



XPages erweitern und ausbauen

© 2016 Sven Hasselbach



XPages erweitern und ausbauen

Über mich

Sven Hasselbach

- Freiberuflicher IT Consultant seit 2003
- IBM Champion 2013
- Blog: <http://blog.hasselba.ch>
- Komme nicht aus der Schweiz



XPages erweitern und ausbauen

Motivation

- Es fehlt eine Komponente
- Der Renderer rendert nicht das, was ich brauche
- Ich bräuchte eine andere Validierung
- Wäre es nicht praktisch, wenn...
- XPages wird Open Source



XPages erweitern und ausbauen

Themen

- Validatoren
- Phase Listener
- Multi-Threading
- Komponenten & Renderer
- Service Locator (JVM übergreifender Datenaustausch)
- OSGi Plugins



XPages erweitern und ausbauen

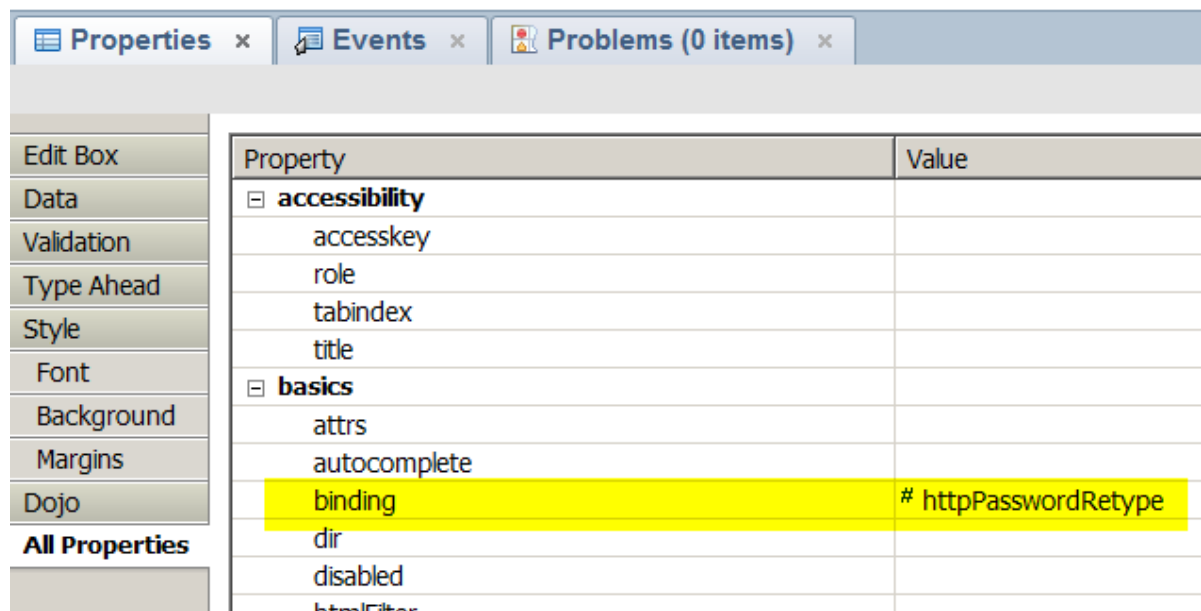
Eigener Validator

Der eigene Validator

XPages erweitern und ausbauen

Eigener Validator: Passwort-Validator

- Zwei Passwort-Felder: Ein Feld für das Passwort, ein Feld für den Retype
 - Verbindung über Binding



The screenshot shows the 'Properties' view in an IDE. On the left is a sidebar with categories: Edit Box, Data, Validation, Type Ahead, Style, Font, Background, Margins, Dojo, and All Properties. The main area displays a table of properties for a selected component. The 'binding' property is highlighted in yellow, showing its value as '# httpPasswordRetype'.

Property	Value
accessibility	
accesskey	
role	
tabindex	
title	
basics	
attrs	
autocomplete	
binding	# httpPasswordRetype
dir	
disabled	
htmlFilter	

XPages erweitern und ausbauen

Eigener Validator: Passwort-Validator (2) - XPage

```
<xp:inputText
    id="httpPassword"
    password="true">

    <xp:this.validators>
        <xp:validator
            validatorId="passwordValidator" />
    </xp:this.validators>

</xp:inputText>

<br />

<xp:inputText
    id="httpPasswordRetype"
    password="true"
    binding="#{httpPasswordRetype}">
</xp:inputText>
```

XPages erweitern und ausbauen

Tips & Tricks: Binding von Component Property vermeiden

Binding von Component Property an Managed Bean-Eigenschaft ist Bad Practice!

```
<xp:inputText
    id="httpPasswordRetype"
    password="true"
    binding="#{myBean.myComponent}">
</xp:inputText>
```

```
public class MyBean {

    private transient UIComponent myComponent;

    public UIComponent getMyComponent(){};
    public void setMyComponent(UIComponent cmp){};

}
```


XPages erweitern und ausbauen

Tips & Tricks: Binding von Component Property vermeiden (2)

Warum?

- Component Tree kann nicht mehr sauber verarbeitet werden
- Auch bei Request Scoped Beans!

Siehe dazu auch:

<http://stackoverflow.com/questions/14911158/how-does-the-binding-attribute-work-in-jsf-when-and-how-should-it-be-used>

XPages erweitern und ausbauen

Eigener Validator: Passwort-Validator (3) - Validator

```
public class PasswordValidator implements Validator {

    public void validate(FacesContext fc, UIComponent component,
        Object value) throws ValidatorException {

        String password = (String) value;

        UIInput confirmComponent = (UIInput) fc.getApplication()
            .getVariableResolver().resolveVariable(fc, "httpPasswordRetype");

        String confirm = (String) confirmComponent.getSubmittedValue();

        if (!password.equals(confirm)) {
            confirmComponent.setValid(false);
            throw new ValidatorException(new FacesMessage(
                "Passwords are not equal."));
        }
    }
}
```

XPages erweitern und ausbauen

Eigener Validator: Passwort-Validator (4) – faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config>
  <validator>
    <description>Password Validator</description>
    <validator-id>passwordValidator</validator-id>
    <validator-class>ec2016.PasswordValidator</validator-class>
  </validator>
</faces-config>
```

XPages erweitern und ausbauen

Tips & Tricks: DocType in faces-config.xml setzen

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE faces-config PUBLIC "-//Sun Microsystems, Inc.//DTD JavaServer
Faces Config 1.0//EN" "http://java.sun.com/dtd/web-facesconfig_1_1.dtd">
```

```
<faces-config />
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE faces-config PUBLIC "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN"
"http://java.sun.com/dtd/web-facesconfig_1_1.dtd">
```

```
<faces-config>
```

```
<
```

```
</fac
```

- <> application
- <> component
- <> converter
- <> factory
- <> lifecycle
- <> managed-bean
- <> navigation-rule
- <> referenced-bean
- <> render-kit
- <> validator

Element : application

===== Definition Elements
===== The
"application" element provides a mechanism to define the
various per-application-singleton implementation artifacts for
a particular web application that is utilizing JavaServer Faces.
For nested elements that are not specified, the JSF
implementation must provide a suitable default.

Content Model : ((action-listener | default-render-kit-id |
message-bundle | navigation-handler | view-handler | state-
manager | property-resolver | variable-resolver | locale-
config)*)



XPages erweitern und ausbauen

Eigener Validator: Passwort-Validator (5) – Demo

Demo

- *example10_password.xsp*

XPages erweitern und ausbauen

Eigener Validator: Passwort-Validator (6) – Regeln

- Validatoren validieren nur, wenn Werte existieren
- Der Required Validator ist „getrickst“:
 - In den Komponenten wurden spezielle Routinen eingebaut
 - Nach Instanzen von *FacesRequiredValidator* wird gesucht
- Klingt verlockend...

XPages erweitern und ausbauen

Eigener Validator: Passwort-Validator (7) - JavaCode

```
import ...;

import com.ibm.xsp.validator.FacesRequiredValidator;

public class RequiredValidator implements FacesRequiredValidator {

    public String getRequiredMessage() {
        return "Required!";
    }

    public void validate(FacesContext paramFacesContext,
        UIComponent paramUIComponent, Object paramObject)
        throws ValidatorException {

        FacesMessage msg = new FacesMessage("Required!");
        throw new ValidatorException(msg);
    }
}
```

XPages erweitern und ausbauen

Eigener Validator: Passwort-Validator (8) – Regeln

- Aber: Java-Konvertierung zerstört alles:

```
private UIComponent createRequiredtext(FacesContext context,  
    UIComponent parent, PageExpressionEvaluator evaluator) {  
    XspInputText result = new XspInputText();  
    ValidatorImpl validators = new ValidatorImpl();  
    validators.setValidatorId("requiredValidator");  
    result.addValidator(validators);  
    setId(result, "requiredText");  
    return result;  
}
```

- Daher nur programmatisch hinzufügbarm
 - siehe <http://hasselba.ch/blog/?p=764>

XPages erweitern und ausbauen

Eigener Validator: Passwort-Validator (9) – Bean Validierung

- Validator-Attribut ruft MethodBinding auf

```
<xp:inputText  
    id="httpPassword"  
    password="true"  
    validator="#{validatorBean.validatePassword}" >  
</xp:inputText>
```

- Methode in Bean mit Parametern
 - FacesContext, UIComponent, Object
 - Rückgabewert void

XPages erweitern und ausbauen

Eigener Validator: Passwort-Validator (10) – Bean Validierung

```
private String binding;

public void validatePassword(FacesContext fc, UIComponent uiComponent,
    Object parameters) {

    String pwd = (String) parameters;
    UIInput confirmComponent = (UIInput) JSFUtils.resolveVariable(binding);

    String confirm = (String) confirmComponent.getSubmittedValue();

    if (!parameters.equals(confirm)) {
        FacesMessage msg = new FacesMessage("Passwords are not equal.");
        throw new ValidatorException(msg);
    }

}

public void setBinding(String binding) {
    this.binding = binding;
}

public String getBinding() {
    return binding;
}
```

XPages erweitern und ausbauen

Eigener Validator: Passwort-Validator (11) – Bean Validierung

```
<managed-bean>
  <managed-bean-name>validatorBean</managed-bean-name>
  <managed-bean-scope>none</managed-bean-scope>
  <managed-bean-class>ec2016.ValidatorBean</managed-bean-class>

  <managed-property>
    <property-name>binding</property-name>
    <value>httpPasswordRetype</value>
  </managed-property>
</managed-bean>
```



XPages erweitern und ausbauen

Eigener Validator: Passwort-Validator (11) – Bean Validierung

Demo

- *example11_beanValidator.xsp*



XPages erweitern und ausbauen

Tipps & Tricks: Managed Properties

- Werden beim Instanzieren der Bean aufgerufen
- Können alle möglichen Objekt-Arten sein, auch Listen
- Auch EL möglich
- Beans können auch Managed Properties sein (Scope muss passen)

XPages erweitern und ausbauen

Tipps & Tricks: Managed Properties (2)

```
<managed-bean>
    <managed-bean-name>myBean</managed-bean-name>
    <managed-bean-scope>requestScope</managed-bean-scope>
    <managed-bean-class>ec2016.myBean</managed-bean-class>
</managed-bean>

<managed-bean>

    <managed-bean-name>myOtherBean</managed-bean-name>
    <managed-bean-scope>requestScope</managed-bean-scope>
    <managed-bean-class>ec2016.myOtherBean</managed-bean-class>

    <managed-property>
        <property-name>refToMyBean</property-name>
        <value>#{myBean}</value>
        <property-class>ec2016.myBean</property-class>
    </managed-property>

</managed-bean>
```



XPages erweitern und ausbauen

Phase Listener

- Phase Listener



XPages erweitern und ausbauen

Phase Listener – Was ist das?

- Phase Listener erlauben den Eingriff in den JSF Lifecycle
- Sind in der ganzen Applikation verfügbar
- Sehr praktisch



XPages erweitern und ausbauen

Phase Listener: Praktische Beispiele

- URL-Überwachung über die ganze Applikation
 - <http://hasselba.ch/blog/?p=1243>
- Eingreifen in den ComponentTree (z.B. um Validierung im FileDownload Control abzuschalten)
 - <http://hasselba.ch/blog/?p=1363>
- Zugriffskontrolle per TokenController



XPages erweitern und ausbauen

Phase Listener: TokenController – Anforderung

- Anforderung: Umfrage-Formular soll anonym ausfüllbar sein
 - jedoch nur „eingeladene“ Benutzer sollen teilnehmen



XPages erweitern und ausbauen

Phase Listener: TokenController – Lösung

- Per Mail wird uniquer Link mit Token erzeugt
- In XPage Applikation wird Link validiert

XPages erweitern und ausbauen

Phase Listener: TokenController (2) – Java Code

```
public class TokenController implements PhaseListener {

    public void beforePhase(PhaseEvent event) {

        FacesContextExImpl fc = (FacesContextExImpl)
FacesContextExImpl.getCurrentInstance();
        ExternalContextEx ec = (ExternalContextEx) fc.getExternalContext();

        String uri = ((HttpServletRequest) ec.getRequest()).getRequestURI();
        if( !(uri.contains( "/example20_survey.xsp" ) ) )
            return;

        Map<String, String> parameterMap = ec.getRequestParameterMap();
        String paramId = (String) parameterMap.get( "key" );
        String paramHash = (String) parameterMap.get( "hash" );

        if( !(MD5Utils.isValidHash( paramId, paramHash ))){
            JSFUtils.redirectToPage( "/example21_generateKey.xsp" );
            return;
        }

    }

    public PhaseId getPhaseId() {
        return PhaseId.RENDER_RESPONSE;
    }

}
```

XPages erweitern und ausbauen

Phase Listener: TokenController (3) – faces-config.xml

```
<faces-config>
    <lifecycle>
        <phase-listener>ec2016.phaselisteners.TokenController</phase-listener>
    </lifecycle>
</faces-config>
```



XPages erweitern und ausbauen

Phase Listener: TokenController (4) – Demo

Demo

- *example21_survey.xsp*



XPages erweitern und ausbauen

Phase Listener: Try/Catch behandeln

- Fehler im Phase Listener müssen selbst behandelt werden (try/catch)
- Sonst wird Phase Listener ohne Rückmeldung deaktiviert



XPages erweitern und ausbauen

Eigene Threads

- Eigene Threads



XPages erweitern und ausbauen

Eigene Threads

- Wozu Threads?
 - Logging
 - Aufräumarbeiten
 - Asynchrone Tätigkeiten
 - Long Running Operations



XPages erweitern und ausbauen

Eigene Threads: XpagesExecutor Service

- Für eigene Threads bietet sich der XPagesExecutor Service an:

com.ibm.xsp.application.XPagesExecutor



XPages erweitern und ausbauen

Eigene Threads: XpagesExecutorService (3) - Demo

Demo

- *example40_XPagesExecutor.xsp*



XPages erweitern und ausbauen

Eigene Komponenten & Renderer

- Komponenten & Renderer



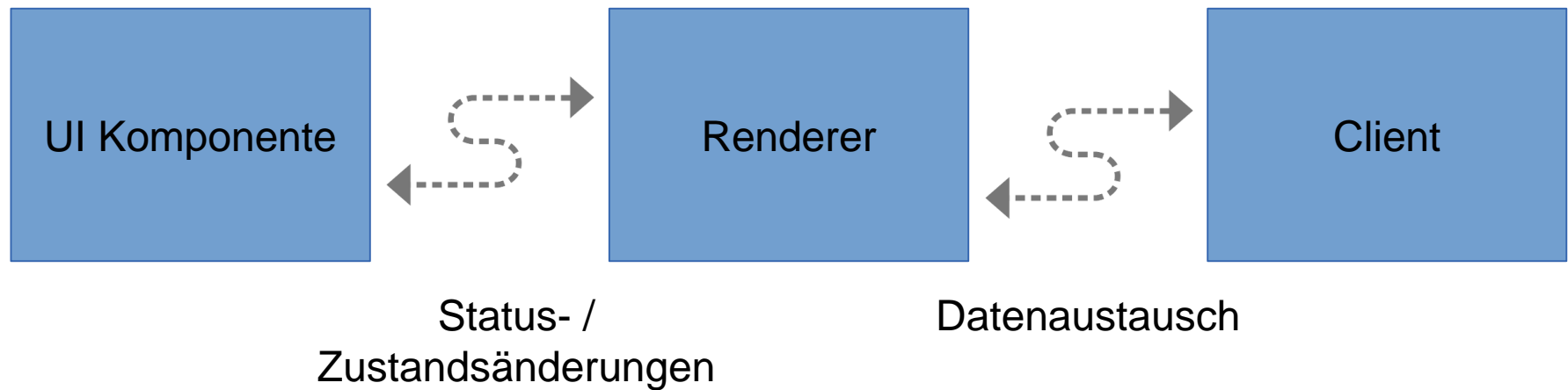
XPages erweitern und ausbauen

Eigene Komponenten & Renderer: Kurzer Überblick

- Komponente besteht aus UIComponent & Renderer
- UIComponent ist die Backend-Komponente
- Der Renderer generiert die Ausgabe (HTML, XML, JSON, etc)
- Kann voneinander getrennt sein, muss aber nicht

XPages erweitern und ausbauen

Eigene Komponenten & Renderer: Grobkonzept





XPages erweitern und ausbauen

Eigene Komponenten & Renderer

- Renderer ist Klasse, die einer Komponente zugeordnet ist
- Liest auch die Daten, die vom Client kommen
 - In XPages „aufgeweicht“:
UIKomponenten reagieren auch auf
AJAX Requests



XPages erweitern und ausbauen

Eigene Renderer

- Eigene Renderer



XPages erweitern und ausbauen

Tips & Tricks: Details einer Komponente

- `UIComponent.getFamily()`

Liefert Familie der Komponente zurück

- `UIComponent.getRendererType()`

Liefert Renderer der Klasse zurück

- `UIComponent.getStyleKitFamily()`

Liefert Theme Id der Komponente zurück



XPages erweitern und ausbauen

Tips & Tricks: Details einer Komponente (2)

Demo

- *example01_getClass.xsp*

Idee von Cameron Gregor

<http://camerongregor.com>

XPages erweitern und ausbauen

Renderer Basics

- `public void decode(FacesContext context, UIComponent component)`
Verarbeitet Request und setzt den Status neu in der UIKomponente
- `public void encodeBegin(FacesContext context, UIComponent component)`
Startet den Output der UIComponent
- `public void encodeChildren(FacesContext context, UIComponent component)`
Verarbeitet ggf. Kinder der UIComponent
- `public void encodeEnd(FacesContext context, UIComponent component)`
Ist das Ende der Verarbeitung

XPages erweitern und ausbauen

Eigener Renderer: Java Klasse

```
public class DemoRenderer extends Renderer {  
  
    public DemoRenderer() {  
        super();  
    }  
  
    @Override  
    public void encodeEnd(FacesContext fc, UIComponent uiComponent)  
        throws IOException {  
  
        ResponseWriter rw = fc.getResponseWriter();  
        rw.startElement("div", uiComponent);  
        rw.writeAttribute("style",  
            "background-color:red; width:100%;  
height:100%", "style");  
  
        super.encodeEnd(fc, uiComponent);  
  
        rw.endElement("div");  
    }  
}
```

XPages erweitern und ausbauen

Eigener Renderer: Registrierung in faces-context.xml

```
<render-kit>
  <renderer>
    <renderer-class>ec2016.renderer.DemoRenderer</renderer-class>
    <renderer-type>ec2016.renderer.DemoRenderer</renderer-type>
    <component-family>javax.faces.Input</component-family>
  </renderer>
</render-kit>
```

XPages erweitern und ausbauen

Eigener Renderer: Verwenden in einer UIComponent

```
<?xml version="1.0" encoding="UTF-8"?>
<xp:view
    xmlns:xp="http://www.ibm.com/xsp/core">

    <xp:inputText
        id="inputText1"
        rendererType="ec2016.renderer.DemoRenderer">
    </xp:inputText>
</xp:view>
```

	id	inputText1
Edit Box	immediate	
Data	lang	
Validation	loaded	
Type Ahead	maxlength	
Style	multipleSeparator	
Font	multipleTrim	
Background	password	
Margins	readonly	
Dojo	redisplay	
	rendered	
All Properties	rendererType	ec2016.renderer.DemoRenderer
	required	



XPages erweitern und ausbauen

Renderer Basics

Demo

- *example30_demoRenderer.xsp*

XPages erweitern und ausbauen

Renderer Basics: Den richtigen Renderer finden

- **core-faces-config.xml**

\osgi\shared\eclipse\plugins\com.ibm.xsp.core_9.0.0.20130301-1431\lwpd.xsp.core.jar!\META-INF\core-faces-config.xml

- **extsn-faces-config.xml**

\osgi\shared\eclipse\plugins\com.ibm.xsp.extsn_9.0.0.20130301-1431\lwpd.xsp.extsn.jar!\META-INF\extsn-faces-config.xml

Beispiel

```
<renderer>
  <component-family>javax.faces.Input</component-family>
  <renderer-type>javax.faces.Text</renderer-type>
  <renderer-class>com.ibm.xsp.renderkit.html_basic.InputTextRenderer</renderer-class>
</renderer>
```


XPages erweitern und ausbauen

Eigener Renderer: Java Klasse

```
public class DemoRenderer extends InputTextRenderer {

    public DemoRenderer() {
        super();
    }

    @Override
    public void encodeEnd(FacesContext fc, UIComponent uiComponent)
        throws IOException {

        ResponseWriter rw = fc.getResponseWriter();
        rw.startElement("div", uiComponent);
        rw.writeAttribute("style",
            "background-color:red; width:100%; height:100%",
            "style");

        super.encodeEnd(fc, uiComponent);

        rw.endElement("div");
    }
}
```



XPages erweitern und ausbauen

Renderer Basics

Demo

- `example31_demoRendererFixed.xsp`

XPages erweitern und ausbauen

Tips & Tricks: Setzen des Renderes über Theme

```
<!-- Theme pager -->

<control>
    <name>InputField.EditBox</name>
    <property>
        <name>rendererType</name>

        <value>ec2016.renderer.DemoRendererFixed</value>
    </property>
</control>
```



XPages erweitern und ausbauen

Eigene Komponenten

- Eigene Komponenten



XPages erweitern und ausbauen

Eigene Komponenten

Komponenten bestehen aus zwei
Teilen:

1. Java Klasse
2. Eigene faces-config.xml

XPages erweitern und ausbauen

Eigene Komponenten: Java Klasse

```
package ec2016.component;

import java.io.Serializable;
import javax.faces.component.UIComponentBase;

public class DemoComponent extends UIComponentBase implements
Serializable {

    private static final long serialVersionUID = 1L;

    public DemoComponent() {
        super();
    }

    @Override
    public String getFamily() {
        return "ec2016.component";
    }
}
```

XPages erweitern und ausbauen

Eigene Komponenten: Eigene faces-config.xml

```
<faces-config>

    <faces-config-extension>
        <namespace-uri>http://entwicklercamp/xsp/control</namespace-uri>
        <default-prefix>ec2016</default-prefix>
    </faces-config-extension>

    <component>
        <component-type>ec2016.component.DemoComponent</component-type>
        <component-class>ec2016.component.DemoComponent</component-class>

        <component-extension>
            <component-family>ec2016.component</component-family>
            <renderer-type>ec2016.component</renderer-type>

            <tag-name>DemoComponent</tag-name>

            <designer-extension>
                <in-palette>true</in-palette>
                <category>DemoComponent</category>
            </designer-extension>

        </component-extension>

    </component>

</faces-config>
```

XPages erweitern und ausbauen

Eigene Komponenten: In XPage verwenden

```
<?xml version="1.0" encoding="UTF-8"?>
<xp:view
  xmlns:xp="http://www.ibm.com/xsp/core"
  xmlns:ec2016="http://entwicklercamp/xsp/control">

  <ec2016:DemoComponent id="DemoComponent1"></ec2016:DemoComponent>

</xp:view>
```

Select
Other...

Core Controls

Container Controls

Data Access

Dojo Form

Dojo Layout

Extension Library

iNotes

Mobile

Other Controls

DemoComponent

EC2016 Component

XPages erweitern und ausbauen

Eigene Komponenten: Werte in Komponenten

```
package ec2016.component;

import javax.faces.component.UIComponentBase;
import javax.faces.context.FacesContext;

public class DemoComponentWithState extends UIComponentBase {

    private static final long serialVersionUID = 1L;
    private String myValue;

    public DemoComponentWithState() {
        super();
    }

    @Override
    public String getFamily() {
        return "ec2016.component";
    }

    @Override
    public Object saveState(FacesContext fc) {
        Object[] obj = new Object[1];
        obj[0] = super.saveState(fc);
        obj[1] = myValue;
        return obj;
    }

    @Override
    public void restoreState(FacesContext fc, Object obj) {
        Object[] values = (Object[]) obj;
        super.restoreState(fc, values[0]);
        this.myValue = ((String) values[1]);
    }
}
```

XPages erweitern und ausbauen

Eigene Komponenten: Werte in Komponenten (2)

```
<component>
  <component-type>ec2016.component.DemoComponentWithState</component-
type>
  <component-class>ec2016.component.DemoComponentWithState</component-
class>

  <component-extension>
    <component-family>ec2016.component</component-family>
    <renderer-type>ec2016.component</renderer-type>

    <tag-name>DemoComponentWithState</tag-name>

    <designer-extension>
      <in-palette>true</in-palette>
      <category>DemoComponent</category>
    </designer-extension>
  </component-extension>

  <property>
    <property-name>myValue</property-name>
    <property-class>java.lang.String</property-class>
  </property>
</component>
```

XPages erweitern und ausbauen

Eigene Komponenten: Werte in Komponenten (3)

```
<?xml version="1.0" encoding="UTF-8"?>
<xp:view
  xmlns:xp="http://www.ibm.com/xsp/core"
  xmlns:ec2016="http://entwicklercamp/xsp/control">
```

```
<ec2016:DemoComponentWithState
  id="DemoComponentWithState1"
  myValue="xx">
</ec2016:DemoComponentWithState>
```

```
</xp:view>
```

esign Source

Properties x Events x Problems (0 items) x

All Properties

Property	Value
basics	
binding	
id	DemoComponentWithState1
loaded	
rendered	
rendererType	
other	
myValue	xx
styling	
disableTheme	
themeId	

XPages erweitern und ausbauen

Eigene Komponenten: XPages Interfaces (Auszug)

Interface	Verwendung für
FacesAjaxComponent	AJAX Requests
FacesAttrsObject	Extra Attribute für Komponente
FacesComponent	Für Initialisierung vor / nach den Child Components
FacesInputComponent	Handling von Validierung & Co
ThemeControl	Für Support von Themes

Weitere Interfaces im Mastering XPages (2. Auflage), Kapitel 5!



XPages erweitern und ausbauen

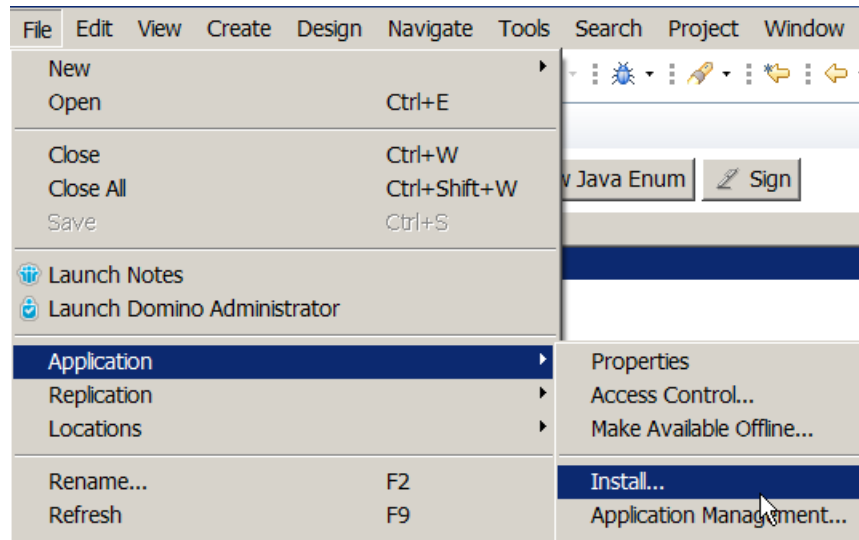
Tipps & Tricks: JD GUI im Designer

- Java Decompiler
 - Als Standalone oder als Eclipse-Plugin
 - <http://jd.benow.ca/>
 - Kein Suchen mehr nach der Spezifikation!

XPages erweitern und ausbauen

Tipps & Tricks: JD GUI im Designer (2)

- Installation des Plugins



XPages erweitern und ausbauen

Tips & Tricks: JD GUI im Designer (3)

```
package ch.hasselba.xpages;
```

```
import javax.faces.context.FacesContext;
```

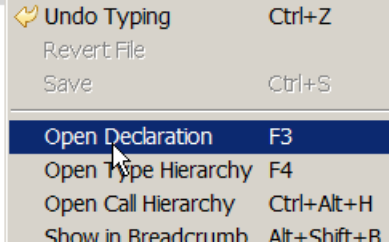
```
public class JDGuiDemo {
```

```
    public void foo() {
```

```
        FacesContext fc = FacesContext.getCurrentInstance();
```

```
    }
```

```
}
```



EC2016.nsf - Java x JDGuiDemo.java x FacesContext.class x

```
package javax.faces.context;
```

```
import java.util.Iterator;
```

```
import javax.faces.application.Application;
```

```
import javax.faces.application.FacesMessage;
```

```
import javax.faces.application.FacesMessage.Severity;
```

```
import javax.faces.component.UIViewRoot;
```

```
import javax.faces.render.RenderKit;
```

```
public abstract class FacesContext
```

```
{
```

```
    private static ThreadLocal instance = new ThreadLocal() {
```

```
        protected Object initialValue() {
```

```
            return null;
```

```
        }
```

```
    };
```

```
    public abstract Application getApplication();
```

```
    public abstract Iterator getClientIdsWithMessages();
```

```
    public abstract ExternalContext getExternalContext();
```

```
    public abstract FacesMessage.Severity getMaximumSeverity()
```



XPages erweitern und ausbauen

OSGi: ServiceLocator

- OSGi: ServiceLocator



XPages erweitern und ausbauen

OSGi: ServiceLocator

ServiceLocator erlaubt Kommunikation über die Grenzen einer XPages hinaus:

- verschiedene XPages Applikationen untereinander
- Kommunikation innerhalb der JVM (z.B. mit einem Equinox Servlet)



XPages erweitern und ausbauen

OSGi: ServiceLocator

Factory Pattern

- Factory kontrolliert, dass es nur eine Instanz gibt
- Static sorgt dafür, dass Object „ewig“ existiert

XPages erweitern und ausbauen

OSGi: ServiceLocator - ServiceLocatorFactory

```
package ch.hasselba.xpageslocator;

public class ServiceLocatorFactory {

    static private ServiceLocatorFactory instance;
    static private ServiceLocator serviceLocator;

    static {
        synchronized (ServiceLocatorFactory.class) {
            if (instance == null) {
                instance = new ServiceLocatorFactory();
                serviceLocator = new ServiceLocator();
            }
        }
    }

    static public ServiceLocatorFactory getInstance() {
        return instance;
    }

    public static ServiceLocator getServiceLocator() {
        return serviceLocator;
    }

    public static void setServiceLocator(ServiceLocator serviceLocator) {
        ServiceLocatorFactory.serviceLocator = serviceLocator;
    }

}
```

XPages erweitern und ausbauen

OSGi: ServiceLocator - ServiceLocator

```
package ch.hasselba.xpageslocator;  
  
public class ServiceLocator {  
  
    private ServerMap servermap = new ServerMap();  
  
    public synchronized ServerMap getServerMap() {  
        return servermap;  
    }  
  
    public synchronized void setServerMap(ServerMap map) {  
        servermap = map;  
    }  
  
}
```

XPages erweitern und ausbauen

OSGi: ServiceLocator - ServerMap

```
@SuppressWarnings("unchecked")
public class ServerMap implements Map<String, Object>, Serializable {

    private static final long serialVersionUID = -1L;
    private static long lastAccessed = 0;
    private static ConcurrentHashMap<String, Object> map = new ConcurrentHashMap<String, Object>();

    public static long getLastAccessed() {
        return lastAccessed;
    }

    public static synchronized void setLastAccessed(long lastAccessed) {
        ServerMap.lastAccessed = lastAccessed;
    }

    @Override
    public void clear() {
        setLastAccessed(System.currentTimeMillis());
        map.clear();
    }

    // usw. → alle Methoden überschreiben, immer setLastAccessed setzen
}
```

XPages erweitern und ausbauen

OSGi: ServiceLocator – Plugin ausrollen & in NSF aktivieren

Page Generation Properties

HTML Generation

Client validation: ☒ Server default ☐ On ☐ Off

Compression:

Encoding:

HTML doctype:

☐ Generate meta tags in the HTML header

☐ Force content type to application/xhtml+xml

☒ Page redirect mode

☐ Allow zero rows in repeat controls

Active Content Filtering

Rich Text Options

Save links in: ☒ Notes format ☐ Web format

XPage Libraries

Select the libraries of extended XPage controls to use in this application.

Library ID	
<input checked="" type="checkbox"/>	ch.hasselba.xpageslocator.library.XPagesServiceLocatorLibrary
<input type="checkbox"/>	ch.hasselba.xpageslocator.library.XPagesServiceLocatorLibrary
<input type="checkbox"/>	ch.hasselba.xpageslocator.library.XPagesServiceLocatorLibrary
<input type="checkbox"/>	ch.hasselba.xpageslocator.library.XPagesServiceLocatorLibrary

When running on the Web, the libraries must be available on the server. When running on the Notes client, the libraries must

XPages erweitern und ausbauen

OSGi: ServiceLocator - XPage

```
<?xml version="1.0" encoding="UTF-8"?>
<xp:view
  xmlns:xp="http://www.ibm.com/xsp/core">
  <h1>Service Locator A</h1>

  <xp:button
    id="button1"
    value="Setze ServerMap">
    <xp:eventHandler
      event="onclick"
      submit="true"
      refreshMode="complete">
      <xp:this.action>
        <![CDATA[#{javascript:importPackage( ch.hasselba.xpages servicelocator );
          var msg = "ServiceLocator: " + java.lang.System.currentTimeMillis();
          ch.hasselba.xpages servicelocator.ServiceLocatorFactory.getInstance().getServiceLocator().getServerMap().put("ec2016", msg );
        }]]>
      </xp:this.action>
    </xp:eventHandler>
  </xp:button>

  <br />
  <br />

  <xp:label
    id="labelSubscriberMessage">
    <xp:this.value>
      <![CDATA[#{javascript:importPackage( ch.hasselba.xpages servicelocator );
        ch.hasselba.xpages servicelocator.ServiceLocatorFactory.getInstance().getServiceLocator().getServerMap().get("ec2016");
      }]]>
    </xp:this.value>
  </xp:label>
</xp:view>
```



XPages erweitern und ausbauen

OSGi: ServiceLocator – Demo 1

Demo!



XPages erweitern und ausbauen

OSGi: ServiceLocator – JVM Grenzen überwinden

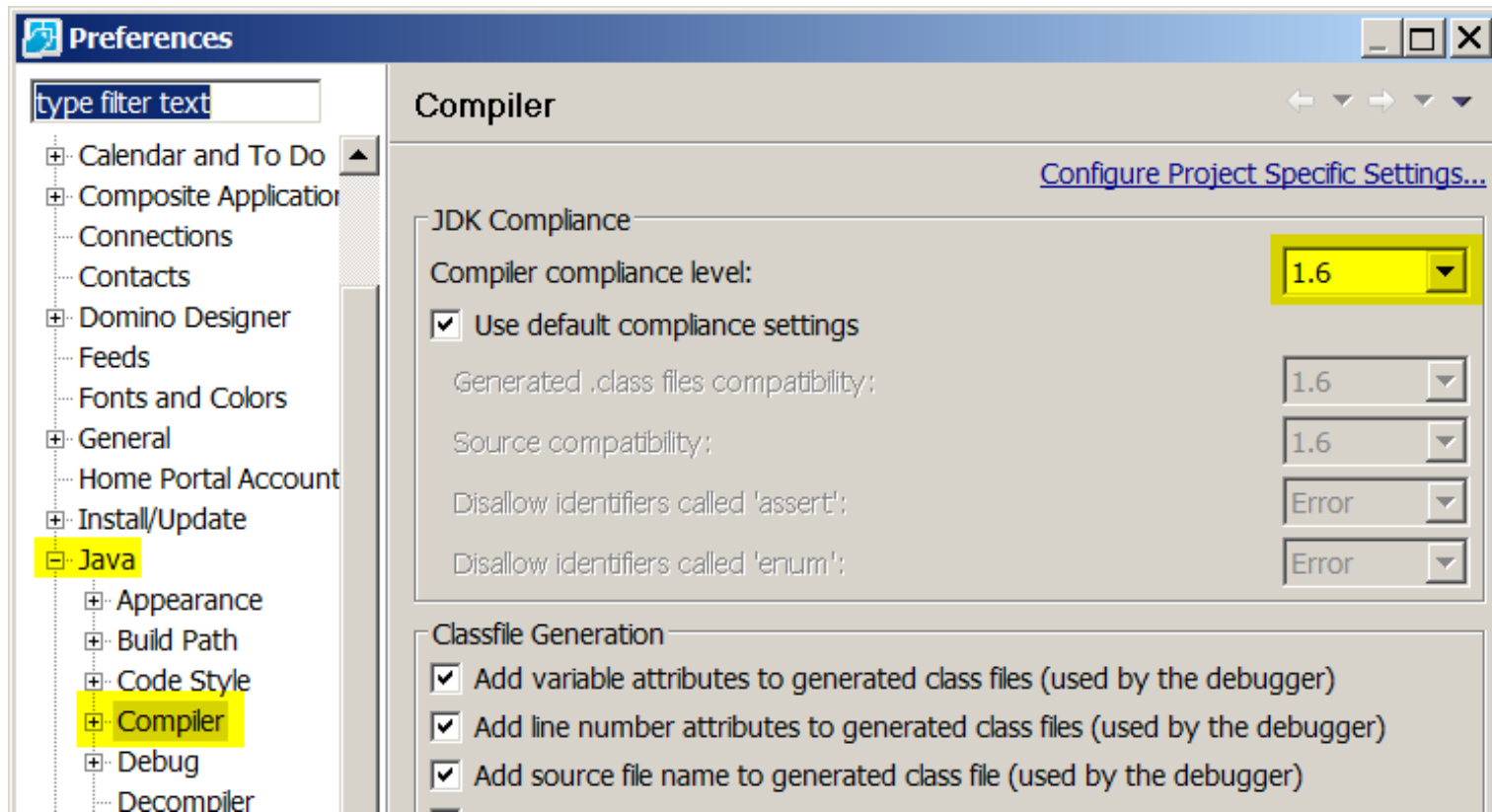
Zugriff von außerhalb der JVM per RMI

- Kommunikation von Agenten mit XPages
- Auch „fremde“ JVMs haben Zugriff

XPages erweitern und ausbauen

OSGi: ServiceLocator

Java Code Compliance auf 1.6 setzen



XPages erweitern und ausbauen

OSGi: ServiceLocator – RMI Interface

```
package ch.hasselba.xpages servicelocator;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.HashMap;

public interface ServerMapRemoteInterface extends Remote {

    HashMap<String, Object> getServerMapRemote() throws RemoteException;

    void setServerMapRemote( HashMap<String, Object> serverMap )
        throws RemoteException;
}
```

XPages erweitern und ausbauen

OSGi: ServiceLocator – ServiceLocator erweitern

```
public class ServiceLocator extends java.rmi.server.UnicastRemoteObject
    implements ServerMapRemoteInterface {

    private static final long serialVersionUID = 1L;
    private int thisPort = 3232;
    private String thisAddress;
    private Registry registry;

    protected ServiceLocator() throws RemoteException {

        try {
            thisAddress = (InetAddress.getLocalHost()).toString();
        } catch (Exception e) {
            throw new RemoteException("can't get inet address.");
        }

        System.out.println("this address=" + thisAddress + ",port=" +
thisPort);

        try {
            registry = LocateRegistry.createRegistry(thisPort);
            registry.rebind("rmiServer", this);
        } catch (RemoteException e) {
            throw e;
        }

    }

    ...

}
```

XPages erweitern und ausbauen

OSGi: ServiceLocator – ServiceLocator erweitern (2)

```
@Override
public HashMap<String, Object> getServerMapRemote()
    throws RemoteException {
    HashMap map = new HashMap<String, Object>();
    for( Entry<String, Object> entry  : getServerMap().entrySet() ){
        map.put(entry.getKey(), entry.getValue());
    }
    return map;
}

@Override
public void setServerMapRemote(HashMap<String, Object> serverMap)
    throws RemoteException {

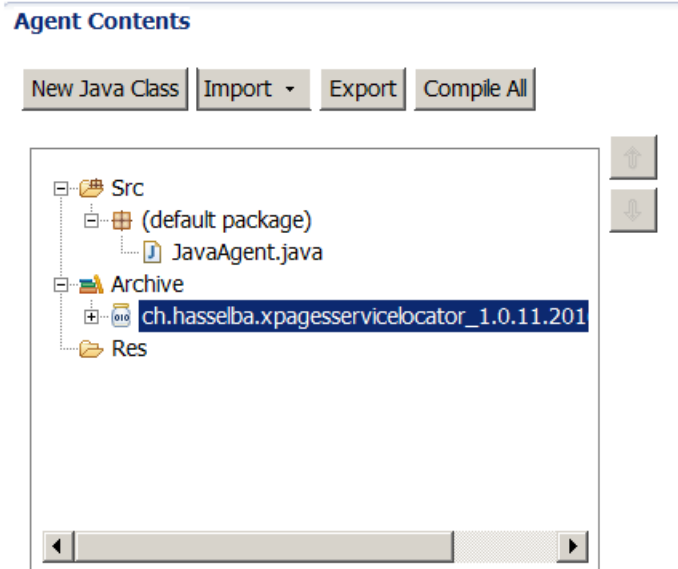
    for( Entry<String, Object> entry  : serverMap.entrySet() ){
        servermap.put(entry.getKey(), entry.getValue());
    }

}
```

XPages erweitern und ausbauen

OSGi: ServiceLocator

JAR in Agenten importieren



XPages erweitern und ausbauen

OSGi: ServiceLocator - Agent

```
public class JavaAgent extends AgentBase {

    @Override
    public void NotesMain() {

        try {

            // Registrierung öffnen
            Registry registry = LocateRegistry.getRegistry("192.168.2.108",
                3232);

            // Holen des RemoteInterfaces
            ServerMapRemoteInterface rmiServer =
                (ServerMapRemoteInterface)
(registry.lookup("rmiServer"));

            // auslesen der ServerMap
            HashMap<String, Object> serverMap =
rmiServer.getServerMapRemote();

            // Ausgabe des Wertes auf Konsole
            System.out.println("Wert: " + serverMap.get("ec2016"));

            // Wert neu setzen
            serverMap.put("ec2016", "Agent: " + System.currentTimeMillis());
            rmiServer.setServerMapRemote(serverMap);

        } catch (RemoteException e) {
            e.printStackTrace();
        } catch (NotBoundException e) {
            e.printStackTrace();
        }

    }

}
```



XPages erweitern und ausbauen

OSGi: ServiceLocator – Demo 2

Demo!



XPages erweitern und ausbauen

Vielen Dank!

Danke!

XPages erweitern und ausbauen

AddOn: SessionListener

```
package ec2016;

import javax.servlet.http.HttpSessionEvent;

import com.ibm.xsp.application.ApplicationEx;
import com.ibm.xsp.application.events.SessionListener;

public class ApplicationSessionListener implements SessionListener{

    public void sessionCreated(ApplicationEx app, HttpSessionEvent
event) {
        System.out.println( "session Created!");
    }

    public void sessionDestroyed(ApplicationEx app, HttpSessionEvent
event) {
        System.out.println( "session Destroyed!");
    }

}
```

XPages erweitern und ausbauen

AddOn: ApplicationListener

```
package ec2016;

import com.ibm.xsp.application.ApplicationEx;
import com.ibm.xsp.application.events.ApplicationListener2;

public class ApplicationListener implements ApplicationListener2 {


    public void applicationRefreshed(ApplicationEx app) {
        System.out.println( "App refreshed!");
    }


    public void applicationCreated(ApplicationEx app) {
        System.out.println( "App created!");
    }

    public void applicationDestroyed(ApplicationEx app) {
        System.out.println( "App destroyed!");
    }
}
```

XPages erweitern und ausbauen

AddOn: Als Service registrieren

 META-INF/services/com.ibm.xsp.core.events.ApplicationListener

 META-INF/services/com.ibm.xsp.core.events.SessionListener

