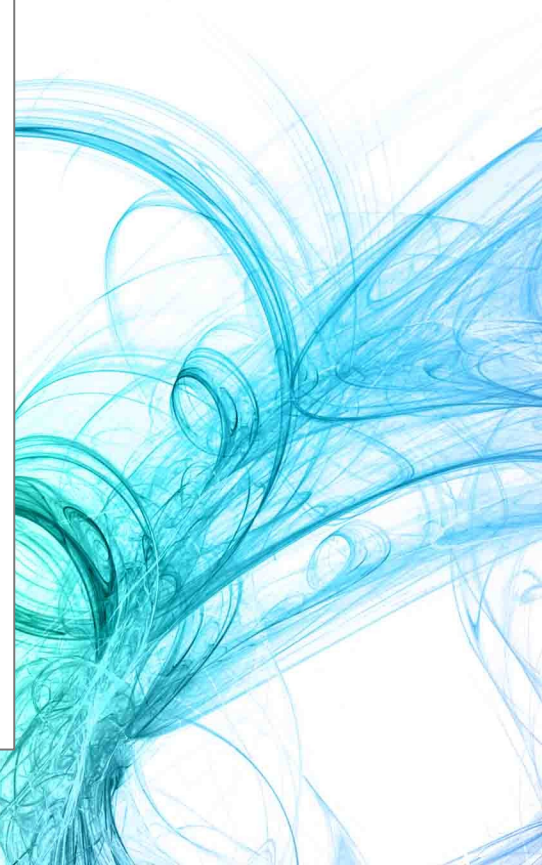


indra

Tapestry 5

Tapestry 5 – inversion of control and presentation framework

Jan Jirout
CZJUG / 30.5.2010 / Praha

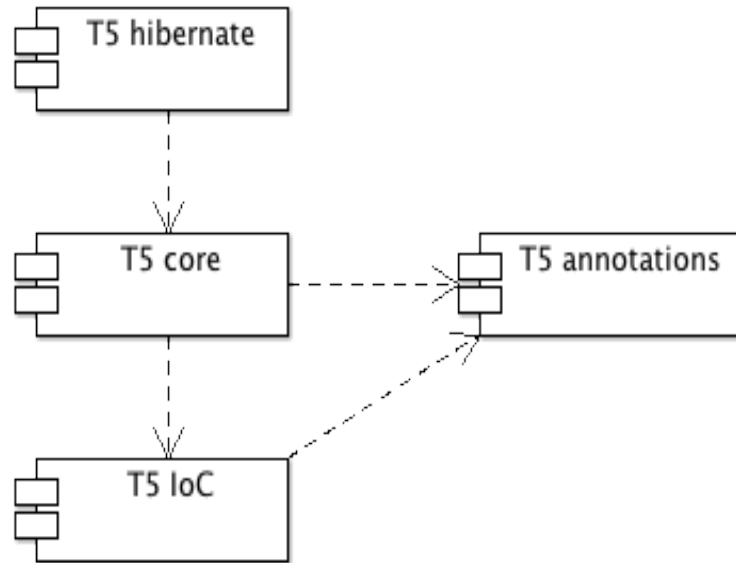


Index

- Inversion of control
 - Service life cycle
 - Configuration
- Request Response processing
- Pages – class, template and properties
- Components
- UI components
 - Grid
 - Bean edit form
- Testing

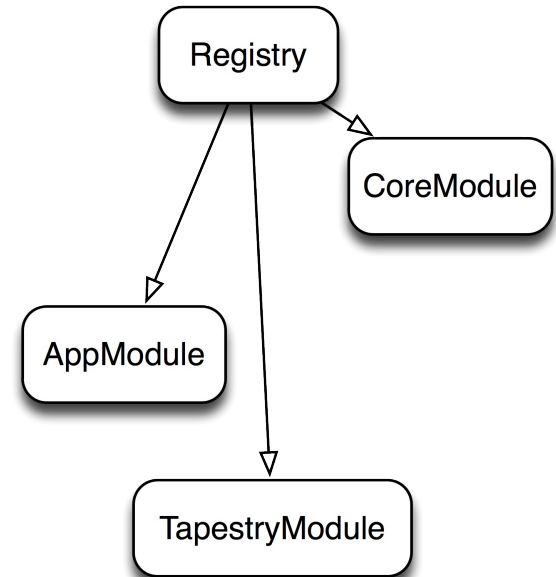
Tapestry modules

- T5 core
 - Web framework
 - UI components
- T5 IoC
 - Background
- T5 hibernate
 - Integration
 - Session in view



Application module

- Application unit
- AppModule java class
 - Configuration
 - Object constructing
 - Configuration contribution
- Module loading
 - Automatic META-INF manifest file
 - Manually
- Maven
 - More than one in jar



```

public static void bind(ServiceBinder binder) {
    binder.bind(DocumentGetter.class, BlogDocumentGetter.class).withId(
        "blogDocumentGetter");
}

```

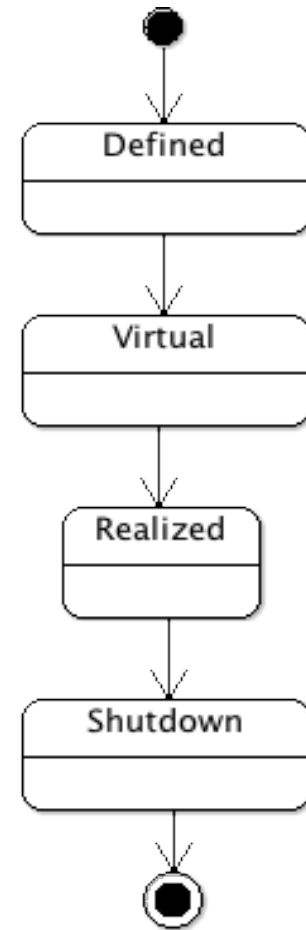
Inversion of control

```
⊖ @Inject  
    private HibernateService hibernateService;  
  
⊖ private final static Logger logger = Logger
```

- @InjectService("someServiceId")
- @InjectPage("somePage")
- Application module class
- Objects
- Lazy references
- Unique service id
 - Inject versus InjectService

Service life cycle

- Lazy loading
- Life cycle
 - Defined
 - Virtual
 - Realized
 - Shutdown
- Scopes
 - Per-thread
 - Singleton



Service instance creating

■ Bind

- In app module class bind method

```
binder.bind(CaptchaService.class, CaptchaServiceImpl.class);
```

■ Build

- Allows complex initialization / configuration

```
public static CacheManager buildCacheManager() {  
    return new CacheManagerImpl("META-INF/jboss-cache.xml");  
}
```

■ Constructor

- Referenced services are not lazy
- Most params rule

```
public ArticleServiceImpl(final AppInitializationService appInitializationService) {
```

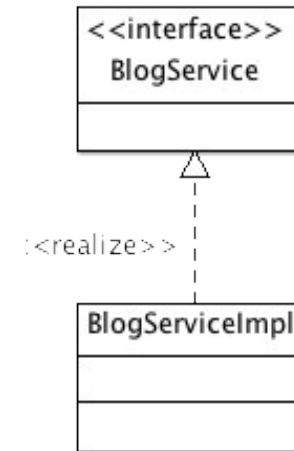
Service shutdown

- Before shutdown code could be executed some code
- Code is defined in app module
- RegistryShutdownHub

```
shutdownHub.addRegistryShutdownListener(new RegistryShutdownListener() {  
    @Override  
    public void registryDidShutdown() {  
        hibernateService.destroy();  
    }  
});
```

Service ID

- Unique identification
- Valid in all initialized modules
- @Local
- Proxy – implementation access
- Implementation is not accessible
- Different ids for same service class
 - GridDataSource
 - Request filter
 - Dispatcher
 - ...



Hibernate integration

- Simplest approach
 - T5 Object coercion
 - Session and transaction boundaries

```
@EagerLoad
public static HibernateService buildHibernateService(
    RegistryShutdownHub shutdownHub,
    AppInitializationService appInitializationService) {
    final HibernateService hibernateService = new HibernateServiceImpl(
        appInitializationService
            .loadConfiguration(AppInitializationService.CONFIGURATION_HIBERNATE));
    shutdownHub.addRegistryShutdownListener(new RegistryShutdownListener() {
        @Override
        public void registryDidShutdown() {
            hibernateService.destroy();
        }
    });
    return hibernateService;
}
```

Configuration of service 1/3

- Contribute methods
 - Configuration specification
 - In many modules
 - Unordered executing
- Application module

```
public static void contributePageLinkTransformerHtml(  
    MappedConfiguration<String, String> configuration) {  
    configuration.add("/rss.xml", "/infr/siteRss");  
}
```

- Class constructor

```
public PageLinkTransformerHtml(final Map<String, String> fixPathMapping) {  
    this.fixPathMapping = fixPathMapping;  
}
```

Configuration of service 2/3

- Configuration types
 - Unordered collection
 - Injected is collection, configured is Configuration
 - Method add(configured object)
 - Ordered list
 - Allows order
 - Requires configuration id
 - Order by “before” and “after”
 - Map
 - Previous slide

Configuration of service 3/3

- Ordered configuration

```
public static void contributeMasterDispatcher(  
    OrderedConfiguration<Dispatcher> configuration,  
    @InjectService("ImageDispatcher") Dispatcher imageDispatcher,  
    @InjectService("AccessController") Dispatcher accessController,  
    @InjectService("RedirectDispatcher") Dispatcher redirectDispatcher) {  
    configuration.add("ImageDispatcher", imageDispatcher,  
        "before:PageRender");  
    configuration.add("AccessController", accessController,  
        "before:PageRender");  
    configuration.add("RedirectDispatcher", redirectDispatcher,  
        "before:ComponentEvent");  
    configuration.overrideInstance("ComponentEvent",  
        ComponentEventDispatcherExtension.class, "before:PageRender");  
}
```

Request/Response

- Extends javax.servlet.Filter
- Cover *
- Basic object types
 - Page
 - Component
 - Mixin
- Component
 - Template
 - Class
 - Messages

Project structure – maven 1/2

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   |-- java
    |   |-- webapp
    |   |   `-- WEB-INF
    |   `-- resources
    |-- test
    |   |-- java
    |   `-- resources
```

Project structure – maven 2/2

```
|-- java
|   |-- com
|   |   |-- mycompany
|   |   |   |-- app
|   |   |   |   |-- components
|   |   |   |   |-- pages
|   |   |   |   |   |-- Index.java
|   |   |   |   |-- services
|   |   |   |   |   |-- AppModule.java
|-- resources
|   |-- com
|   |   |-- mycompany
|   |   |   |-- app
|   |   |   |   |-- components
|   |   |   |   |-- pages
|   |   |   |   |   |-- Index.properties
|-- webapp
|   |-- Index.html
|   |-- WEB-INF
|       |-- web.xml
```

MasterDispatcherService

- Chain of command
 - True when request was processed otherwise false
- Ordered configuration
- Could be used for
 - Access controller
 - Redirecting
 - Content serving

```
boolean dispatch(Request request, Response response) throws IOException;
```

RequestHandle

- Generic changes in request / response
 - Set correct page encoding
 - Adjust page request/response
 - Closing hibernate

```
public static void contributeRequestHandler(  
    OrderedConfiguration<RequestFilter> configuration,  
    @InjectService("hibernateFilter") RequestFilter hibernateFilter) {  
    configuration.add("hibernateFilter", hibernateFilter,  
        "before:StaticFiles");  
}
```

Page 1/5 – java class

- Ordinary java class
 - Inject, injectService
- Per request related data
 - persist
- Project structure
- Files tml, properties, class
 - Loaded
 - Attached – to thread
 - Detached

Page 2/5 - properties

- Encoding UTF8
- Localization of page texts
- Extension properties

```
msgAccountInactive="Account is inactive!"
msgIdentityAlreadyLoggedIn=Identity is already logged in!
msgRetrievingArgumentError="No arguments."
# global regular expressions
email-regexp = ^\\s*[a-zA-Z0-9!#$%&'*/-?^_`{|}~-]+(?:\\. [a-zA-Z0-9!#$%&'*/-?^_`{|}~-]+)*$
email-regexp-message = Email is in wrong format.
textContactEmail-regexp=^\\s*[a-zA-Z0-9!#$%&'*/-?^_`{|}~-]+(?:\\. [a-zA-Z0-9!#$%&'*/-?^_`{|}~-]+)*$
firstName-regexp=^.{0,40}$
lastName-regexp=^.{0,40}$
street-regexp=^.{0,40}$
```

Page 3/5 Binding expressions

- Get values into components
- Build in bindings
 - Asset – images
 - Block – id of block
 - Component – id of component
 - Context – root relative path
 - Literal – string value
 - Message – messages from resources
 - Prop – property

```
<t:select t:id="category" t:model="categories" value="articleForm.category"/>
```

Page 4/5 - template

- Encoding UTF8
- Localization of page texts,
- Extension tml

```
<html t:type="layout" t:id="layout"  
      t:title="message:title" t:section="blogs"  
      xmlns:t="http://tapestry.apache.org/schema/tapestry_5_1_0.x  
      xmlns:p="tapestry:parameter">  
  <t:beaneditform t:id="form" t:object="user" >  
  </t:beaneditform>  
</html>
```

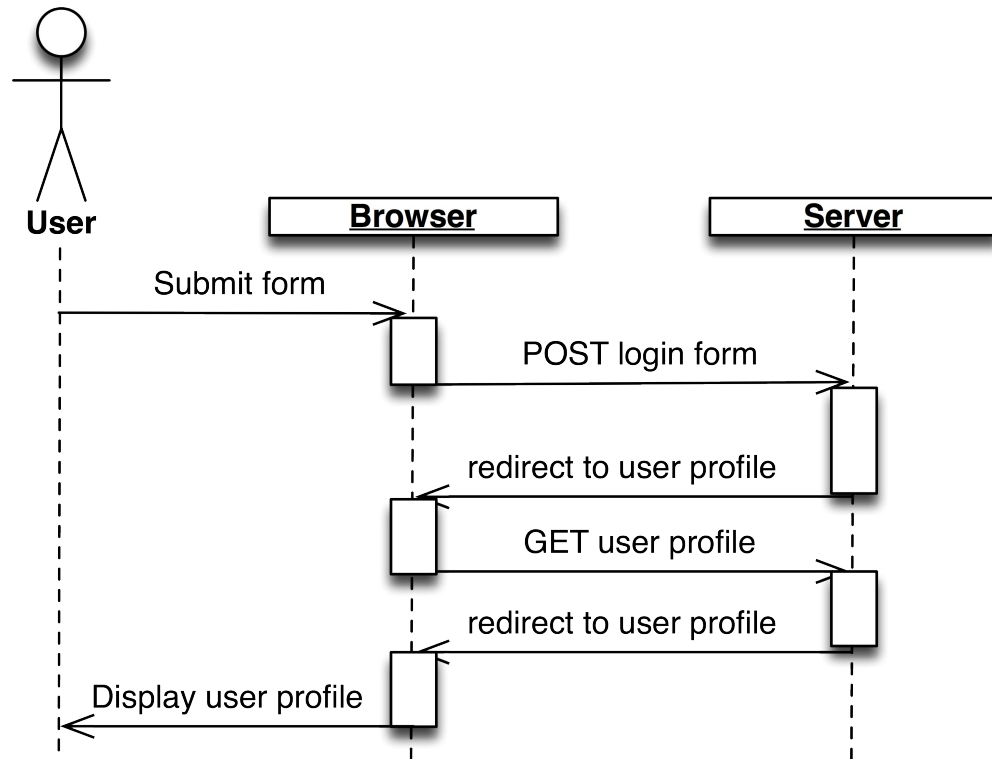
Expressions 5/5

- `${car.name}` → `getCar().getName()`
- Default binding “prop:”
- `{message:caption}`
- Output is formatted
- `<t:outputRaw value="car.name" />`

```
<t:if test="userSigned">
  <t:if test="userHaveApiKey">
    <t:pagelink page="blog/api" context="{blog.webName}">Your API key</t:pagelink>
  <p:else>
    <t:actionLink t:id="generateApiKey">Generate API key</t:actionLink>
  </p:else>
</t:if>
<p:else>
  <t:pagelink page="registration">Register to get you own API key</t:pagelink>
</p:else>
</t:if>
```

Redirect after post

- Prevent multiple submissions
- Separate action and rendering
- Nice URL
- Back button



Activation context

- onActivate / onPassivate method
- Persist page state
- Pass parameters between pages
 - /example/foo.bar:magic/99

```
void onActivate(long productId)
{
    product = productDAO.getById(productId);
}

long onPassivate() { return product.getId(); }
```

Component

- Similar to page objects
 - Template is in resources

```
⊖ @SuppressWarnings("unused")
    @Component(parameters = { "encoder=prop:categoryEncoder",
                             "blankOption=never" })
    private Select category;
```

```
<t:select t:id="category" t:model="categories" value="articleForm.category"/>
```

Component events

- Naming convention
- Return values from form submission
 - Null
 - String
 - Class
 - Page object
 - Link
 - Stream
 - URL
 - Object
- Can throw any exception
- Multiple method matches
- Event context

Events heritage

- Page contains component
- Event bubble through component hierarchy
- Comments component
 - Comments contains event “onSuccess”
 - Events are not terminated in component
 - In page could be used “onSuccessFromComments”

Grid 1/2

1 2 3 4 5 6 7 8 9 10 11 ... 31 32

<u>Id</u> ⌵	<u>Email</u> ⌵	<u>User Name</u> ⌵	<u>App Right</u> ⌵	Edit
1	qqq@qqq.qq	qqq	User	Detail
2	www@www.ww	www	User	Detail
3	sss@sss.ss	sss	User	Detail

- Grid component
 - GridDataSource
 - Model
 - Row

```
<t:grid source="source" row="user" model="model" exclude="idUser, birthDateYear, birthDateMonth,
    birthDateDay, password, password2, loginName, publicProfile, sessionId, lastLogin,
    registrationDate, birthDate">
    <t:parameter name="editCell">
        <t:pagelink page="admin/AdminUser" context="user.idUser">Detail</t:pagelink>
    </t:parameter>
</t:grid>
```

more

Ta

Grid 2/2

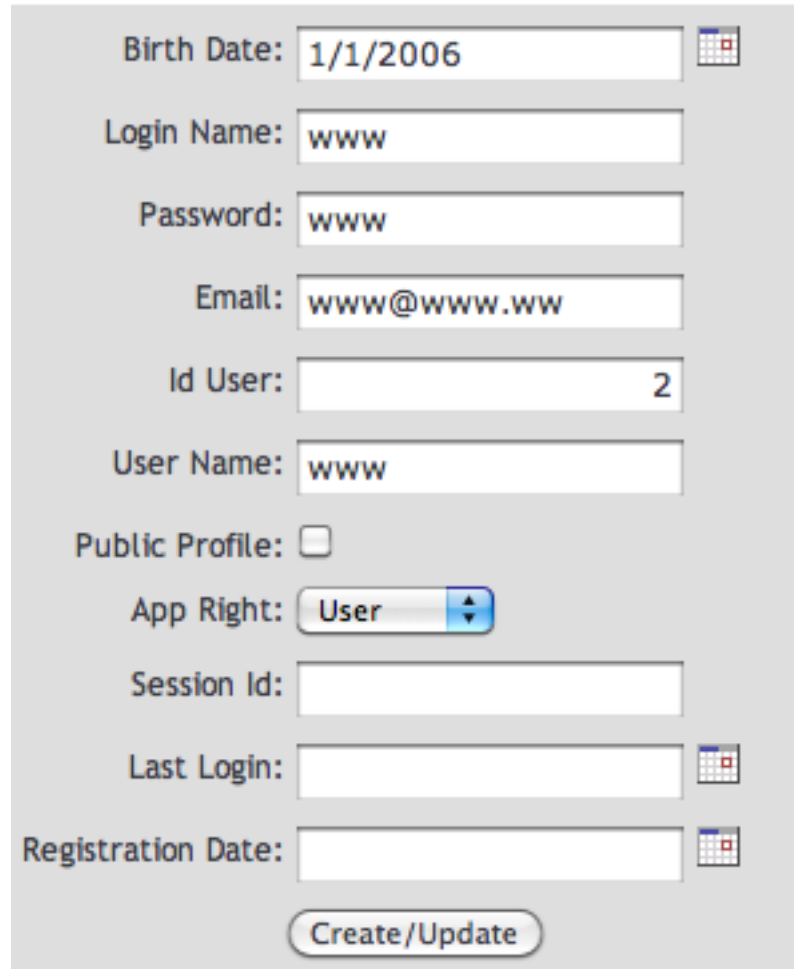
- GridDataSource
 - Support pagination
 - Get total number of records
 - Get limited list of records
- Model
 - Define table structure, columns

```
@SuppressWarnings("unused")  
@Inject  
@Service("userGds")  
@Property  
private GridDataSource source;
```

```
public BeanModel getModel() {  
    BeanModel model = beanModelSource.createDisplayModel(User.class,  
        resources.getMessage());  
    model.add("edit", null);  
    return model;  
}
```

BeanEditForm 1/2

- Generate bean editing form
 - String
 - Boolean
 - Date
 - Integer
 - Float
 - Enumeration
 - User can contribute
- UI is generated



Birth Date: 1/1/2006

Login Name: www

Password: www

Email: www@www.ww

Id User: 2

User Name: www

Public Profile:

App Right: User

Session Id:

Last Login:

Registration Date:

Create/Update

BeanEditForm 2/2

- Page context contains entity id

```
Object onActivate(final Integer idUsers) {  
    this.idUsers = idUsers;  
    user = serviceLocator.getUserService().get(idUsers);  
    if (user == null) {  
        return new ErrorCode(404, null);  
    }  
    return null;  
}
```

- Publish property

```
@Property  
private User user;
```

- Page

```
<t:beaneditform t:id="form" t:object="user" >  
</t:beaneditform>
```

Edit block contribution

- Edit user provided type
- Contribute default data type analyzer
 - Allows to identify bean field type
- Contribute bean block source
 - Which component handle which data type
- Edit component
 - Could extends already existing “Select”
 - Could define new

Form validation 1/2

- Defined
 - Bean annotations
 - Page template
- Event validateForm
 - “success”
 - “fails”
- ValidationTracker collect validation problems
- Message are from properties file
 - FormId-fieldId-validatorName-message.
- Validators
 - Email, max/min (integer value), maxLength, minLength, regexp, required

Form validation 2/2

- LoginForm
 - Fields
 - Annotated
- On validateForm event
 - Allows run non standard validation
 - Can adjust validation tracker from form object
- On success event
 - Page navigation
 - Object persisting
- Regexp definition
 - form-email-regexp=`^[a-zA-Z0-9._-]+@[a-zA-Z0-9-]+\.[a-zA-Z]{2,5}$`

```
@Property  
@Persist  
@Validate("required,regexp")  
private String email;
```

```
@Property  
@Validate("required")  
private String password;
```

Field persistence

- Page is cleaned
- Persist annotation
- Storing strategy
 - Session – server HTTP session, default value
 - Client – URL rewriting
 - Flash – server HTTP session one usage
- Strategy hierarchy
- Clustering
- Immutable object

```
⊖ @SuppressWarnings("unused")  
  @Persist("client")  
  private int pageNo;
```

Testing 1/2

- Junit / testNG

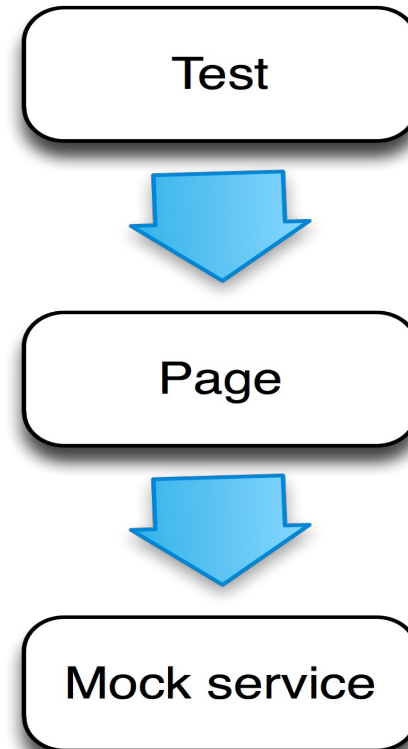
- ```
String appPackage = "org.example.app";
String appName = "LocaleApp";
PageTester tester =
 new PageTester(appPackage, appName,
 "src/main/webapp");
Document doc = tester.renderPage("MyPage");
AssertEquals(
 doc.getElementById("id1").getChildText(),
 "hello");
```

- Test

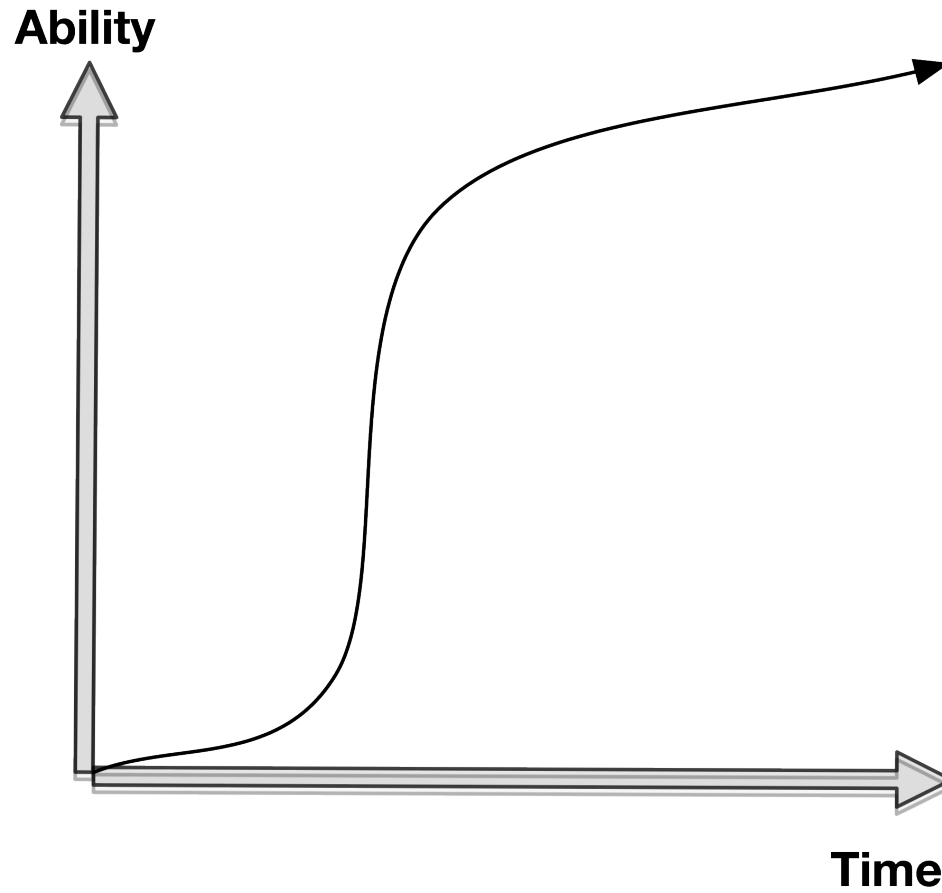
- Submit form
- Form validation
- Action links

# Testing 2/2

- Service mocking – EasyMock
- Pattern usage
  - Record behavior
  - Execute page call
  - Verify service calls



# Learning curve



# Tapestry 5 – Why not?

- Complexity
- Learning curve
- Howard Lewis Ship
  - Inconsistent changes
  - Unsure progress
- Community
- Job opportunities

# Resources

- <http://tapestry.apache.org>
- <http://www.infoq.com/articles/tapestry5-intro>
- <http://tapestryjava.blogspot.com/>
- <http://www.chenillekit.org/>
- <http://wiki.apache.org/tapestry/Tapestry5HowTos>
- <http://www.slideshare.net/mraible/comparing-jsf-spring-mvc-stripes-struts-2-tapestry-and-wicket-presentation>

# Questions ?



*apache*  
**tapestry 5**  
*Code less, deliver more.*



# indra

Jan Jirout  
Senior Architect/SW Engineer  
TEL Praha (OS)  
[jjirout@indracompany.com](mailto:jjirout@indracompany.com)

INDRA Czech Republic s.r.o  
Karolinská 650/1  
186 00 Praha 8  
T +420 246 085 700  
F +420 246 085 701  
[www.indra.es](http://www.indra.es)

