MVVM Databinding CheatSheet

MVVM Concept

Model	*data model*
View	*zul, zhtml page*
ViewModel (VM)	*view's state and data model behavior*

Binding Syntax - ViewModel

@Init

executed when Binder initializes a VM class

```
public class VM {
  @Init
  public void init() {}
}
public class FooVM extends VM {
  @Init(superclass=true)
  public void init() {}
}
@Init(superclass=true)
public class BarVM extends FooVM {}
```

@AfterCompose 6.0.2

invoked in BindComposer's doAfterCompose()

```
public class VM {
  @AfterCompose
  public void afterCompose() {}
}
public class FooVM extends VM {
  @AfterCompose (superclass=true)
  public void afterCompose() {}
}
@Init(superclass=true)
public class BarVM extends FooVM {}
```

@NotifyChange

notify binder that properties changed

```
public class VM {
  @NotifyChange("prop")
  public void cmd1() {}

  @NotifyChange({"prop1","prop2"})
  public void cmd2() {}

  @NotifyChange(*) //notify all properties
  public void cmd3() {}

  @NotifyChange(.) //enforce reloading
    object
  public void cmd4() {}
}
```

@NotifyChangeDisabled

disables default notification behavior

```
public class VM {
  @NotifyChangeDisabled
  public void setProp() {}
}
```

@SmartNotifyChange 8.0

notify value change once it changed

```
public class VM {
  @SmartNotifyChange({"prop1", "prop2"})
  public void setProp() {}
}
```

@DependsOn

notify change upon property's dependency

```
public class VM {
  String prop1, prop2;
  @DependsOn({"prop1", "prop2"})
  public String setProp() {
    return prop1 + " " + prop2;
  }
}
```

@Command

identify command method defined in view

```
public class VM {
  @Command
  public void cmd() {}
  @Command("cmd")
  public void doCmd() {}
}
```

@DefaultCommand 6.5.1

mark a default command method

```
public class VM {
  @DefaultCommand
  public void defaultCmd() {}
}
```

@GlobalCommand

identify global command method defined in view

```
public class VM {
  @GlobalCommand
  public void cmd() {}
  @GlobalCommand("cmd")
  public void doCmd() {}
}
```

@DefaultGlobalCommand 6.5.1

mark a default global command method

```
public class VM {
  @DefaultGlobalCommand
  public void defaultGlobalCmd() {}
}
```

@Immutable

indicate immutable class or method

```
@Immutable
public class VM {
  @Immutable
  public String getFoo() {}
}
```

@ImmutableElements

indicate elements of the collection are immutable class

```
public class VM {
  @ImmutableElements
  private Map<String, Object> foo;
}
```

@Transient 8.0

mark element transient (not cached, in a form proxy)

```
public class VM {
  @Transient
  public Object getTotal() {
    return getFoo() * getbar();
  }
}
```

@NotifyCommand 8.0

trigger command in VM when the property changes at server

```
@NotifyCommand(value="cmd",
  onChange="_vm_.foo")
public class VM {
  private Map<String, Object> foo;
  @Command
  public void cmd() {foo.put("key",
    val);}
}
```

@NotifyCommands 8.0

trigger array of commands in VM when the property changes at server

```
@NotifyCommands({
  @NotifyCommand(value="cmd1",
    onChange="_vm_.foo")
  @NotifyCommand(value="cmd2",
    onChange="_vm_.bar")
})
public class VM {
  private Map<String, Object> foo;
  private Map<String, Object> bar;
  @Command
  public void cmd1() {foo.put("key",
    val);}
  @Command
  public void cmd2() {bar.put("key",
    val);}
}
```

@ToClientCommand 8.0

trigger callback function defined in binder.after at client

```
@ToClientCommand("doClientCmd")
public class VM {
  @Command
  public void doClientCmd() {}
}

<script defer="true">
var binder = zkbind.$('$dom_id');
binder.after('doClientCmd', function () {
  console.log('after doClientCmd');
});
</script>
```

@ToServerCommand 8.0

execute server command that triggered at client

```
<script defer="true">
var binder = zkbind.$('$dom_id');
dom.onclick = function() {
  binder.command('doServerCmd');
}
</script>

@ToServerCommand("doServerCmd")
public class VM {
  @Command
  public void doServerCmd() {
    System.out.println("triggered at
      client click");
  }
}
```

Binding Syntax - ViewModel Parameters

@BindingParam

retrieve parameter with key from command binding on ZUL

```
<button onClick="@command('cmd',
  key='value')"/>

public class VM {
  @Command
  public void cmd(@BindingParam("key")
    String value) {}
}
```

@QueryParam

retrieve parameter with key from HTTP request

```
URL: http://domain/app/foo.zul?key=val

public class VM {
  @Init
  public void init(@QueryParam("key")
    String value) {}
}
```

@HeaderParam

retrieve parameter with key from HTTP request header

```
public class VM {
  @Init
  public void init(@HeaderParam("user-
    agent") String userAgent) {}
}
```

@CookieParam

retrieve parameter with key from HTTP request cookie

```
public class VM {
  @Init
  public void init(@CookieParam("token")
    String token) {}
}
```

@ExecutionParam

retrieve parameter with key from current execution's attribute

```
Executions.getCurrent().setAttribute(
  "key", "value");

public class VM {
  @Init
  public void init(@ExecutionParam("key")
    String value) {}
}
```

@ExecutionArgParam

retrieve parameter with key from current execution's argument

```
<include src="foo.zul" arg="value"/>

public class VM {
  @Init
  public void
    init(@ExecutionArgParam("arg") String
    value) {}
}
```

@ScopeParam

retrieve value from specified scope's attribute

```
public class VM {
  @Init
  public void

  init(@ScopeParam(scopes=Scope.SESSION,
      value="user") User user) {}
}
List of Scope: COMPONENT, SPACE, PAGE,
DESKTOP, SESSION, APPLICATION, AUTO
```

@SelectorParam

retrieve component from selector syntax

```
public class VM {
  @Command
  public void
    cmd(@SelectorParam("#compId")
    Component comp) {}
}
```

@ContextParam

retrieve context object

```
public class VM {
  @Command
  public void
    cmd(@ContextParam(ContextType.Binder)
    Binder binder) {}
}
List of Context: BIND_CONTEXT, BINDER,
TRIGGER_EVENT6.0.1, COMMAND_NAME6.0.1,
EXECUTION, COMPONENT, SPACE_OWNER, VIEW,
PAGE, DESKTOP, SESSION, APPLICATION
```

@Default

assign default value when it's null

```
public class VM {
  @Command
  public void cmd(@BindingParam("key")
    @Default("3") Integer value) {}
}
```

Binding Syntax - View

@id

give an id to current binding target

```
<div viewModel="@id('vm')
  @init('pkg.VM')"
  validationMessage="@id('vmsg')" />
<div form="@id('fx')" />
```

@init

initialize an attribute's value, no reload

```
<div viewModel="@id('vm')
  @init('pkg.VM')"/>
<textbox value="@init(vm.value)"/>
<button disabled="@init(vm.disabled)"/>
```

@load

load data from VM, no save back

```
<textbox value="@load(vm.value)"/>
<intbox value="@load(vm.value,
  after='cmd')"/>
<button disabled="@load(vm.disabled)"/>
```

@save

save data to VM, no loading

```
<textbox value="@save(vm.value)"/>
<textbox value="@save(vm.value,
  before='cmd')"/>
```

@bind

load/save data from/to ViewModel

```
<textbox value="@bind(vm.value)"/>
```

@ref 6.0.1, PE/EE

create reference to an EL

```
<div p="@ref(vm.person)">
  <label value="@load(p.name)"/>
</div>
```

@command

execute when the event fires

```
<button onClick="@command('cmd')"/>
<button onClick="@command('cmd',
  arg=value)"/>
```

@global-command

execute when the event fires

```
<button onClick="@global-command('cmd')"/>
```

@converter

convert data between UI and VM

```
<label value="@load(vm.date)
  @converter('formatteddate',
  format='yyyy/MM/dd')"/>
<label value="@load(vm.date)
  @converter(vm.dateConverter)"/>
```

@validator

validate data when saving to VM

```
<textbox value="@load(vm.date)
  @validator('beanValidator')"/>
<textbox value="@load(vm.date)
  @validator(vm.dateValidator)"/>
```

@template

determine which template to render child components

```
<grid model="@load(vm.model)
  @template('editable')">
  <template name="editable" var="bean">
  </template>
</grid>
```