



**Zentrum für sichere Informationstechnologie – Austria**  
**Secure Information Technology Center – Austria**

A-1040 Wien, Weyringergasse 35  
 Tel.: (+43 1) 503 19 63-0  
 Fax: (+43 1) 503 19 63-66

A-8010 Graz, Inffeldgasse 16a  
 Tel.: (+43 316) 873-5514  
 Fax: (+43 316) 873-5520

http://www.a-sit.at  
 E-Mail: office@a-sit.at

**IT-SICHERHEITSHANDBUCH**  
**VERBESSERUNG DER ANWENDBARKEIT IM BETRIEBLICHEN UMFELD**  
**VERSION 1.1, 22. DEZEMBER 2006**

DI Edgar Neuherz – [edgar.neuherz@a-sit.at](mailto:edgar.neuherz@a-sit.at)

**Zusammenfassung:** Das österreichische Sicherheitshandbuch stellt ein auf Kompaktheit bedachte und auf österreichische Verhältnisse Rücksicht nehmendes Standardwerk zur Informationssicherheit in Organisationen. Umfang und technische Ausprägung als XML-Dokument waren für kleine und mittlere Unternehmen (KMUs) jedoch bisher teilweise schwerhandhabbar. Für den praktischen Einsatz in Wirtschaftsunternehmen wurde das Sicherheitshandbuch um Werkzeuge und Funktionen erweitert, die diesen Umstand verbessern sollen.

## Inhaltsverzeichnis

Inhaltsverzeichnis .....	1
Abbildungsverzeichnis .....	3
1 Einleitung.....	4
1.1 Projektinhalt .....	4
1.2 Anforderungen .....	4
1.2.1 Offline-Verfügbarkeit.....	5
1.2.2 Plattformunabhängigkeit.....	5
1.2.3 Personalisierung.....	5
1.2.4 Modularität und einfache Erweiterbarkeit.....	5
1.2.5 Skalierbarkeit.....	5
1.2.6 Zugriff auf verschieden Datenquellen .....	5
1.3 Implementierung .....	5
2 Cocoon .....	7
2.1 Umgesetzte Technologien .....	7
2.1.1 XLST.....	7
2.1.2 XSL-FO.....	7
2.1.3 XPATH.....	7
2.2 Prinzip .....	7
2.2.1 Sitemap .....	8
2.2.2 Pipeline(s).....	8
2.2.3 Generator .....	9
2.2.4 Von Cocoon mitgelieferte Generatoren .....	10
2.2.5 Transformer .....	10
2.2.6 Von Cocoon mitgelieferte Transformer .....	11
2.2.7 Serializer.....	11
2.2.8 Von Cocoon mitgelieferte Serializer.....	12
3 Update Sicherheitshandbuch .....	13
3.1 Trennung von Inhalt und Layout .....	13
3.2 Sicherheitshandbuch .....	13
3.2.1 Teil 1.....	13
3.2.2 Sitemap .....	14
3.2.3 Teil 2.....	14
3.3 Checkliste .....	15

3.4	Konfiguration	15
3.4.1	Verantwortlichkeit(en)	16
3.4.2	Auswahl Sicherheitsmaßnahmen	16
3.5	Ausgabeformate	18
4	Praktische Umsetzung	19
4.1	Verzeichnisstruktur	19
4.2	Server Lösung (Servlet Container vorhanden)	19
4.2.1	Systemanforderungen	19
4.2.2	Installation	20
4.2.3	Verzeichnisstruktur	20
4.3	Client Lösung	20
4.3.1	Systemanforderungen	20
4.3.2	Installation	21
5	Zusammenfassung und Ausblick	22
5.1	Projekthalte	22
5.1.1	Inhaltliche Anpassung an das Betriebliche Umfeld	22
5.1.2	Weiterentwicklung der Online-Version	22
5.1.3	PDF-Generator (Multichannel Publishing)	22
5.1.4	Ausblick	23
	Glossar	24
	Referenzen	25
	Historie	26

## Abbildungsverzeichnis

Abb. 1: Architektur Cocoon .....	8
Abb. 2: Cocoon Pipeline .....	9
Abb. 3: Cocoon-Generator .....	9
Abb. 4: Cocoon-Transformer .....	10
Abb. 5: Cocoon-Serializer .....	11
Abb. 6: Ausschnitt Korrekturen in Transformationsdatei .....	13
Abb. 7: Pipeline OE-SIHA_I.html .....	14
Abb. 8: Ausschnitt aus sitemap.xmap .....	14
Abb. 9: Pipeline OE-SIHA_II.html .....	15
Abb. 10: Pipeline Checkliste.html .....	15

# 1 Einleitung

Das Österreichische IT Sicherheitshandbuch wurde ursprünglich für den Einsatz im öffentlichen Sektor entworfen. So basieren Security Policies von einigen Bundesministerien auf diesem Standardwerk. Für den praktischen Einsatz im Unternehmen vor Ort war es jedoch notwendig, die Bedürfnisse und Voraussetzungen von KMUs zu berücksichtigen. Deshalb wurde das IT Sicherheitshandbuch angepasst, um sowohl die Installation als auch die Anwendung einfacher zu gestalten.

Dies soll auch Unternehmen mit geringen Vorkenntnissen im IT-Bereich ermöglichen, adäquates Informationssicherheitsmanagement für das eigene Unternehmen zu realisieren. Weiters wurde die Möglichkeit vorgesehen, das Sicherheitshandbuch und Checklisten dem „**Corporate Identity**“ des Unternehmens anzupassen (Branding).

Da das Sicherheitshandbuch im XML-Format vorliegt, sowie in Zukunft mit Erweiterungen und Änderungen der Funktionalität zu rechnen ist, wird ein XML-Publishing-Framework (Cocoon) eingesetzt.

## 1.1 Projektinhalt

Gemäß des Förderantrages „Sicherheitsinfrastruktur -Teilprojekt IT-Sicherheitshandbuch“ wurden die Projektinhalte wie folgt definiert:

*„Das Sicherheitshandbuch soll in drei Aspekten weiterentwickelt werden:*

- a) Inhaltliche Anpassung an neue IT-Szenarien, Technologien und Best Practices generell sowie branchenspezifische Bedürfnisse der Wirtschaft*
- b) Technisch sind die Funktionen und Tools der Online-Version weiterzuentwickeln, damit solche Bedürfnisse auch umgesetzt werden können.*
- c) Übernahme des PDF-Generators aus der WKÖ-Version auf das generelle Handbuch*
- d) Analyse von aufkommenden XML-basierenden Standard-Werkzeugen zum Editieren des Sicherheitshandbuches*
- e) Awareness, um den Nutzen darzustellen und die Bekanntheit und Verwendung zu fördern“*

## 1.2 Anforderungen

Das IT Sicherheitshandbuch hat sich in den vergangenen Jahren bei den Behörden und öffentlichen Institutionen als Standard etabliert. Für die praktische Anwendung des IT Sicherheitshandbuchs im breiteren Einsatz, insbesondere auch für Wirtschaftsunternehmen ist es jedoch notwendig, folgende Anforderungen zu berücksichtigen:

- Offline-Verfügbarkeit (z.B. in der lokalen Umgebung eines KMU)
- Plattformunabhängigkeit
- Personalisierung
- Modularität und einfache Erweiterbarkeit
- Skalierbarkeit
- Zugriff auf verschiedene Datenquellen

### **1.2.1 Offline-Verfügbarkeit**

Das Sicherheitshandbuch ist derzeit für die ONLINE-Anwendung, etwa im Intranet einer Organisation, konzipiert. Für die praktische Anwendung wäre es sinnvoll, das Handbuch über ein Front-End auch lokal (z.B. am Notebook ohne Netzwerkzugang) verfügbar zu machen.

### **1.2.2 Plattformunabhängigkeit**

Aufgrund der gegebenen Infrastruktur in den KMUs kommen verschiedene Betriebssysteme zum Einsatz. Eine in Java realisierte Lösung kann auf jedem Betriebssystem für die es eine aktuelle Java-Laufzeitumgebung gibt (z.B. Windows, Linux, Unix und MacOSX), verwendet werden.

### **1.2.3 Personalisierung**

Für den praktischen Einsatz ist es notwendig, auf die unterschiedlichen Anforderungen der KMUs Rücksicht zu nehmen. Deshalb ist die Möglichkeit, verschiedene Komponenten den eigenen Bedürfnissen anzupassen, eine unverzichtbare Funktionalität.

### **1.2.4 Modularität und einfache Erweiterbarkeit**

Durch die Verwendung von unabhängigen Bausteinen, die unter bestimmten Bedingungen beliebig und einfach kombiniert werden können, wird die Basis-Funktionalität einfach erweiterbar.

### **1.2.5 Skalierbarkeit**

Die Anforderungen in KMUs und im IT-Bereich ändern sich ständig. Deshalb soll die Applikation mit den Anforderungen mitwachsen können. Durch Verwendung einer erweiterbaren Komponentenarchitektur und standardisierten Schnittstellen skaliert die Applikation..

### **1.2.6 Zugriff auf verschieden Datenquellen**

Um Daten wieder verwenden zu können, ist eine Persistenz-Schicht notwendig. Je nach vorhandenem Know-How und Anforderungen können die Informationen einfach in einer Datei (Filesystem, XML-Datei o.ä.) oder in einer Datenbank abgespeichert werden.

In manchen Fällen ist es notwendig mit anderen Datenquellen, wie z.B. LDAP-Server, zu kommunizieren.

## **1.3 Implementierung**

Da das Sicherheitshandbuch XML als Standard verwendet, ist es sinnvoll, ein Framework zu verwenden, welches diesen Standard unterstützt und Anwendungen durch vorhandene Komponenten effizient umsetzt.

Es wurde im gegenständlichen Projekt eine Migration zu Cocoon gewählt. Cocoon bietet eine offene Architektur mit optimaler Implementierung betreffend der Performance, Stabilität, Skalierbarkeit und Modularität. „Cocoon 2“ als „abstrakte Maschine“ konzipiert, ist sowohl in einer Servlet-Umgebung (Server mit Servlet-Container) als auch als Standalone-Applikation lauffähig. Es unterstützt die performante Verarbeitung von XML-Dateien durch die Verwendung von SAX statt DOM und die Verwendung von performanten Cache-Algorithmen.

Durch die Einführung eines Formular-Frameworks in der Version Cocoon 2.1.5 (Cocoon Forms) wird die Erstellung von Eingabe-Masken unterstützt. Weiters ist es mit Cocoon möglich, Daten in den verschiedensten Formaten (HTML, PDF, Postscript, Excel, ...) ohne aufwändige Programmierung auszugeben.

## 2 Cocoon

Cocoon ist ein universelles XML-Publishing-System, mit dessen Hilfe XML-Dokumente in fast jedes gewünschte Zielformat umgewandelt und via Webbrowser ausgegeben werden können. Es wurde 1999 von Stefano Mazzocchi als Open Source Projekt unter der Apache Software Foundation gegründet.

Cocoon basiert auf der Programmiersprache Java und bietet damit eine hohe Plattformunabhängigkeit. Das Framework versetzt den Anwender in die Lage, aus einem XML-Datenfundus fast jedes gewünschte Ausgabeformat zu produzieren. Dies können zum Beispiel klassische XHTML-Dokumente für das Web ebenso sein wie Formate für Mobile Devices sowie Word- (rtf-Format), Open-Office- oder PDF-Dokumente.

Da die Umwandlung eines XML-Dokuments in das jeweilige Ausgabeformat über XSL erfolgt, lässt sich die Formatvielfalt beliebig ausbauen. Daten, Layout und Logik werden bei Cocoon klar getrennt.

### 2.1 Umgesetzte Technologien

Für die Verarbeitung der XML-Daten werden folgende Technologien eingesetzt:

#### 2.1.1 XLST

XSLT (Extensible Stylesheet Language Transformation) wird für die Transformation von XML in eine andere Struktur verwendet. Die Regeln für eine solche Transformation werden in sogenannten XSLT-Stylesheet (in der Regel mit Endung \*.xsl) definiert.

#### 2.1.2 XSL-FO

XSL-FO (Extensible Stylesheet Language Formatting Objects) wird vor allem für die Transformation von XML in PDF verwendet. Hierbei erfolgt die Umwandlung in zwei Schritten. Im ersten Schritt wird das XML-Dokument in ein xsl-fo-Dokument umgewandelt. Dieses wird im zweiten Schritt von einem FO-Prozessor in das gewünschte Zielformat (in der Regel PDF) übergeführt

#### 2.1.3 XPATH

XPATH (XML Path Language) ist eine Abfragesprache. Damit können Elemente in einem XML-Dokument (z.B. Elementknoten, Attributknoten ...) für die Weiterverarbeitung eindeutig identifiziert werden.

XPATH wird vor allem innerhalb von XSLT-Stylesheets intensiv verwendet um bestimmte Information zu extrahieren und weiter zu verarbeiten.

### 2.2 Prinzip

Da Cocoon in einen Servlet-Container eingebettet ist, basiert seine Arbeitsweise auf den von (Web)Servern bekannten Request/Response-Zyklus. Ein Request von einem beliebigen Client initiiert eine Reihe von Prozessen, um die Anfrage zu bearbeiten und dann den gewünschten Output als Response an den Client zu senden.

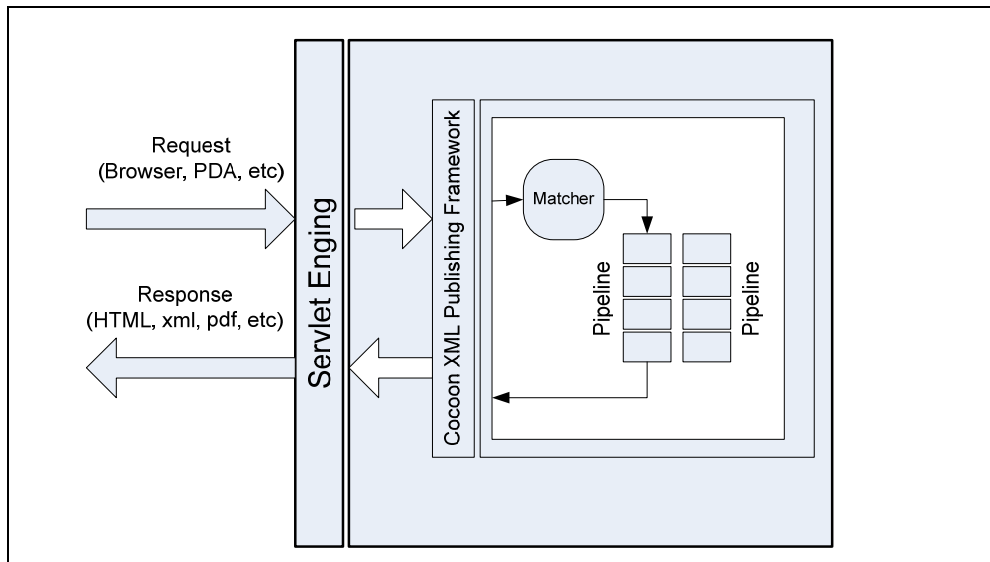


Abb. 1: Architektur Cocoon

### 2.2.1 Sitemap

Die Sitemap spielt eine zentrale Rolle bei Cocoon. In ihr werden Komponenten und Aufrufe konfiguriert und logische Abläufe modelliert. Hier kann zum Beispiel für einen bestimmten eingehenden URI definiert werden, welche Aktion daraufhin geschehen soll.

Bei der Sitemap handelt es sich wiederum um eine XML-Datei, die sich standardmäßig im Wurzelverzeichnis von Cocoon befindet und den Namen sitemap.xmap besitzt. Der Inhalt umfasst drei Hauptbereiche: Der Komponenten-Bereich definiert alle Komponenten, auf die Cocoon Zugriff hat und die eine bestimmte Arbeit verrichten.

Die Hauptaufgaben der Komponenten sind das Lesen, Interpretieren und Umwandeln von XML-Dokumenten und das Erzeugen von bestimmten Zielformaten. Auch die Entscheidungslogik wird im Komponentenbereich der Sitemap festgelegt.

Im Bereich Gruppierungen lassen sich gegebenenfalls einzelne Komponenten auf verschiedene Weisen für eine spezielle Aufgabe kombinieren.

### 2.2.2 Pipeline(s)

Innerhalb des Cocoon Frameworks werden alle Anfragen innerhalb einer sogenannten Pipeline verarbeitet. Eine Pipeline stellt also eine Aneinanderreihung von verschiedenen Verarbeitungsstufen dar. Die Übergabe der Daten zwischen den einzelnen Stufen findet immer mit XML statt und die Verarbeitung wird durch die SAX-Events gesteuert. Man kann sich eine Pipeline als eine Sammlung von Vorschriften vorstellen, die festlegen, wie ein Dokument zu erzeugen, umzuwandeln und abzuschließen ist.

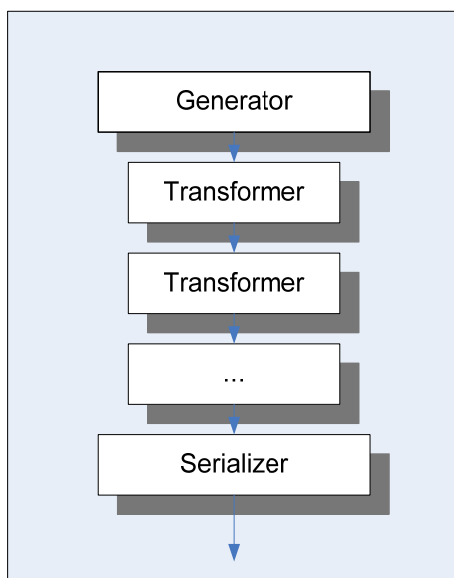


Abb. 2: Cocoon Pipeline

Das Bild zeigt eine typische Pipeline. Zu Beginn werden mittels eines Generators SAX-Events aus einer Datenquelle generiert. Dabei muss die Datenquelle nicht zwangsläufig eine XML-Datei sein. Innerhalb einer Pipeline kann es höchstens einen Generator geben. Darauf können mehrere Transformatoren folgen, deren Anzahl ist beliebig. Das Ende einer Pipeline bildet ein Serializer, welcher aus den SAX-Events wieder ein für den Benutzer lesbares Format generiert. Eine Pipeline kann ebenfalls nur einen Serializer enthalten.

### 2.2.3 Generator

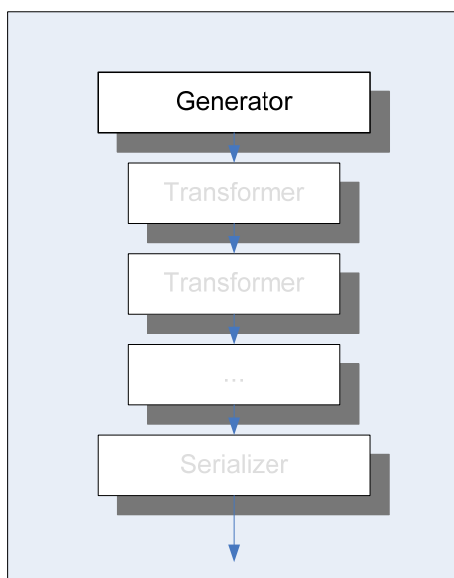


Abb. 3: Cocoon-Generator

Ein Generator ist immer der Startpunkt einer Pipeline und hat die Aufgabe, aus beliebig strukturierten Daten SAX-Events zu erzeugen und in die Pipeline zu senden. Andere Komponenten wiederum können diese SAX-Events benutzen, um bestimmte Aktionen auszuführen.

Ein Generator steht immer zu Beginn der Eventverarbeitung, da ohne XML-Informationen keine Transformation erfolgen kann. Dabei müssen die Daten nicht bereits in einer XML-Struktur vorliegen. Die Daten werden vom Generator in SAX-Events und damit in eine XML-Struktur umgewandelt.

Jede Pipeline, die mit einem Generator beginnt, muss mit einem Serializer abgeschlossen werden.

### 2.2.4 Von Cocoon mitgelieferte Generatoren

Generator	Funktion
(Image-) File-Generator	Liest ein Verzeichnis und generiert daraus ein XML-Dokument. Elemente enthalten z.B. Informationen über Name, Größe und den letzten Zugriff einer Datei
File-Generator	Der Default-Generator. Liest ein XML-Dokument vom lokalen Dateisystem oder einer URL und konvertiert dieses in SAX-Events
HTML-Generator	Dieser Generator liest eine HTML-Datei und generiert SAX-Events im XHTML-Format
Status-Generator	Liefert aktuelle Systeminformationen über Cocoon2 als SAX-Events

### 2.2.5 Transformer

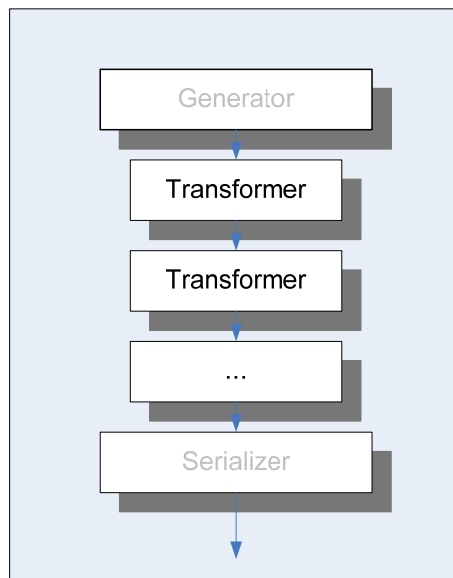


Abb. 4: Cocoon-Transformer

Ein Transformer ist der zentrale Punkt in einer Pipeline und transformiert ein vom Generator in SAX-Events umgewandeltes XML-Dokument in beliebige andere SAX-Events und sendet diese an die nächste Verarbeitungskomponente. Dabei kann das gesamte XML-Dokument verändert (z.B. Seitenlayout erstellen) oder nur auf bestimmte Tags (z.B. nur SQL-Abfragen ausführen) reagiert werden.

Gesteuert wird die Transformation oft über eine externe Datei (z.B. XSLT) die eingelesen wird. In einer Pipeline können mehrere Transformatoren aufgerufen werden.

### 2.2.6 Von Cocoon mitgelieferte Transformer

Transformer	Funktion
XSLT-Transformer	Default-Transformer. Liest ein XSL-Dokument vom lokalen Dateisystem oder einer URL und transformiert den SAX-Stream unter Verwendung des XSL-Dokuments
SQL-Transformer	Führt eine SQL-Abfrage durch und übersetzt das Ergebnis in XML

### 2.2.7 Serializer

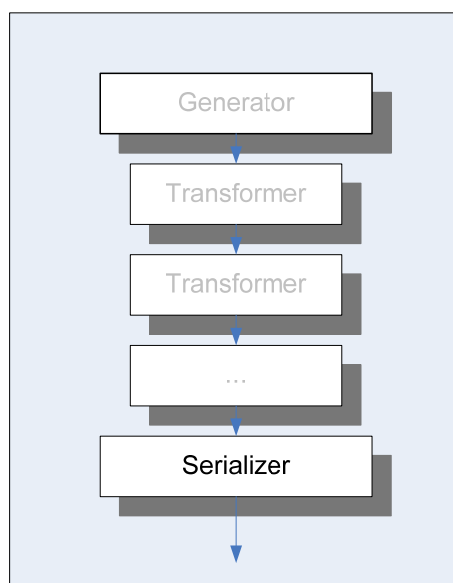


Abb. 5: Cocoon-Serializer

Der Serializer ist der Schlusspunkt in einer Verarbeitungspipeline. Mit den bisherigen Komponenten konnten XML-Dokumente gelesen und bearbeitet werden, allerdings würde dies nichts nützen, da man sich die Daten so nicht betrachten kann. Der Serializer wandelt die SAX-Events wieder in ein statisches Format um (z.B. in eine Datei). Jede Pipeline, die einen Generator enthält, muss mit einem Serializer beendet werden.

## 2.2.8 Von Cocoon mitgelieferte Serializer

Generator	Funktion
(X)HTML-Serializer	Dies ist der Default-Serializer. Serialisiert XML nach (X)HTML
XML-Serializer	Der einfachste Serializer, generiert ein XML-Dokument aus SAX-Events. Wird benutzt, um verschiedene XML-Formate (z.B. SVG, WML) zu serialisieren.
Text-Serializer	Erzeugt als Ausgabe ein einfaches Textdokument
SVG/JPEG-Serializer	Serialisiert ein XML/SVG-Dokument in ein JPEG-Bild
PDF-Serializer	Der PDF-Serializer nimmt XSL-FO SAX-Events als Eingabe und generiert daraus ein PDF-Dokument

## 3 Update Sicherheitshandbuch

In Folge werden die im Rahmen des Projekts durchgeführten Aktualisierungen am IT Sicherheitshandbuch beschrieben. Dabei wurden die Originaldaten des IT Sicherheitshandbuches (SiHa\_Teil\_1.XML, SiHa\_Teil\_2.XML, \*.xsd) übernommen – es wurden also keine inhaltlichen Anpassungen vorgenommen, sondern technische Weiterentwicklungen hinsichtlich der Anwendbarkeit.

### 3.1 Trennung von Inhalt und Layout

Die Transformationsdateien (\*.xsl) wurden überarbeitet, d.h. alle Strukturen die nur eingefügt wurden, um das Layout zu beeinflussen, wie z.B. <br>, <p>, <hr> oder <center>, wurden entfernt.

```
<xsl:template name="createTitlePage">
  <xsl:element name="div">
    <xsl:attribute name="id">titlepage</xsl:attribute>
    <center>
      <xsl:call-template name="createBookTitle"></xsl:call-template>
      <xsl:call-template name="createSubTitle"></xsl:call-template>
      <xsl:text>Version </xsl:text>
      <xsl:value-of select="/descendant::siha:meta/siha:bookversion"/>
      <br/>
      <xsl:value-of select="/descendant::siha:meta/siha:bookdate"/>
      <br/>
      <br/>
      <br/>
      <br/>
      <br/>
      <br/>
      <br/>
      <br/>
      
      <br/>
      <br/>
    </center>
  </xsl:element>
  <hr/>
</xsl:template>
```

Abb. 6: Ausschnitt Korrekturen in Transformationsdatei

Die Transformation erfolgt nun in mehreren Schritten, wobei auf eine saubere Trennung von Inhalt, logischer Struktur und Anzeigeeinformationen (Layout) geachtet wurde.

### 3.2 Sicherheitshandbuch

Die Originaldaten sind nun im Cocoon Framework eingebettet, welches die notwendigen Transformationen für die Darstellung ausführt. Nachfolgend ist die prinzipielle Arbeitsweise dargestellt.

#### 3.2.1 Teil 1

Beim Aufruf der Seite [http://.../OE-SIHA\\_I.html](http://.../OE-SIHA_I.html) wird dieser Request von Cocoon übernommen. Die Verarbeitung des Requests erfolgt in folgenden Schritten:

- Einlesen der XML-Datei (OE-SIHA\_I.xml)
- XML-Transformation mit „2filtered.xsl“
- XML-Transformation mit „filtered2html.xsl“
- Ausgabe des OE-SIHA im html-Format (OE-SIHA\_I.html)

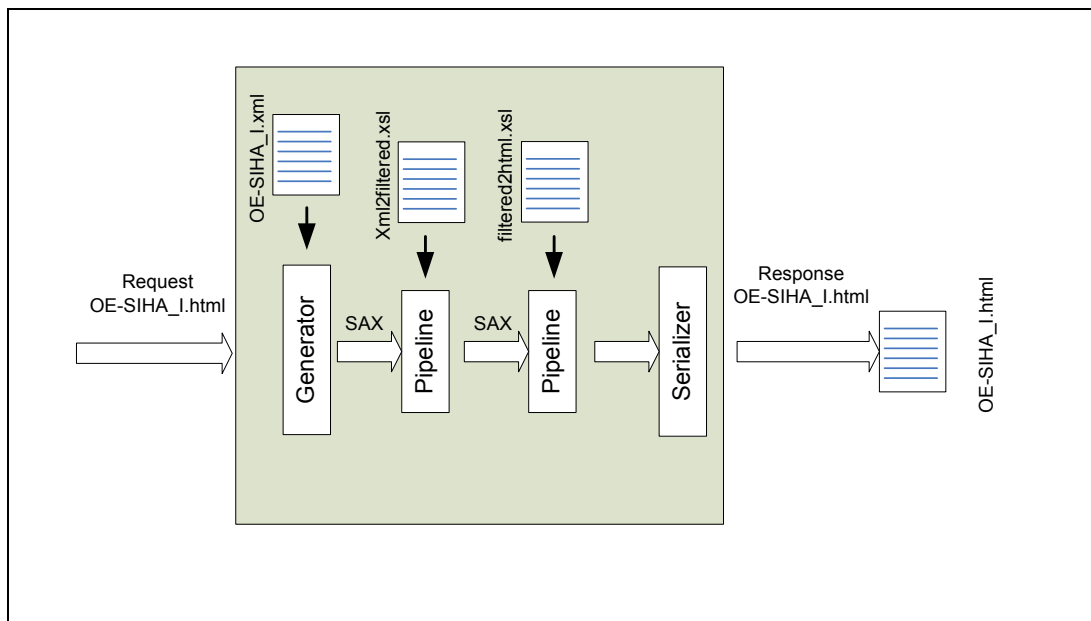


Abb. 7: Pipeline OE-SIHA\_I.html

### 3.2.2 Sitemap

```

<map:match pattern="OE-SIHA_*.html">
  <map:generate src="src/xml/OE-SIHA_{1}.xml"/>
  <map:transform src="src/xsl/xml2filtered.xsl">
  </map:transform>
  <map:transform src="src/xsl/filtered2html.xsl">
    <use-request-parameters>true</use-request-parameters>
    <map:parameter name="resources" value="" />
  </map:transform>
  <map:serialize type="html" />
</map:match>
  
```

Abb. 8: Ausschnitt aus sitemap.xml

### 3.2.3 Teil 2

Beim Aufruf der Seite [http://.../OE-SIHA\\_II.html](http://.../OE-SIHA_II.html) erfolgt die Verarbeitung ähnlich wie beim Aufruf des Teil 1. Es sind die gleichen Komponenten, sowie die gleichen xsl-Dateien beteiligt:

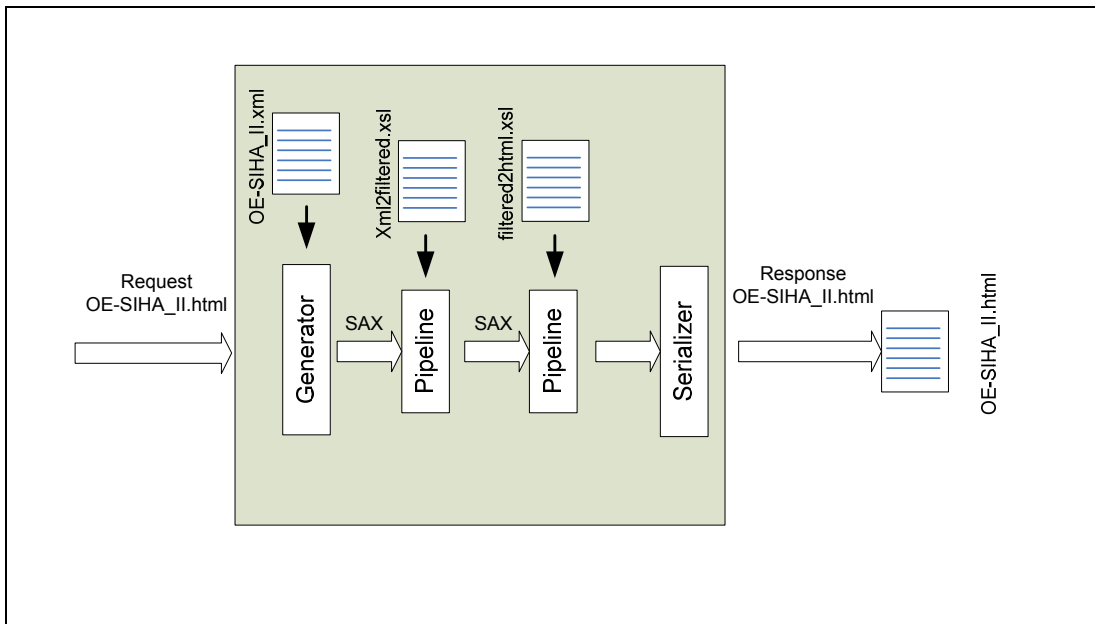


Abb. 9: Pipeline OE-SIHA\_II.html

### 3.3 Checkliste

Für die Generierung der Checkliste werden den xsl-Dateien zusätzlich Parameter übergeben, die spezifizieren, welche Inhalte die Checkliste darstellen soll.

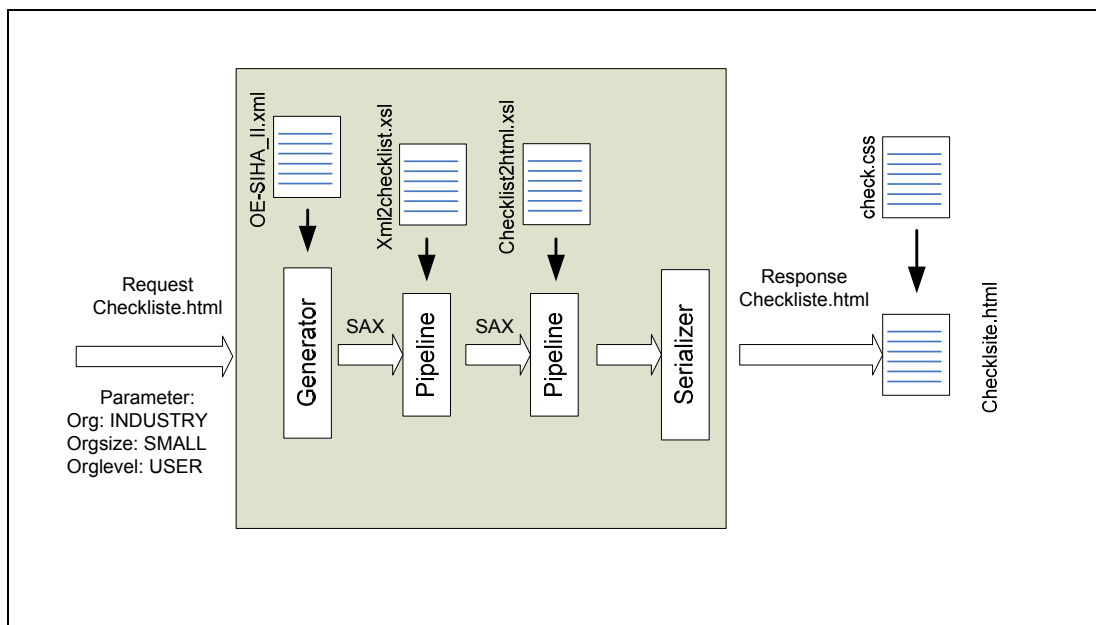


Abb. 10: Pipeline Checkliste.html

### 3.4 Konfiguration

Für die praktische Anwendung im betrieblichen Umfeld ist es sinnvoll, Daten und Anforderungen des Unternehmens oder der Organisation in die Checkliste zu integrieren. Zu diesem Zweck ist es möglich, in einem eigenen Konfigurations-Menü Unternehmer- bzw. Organisations-bezogene Daten einzugeben.

### 3.4.1 Verantwortlichkeit(en)

Damit wird die Eingabe verantwortlicher Personen ermöglicht.

### 3.4.2 Auswahl Sicherheitsmaßnahmen

Das Sicherheitshandbuch umfasst eine umfangreiche Sammlung von Sicherheitsmaßnahmen. Nicht alle Maßnahmen sind für alle Unternehmen von Bedeutung. Deshalb ist es notwendig, die Sammlung von Sicherheitsmaßnahmen den spezifischen Anforderungen und Bedürfnissen des Unternehmens anzupassen.

Die Konfiguration erfolgt in mehreren Schritten.

#### 3.4.2.1 Schritt 1: Katalogauswahl

Hier können Bereiche aus dem Maßnahmenkatalog selektiert werden, für die im nächsten Schritt feiner selektiert werden kann.

<b>1</b>	<b>Bauliche und Infrastrukturelle Maßnahmen</b>	
<input type="checkbox"/>	1.1	Bauliche Maßnahmen
<input type="checkbox"/>	1.2	Brandschutz
<input type="checkbox"/>	1.3	Stromversorgung, Maßnahmen gegen elektrische und elektromagnetische Risiken
<input type="checkbox"/>	1.4	...
<input type="checkbox"/>	1.5	...
<input type="checkbox"/>	1.6	Weitere Schutzmaßnahmen
<b>2</b>	<b>Personelle Maßnahmen</b>	
<input type="checkbox"/>	2.1	Regelung für Mitarbeiter
<input type="checkbox"/>	2.2	Regelung für Einsatz von Fremdpersonal
<input type="checkbox"/>	2.3	Sicherheitssensibilisierung und -schulung
<b>3</b>	<b>IT-Sicherheitsmanagement</b>	
<input type="checkbox"/>	3	IT-Sicherheitsmanagement
<b>4</b>	<b>Sicherheit in der Systementwicklung</b>	
<input type="checkbox"/>	4.1	Sicherheit im gesamten Lebenszyklus eines IT-Systems
<input type="checkbox"/>	4.2	Dokumentation
<input type="checkbox"/>	4.3	Evaluierung und Zertifizierung
<b>5</b>	<b>Systemsicherheit</b>	
<input type="checkbox"/>	5.1	Berechtigungssysteme, Schlüssel- und Passwortverwaltung
<input type="checkbox"/>	5.2	Betriebsmittel und Datenträger
<input type="checkbox"/>	5.3	...
<input type="checkbox"/>	5.4	...
<input type="checkbox"/>	5.5	...
<input type="checkbox"/>	5.6	...
<input type="checkbox"/>	5.7	...
<input type="checkbox"/>	5.8	...
<input type="checkbox"/>	5.9	...
<input type="checkbox"/>	5.10	...
<input type="checkbox"/>	5.11	Kryptografische Maßnahmen
<b>6</b>	<b>Aufrechterhaltung der Sicherheit im laufenden Betrieb</b>	
<input type="checkbox"/>	6.1	Wartung
<input type="checkbox"/>	6.2	...
<input type="checkbox"/>	6.3	...
<input type="checkbox"/>	6.4	Incident Handling
<b>7</b>	<b>Disaster Recovery und Business Continuity Planung</b>	
<input type="checkbox"/>	7.1	Datensicherung
<input type="checkbox"/>	7.2	Strategie und Planung
<input type="checkbox"/>	7.3	Umsetzung und Test

### 3.4.2.2 Schritt 2: Grobauswahl

Im diesem Schritt können für jeden selektieren Maßnahmenkatalog die entsprechenden Themen-bezogenen Bereiche selektiert werden, für die dann im letzten Schritt die entsprechenden konkreten Maßnahmen aktiviert werden können (d.h. als nach dem Sicherheitshandbuch umgesetzt) oder deaktiviert werden können (d.h. als nicht umgesetzt markiert, wenn z.B. Maßnahme nicht relevant ist).

Beispiel für „Bauliche und Infrastrukturelle Maßnahmen“:

<b>1.1</b>	<b>Bauliche Maßnahmen</b>		
<input type="checkbox"/>	INF	1.1	Geeignete Standortwahl
	...	...	...
<input type="checkbox"/>	INF	1.8	Perimeterschutz
<b>1.2</b>	<b>Brandschutz</b>		
<input type="checkbox"/>	INF	2.1	Einhaltung von Brandschutzvorschriften und Auflagen
	...	...	...
<input type="checkbox"/>	INF	2.12	Rauchschutzvorkehrungen
<b>1.3</b>	<b>Stromversorgung</b>		

### 3.4.2.3 Schritt 3: Feinauswahl

In diesem Schritt werden alle Maßnahmen des entsprechenden Kataloges angezeigt. Der Anwender kann nun je nach Anforderung die entsprechenden konkreten Maßnahmen einzeln behandeln.

Weiters können die Maßnahmen mit einem entsprechenden Kommentar versehen werden, welcher dann in der generierten Checkliste angezeigt wird.

Beispiel für „Geeignete Standortwahl“:

<b>INF</b>	<b>1.1</b>	<b>Geignete Standortwahl</b>	
<input type="checkbox"/>	INF	1.1.1	...
<input type="checkbox"/>	INF	1.1.2	...
<input type="checkbox"/>	INF	1.1.3	...
<input type="checkbox"/>	INF	1.1.4	...
<b>INF</b>	<b>...</b>	<b>...</b>	
<input type="checkbox"/>	INF	1.2.1	.
<input type="checkbox"/>	INF	1.2.2	.
	...	...	...
<b>INF</b>	<b>1.8</b>	<b>Perimeterschutz</b>	
<input type="checkbox"/>	INF	1.8.1	.
<input type="checkbox"/>	INF	1.8.2	.
	...	...	

### **3.5 Ausgabeformate**

Für den praktischen Einsatz vor Ort ist es möglich, zwischen verschiedenen Ausgabeformaten für die Checkliste auszuwählen. Folgende Formate werden dabei unterstützt:

- html
- pdf
- rtf
- xhtml
- XML

Weiters ist es für das Unternehmen / die Organisation möglich, durch Anpassung der css-Datei (check.css) ein passendes Layout (Ausgabe Browser, PDA, Drucker o.ä.) zu verwenden. Dabei können auch Unternehmer-spezifische Grafiken für ein Layout im Sinne des Corporate Identity des Unternehmens verwendet werden.

## 4 Praktische Umsetzung

Abhängig von der Größe des Unternehmens und der vorhandenen IT-Infrastruktur, kann das IT-Sicherheitshandbuch sowohl auf einem vorhandenen Servlet-Container in einer Intranet-Umgebung installiert werden, als auch als Stand-Alone Lösung lokal auf einem Rechner. Für die zweite Variante wird ein Servlet-Container in einer Minimal-Ausführung (Tomcat oder Jetty) im Installer mitgeliefert.

### 4.1 Verzeichnisstruktur

Unabhängig welche Lösung Verwendung findet, wird folgende Verzeichnis-Struktur verwendet:

```
.../webapps
  \ ROOT                                // beinhaltet HTML-Dokumente (Frontend)
  | |-- css
  | |-- home
  | |-- images
  | |-- teil_I
  | |-- teil_II
  | |-- teil_III
  | |-- WEB-INF
  | \ index.html
  |-- siha                               // beinhaltet Cocoon Komponenten
  | |-- resources
  | |-- stylesheets
  | |-- teil-1                            // Cocoon-Komponenten Teil 1
  |   |-- src
  |     |-- xml
  |       \ OE-SIHA_I.xml                // Sicherheitshanduch Teil 1
  |       \ OE-SIHA_II.xml              // Sicherheitshanduch Teil 2
  |     |-- xsd
  |       \ OE_IT_SIHB_UPDATE_OE.xsd
  |     |-- xsl
  |       |-- filtered
  |       |-- fo
  |       |-- html
  |       \ checklist2html.xsl
  |       \ filtered2fo.xsl
  |       \ filtered2html.xsl
  |       \ xml2filtered.xsl
  | \ sitemap.xmap
  |-- teil-2                              // Cocoon-Komponenten Teil 2
  |-- teil-3
```

Abbildung 11: webapps-Verzeichnis

### 4.2 Server Lösung (Servlet Container vorhanden)

Sind im Unternehmen / der Organisation eine Server-Infrastruktur sowie das technische Know-How vorhanden, kann das OE-SIHA auf einen Servlet-Container, wie z.B. Tomcat, installiert werden. Da nur Standard-Schnittstellen verwendet werden, kann das Framework dahingehend erweitert werden, dass auf eine vorhandene Datenbank-Infrastruktur zugegriffen werden kann.

#### 4.2.1 Systemanforderungen

- Java Environment

- Java 1.4 oder höher
- Servlet Container
  - Servlet API 2.3
  - z.B. Tomcat 5.20 oder höher

#### 4.2.2 Installation

Für die Installation auf einen Server müssen folgende Dateien installiert werden

- siha.war (cocoon-siha webapplikation)
- siha-html.zip (html Seiten als Frontend zur Webapplikation)

Die \*.war-Datei enthält die Webapplikation und muss im entsprechenden webapps-Verzeichnis des Tomcat installiert werden.

Die \*.zip-Datei enthält alle notwendigen html-Seiten welches als Front-End für die Webapplikation verwendet wird. Diese Dateien können sowohl auf einem Webserver (soweit vorhanden) installiert werden, oder wie bei der Stand-Alone Variante auch im Tomcat eingerichtet werden.

#### 4.2.3 Verzeichnisstruktur

```

.../Tomcat-5.5.20
|-- bin
|-- common
|-- conf
|-- ...
|-- webapps
|   |-- ROOT
|   |-- ...
|   \ siha.war

```

Abbildung 2: Tomcat-Verzeichnis

### 4.3 Client Lösung

Ist im Unternehmen weder eine Server-Infrastruktur noch das Technische Know-How vorhanden, kann durch entpacken eine Minimal-Variante des OE-SIHA installiert werden. Diese Minimal-Lösung installiert auf dem Client-System alle notwendigen Komponenten in einem Verzeichnis. Diese beinhaltet unter anderem einen Servlet-Container, welcher ohne Administration verwendet werden kann, sowie die benötigte Java-Umgebung. Durch die Verwendung des System-Browser ist das OE-SIHA unter der URL

<http://localhost:8085/>

erreichbar. Der Anwender bemerkt zu der Server-Lösung keinen Unterschied.

#### 4.3.1 Systemanforderungen

Für die lokale Installation werden alle benötigten Komponenten mitgeliefert.

### **4.3.2 Installation**

Für die lokale Installation wird ein Installer im msi-Format (Microsoft Software Installation) zur Verfügung gestellt. Diese stellt eine Ausführungsumgebung (Runner) für Installationsroutinen und Microsoft Windows bereit.

Beim Ausführen der Installation werden alle benötigten Komponenten für die lokale Verwendung auf dem System installiert.

## 5 Zusammenfassung und Ausblick

Gerade im Bereich der Informationstechnologien (IT) ist die technologische Entwicklung ständig Änderungen unterworfen. Alte Technologien werden von neuen Entwicklungen abgelöst, die wiederum neue Rahmenbedingungen nach sich ziehen.

Deshalb ist es wichtig, für die Anwendung des Sicherheitshandbuches diesen schnelllebigen Änderungen zu tragen und Möglichkeiten anzubieten auf Änderungen schnell reagieren zu können.

### 5.1 Projektinhalte

#### 5.1.1 Inhaltliche Anpassung an das Betriebliche Umfeld

Für die praktische Anwendung des Sicherheitshandbuches war es notwendig die Inhalte an die relevanten Anforderungen des Unternehmens anzupassen. In der aktuellen Version ist es möglich, über das Konfigurations-Menü nicht benötigte Inhalte der Checkliste auszublenden. Jene Inhalte die für das Unternehmen relevant sind können um eigene Kommentare ergänzt werden.

Parallel dazu ist es möglich benutzerdefinierte Texte in eigenen xml-Dateien in das Cocoon-Framework einzupflegen und diese dann auch in der Ansicht anzuzeigen. Durch diesen Ansatz werden die Standard- und Benutzerdefinierten-Texte getrennt voneinander verwaltet. Damit ist es in Zukunft möglich neue Inhalte durch Ergänzungen in der Sitemap ins Sicherheitshandbuch zu integrieren.

#### 5.1.2 Weiterentwicklung der Online-Version

Durch die Verwendung von Servlet-Technogien war es möglich, in Java und den damit verbundenen Technologien (J2EE, struts, jsp, ...) individuelle Lösungen zu realisieren. Nachteilig für KMUs war bisher das benötigte Know-How für die Programmierung.

Durch die Einführung von Cocoon als Framework bieten sich für die KMUs folgende Vorteile:

- Entwicklung eigener Lösungen ohne Java-Kenntnisse
- Optionale Anbindung verschiedener Datenquellen (DB, LDAP, ...)
- Anpassungen des Layouts des Sicherheitshandbuches an das eigene Unternehmen (Ausstausch der Stylesheets \*.css, Anpassung des Frontends d.h. html-Seiten, Austausch Grafiken)
- Lokale Installation

#### 5.1.3 PDF-Generator (Multichannel Publishing)

Zu den größten Stärken von Cocoon gehört die Fähigkeit, Inhalte in verschiedensten Formaten ohne aufwändige Programmierung oder Erweiterung der Web-Applikation darzustellen.

Normale Servlet-Technologien bieten speziell programmierte Lösungen, die in anderen Fällen nicht mehr verwendet werden können. Cocoon integriert über eigene Schnittstellen vorhandene Lösungen wie z.B. „Apache-XML-Projekt FOP“. Diese werden unabhängig von Cocoon kontinuierlich entwickelt.

Standardmäßig unterstützt Cocoon folgende Ausgabeformate:

- HTML/XHTML
- PDF
- PostScript
- XLS (Excel)
- XML

#### **5.1.4 Ausblick**

Durch die Einführung von einem Framework als Basis für das Sicherheitshandbuch wurde erstmals die Möglichkeit geschaffen, mit einfachen Mitteln die Inhalte an eigene Bedürfnisse anzupassen.

Dadurch können KMUs nun eigene einfache Lösungen für ihr Unternehmen realisieren. Durch die Unterstützung offener Standards und verschiedenen Datenquellen ist es möglich, auch komplexere Anwendungsfälle mit einfachen Mitteln zu realisieren.

## Glossar

Abkürzung	Bedeutung
ASF	<b>Apache Software Foundation</b> Open Source Gemeinschaft, unterstützt zahlreiche Projekte wie z.B. Apache (http-Server), Tomcat (Servlet-Container), Cocoon ...
URI	<b>Uniform Resource Identifier</b> Eindeutige Name (String) für Identifizierung einer Internet-Resource
URL	<b>Uniform Resource Location</b> Eindeutige Pfadangabe zur Adressierung eines Internet-Objekts
A-SIT	<b>Secure Information Technology Center – Austria</b>
PDF	<b>Portable Document Format</b> Offenes Dateiformat von der Firma Adobe Systems
Persistenz	Speicherung von Daten, die nach Abschalten des Rechner-Systems erhalten bleiben

## Referenzen

[COC-1]	<i>Offizielle Website des Cocoon-Projekts</i> <a href="http://cocoon.apache.org">http://cocoon.apache.org</a> abgerufen aus dem WWW am 17.11.2006
[COC-2]	<i>User Guide der Cocoon-Website</i> <a href="http://cocoon.apache.org/2.1/userdocs/index.html">http://cocoon.apache.org/2.1/userdocs/index.html</a> abgerufen aus dem WWW am 17.11.2006
[COC-3]	<i>Cocoon Wiki-Site</i> <a href="http://wiki.apache.org/cocoon">http://wiki.apache.org/cocoon</a> abgerufen aus dem WWW am 17.11.2006
[TOMC-1]	<i>Offizielle Website des Tomcat-Projekts</i> <a href="http://tomcat.apache.org">http://tomcat.apache.org</a> abgerufen aus dem WWW am 17.11.2006

# Historie

<b>Version</b> 0.1	<b>Datum</b> 20.11.2006	<b>Kommentar</b> Entwurf, Festlegung der Inhalte
<b>Autor</b> Edgar Neuherz		
<b>Version</b> 0.2	<b>Datum</b> 19.12.2006	<b>Kommentar</b> Überarbeitung der Inhalte
<b>Autor</b> Edgar Neuherz		
<b>Version</b> 1.0	<b>Datum</b> 21.12.2006	<b>Kommentar</b> Korrektur
<b>Autor</b> Herbert Leitold		
<b>Version</b> 1.1	<b>Datum</b> 22.12.2006	<b>Kommentar</b> Fertigstellung der Inhalte
<b>Autor</b> Edgar Neuherz		