PDFDoclet Test PDF

Javadoc

Marcel Schön
Guggenbühlstr. 26
8953 Dietikon
Switzerland

marcelschoen@users.sourceforge.net

2.1 Ausgangslage

IT-RSQ besitzt als IT-Unternehmen keinen eigenen Webauftritt. Die Administration hat viele einfache telefonische Anfragen zu Kursangeboten, Offerten und weiteren Angeboten zu bewältigen.

2.2 Aufgabenstellung

Aus Image-Gründen ist ein Internetauftritt für eine IT-Firma unabdingbar.
Zunächst ist eine Vorstudie mit groben Designvorschlägen zum Webauftritt zu erarbeiten. Auf dieser Grundlage wird die Design-Spezifikation und ein Prototyp erstellt. Nach der Realisierung und der Live-Schaltung soll der Web-Content durch die Geschäftsleitung selbstständig unterhalten werden.

3. Abgrenzung

3.1 Der Inhalt der Website umfasst:

- detailliertes Kursangebot
- Online-Kursanmeldung mit tagesaktuellen Informationen zu freien Plätzen
- Bestellung von Unterlagen, Beratung und Offertanfragen
- Information über die Firma IT-RSQ GmbH
- Referentengalerie
- virtueller Rundgang durch die Kursräume
- Links zu Partnerorganisationen

3.2 Folgendes wird von der Website nicht abgedeckt:

- Online-Beratungswizard
- Online-Offerten

Der Webauftritt muss auch offline für Kunden ohne Internetanschluss auf CD/DVD verfügbar sein.

4. Ziele

- Reduktion der telefonischen Anfragen innert 3 Monaten nach Einführung um 30 %
- Erhöhung des Bekanntheitsgrades (1500 Page-Klicks pro Monat)
- Vergrösserung des Kundenstamms um 20 % innerhalb eines Jahres
- Nach 3 Monaten sollen 20 % aller Kursanmeldungen online erfolgen

5. Wirtschaftlichkeit und Budget

Für die Realisierung einer Lösung steht ein Budget von maximal CHF 65'000 zur Verfügung. Die Kosten für die Vorstudie und den Prototyp dürfen CHF 10'000 nicht übersteigen. Der "return on investment" soll in 2 Jahren erreicht sein.

8. Projekt-Organisation

- Organisationsform "Matrix"
- Auftraggeber ist Niklaus, CEO von IT-RSQ
- Der Projektausschuss setzt sich zusammen auf der Geschäftsleitung und dem Projektleiter
- Der Projektleiter ist Patrik
- Der Bedarf an internen und externen Ressourcen ist im Rahmen der Vorstudie abzuklären

9. Dokumentation und Kommunikation

- Die Projektdokumentation gliedert sich in Planung, Protokolle, monatliche Berichterstattung über Projektfortschritt und lösungsbezogene Dokumentation
- Art, Häufigkeit, Teilnehmer und Ziele der notwendigen Projektmeetings werden in der Vorstudie festgelegt.

10. Risiken

- Keine geeignete Lösung im Rahmen des Projektbudgets
- Häufige grosse Designänderungen können den Einführungstermin verzögern
- Technische Überforderung der Administratoren nach Einführung

11. Auswirkungen

11.1 Bei Realisierung

- Entlastung der Administration ➔ Kosteneinsparung
- Steigerung des Bekanntheitsgrades

11.2 Bei Nichtrealisierung

- Image-Verlust
- zusätzliches Personal in der Administration (mittelfristig)

12. Auftraggeber

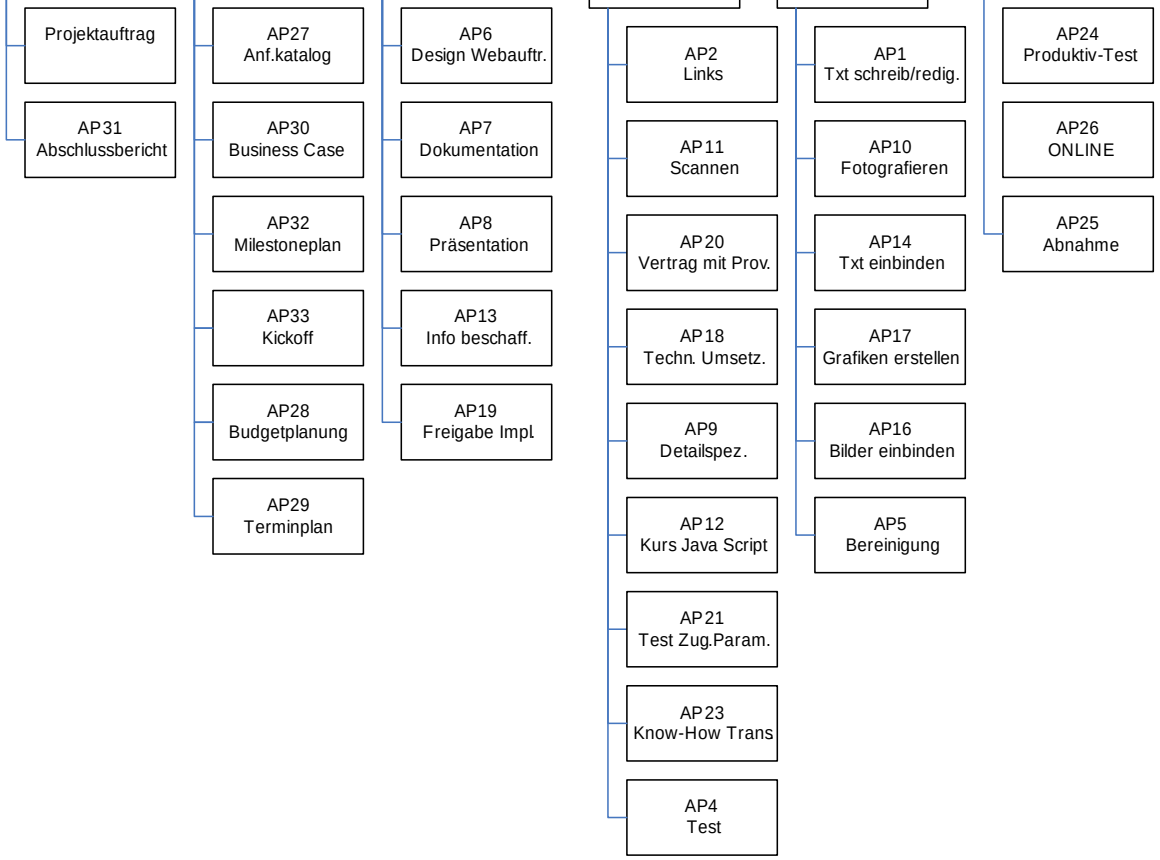
Niklaus, CEO der IT-RSQ GmbH

13. Antrag

Antrag auf Freigabe der Phase Initialisierung und der dazu benötigten Ressourcen.

14. Unterschriften

Projektleiter:	Auftraggeber:
.....	.....



marco.buergisser@ubs.com, adrian.glaettli@ubs.com,
christian.kohnle@ubs.com, marcel.schoen@tetrade.ch

My Overview

An overview of the JLaby

and an image

(continued on next page)

A definition list:

This is the first term
: This is some text belonging to that term.

This is the second term, bold
: This is some text belonging to that term.

This is the third, strong term
: This is some text belonging to that term.

A simple bullet list:

- The first element
- The second element

Now an ordered list with unclosed list tags:

1. The first element
2. The second element

And a HR line

Then some nested lists:

- Outer unordered list, first element
  1. Nested ordered list, first element
  2. Nested ordered list, some **bold text** in the second element
- And now a nested ordered list:
  1. Nested ordered list, this link points to LabyAction
  2. Nested ordered list, and an external link to the Yahoo website.

Then another HR line

---

1. First entry is nested bullet list
   - Bullet point one
   - Bullet point two
2. Second entry is nested ordered list
   - Number one
   - Number two

(continued on next page)

Now, this is a link to anchor one, while this is an invalid link to a non-existent anchor. Because the target anchor does not exist, the link leads to the last page of the document instead.

Now ordered lists of different type:

- Type "a"
  a. Entry one
  b. Entry two
- Type "i"
  i. Entry one
  ii. Entry two
- Type "l"
  1. Entry one
  2. Entry two

Try a member method link: `com.test.TestClass.getFormattedText()`
Try a member variable link: `com.test.ITestInterface.WEST`

Anchor one

Here, a page break is enforced by inserting a specific comment into the HTML code.

(continued on next page)

Table support

Coloured, right aligned table

header 1	header 2
cell 1	cell 2
spanning cell	
A nested table:	

inner cell 1	inner cell 2	inner cell 3
inner cell 1	AND A YELLOW CELL	inner cell 3
inner cell 1	inner cell 2	inner cell 3
inner cell with list: • one • two	inner cell 2	inner cell 3

A very large, page-breaking table

Note that the header cells are repeated automatically on a new page.

header 1	header 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2

(continued on next page)

header 1	header 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2

(continued on next page)

header 1	header 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2
cell 1	cell 2

(continued on next page)

Link Examples

A link to a package.

A link to an appendix file..

A link to a PDF appendix file..

A link to another PDF appendix file..

Some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here.

Heading 1

Some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here.

Heading 2

Some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here.

Heading 3

Some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here.

Heading 4

Some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here.

Heading 5

(continued on next page)

Some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here, some regular body text here.

(continued on next page)

Preformatted Text

A link to a package, and then comes a preformatted text region:

```
String permission = ...
ConfigurationProvider cp = <ConfigurationProvider>
UserDataProvider up = <UserDataProvider>
SecurityConfiguration sc = SecurityConfiguration.getInstance(permission, cp, up);
Security security = SecurityFactory.getInstance(sc)
Ticket ticket = security.getTicket(<Permission(permission)>);
```

And more text...

And now an inline link to class `com.test.TestClass` and an inline <a href> Yahoo web URL link.

Font modifiers: *emphasized*, ~~strike~~, ~~strike again~~, **bold**, underline, *italic*, `code`, ***italic bold***, and mixed up: **bold and *italic* is pretty cool, man.**

And special characters: ä, ü, ö, Ä, Ü, Ö, ß, &, ", >, <

Preformatted Text With Font Modifiers

```
String permission = ...
ConfigurationProvider cp = <ConfigurationProvider>
UserDataProvider up = <UserDataProvider>
SecurityConfiguration sc = SecurityConfiguration.getInstance(permission, cp, up);
Security security = SecurityFactory.getInstance(sc)
Ticket ticket = security.getTicket(<Permission(permission)>);
```

Package
com.dummy

com.dummy
Class Dummy

```
java.lang.Object
   |
   +-com.dummy.Dummy
```

---

public class **Dummy**
extends Object

This class exists only to fill the "stuff" package with something.

aBcDeF:
Mixed case tag

ABCDEF:
Upper case tag

Constructor Summary

public	Dummy() Creates an instance of this dummy class.

Method Summary

static String[]	getSomething(ITestInterface receiver, Object[] args) Gets some dummy text.

Constructors

Dummy

```
public Dummy()
```

Creates an instance of this dummy class.

Methods

getSomething

```
public static String[] getSomething(ITestInterface receiver,
        Object[] args)
```

Gets some dummy text.

Parameters:

`receiver` - This is an absolutely useless parameter.
`args` - And these are the even more useless arguments.

Returns:

The most useless text strings.

Package
com.other

Some package description blah blah.

Here comes a link to an anchor.

This package is divided into three categories:

1. Classes for the netbanks
2. Classes for the content providers
3. Shared classes

A dummy table to fill the page:

X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y

X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y

And this is the anchor.

com.other
Interface IOneInterface

All Subinterfaces:
ISomeInterface

---

public interface **IOneInterface**
extends

Interface one.

Method Summary

String	doIt(String name) Do something.

Methods

doIt

```
public String doIt(String name)
```

Do something.

Parameters:
`name` - The name value.

Returns:
The new value.

com.other
Interface ISomeInterface

All Superinterfaces:
ITwoInterface

---

public interface **ISomeInterface**
extends IOneInterface, ITwoInterface

This interface extends TWO other interfaces.

Methods inherited from interface com.other.IOneInterface
doIt

Methods inherited from interface com.other.ITwoInterface
doItNow

---

com.other
Interface ISubInterface

All Superinterfaces:
ITestInterface

All Subinterfaces:
IThirdInterface

---

public interface **ISubInterface**
extends ITestInterface

This is a sub-interface of another interface.

Fields inherited from interface com.test.ITestInterface
EAST, NORTH, SOUTH, WEST

Method Summary

boolean	isAnything() Some method for something.

Methods inherited from interface com.test.ITestInterface
handleAction, initialize

Methods

isAnything

```
public boolean isAnything()
```

Some method for something.

Returns:

true if there is anything.

com.other
Interface IThirdInterface

All Superinterfaces:
ISubInterface, ITestInterface

All Known Implementing Classes:
OtherClass

---

public interface **IThirdInterface**
extends ISubInterface

This is a sub-interface of another interface.

Fields inherited from interface com.test.ITestInterface
EAST, NORTH, SOUTH, WEST

Method Summary

boolean	isAnnyoing() And a method again.

Methods inherited from interface com.other.ISubInterface
isAnything

Methods inherited from interface com.test.ITestInterface
handleAction, initialize

Methods

isAnnyoing

```
public boolean isAnnyoing()
```

And a method again.

Returns:

true if it is annyoing

com.other
Interface ITwoInterface

All Subinterfaces:
ISomeInterface

---

public interface **ITwoInterface**
extends

Interface two.

Method Summary

int	doItNow(String name) Do something.

Methods

doItNow

```
public int doItNow(String name)
```

Do something.

Parameters:
`name` - The name value.

Returns:
The new value.

com.other
Class OtherClass

```
java.lang.Object
   ┗-com.test.TestClass
      ┗-com.test.SubClass
         ┗-com.other.OtherClass
```

All Implemented Interfaces:
IThirdInterface, ISecondInterface, ITestInterface

---

public class **OtherClass**
extends SubClass
implements ITestInterface, ISecondInterface, IThirdInterface

Blah blah blah blah...

Fields inherited from class com.test.TestClass
SOME_CONSTANT

Fields inherited from interface com.test.ITestInterface
EAST, NORTH, SOUTH, WEST

Fields inherited from interface com.test.ISecondInterface
ANORTH, BSSOUTH

Fields inherited from interface com.test.ITestInterface
EAST, NORTH, SOUTH, WEST

Constructor Summary

public	OtherClass() Creates new other class, SubClass also helps.

Method Summary

ISubInterface	doAction(IThirdInterface someValue) This method takes a parameter of a type which is part of this API.
void	inheritsDocMethod() This is the otherclass method.
boolean	isAnnyoing()
void	someOtherMethod(String label) This is another method for something.
void	useFourParms(String one, int two, Object three, char four)

Methods inherited from class com.test.SubClass

inheritsDocMethod, sayHello, useFourParms

Methods inherited from class com.test.TestClass

aTestMethod, doAnyhing, doit, doit, doit, doit, doNothing, getFormattedText, getNames, inheritsDocMethod, no, sayHello, sayHello, sayHello, sayHello, sayNothing

Methods inherited from interface com.test.ITestInterface

handleAction, initialize

Methods inherited from interface com.test.ISecondInterface

doNothing

Methods inherited from interface com.other.IThirdInterface

isAnnyoing

Methods inherited from interface com.other.ISubInterface
isAnything

Methods inherited from interface com.test.ITestInterface
handleAction, initialize

Constructors

OtherClass

```
public OtherClass()
```

Creates new other class, SubClass also helps.

See Also:

TestClass.sayHello(long)
ITestInterface.initialize()
ITestInterface
ITestInterface

Methods

isAnnyoing

```
public boolean isAnnyoing()
```

someOtherMethod

```
public void someOtherMethod(String label)
```

This is another method for something.

Parameters:

`label` - Says everything, doesn't it?

---

doAction

```
public ISubInterface doAction(IThirdInterface someValue)
```

This method takes a parameter of a type which is part of this API. That should create an internal link.

Parameters:

`someValue` - The parameter value.

Returns:

The return value.

---

useFourParms

```
public void useFourParms(String one,
         int two,
         Object three,
         char four)
```

This method uses four parameters.

---

inheritsDocMethod

```
public void inheritsDocMethod()
```

This is the otherclass method. The following paragraph shows the doc inherited from the SubClass.

This is the subclass method. The next paragraph shows the doc inherited from the TestClass.

This is the original text which will be inherited by the subclasses.

Package
com.single

com.single
Class SingleClass

```
java.lang.Object
   |
   +-com.single.SingleClass
```

---

public class **SingleClass**
extends Object

This is a poor, single class. It is used to test the PDFDoclet's ability to create doc for single classes, and not only for packages.

- Number one
- Number two

And then some text after the list. Now a nested list.

1. List one
    - James
    - Jones
2. Second
3. List three
    - James
    - Jones
4. Fourth

Field Summary

public static final	FIRST_FLAG The first flag. Value: **200**
public static final	SECOND_FLAG Yet another flag. Value: **200**

public static final	SOME_FLAG A public flag here. Value: **100**

Constructor Summary

public	SingleClass() Returns NOT a java.util.List with anything.

Method Summary

org.w3c.domElement	createElementNS(String namespaceURI, String qualifiedName) Creates an element of the given qualified name and namespace URI.

Fields

SOME_FLAG

`public static final int` **`SOME_FLAG`**

A public flag here.

- Number one
- Number two

And some text.
Constant value: **100**

SECOND_FLAG

`public static final int` **`SECOND_FLAG`**

Yet another flag.
Constant value: **200**

FIRST_FLAG

```
public static final int FIRST_FLAG
```

The first flag.
Constant value: **200**

Constructors

SingleClass

```
public SingleClass()
```

Returns NOT a java.util.List with anything. Default constructor. List:

- Number one
- Number two

And some text.

Methods

createElementNS

```
public org.w3c.domElement createElementNS(String namespaceURI,
        String qualifiedName)
  throws org.w3c.dom.DOMException
```

Creates an element of the given qualified name and namespace URI.

Parameters:

`namespaceURI` - The namespace URI of the element to create.
`qualifiedName` - The qualified name of the element type to instantiate.

Returns:

A new table:

Column one	Column two
Column one	Column two

Throws:

`DOMException`

Package
com.stuff

com.stuff
Class StuffClass

```
java.lang.Object
   |
   +-com.stuff.StuffClass
```

---

public class **StuffClass**
extends Object

This class exists only to fill the "stuff" package with something.

Field Summary

public static final	SOME_FLAG A public flag here. Value: **100**

Constructor Summary

public	StuffClass() Returns NOT a java.util.List with anything.

Fields

SOME_FLAG

```
public static final int SOME_FLAG
```

A public flag here.

- Number one
- Number two

And some text.
Constant value: **100**

Constructors

StuffClass

```
public StuffClass()
```

Returns NOT a java.util.List with anything. Default constructor. List:

- Number one
- Number two

And some text.

Package
com.test

Here comes a link to an anchor.

And another image:

And now a line break:
This is a new line

A dummy table to fill the page:

X	Y
X	Y
X	Y
X	Y

X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y
X	Y

And this is the anchor.

com.test
Class AnotherClass

```
java.lang.Object
   |
   +-com.test.AnotherClass
```

All Implemented Interfaces:
ITestInterface

---

public class **AnotherClass**
extends Object
implements ITestInterface

Fields inherited from interface com.test.ITestInterface
EAST, NORTH, SOUTH, WEST

Constructor Summary

public	AnotherClass()

Method Summary

Object	handleAction(Object action)
void	initialize()

Methods inherited from interface com.test.ITestInterface
handleAction, initialize

Constructors

AnotherClass

```
public AnotherClass()
```

Methods

initialize

```
public void initialize()
  throws RuntimeException
```

---

handleAction

```
public Object handleAction(Object action)
  throws Exception
```

com.test
Class CustomException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-com.test.CustomException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **CustomException**
extends Exception

This is a dummy exception.

Constructor Summary

public	CustomException()

Constructors

CustomException

public **CustomException**()

com.test
Interface ISecondInterface

All Known Implementing Classes:
TestClass

---

public interface **ISecondInterface**
extends

And the second interface.

Field Summary

public static final	ANORTH Constant value for northern direction. Value: **1**
public static final	BSSOUTH Constant value for southern direction. Value: **2**

Method Summary

void	doNothing() This method does something.

Fields

ANORTH

```
public static final int ANORTH
```

Constant value for northern direction.

Constant value: **1**

BSSOUTH

```
public static final int BSSOUTH
```

Constant value for southern direction.
Constant value: **2**

Methods

doNothing

```
public void doNothing()
```

This method does something.

com.test
Interface ITestInterface

All Subinterfaces:
ISubInterface, IThirdInterface

All Known Implementing Classes:
AnotherClass, TestClass

Deprecated. *use the new interface for now*

public interface **ITestInterface**
extends

Simple interface for pdfdoclet testing.

Field Summary

public static final	EAST **Deprecated.** Value: **3**
public static final	NORTH **Deprecated.** Value: **1**
public static final	SOUTH **Deprecated.** Value: **2**
public static final	WEST **Deprecated.** Value: **4**

Method Summary

Object	handleAction(Object action) **Deprecated.**
void	initialize() **Deprecated.**

Fields

NORTH

```
public static final int NORTH
```

Deprecated.

Constant value for northern direction.
Constant value: **1**

SOUTH

```
public static final int SOUTH
```

Deprecated.

Constant value for southern direction.
Constant value: **2**

EAST

```
public static final int EAST
```

Deprecated.

Constant value for eastern direction.
Constant value: **3**

WEST

```
public static final int WEST
```

Deprecated.

Constant value for western direction.
Constant value: **4**

Methods

initialize

```
public void initialize()
  throws RuntimeException
```

Deprecated.

Initializes the world (sounds nice, doesn't it?). In other words, it does in a few milliseconds what took god six days... so to speak...

Throws:

`RuntimeException` - if something went wrong and the world could not be created.

---

handleAction

```
public Object handleAction(Object action)
  throws Exception
```

Deprecated.

This method has to deal with an action created by a "client" (local or remote). It has to have the action handled by the appropriate handler and then create an answer to be returned to the client.

Parameters:

`action` - the action created /sent by a client.

Returns:

the answer for the client.

Throws:

`LabyException` - thrown if the action could not be handled.

com.test
Interface MyCriteria

public interface **MyCriteria**
extends

This is just some strange class.

Method Summary

boolean	`contains(Target target)` Method contains.

Methods

contains

`public boolean **contains**(Target target)`

Method contains. This method returns if the track is in the geographic criteria.

Parameters:

`target` - The object to test.

Returns:

The boolean is true if the filter contains the object.

com.test
Class SubClass

```
java.lang.Object
   |-com.test.TestClass
      |-com.test.SubClass
```

All Implemented Interfaces:
ISecondInterface, ITestInterface

Direct Known Subclasses:
OtherClass

---

public class **SubClass**
extends TestClass

This is a subclass. The following "see" tag refers to a method of a super class. This is quite a challenge. Here's an inline link first: `Die Testklasse` And then some more stuff.

Now we try a table with 3 rows and 3 columns...

Column 1	Column 2	Column 3
Row 1, Column 1	Row 1, Column 2	Row 1, Column 3
Row 2, Column 1	A nested table: FIELD ONE / AND TWO / AND THREE	
Row 3, some **bold** text	Row 2, a list • one • two as well	Row 2, a `link` to some class

See Also:
`TestClass.no()`, `TestClass.doit()`, `TestClass.doit(int, String)`, `TestClass.no()`, `TestClass.no()`

Fields inherited from class `com.test.TestClass`

SOME_CONSTANT

Fields inherited from interface com.test.ITestInterface

EAST, NORTH, SOUTH, WEST

Fields inherited from interface com.test.ISecondInterface

ANORTH, BSSOUTH

Constructor Summary

public	SubClass() Creates new TestClass

Method Summary

void	inheritsDocMethod() This is the subclass method.
ISecondInterface	sayHello(String name) Does the same as the overriden method (), but with a different prefix.
void	useFourParms(String one, int two, Object three, char four) This method uses four parameters.

Methods inherited from class com.test.TestClass

aTestMethod, doAnyhing, doit, doit, doit, doit, doNothing, getFormattedText, getNames, inheritsDocMethod, no, sayHello, sayHello, sayHello, sayHello, sayNothing

Methods inherited from interface com.test.ITestInterface

handleAction, initialize

Methods inherited from interface com.test.ISecondInterface

doNothing

Constructors

SubClass

`public` **`SubClass`**`()`

Creates new TestClass

Methods

sayHello

`public` `ISecondInterface` **`sayHello`**`(String name)`

Does the same as the overriden method (), but with a different prefix. Now we try a table with 3 rows and 3 columns...

Column 1	Column 2	Column 3
Row 1, Column 1	Row 1, Column 2	Row 1, Column 3
Row 2, Column 1	A nested table: FIELD ONE | AND TWO	
Row 3, some **bold** text	Row 2, a list • one • two as well	Row 2, a `link` to some class

Parameters:

`name` - Same as before ({@inheritDoc}) but with a totally different meaning

useFourParms

```
public void useFourParms(String one,
         int two,
         Object three,
         char four)
```

This method uses four parameters.

Parameters:

`one` - The first parameter
`two` - The second parameter
`three` - The third parameter
`four` - The fourth parameter

---

inheritsDocMethod

```
public void inheritsDocMethod()
```

This is the subclass method. The next paragraph shows the doc inherited from the TestClass.

This is the original text which will be inherited by the subclasses.

com.test
Class TestClass

```
java.lang.Object
   |
   +-com.test.TestClass
```

All Implemented Interfaces:
ISecondInterface, ITestInterface

Direct Known Subclasses:
SubClass

---

public class **TestClass**
extends Object
implements ITestInterface, ISecondInterface

This is a test class with an inner class. Now let's see a simple bulleted list:

- date/time
- severity (as string: `INFO/WARNING/ERROR/FATAL`)
- text which describes the severity

and an image:

And some preformatted text:

```
String permission = ...
 ConfigurationProvider cp = <ConfigurationProvider>
 Security security = SecurityFactory.getInstance(sc)
 Ticket ticket = security.getTicket(<Permission(permission)>);
```

See Also:
Andere Klasse, OtherKlasse, OtherClass, OtherClass.someOtherMethod(String), someOtherMethod(), someOtherMethod(label), OtherClass.someOtherMethod(String), OtherClass, sayHello(int), sayHello(String)

To Do:

This is a custom tag.

Cool stuff:

Or is it not?

Nested Class Summary

class	TestClass.AnotherInnerClass TestClass.AnotherInnerClass
class	TestClass.MyInnerClass TestClass.MyInnerClass

Field Summary

public static final	SOME_CONSTANT This is a constant for something. Value: **3**

Fields inherited from interface com.test.ITestInterface

EAST, NORTH, SOUTH, WEST

Fields inherited from interface com.test.ISecondInterface
ANORTH, BSSOUTH

Constructor Summary

public	TestClass() Creates new TestClass,
public	TestClass(String value) Another constructor.
public	TestClass(int value) Another constructor.
public	TestClass(int value, long otherValue) Another constructor.

Method Summary

ITestInterface	aTestMethod(ISecondInterface parm, IThirdInterface parmTwo) This is a method to test lots of things (printing of tags, linking of types etc.).
void	doAnyhing() Base-class method that will be overridden in the subclass.
void	doit() Some test method.
void	doit(int value, int value) Another method with a very short name.
void	doit(int value, int value, String value) Another method with a very short name.

void	doit(int value, String value) Another method with a very short name.
static void	doNothing(String[] arrayArgs) Lets see a simple bulleted list:
String	getFormattedText(String record) This method returns the text of the logrecord.
String[]	getNames() Lets see a simple bulleted list:
void	inheritsDocMethod() This is the original text which will be inherited by the subclasses.
void	no() A method with a very short name.
void	sayHello(int value) Another say hello method.
void	sayHello(long value) Another say hello method.
void	sayHello(Object value, Hashtable store) Another say hello method.
void	sayHello(String name) Says hello to someone.
void	sayNothing(String text) Does something new.

Methods inherited from interface com.test.ITestInterface

handleAction, initialize

Methods inherited from interface com.test.ISecondInterface

doNothing

Fields

SOME_CONSTANT

`public static final int` **`SOME_CONSTANT`**

This is a constant for something.
Constant value: **3**

Constructors

TestClass

`public` **`TestClass`**`()`

Creates new TestClass,

TestClass

`public` **`TestClass`**`(String value)`

Another constructor.

Parameters:
`value` - Some value.

TestClass

`public` **`TestClass`**`(int value)`

Another constructor.

Parameters:
`value` - Some value.

TestClass

```
public TestClass(int value,
                 long otherValue)
```

Another constructor.

Parameters:

`value` - Some value.
`otherValue` - some other value.

Methods

no

```
public void no()
```

A method with a very short name.

---

doit

```
public void doit()
```

Some test method.

---

doit

```
public void doit(int value,
         int value)
```

Another method with a very short name.

Parameters:

`value`
`value`

---

doit

```
public void doit(int value,
         int value,
         String value)
```

Another method with a very short name.

Parameters:

`value`
`value`
`value`

---

doit

```
public void doit(int value,
         String value)
```

Another method with a very short name.

Parameters:

`value`
`value`

---

doAnyhing

```
public void doAnyhing()
  throws RuntimeException,
         ClassCastException
```

Base-class method that will be overridden in the subclass.

Throws:

`IllegalArgumentException` - Blablabla
`RuntimeException` - This tests support for the 'exception' tag.
`ClassCastException` - If there was an invalid class.

---

getNames

```
public String[] getNames()
  throws RuntimeException,
         IllegalStateException
```

Lets see a simple bulleted list:

- First item here
- Second item

And then some more text.

Returns:

An array with all names or null.

Throws:

`RuntimeException` - Sometimes if something failed.
`IllegalStateException` - If it feels like it.

---

doNothing

```
public static void doNothing(String[] arrayArgs)
```

Lets see a simple bulleted list:

- First item here
- Second item

And then some more text.

Parameters:

`arrayArgs` - This is an array of arguments, doh!

---

sayHello

```
public void sayHello(String name)
```

Says hello to someone.

Parameters:

`name` - The name

---

sayHello

```
public void sayHello(int value)
```

Another say hello method.

Parameters:

`value` - Some value.

---

sayHello

```
public void sayHello(long value)
```

Another say hello method.

Parameters:

`value` - Some value.

---

sayHello

```
public void sayHello(Object value,
         Hashtable store)
```

Another say hello method.

Parameters:

`value` - Some value.
`store` - A hashtable for something.

---

getFormattedText

```
protected String getFormattedText(String record)
```

This method returns the text of the logrecord.

Parameters:

`record` - the String containing the log information

Returns:

the text string of the log record

---

aTestMethod

```
public ITestInterface aTestMethod(ISecondInterface parm,
                                  IThirdInterface parmTwo)
  throws CustomException,
         RuntimeException
```

This is a method to test lots of things (printing of tags, linking of types etc.).

Parameters:

`parm` - The first parameter.
`parmTwo` - The second parameter.

Returns:

Something of a certain type.

Throws:

`CustomException` - The is our own exception.
`RuntimeException` - The is a very ugly exception.

---

sayNothing

```
public void sayNothing(String text)
```

Does something new.

Parameters:

`text` - The text to say.

---

inheritsDocMethod

```
public void inheritsDocMethod()
```

This is the original text which will be inherited by the subclasses.

com.test
Class TestClass.MyInnerClass

```
java.lang.Object
   |
   +-com.test.TestClass.MyInnerClass
```

---

public class **TestClass.MyInnerClass**
extends Object

Some inner class for anything.

Nested Class Summary

class	TestClass.MyInnerClass.MyNestedInnerClass TestClass.MyInnerClass.MyNestedInnerClass

Constructor Summary

public	TestClass.MyInnerClass()

Method Summary

int	getValue(String key) Method which returns some numeric value based on some String key.

Constructors

TestClass.MyInnerClass

public **TestClass.MyInnerClass**()

Methods

getValue

```
public int getValue(String key)
```

Method which returns some numeric value based on some String key.

Parameters:

`key` - The key string.

Returns:

The numeric value.

com.test
Class TestClass.MyInnerClass.MyNestedInnerClass

```
java.lang.Object
   |
   +-com.test.TestClass.MyInnerClass.MyNestedInnerClass
```

---

public class **TestClass.MyInnerClass.MyNestedInnerClass**
extends Object

Not that this is often used. But theoretically, inner classes can even be nested into other inner classes..

Constructor Summary

public	TestClass.MyInnerClass.MyNestedInnerClass()

Method Summary

String	getSomeValue() This method returns a text value.

Constructors

TestClass.MyInnerClass.MyNestedInnerClass

public **TestClass.MyInnerClass.MyNestedInnerClass**()

Methods

getSomeValue

```
public String getSomeValue()
```

This method returns a text value.

Returns:

The text value or null.

com.test
Class TestClass.AnotherInnerClass

```
java.lang.Object
   |
   +-com.test.TestClass.AnotherInnerClass
```

---

public class **TestClass.AnotherInnerClass**
extends Object

Another inner class for testing how a second inner class is handled.

Constructor Summary

public	TestClass.AnotherInnerClass()

Method Summary

int	getAnotherValue(String key) Method which returns some numeric value based on some String key.

Constructors

TestClass.AnotherInnerClass

```
public TestClass.AnotherInnerClass()
```

Methods

getAnotherValue

```
public int getAnotherValue(String key)
```

Method which returns some numeric value based on some String key.

Parameters:

`key` - The key string.

Returns:

The numeric value.

	Pixelabstand	0.264 mm (H x V)
	Kontrastverhältnis	600 : 1 (max.)
	Reaktionszeit	12 ms (typisch)
	Helligkeit	300 cd/m² (typisch)
Eingangssignal	**Video**	R.G.B Analog / Digital Signal (DVI-D, inkl. DVI-Kabel)
	Synchronisation	H/V - Separat / TMDS Signal
	Anschluß	15 pin D-sub / DVI-D Anschluß
Bildwiederholrate	**horizontal**	31 ~ 80 kHz
	vertikal	56 ~ 75 Hz
Auflösung	**max.**	1280 x 1024 @ 75Hz
	empfohlen	1280 x 1024 @ 60Hz
Aktive Display Größe		338 x 270 mm
Farben		16.7 Mio.
Plug-and-Play		VESA DDC 1/2B
Voreingestellte Auflösungen	**Analog**	720 x 400 @ 70 Hz 640 x 480 @ 60/75 Hz 800 x 600 @ 56/60/72/75 Hz 1024 x 768 @ 60/70/75 Hz 1280 x 1024 @ 60/75 Hz
	Digital	640 x 480 @ 60 Hz 800 x 600 @ 60 Hz 1024 x 768 @ 60 Hz 1280 x 1024 @ 60 Hz
Einstellmöglichkeiten	**Analog**	Helligkeit, Kontrast, H/V-Position, Taktrate, Farbton Farbeinstellungen, Sprache, Miscellaneous, Auto Adjust
	Digital	Helligkeit, Farben, Sprache, OSD Adjust, Eingangssignal-Auswahl, Farbton, Flesh Tone
Power Management		Energy Star/VESA DPMS/NUTEK
Stronmversorgung	**Quelle**	100 - 240 V AC
	Verbrauch (max.)	Betrieb: < 40 Watt / Standby: ≤ 1 Watt
Netzteil		intern
Ergonomie Standards		TCO 03
Zertifizierungen		cULus,TUV-GS SEMKO,FCC Class B,CE,EN55022-B, VCCI
Weitere Funktionen	**USB-Hub**	USB 2.0 / 1up x 2down / inkl. USB-Kabel
	Audio System	2 x 2 Watt max., Kopfhöreranschluß
	VESA Standard	kompatibel für Schwenkarm / Wandhalterung (100 x 100 mm)
	Kensington-ready	Kensington-Sicherheits-Slot auf der Geräterückseite
	Höhenverstellbar	höhenverstellbarer und faltbarer Standfuß
Maße (B x H x T)	**Netto**	396 x 414 x 200 mm
	Verpackt	470 x 370 x 173 mm
Gewicht	**Netto**	4.8 kg
	Verpackt	6.0 kg
Garantie	**Garantiedauer**	60 Monate
	Service Level	Vor-Ort-Service

Technische Änderungen und Irrtümer vorbehalten.

HYUNDAI ImageQuest
Europe GmbH

Geheimrat-Hummel-Platz 2
D-65239 Hochheim
Germany

URL: www.hyundaiQ.de
Email: info@hyundaiQ.de

Telefon: +49 (0)6146-904-0
Fax: +49 (0)6146-904-410
Service Hotline: 00800-49863247

V.1.0 / September 2004

Appendix B: Some Stuff

Additional Stuff

This HTML file contains some meaningless blah blah which only serves as an example for using appendixes. This HTML file contains some meaningless blah blah which only serves as an example for using appendixes. This HTML file contains some meaningless blah blah which only serves as an example for using appendixes.

This HTML file contains some meaningless blah blah which only serves as an example for using appendixes. This HTML file contains some meaningless blah blah which only serves as an example for using appendixes. This HTML file contains some meaningless blah blah which only serves as an example for using appendixes.

This HTML file contains some meaningless blah blah which only serves as an example for using appendixes. This HTML file contains some meaningless blah blah which only serves as an example for using appendixes. This HTML file contains some meaningless blah blah which only serves as an example for using appendixes.

This HTML file contains some meaningless blah blah which only serves as an example for using appendixes. This HTML file contains some meaningless blah blah which only serves as an example for using appendixes. This HTML file contains some meaningless blah blah which only serves as an example for using appendixes.

This HTML file contains some meaningless blah blah which only serves as an example for using appendixes. This HTML file contains some meaningless blah blah which only serves as an example for using appendixes. This HTML file contains some meaningless blah blah which only serves as an example for using appendixes.

This HTML file contains some meaningless blah blah which only serves as an example for using appendixes. This HTML file contains some meaningless blah blah which only serves as an example for using appendixes. This HTML file contains some meaningless blah blah which only serves as an example for using appendixes.

2.1 Ausgangslage

IT-RSQ besitzt als IT-Unternehmen keinen eigenen Webauftritt. Die Administration hat viele einfache telefonische Anfragen zu Kursangeboten, Offerten und weiteren Angeboten zu bewältigen.

2.2 Aufgabenstellung

Aus Image-Gründen ist ein Internetauftritt für eine IT-Firma unabdingbar.
Zunächst ist eine Vorstudie mit groben Designvorschlägen zum Webauftritt zu erarbeiten. Auf dieser Grundlage wird die Design-Spezifikation und ein Prototyp erstellt. Nach der Realisierung und der Live-Schaltung soll der Web-Content durch die Geschäftsleitung selbstständig unterhalten werden.

3. Abgrenzung

3.1 Der Inhalt der Website umfasst:

- detailliertes Kursangebot
- Online-Kursanmeldung mit tagesaktuellen Informationen zu freien Plätzen
- Bestellung von Unterlagen, Beratung und Offertanfragen
- Information über die Firma IT-RSQ GmbH
- Referentengalerie
- virtueller Rundgang durch die Kursräume
- Links zu Partnerorganisationen

3.2 Folgendes wird von der Website nicht abgedeckt:

- Online-Beratungswizard
- Online-Offerten

Der Webauftritt muss auch offline für Kunden ohne Internetanschluss auf CD/DVD verfügbar sein.

4. Ziele

- Reduktion der telefonischen Anfragen innert 3 Monaten nach Einführung um 30 %
- Erhöhung des Bekanntheitsgrades (1500 Page-Klicks pro Monat)
- Vergrösserung des Kundenstamms um 20 % innerhalb eines Jahres
- Nach 3 Monaten sollen 20 % aller Kursanmeldungen online erfolgen

5. Wirtschaftlichkeit und Budget

Für die Realisierung einer Lösung steht ein Budget von maximal CHF 65'000 zur Verfügung. Die Kosten für die Vorstudie und den Prototyp dürfen CHF 10'000 nicht übersteigen. Der "return on investment" soll in 2 Jahren erreicht sein.

8. Projekt-Organisation

- Organisationsform "Matrix"
- Auftraggeber ist Niklaus, CEO von IT-RSQ
- Der Projektausschuss setzt sich zusammen auf der Geschäftsleitung und dem Projektleiter
- Der Projektleiter ist Patrik
- Der Bedarf an internen und externen Ressourcen ist im Rahmen der Vorstudie abzuklären

9. Dokumentation und Kommunikation

- Die Projektdokumentation gliedert sich in Planung, Protokolle, monatliche Berichterstattung über Projektfortschritt und lösungsbezogene Dokumentation
- Art, Häufigkeit, Teilnehmer und Ziele der notwendigen Projektmeetings werden in der Vorstudie festgelegt.

10. Risiken

- Keine geeignete Lösung im Rahmen des Projektbudgets
- Häufige grosse Designänderungen können den Einführungstermin verzögern
- Technische Überforderung der Administratoren nach Einführung

11. Auswirkungen

11.1 Bei Realisierung

- Entlastung der Administration ➔ Kosteneinsparung
- Steigerung des Bekanntheitsgrades

11.2 Bei Nichtrealisierung

- Image-Verlust
- zusätzliches Personal in der Administration (mittelfristig)

12. Auftraggeber

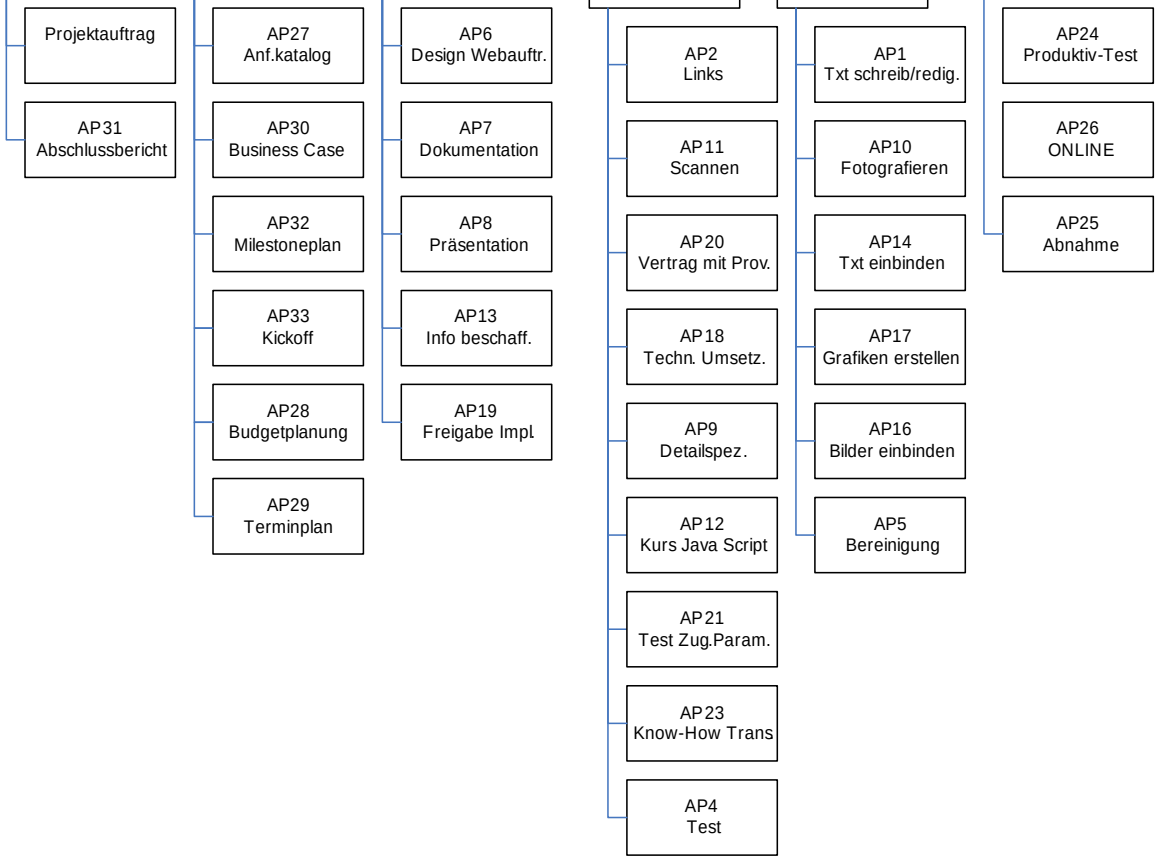
Niklaus, CEO der IT-RSQ GmbH

13. Antrag

Antrag auf Freigabe der Phase Initialisierung und der dazu benötigten Ressourcen.

14. Unterschriften

Projektleiter:	Auftraggeber:
.....	.....


Projektauftrag
AP31
Abschlussbericht
AP27
Anf.katalog
AP30
Business Case
AP32
Milestoneplan
AP33
Kickoff
AP28
Budgetplanung
AP29
Terminplan
AP6
Design Webauftr.
AP7
Dokumentation
AP8
Präsentation
AP13
Info beschaff.
AP19
Freigabe Impl
AP2
Links
AP11
Scannen
AP20
Vertrag mit Prov.
AP18
Techn. Umsetz.
AP9
Detailspez.
AP12
Kurs Java Script
AP21
Test Zug.Param.
AP23
Know-How Trans
AP4
Test
AP1
Txt schreib/redig.
AP10
Fotografieren
AP14
Txt einbinden
AP17
Grafiken erstellen
AP16
Bilder einbinden
AP5
Bereinigung
AP24
Produktiv-Test
AP26
ONLINE
AP25
Abnahme

My Index