Web 2.0 Anwendung mit ADF barrierefrei entwickeln

Tobias Huber
Virtual7 GmbH
Karlsruhe

Keywords:

ADF Faces, Barrierefreiheit, Accessibility, Screenreader, JAWS, Lunar, Supernova, Best Practice

Einführung

Die Zusammenarbeit mit einem unserer Kunden, der gesetzlich dazu verpflichtet ist barrierefreie Anwendungen zu entwickeln, haben uns als Unternehmen auf diesem Gebiet zu Experten heranreifen lassen. Der Vortrag geht darauf ein moderne ADF Webanwendungen barrierefrei zu entwickeln. Die Anwendung soll sowohl für blinde Menschen mit Hilfe eines Screenreader (JAWS) als auch für schlecht sehende Menschen mit Hilfe einer entsprechenden Anwendung (Supernova) verständlich sein und dennoch dem heutigen Standard einer modernen Webanwendung entsprechen (Cooles Design, Ajax). Regelmäßige Telefonkonferenzen mit dem "Oracle Webcenter A-Team" und weiteren "Oracle Experten" aus dem Bereich Barrierefreiheit haben dazu geführt, dass wir eine Liste mit Best Practices und ToDos zusammenstellen konnten, die notwendig sind für eine barrierefreie Entwicklung. Unsere ADF Anwendungen haben zum einen den kundeninternen Test, der von blinden Menschen durchgeführt wurde, als auch den externen „Barrierefreie Informationstechnik-Verordnungs“ (BItV) Test bestanden.

Ich werde eine Liste mit den Best Practices und ToDos vorstellen, die eine unkomplizierte und schnelle Entwicklung von barrierefreien Anwendungen ermöglicht. Gerade zu Beginn unserer Arbeit hatten wir mit Barrierefreiheit unsere Probleme, weil uns das Wissen dieser Best Practices fehlte. Außerdem möchte ich auf den ADF Screenreader Modus eingehen, der in unserem Projekt vom Kunden zwar nicht gewünscht war, aber dennoch eine sehr gute Möglichkeit bietet, um dem Kunden noch mehr Qualität in Puncto Barrierefreiheit zu bieten.

Was bedeutet Barrierefreiheit im Internet?

Die Barrierefreiheit im Internet bedeutet, dass von jedem auf alle Inhalte Ihrer Website zugegriffen werden kann. Dabei gilt es in Zeiten des Web 2.0 einen Spagat zu schaffen zwischen HTML Code, der cool, dynamisch und frisch aussieht, aber gleichzeitig auch noch die benötigten HTML Richtlinien einhält um von Screenreadern wie JAWS interpretiert werden zu können.

HTML bietet bereits Auszeichnungsmöglichkeiten, die es ermöglichen die Seite für Screenreader verständlicher zu gestalten indem man beispielsweise die Seite in Segmente aufteilen kann oder Navigationselemente gezielt auszeichnen kann. Solche Auszeichnungsmöglichkeiten werden von Oracle ADF momentan noch nicht unterstützt.

Barrierefreie Internetseiten werden in erster Linie für Menschen entwickelt, deren Sehfähigkeiten eingeschränkt sind. So sind es z.B. blinde Menschen, die einen Screenreader wie Jaws als

Unterstützung verwenden. Zusätzlich gibt es auch Anwendungen wie Lunar, die es dem Endanwender mit einer Sehschwäche ermöglichen den Bildschirminhalt zu vergrößern.

Was bietet Oracle ADF um Barrierefreiheit zu unterstützen?

Oracle bietet alle ADF Faces auch in barrierefreier Form an. In dieser barrierefreien Variante verlieren die Komponenten teilweise allerdings an Funktionalität. Es gilt als ein gängiges Verfahren, dass man dem Endanwender einen Link anbietet, der es erlaubt zwischen barrierefreiem Modus und normalem Modus hin- und herzuwechseln. Dieses Verfahren kam für unseren Kunden nicht in Frage, weil man allen Endanwendern die gleiche Seite bieten wollte, unabhängig von einer Behinderung. Dies führte dazu, dass wir uns mit Möglichkeiten beschäftigt haben, ADF auch im nicht barrierefreien Modus, barrierefrei zu gestalten.

Oracle hat uns bei diesem Vorgehen dahingegehend unterstützt, dass es einen regelmäßigen Austausch mit dem WCP A-Team und zusätzlich mit Entwicklern aus dem Barrierefreiheitsumfeld gab. So wurde gemeinsam an Lösungen für verschiedene Probleme gearbeitet. Unter anderem hat Oracle auch einen Patch geliefert, der es uns ermöglicht zu definieren welche ADF Komponenten barrierefrei angezeigt werden sollen und welche im normalen Modus angezeigt werden sollen. Das hat den Vorteil, dass man diese Entscheidung nicht global für die komplette Seite treffen muss.

Wie kann man die Barrierefreiheit testen?

Bei der Prüfung ob eine Anwendung barrierefrei ist können externe Tester mit ins Boot geholt werden, die prüfen ob eine Anwendung den BItV Richtlinien entspricht. Die BItV ist eine Richtlinie, die in mehreren Punkten vorgibt was umgesetzt werden muss, damit eine Seite das Siegel barrierefrei enthält. Dabei unterscheidet die BItV in drei verschiedene Klassen abhängig von der Wichtigkeit eines Punktes. Zusätzlich gab es in unserem Projekt noch Usability Tests, die von blinden Menschen durchgeführt wurden. Das Feedback von diesen Tests ist besonders wichtig, weil es eben zeigt ob eine Anwendung auch von blinden Menschen bedient werden kann. Es war definitiv eine Herausforderung die Anforderungen von beiden zu erfüllen, weil sich die Fehler eben nicht einfach überschnitten haben.

Es ist ratsam, wenn sich auch der Entwickler Werkzeuge wie Jaws oder Lunar zulegt, damit er seine Anwendung auf Barrierefreiheit testen kann. Diese Werkzeuge sind in der Regel sehr teuer, weil der große Abnehmermarkt fehlt. Von den meisten Anbietern gibt es allerdings Demoversionen speziell für Entwickler. So gibt es z.B. von Jaws eine Demoversion, die nach jedem Rechnerneustart für 40 Minuten verwendet werden kann. Von Lunar gibt es eine Version, die 30 Tage lang gültig ist.

Best Practices auf dem Weg zur Barrierefreiheit?

Um eine gute Grundlage einer barrierefreien Website zu erreichen haben wir uns eine Liste mit Punkten erstellt, die wir erfüllt haben, bevor es in den ersten Test ging. Um das Siegel

barrierefrei von dem Test allerdings letzendlich zu erhalten wurden im Anschluss an den ersten Test noch die Verbesserungsvorschläge der Tester implementiert. Dabei handelte es sich im optimalen Fall um kleinere Anpassungen an Textpassagen. Im schlechten Fall wurden Dinge gefordert, für die ARIA (Accessible Rich Internet Application) benötigt wird. Bei ARIA handelt es sich um einen Standard Entwurf des W3C um beispielsweise Auszeichnungen von Navigationselementen oder der Seitenstrukturierung vorzunehmen, damit diese von Screenreadern wie JAWS auch als solche erkannt werden.

Punkt 1 – Fokussieren
Es handelt sich dabei um den Einsatz der Javascript Funktion *focus()*. Fokussieren ist aus zwei Gründen wichtig. Zum einen erlaubt es dem Entwickler einen beschreibenden Text zu definieren, der beim Betreten der Seite vorgelesen wird. Zum anderen dient ein gut gesetzter Fokus auch dazu, dass der Endanwender nach einer durchgeführten Aktion auf das Ergebnis der Aktion verwiesen werden kann. So ist es beispielsweise üblich im Anschluss an eine Suche das Suchergebnis zu fokussieren.

Punkt 2 – Bedienbarkeit per Tastatur
Nach dem Starten von Screenreadern kann der Endanwender mit den Pfeiltasten und der Tabtaste durch die Seite navigieren. Zusätzlich bieten Screenreader verschiedene Shortcuts um Überschriften, Formulare oder Links direkt anzuspringen. Dieser Punkt beschreibt, dass darauf geachtet werden sollte, dass die Seite mit den Pfeiltasten und Tabtasten bedient werden kann. Dies bedeutet, dass alle Links, Formularfelder und Buttons mit der Tastatur angesteuert werden können. Formulare sollten das ADF Attribut “Autosubmit” enthalten, damit festgelegt ist, welche Aktion ausgeführt werden soll, falls der Endanwender in einem Formular auf die Eingabetaste drückt. Zusätzlich bedeutet es, dass darauf geachtet werden sollte, dass die Reihenfolge in der die einzelnen Seitenelemente vorgelesen werden auch einen Sinn ergibt.

Punkt 3 – Einsatz von Überschriften
Da ADF ARIA nicht unterstützt ist es umso wichtiger mit Überschridten zu arbeiten um der Seite eine gewisse Struktur zu vermitteln. Mit dem ADF Tag “PanelHeder” können Überschriften definiert werden. Falls ein Kundenkonzept an einer gewissen Stelle keine Textausgabe vorsieht, sollten versteckte Überschriften werden. Der Inhalt und die Platzierung dieser versteckten Überschriften kann mit Hilfe der Testabteilung erfolgen.

Punkt 4 – Benennung von Bildern
Bei der Verwendung von Bildern muss immer das ADF Attribut “ShortDesc” mit ausfeüllt sein. Es ist besser “ShortDesc” auf einen leeren Wert zu setzen anstatt es nicht zu definieren. Das Problem ist, dass die Screenreader den Dateinamen der Bilder vorlesen, falls “ShortDesc” nicht gesetzt ist. Ist es auf einen leeren Wert gesetzt wird das Bild übersprungen.

Punkt 5 – Benennung von Links
Da Screenreader einen Shortcut anbieten mit dem Links direkt angesprungen werden können gilt es darauf zu achten, dass die Namen der Links nicht identisch sind. Dies tritt zum Beispiel bei einer Suchergebnisliste auf, bei der am Ende eines Suchergebnis oftmals ein Link mit dem Namen “mehr Informationen” steht. Eine technische Lösung haben wir zu diesem Punkt noch nicht. Man kann sich hier mit der Konzeption nochmals zusammensetzen um über eine

konzeptionelle Lösung nachzudenken indem man aussagekräftigere Links verwendet. Diese könnten zum Beispiel “mehr Infos zu Ergebnis x” lauten.

Punkt 6 – Formularfelder
Bei allen Eingabefeldern sollte entweder das Attribut “Label” definiert sein oder es sollte das ADF Tag “OutputLabel” verwendet werden. Beim Label handelt es sich um den Text, der vorgelesen wird, falls der Endanwender mit dem Screenreader ein Formularfeld ansteuert. Mit dem “OutputLabel” Text ist es beispielsweise auch möglich eine Warnungsinformation mit einem Eingabefeld zu verknüpfen, damit diese auch vorgelesen wird. Mit Warnungsinformation ist der Text gemeint, der definiert warum ein Eingabefeld die Validierung nicht bestanden hat.

Punkt 7 – Aktiven Bereich definieren
Bei der Verwendung von Trains, Reitern oder Akkordeonen muss definiert werden welcher Bereich gerade aktiv ist. Dies wurde von uns meist mit einem versteckten Text vor der Komponente gemacht. Dieser Text würde bei Reitern bspw. alle Reiterelemente vorlesen und definieren welcher Reiter gerade aktiv ist.

Punkt 8 – Arbeiten mit versteckten Texten
Bei komplexen und nicht barrierefreien ADF Elementen, wie beispielsweise Tabellen, Navigationsbäumen oder komplexen Diagrammen gilt es mit versteckten Texten zu arbeiten, die vor dem jeweiligen Element platziert werden und es gut beschreiben. Der Beschreibungstext sollte auch hier wiederum mit dem Test abgesprochen werden. Dies ist normalerweise die zuletzt gewälte Lösung und wird von dem Test auch nicht gerne gesehen. Bei versteckten Texten gilt es technisch zu beachten, dass alles was mit CSS auf unsichtbar gesetzt wird auch nicht vorgelesen wird. Texte, die für das menschliche Auge also nicht sichtbar sind und trotzdem vorgelesen werden sollen müssen mit folgendem CSS Code ausgestattet sein.

```
position:absolute;
top:-9999999px;
```

Punkt 9 – Kontrast
Zu guter letzt gilt es zu prüfen ob die Seite auch im schwarz / weiß Modus sichtbar ist. Wir haben für diesen Test Lunar verwendet. Häufig ist es so, dass Buttons nicht als solche erkennbar waren, weil der Rand nicht mehr sichtbar war oder Texte waren verschwunden, weil der Kontrast zwischen dem Text und dem Hintergrund nicht groß genug waren. In diesem Fall gilt es sich einfach die Seite mit einem Tool wie Lunar im schwarz / weiß Modus anzuschauen und gegebenfalls sich mit der Konzeption des Stylguide nochmals absprechen.

Fazit

Oracle bietet mit dem Screenreadermodus eine gute Grundlage um Seiten barrierefrei zu gestalten. Sollte man aus irgendwelchen Gründen auf diesen Screenreadermodus nicht zurückgreifen können gibt es auch ohne diesen Modus Möglichkeiten barrierefreie Seiten zu erstellen. Um das Ziel einer barrierefreien Seite zu erreichen ist viel Kommunikation und Zeit

erforderlich bis man von den Testern eine Freigabe erhält. Gerade die Usability Tests stellen hier ein Problem dar, weil es eben keine wirklichen Spezifikationen gibt an die man sich als Entwickler halten kann.

Mein Schlussfazit zu diesem Thema lautet: “Man wird mit Oracle ADF die Barrierefreiheit erreichen, allerdings wird man keinen Preis wie den “Biene Award” damit gewinnen.”

Contact address:

Tobias Huber
virtual7 GmbH
Zeppelinstraße 2
76185, Karlsruhe

Phone: +49(0)721-61901737
Fax: +49(0)721-61901729
Email huber@virtual7.de
Internet: http://www.virtual7.de