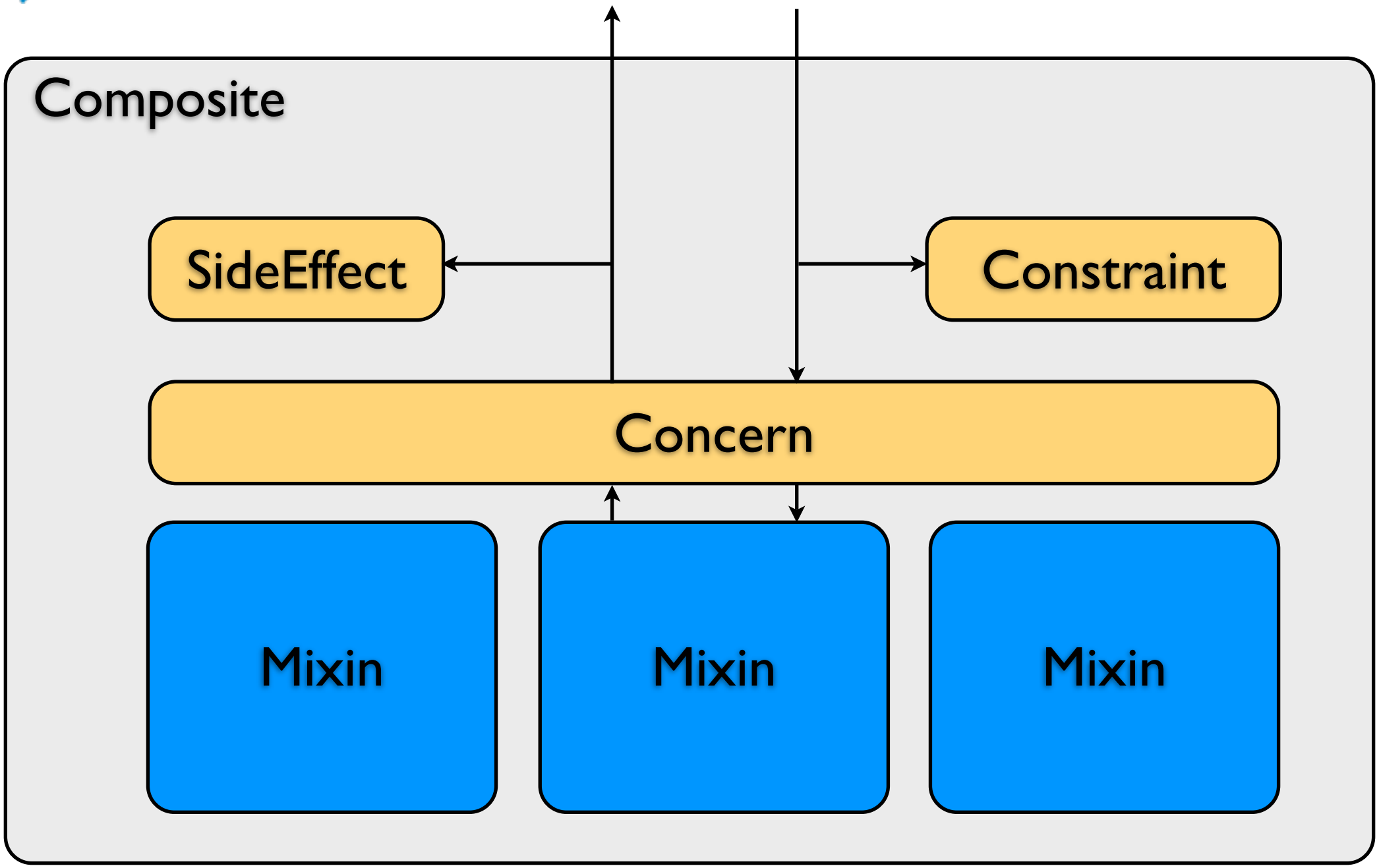
:: The most basic element in Qi4j is the Composite
:: A Composite is created by composing a number of Fragments.
:: Mixins are Fragments that can handle method invocations

Composites

:: The most basic element in Qi4j is the Composite
:: A Composite is created by composing a number of Fragments.
:: Mixins are Fragments that can handle method invocations
:: Modifiers are Fragments that modify method invocations (Decorator pattern)

Composites

:: The most basic element in Qi4j is the Composite
:: A Composite is created by composing a number of Fragments.
:: Mixins are Fragments that can handle method invocations
:: Modifiers are Fragments that modify method invocations (Decorator pattern)
  :: Constraints, Concern, SideEffects


Composite
SideEffect
Constraint
Concern
Mixin
Mixin
Mixin

Example Composite

HelloWorldComposite.java:

```
public interface HelloWorldComposite
    extends StandardComposite
{
    @Cached String say();

    void setPhrase( @NonEmptyString @Contains( "H" )String phrase )
        throws ValidationException;

    String getPhrase();

    void setName( @NotNull @Matches( "Universe|World" )String name )
        throws ValidationException;

    String getName();
}
```

HelloWorldComposite.say.js:

```
function say()
{
    return phrase + " " + name;
}
```

HelloWorldComposite.properties:

```
phrase.not.null=Phrase may not be null
phrase.contains=Phrase "{0}" must contain the string "{1}"
phrase.min.length=Length of phrase "{0}" is too short. It must be at least {1} characters
name.not.null=Name may not be null
name.matches=Name must be either Universe or World
```

Fragments

Fragments

:: Each part of a Composite is called Fragment

Fragments

:: Each part of a Composite is called Fragment
:: Fragments may use Dependency Injection

Mixins

Mixins

:: Implements methods from a domain interface

Mixins

:: Implements methods from a domain interface
:: May have state

Example Mixin

```
public class FooMixin
    implements Foo
{
    String foo;
    public String doIt(String someString)
    {
      foo += someString;
      return foo;
    }
}
```

Modifiers

Modifiers

:: A Modifier modifies method invocations
  :: Constraints
  :: Assertions
  :: SideEffects

Modifiers

- A Modifier modifies method invocations
  - Constraints
  - Assertions
  - SideEffects
- Shared between Composites
  - Must not have instance-specific state

Constraints

Constraints

:: Constraints verifies method parameters

Constraints

:: Constraints verifies method parameters
:: Implements ParameterConstraint<Annotation, Type>

Constraints

:: Constraints verifies method parameters
:: Implements ParameterConstraint<Annotation, Type>
:: Mark parameter with Annotation to apply

Example Constraint

```
interface MyService
{
   void setFoo(@Range(min=3,max=42) int someParameter);
}
```

Example Constraint

```
public class RangeConstraint
    implements ParameterConstraint<Range,int>
{
    public boolean isValid(Range annotation, int parameter)
    {
        return parameter >= annotation.min() && parameter <= annotation.max();
    }
}
```

Example Constraint

```
MyService.properties:
foo.range=Foo value “{0}” is not in the range {1} to {2}
```

Concerns

Concerns

:: Concerns verifies pre/post-conditions

Concerns

:: Concerns verifies pre/post-conditions
  :: Are allowed to change parameters

Concerns

- Concerns verifies pre/post-conditions
  - Are allowed to change parameters
  - Are allowed to throw exceptions

Concerns

- Concerns verifies pre/post-conditions
  - Are allowed to change parameters
  - Are allowed to throw exceptions
  - Are allowed to return value without calling the underlying mixin

@AppliesTo

@AppliesTo

:: Some Concerns should only be used if a certain criteria is met

@AppliesTo

- Some Concerns should only be used if a certain criteria is met
  - Method annotations

@AppliesTo

- Some Concerns should only be used if a certain criteria is met
  - Method annotations
  - Mixin interfaces

@AppliesTo

- Some Concerns should only be used if a certain criteria is met
  - Method annotations
  - Mixin interfaces
  - AppliesToFilter

@AppliesTo

- Some Concerns should only be used if a certain criteria is met
  - Method annotations
  - Mixin interfaces
  - AppliesToFilter
- Add @AppliesTo(<criteria>) annotation to Concerns and SideEffects to control this

Example Concern

```
@AppliesTo( Cached.class )
public class ReturnCachedValueConcern
    implements InvocationHandler
{
    @ThisAs private InvocationCache cache;
    @ConcernFor private InvocationHandler next;

    public Object invoke( Object proxy, Method method, Object[] args ) throws Throwable
    {
        // Try cache
        String cacheName = method.getName();
        if( args != null )
        {
            cacheName += Arrays.asList( args );
        }
        Object result = cache.getCachedValue( cacheName );
        if( result != null )
        {
            if( result == Void.TYPE )
            {
                return null;
            }
            else
            {
                return result;
            }
        }

        // No cached value found - call method
        return next.invoke( proxy, method, args );
    }
}
```

SideEffects

SideEffects

:: SideEffects may interact with external resources

SideEffects

:: SideEffects may interact with external resources
   :: Executed after last Concern has finished

SideEffects

- SideEffects may interact with external resources
  - Executed after last Concern has finished
  - Cannot change invocation result in any way

SideEffects

- SideEffects may interact with external resources
  - Executed after last Concern has finished
  - Cannot change invocation result in any way
  - May be executed asynchronously

Example SideEffect

```
interface MyService
{
    void doStuff();
}

public class CountCallsSideEffect
    implements MyService
{
    private @SideEffectFor MyService next;
    private @ThisAs Counter counter;

    public void doStuff()
    {
        counter.increment();
        return next.doStuff(); // Optional call to doStuff
    }
}
```

Composites

Composites

:: Composites declares how to compose a set of Fragments into an instantiable object

Composites

- Composites declares how to compose a set of Fragments into an instantiable object
  - Implemented as an interface

Composites

- Composites declares how to compose a set of Fragments into an instantiable object
  - Implemented as an interface
  - Extends Composite interface

Composites

- Composites declares how to compose a set of Fragments into an instantiable object
  - Implemented as an interface
  - Extends Composite interface
  - Annotations declare use of Fragments

Example Composite

```
@Constraints( NotNullConstraint.class )
@Mixins( HelloWorldMixin.class )
public interface HelloWorldComposite
    extends HelloWorld, Composite {}

public interface HelloWorld
    extends HelloWorldBehaviour, HelloWorldState {}

public interface HelloWorldBehaviour {
    String say();
}

public interface HelloWorldState {
    void setPhrase( @NotNull String phrase )
        throws IllegalArgumentException;
    String getPhrase();
    void setName( @NotNull String name )
        throws IllegalArgumentException;
    String getName();
}

public class HelloWorldMixin
    implements HelloWorld {
    String phrase;
    String name;

    public String say()
    {
        return getPhrase() + " " + getName();
    }

    ... get/set methods omitted ...
}
```

Fragment declarations

Fragment declarations

:: Fragment declaration annotations may be added to either the Composite interface or Mixins

Fragment declarations

:: Fragment declaration annotations may be added to either the Composite interface or Mixins
:: A Mixin may declare Modifiers that apply to other Mixins

Example

```
public class HelloWorldStateConcern
    implements HelloWorldState
{
    @ConcernFor HelloWorldState next;

    public void setPhrase( String phrase )
        throws IllegalArgumentException
    {
        if( phrase == null )
        {
            throw new IllegalArgumentException( "Phrase may not be null" );
        }

        next.setPhrase( phrase );
    }
...
}

@Concerns( HelloWorldStateConcern.class )
public class HelloWorldStateMixin
    implements HelloWorldState
{
    String phrase;
    String name;

    public void setPhrase( String phrase )
        throws IllegalArgumentException
    {
        this.phrase = phrase;
    }
    ...
}
```

Generic Fragments

Generic Fragments

:: Fragments may implement java.lang.proxy.InvocationHandler instead of a specific domain interface

Generic Fragments

- Fragments may implement java.lang.proxy.InvocationHandler instead of a specific domain interface
- Can handle any method invocation as long as the @AppliesTo() requirements are met

Generic Fragments

:: Fragments may implement java.lang.proxy.InvocationHandler instead of a specific domain interface
:: Can handle any method invocation as long as the @AppliesTo() requirements are met
:: Example Concerns:

Generic Fragments

- Fragments may implement java.lang.proxy.InvocationHandler instead of a specific domain interface
- Can handle any method invocation as long as the @AppliesTo() requirements are met
- Example Concerns:
  - Transaction management

Generic Fragments

- Fragments may implement java.lang.proxy.InvocationHandler instead of a specific domain interface
- Can handle any method invocation as long as the @AppliesTo() requirements are met
- Example Concerns:
  - Transaction management
  - ACL checking

Generic Fragments

- Fragments may implement java.lang.proxy.InvocationHandler instead of a specific domain interface
- Can handle any method invocation as long as the @AppliesTo() requirements are met
- Example Concerns:
  - Transaction management
  - ACL checking
- Example Mixins:

Generic Fragments

- Fragments may implement java.lang.proxy.InvocationHandler instead of a specific domain interface
- Can handle any method invocation as long as the @AppliesTo() requirements are met
- Example Concerns:
  - Transaction management
  - ACL checking
- Example Mixins:
  - Get/set methods

Generic Fragments

- Fragments may implement java.lang.proxy.InvocationHandler instead of a specific domain interface
- Can handle any method invocation as long as the @AppliesTo() requirements are met
- Example Concerns:
  - Transaction management
  - ACL checking
- Example Mixins:
  - Get/set methods
  - Scripting

Generic Fragments

- Fragments may implement java.lang.proxy.InvocationHandler instead of a specific domain interface
- Can handle any method invocation as long as the @AppliesTo() requirements are met
- Example Concerns:
  - Transaction management
  - ACL checking
- Example Mixins:
  - Get/set methods
  - Scripting
  - “findBy*” -> queries

Generic SideEffect Sample

```
interface MyService
{
  @CountCalls void doStuff();
}

@AppliesTo( CountCalls.class )
public class CountCallsSideEffect
    implements InvocationHandler
{
    private @SideEffectFor InvocationHandler next;
    private @ThisCompositeAs Counter counter;

    public Object invoke( Object proxy, Method method, Object[] args ) throws Throwable
    {
        counter.increment();
        return next.invoke( proxy, method, args ); // Optional next call
    }
}
```

Abstract Fragments

Abstract Fragments

:: Fragments that do not implement an entire domain interface may be marked “abstract”

Abstract Fragments

:: Fragments that do not implement an entire domain interface may be marked "abstract"
:: Modifiers can use this to modify only specific methods

Abstract Fragments

:: Fragments that do not implement an entire domain interface may be marked "abstract"
:: Modifiers can use this to modify only specific methods
:: Mixins can use this to indicate that it can only handle part of the interface. Other Mixins must implement the other methods

Abstract Concern Sample

```
public abstract class HelloWorldStateConcern
    implements HelloWorldComposite
{
    @ConcernFor HelloWorldComposite next;

    public void setPhrase( String phrase )
        throws IllegalArgumentException
    {
        if( phrase == null )
        {
            throw new IllegalArgumentException( "Phrase may not be null" );
        }

        next.setPhrase( phrase );
    }

    public void setName( String name )
        throws IllegalArgumentException
    {
        if( name == null )
        {
            throw new IllegalArgumentException( "Name may not be null" );
        }

        next.setName( name );
    }
}
```

Abstract Mixin Sample

```
public abstract class HelloWorldMixin
    implements HelloWorldComposite
{
    @ThisCompositeAs HelloWorldComposite state;

    public String say()
    {
        return state.getPhrase() + " " + state.getName();
    }
}
```

Instantiating Composites

Instantiating Composites

:: Composites are instantiated using CompositeBuilder's

Instantiating Composites

- Composites are instantiated using CompositeBuilder's
- Builder pattern allow for complex instantiation
  - Set properties
  - Set Dependency Injection configuration

Instantiating Composites

- Composites are instantiated using CompositeBuilder's
- Builder pattern allow for complex instantiation
  - Set properties
  - Set Dependency Injection configuration
- Prototype pattern

CompositeBuilder example

```
@Structure CompositeBuilderFactory cbf;

...

// No instantiation settings - just create it
helloWorld = cbf.newCompositeBuilder( HelloWorldComposite.class ).newInstance();

// Set instantiation properties
CompositeBuilder<HelloWorldComposite> builder;
builder = cbf.newCompositeBuilder( HelloWorldComposite.class );
builder.propertiesFor( HelloWorldState.class ).setPhrase( "Hello" );
builder.propertiesFor( HelloWorldState.class ).setName( "World" );

helloWorld = builder.newInstance();
helloWorld = builder.newInstance(); // Prototype pattern
```

Questions?