

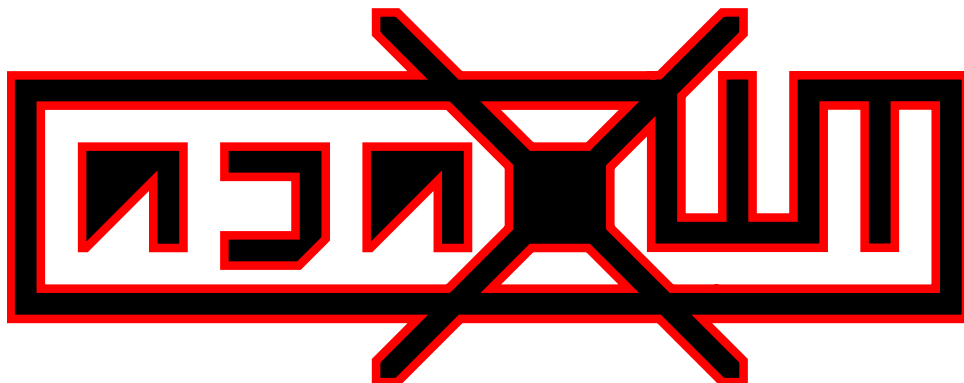
Berufliche Schulen Bretten

JG1

Computertechnik

Lehrer: Herr Hagenlocher

2007/2008



vorgelegt von: **Dennis Felsing**

**Ralf Schaufelberger**

**Andreas Waidler**

2007-12-17

# INHALTSVERZEICHNIS

<b>1</b>	<b>EINLEITUNG</b>	<b>1</b>
<b>2</b>	<b>KOMMUNIKATIONSPROTOKOLL</b>	<b>4</b>
2.1	Events . . . . .	4
2.2	Login und Logout . . . . .	5
2.3	Fenster erzeugen, schließen, und zerstören . . . . .	6
2.4	Synchronisation und Tastendruck . . . . .	7
<b>3</b>	<b>SERVER</b>	<b>8</b>
3.1	Python . . . . .	8
3.2	Startvorgang . . . . .	10
3.2.1	Socket öffnen und lauschen . . . . .	12
3.2.2	Dateien selektieren und laden . . . . .	12
3.3	Sichere Verbindung . . . . .	14
3.3.1	Apache-Konfiguration . . . . .	14
3.4	Beantworten von Anfragen . . . . .	15
3.4.1	Zur Verfügung Stellen von Dateien . . . . .	15
3.4.2	Login . . . . .	15
3.4.3	Logout . . . . .	16
3.4.4	Fenster erzeugen . . . . .	16
3.4.5	Fenster schließen . . . . .	16
3.4.6	Fenster zerstören . . . . .	16
3.4.7	Synchronisation und Tastendruck . . . . .	16
3.5	SSH . . . . .	19
3.6	Kommunikation mit Terminals . . . . .	20
3.7	Beenden des Programms . . . . .	22
<b>4</b>	<b>CLIENT</b>	<b>24</b>
4.1	Verwendete Technologien . . . . .	24
4.1.1	Sprachen und Werkzeuge . . . . .	24
4.1.2	Architekturmodell . . . . .	25
4.1.3	Grundlegende Funktionsweise . . . . .	25
4.2	Fehlerbehebung . . . . .	27

4.3	Startvorgang und Anmeldung . . . . .	29
4.3.1	Grafische Oberfläche . . . . .	29
4.3.2	Anmeldung . . . . .	29
4.3.3	Übergang in den Normalbetrieb . . . . .	30
4.4	Der Desktop . . . . .	31
4.4.1	Virtuelle Desktops . . . . .	31
4.4.2	Fenster . . . . .	31
<b>5</b>	<b>THEMES</b>	<b>35</b>
5.1	Design . . . . .	35
5.1.1	Planung . . . . .	35
5.1.2	Statischer Aufbau . . . . .	37
5.1.3	Stand . . . . .	37
5.2	HTML . . . . .	39
5.3	CSS . . . . .	40
5.4	Umsetzung . . . . .	42
5.4.1	Loginseite . . . . .	42
5.4.2	Desktopseite . . . . .	43
<b>6</b>	<b>QUELLTEXT</b>	<b>44</b>
6.1	Server . . . . .	44
6.1.1	ajaxwm.py . . . . .	44
6.1.2	Server.py . . . . .	47
6.1.3	Files.py . . . . .	52
6.1.4	Session.py . . . . .	55
6.1.5	Loop.py . . . . .	57
6.1.6	Window.py . . . . .	59
6.1.7	Terminal.py . . . . .	62
6.1.8	Singleton.py . . . . .	69
6.1.9	Error.py . . . . .	70
6.2	Client . . . . .	72
6.2.1	AjaxWM_Manager.js . . . . .	72
6.2.2	AjaxWM_Login_Screen.js . . . . .	92
6.2.3	AjaxWM_Window.js . . . . .	97
6.2.4	AjaxWM_Template.js . . . . .	111
6.2.5	index.html . . . . .	121
6.2.6	default.css . . . . .	122
6.3	Themes . . . . .	124
6.3.1	lim . . . . .	124
6.3.2	winxp . . . . .	126



# KAPITEL 1

## EINLEITUNG

*You need to build a system that is futureproof;  
it's no good just making a modular system.  
You need to realize that your system is just  
going to be a module in some bigger system to come,  
and so you have to be part of something else,  
and it's a bit of a way of life.*  
Tim Berners-Lee, at the WWW7 conference

*Am Anfang war TELNET.* Telnet ist ein Netzwerkprotokoll, dass das Administrieren von Rechnern über ein Netzwerk ermöglicht. Der gesamte Datenverkehr ist dabei unverschlüsselt und telnet eignet sich somit nicht zur sicheren Nutzung in unsicheren Netzwerken, wie dem Internet.

*Dann kam SSH.* Eine Verbindung mit Secure Shell ist verschlüsselt. SSH-2 gilt heute noch als sicher. Es benötigt jedoch zum Aufbau einer Verbindung eine Client-Software, die nicht auf jedem Rechner anzutreffen ist und deren Installation nicht möglich ist. Häufig ist auf Rechnern, die man nicht sein Eigen nennt, nur eine restriktive Nutzung möglich. Zum einen hat man nicht immer die Berechtigung eigene Programme auszuführen. Zum anderen bauen die PCs in Schulen, Universitäten, und Firmen häufig über einen Proxy eine Verbindung in das Internet auf. Dieser Proxy kann dann aus Sicherheitsgründen den Download von bestimmten Dateien oder Dateien mit bestimmten Dateieindungen blockieren. Solch ein begrenzter Zugang beschränkt die Internetkonnektivität meist auf wenige Ports, wie zum Beispiel 80 (HTTP), 443 (HTTPS), und 21 (FTP), wodurch man zwar immer noch im Web surfen kann, andere Dienste aber unnutzbar sind. Für SSH vorgesehen ist Port 22. Eine Verbindung ist nur dann möglich, wenn man auf dem Server den Port, auf dem der SSH Daemon lauscht, ändert, wozu man administrative Berechtigungen benötigt.

*Diese Probleme löst AjaxTerm.* AjaxTerm besteht aus einem Server und einem auf HTML, CSS, und AJAX aufgebauten Client. Dieser ist somit in sämtlichen aktuellen graphischen Webbrowsern<sup>1</sup> lauffähig. Das Installieren eines Clients entfällt.<sup>2</sup> Damit

---

<sup>1</sup>Dazu zählen beispielsweise Firefox, Opera, Konqueror, Safari, und Internet Explorer.

<sup>2</sup>Es wird angenommen, dass bereits ein graphischer Browser mit entsprechenden Fähigkeiten auf

eine verschlüsselte und somit sichere Verbindung zustande kommt, muss auf dem als Server fungierenden Computer ein HTTP Server die Rolle eines Proxys annehmen.

Will man in AjaxTerm parallel mehrere Terminals nutzen, so muss für jedes ein neues Browser-Fenster oder, sofern der Webbrowser dazu in der Lage ist, ein neuer Tab geöffnet und erneut eingeloggt werden. Ein mühseliger Prozess, der durch das Wechseln zwischen Fenstern bzw. Tabs zum Erreichen der Terminals noch einen unangenehmen Nachgeschmack erhält.

*Jetzt ist es Zeit für ajaxWM.* AjaxWM basiert auf Ideen und Code von AjaxTerm, ist jedoch ein Window Manager und Terminal Emulator in einem. Die Nutzung ist komfortabler als bei AjaxTerm, da ein einmaliges Einloggen ausreicht um mehrere Fenster erstellen zu können. Das Aussehen des Clients lässt sich mit Themes an die eigenen Vorstellungen anpassen. AjaxWM ist, im Gegensatz zu AjaxTerm, dass nur mit dem latin1-Encoding funktioniert, unicodefähig.

Das ajaxWM-Projekt und sein Subversion-Repository sind zu finden auf Sourceforge. Da es noch keine öffentlichen Releases gibt, muss man die aktuelle Version von ajaxWM per SVN beziehen.<sup>3</sup> Zusätzlich benötigt man einen Theme, zur Zeit steht nur lim zur Verfügung, den man auf die selbe Weise erhält.<sup>4</sup>

Diese Ausarbeitung zum Thema ajaxWM ist in fünf Teile gegliedert. Das Kapitel über das Protokoll stammen von mir, Dennis Felsing, und Andreas Waidler, das über den Server von mir, das über den Client von Andreas Waidler und das über Themes von Ralf Schaufelberger. Es wird auf die Grundlagen der verwendeten Sprachen und primär auf die Funktionsweise eingegangen. Den Abschluss stellen der eigentliche Programmcode auf dem aktuellen Stand und eine Auflistung der Dateien und ihrer Autoren dar.

*Dennis Felsing*

---

dem Client-Rechner installiert ist.

<sup>3</sup>svn co <https://ajaxwm.svn.sourceforge.net/svnroot/ajaxwm/ajaxwm/tags/0.2.0> ajaxwm

<sup>4</sup>svn co <https://ajaxwm.svn.sourceforge.net/svnroot/ajaxwm/themes/lim/tags/0.2.0>  
ajaxwm/src/themes/0.2.0

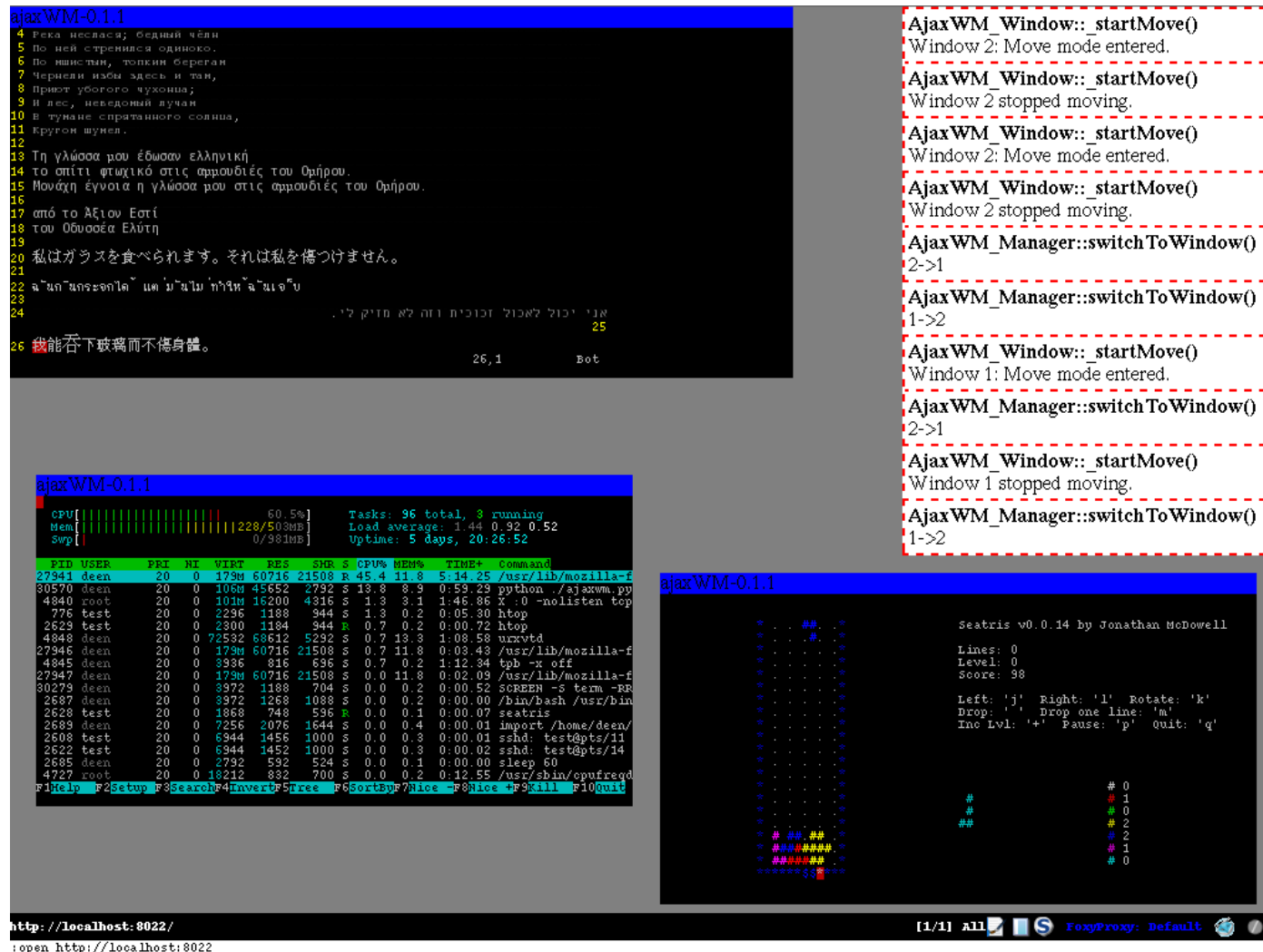


Abbildung 1.1: ajaxWM in Aktion: Eine Textdatei mit verschiedenen Schriftsystemen im Texteditor vim, der Prozessmonitor htop, und der Tetrisklon seatris

## KAPITEL 2

### KOMMUNIKATIONSPROTOKOLL

*Kommunikationswissenschaft -  
die Lehre von den Missverständnissen.*

Markus M. Ronner  
„Treffende Pointen zu Geld und Geist“

#### 2.1 EVENTS

Die Kommunikation zwischen Client und Server wird über Events gesteuert. Es gibt die folgenden sieben Events: Login, Logout, Fenster erzeugen, Fenster schließen, Fenster zerstören, Synchronisation, Tastendruck.



## 2.2 LOGIN UND LOGOUT

Jeder Client, der eine Session aufbauen will, sendet dem Server ein Login-Event. Die Anfrage beinhaltet den Host, auf den zugegriffen werden soll, den Benutzernamen und das Passwort. Der Server beantwortet bei erfolgreicher Anmeldung mit einer Session-ID, andernfalls mit einem der folgenden negativen Fehlercodes: -1 bei falschem Benutzernamen oder Passwort, -2, wenn keine Verbindung zum angegebenen Host aufgebaut werden kann.

Die Session-ID muss bei jeder weiteren Anfrage mitgesendet werden und stellt die Authentizität sicher.

Das Logout-Event wird beim Beenden der Session durch den Client gesendet und enthält lediglich die Session-ID.

## 2.3 FENSTER ERZEUGEN, SCHLIESSEN, UND ZERSTÖREN

Beim Erstellen eines neuen Fensters sendet der Client eine Anfrage. Er erhält bei erfolgreicher Erzeugung eines neuen Fensters dessen ID zurück, ansonsten eine -1.

Die Fenster-ID muss bei jeder weiteren Anfrage, die sich auf ein Fenster bezieht, mitgesendet werden.

Beim Schließen wird das Fenster „gebeten“ sich zu beenden. Bei Erfolg wird eine 1, ansonsten eine 0 zurückgegeben. Ähnlich arbeitet das Zerstören, dabei wird das Fenster jedoch gewaltvoll beendet und ein Rückgabewert entfällt.

## 2.4 SYNCHRONISATION UND TASTENDRUCK

Bei der Synchronisation wird der Inhalt des Fensters in Form eines XML-Dokuments zurückgegeben. Optional kann eine neue Fenstergröße angegeben werden.

Beim Tastendruck werden zusätzlich bei der Anfrage eine oder mehrere Tasten übertragen.

## KAPITEL 3

### SERVER

#### 3.1 PYTHON

*Python is a truly wonderful language.  
When somebody comes up with a good idea  
it takes about 1 minute and five lines  
to program something  
that almost does what you want.  
Then it takes only an hour  
to extend the script to 300 lines,  
after which it still does almost what you want.*  
Jack Jansen, 1992-07-08

Der Server von ajaxWM wurde und wird komplett in Python geschrieben. Python ist eine hauptsächlich von Guido van Rossum seit Anfang der 1990er Jahre entwickelte Scriptsprache. Der Name stellt eine Anlehnung an Monty Python dar.

Der Slogan von Python beschreibt bereits einen der größten Vorteile gegenüber viele anderen Sprachen: “Batteries included“ - Dank einer sehr großen Standardbibliothek werden Module für verschiedenste Einsatzgebiete direkt mitgeliefert. Dadurch reduzieren sich die Abhängigkeiten des eigenen, in Python geschriebenen Programms.

Aufgrund der einfachen Syntax ist es ein Leichtes in Python geschriebene Programme übersichtlich zu halten.

Die Sprache bietet Duck Typing<sup>1</sup>, beziehungsweise verfügt nicht über statische Typisierung. Der Typ einer Variable wird erst während der Laufzeit aus dem zugewiesenen Inhalt festgelegt. Dies erleichtert die Programmierung, zerzt jedoch gleichermaßen an der Performance. Desweiteren stellt dieses Verhalten eine schwierig aufzuspürende Fehlerquelle dar, weil die Typen der Parameter, die einer Methode beim Aufruf übergeben werden, nicht festgelegt werden können. Ändert man nun die Typen der zu übergebenden Parameter, nicht aber den Aufruf, so wird nicht sofort ein Fehler

---

<sup>1</sup>“When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.“ - James Whitcomb Riley

ausgegeben und man erkennt häufig nicht den Grund für den erst später auftretenden Fehler.

## 3.2 STARTVORGANG

*This story gives a straightforward account of  
the three Minami sisters' normal, everyday lives.*

*Please have no expectations beyond that.*

*Also, please watch with the lights on  
and sit three metres from the TV.*

Chiaki Minami from Minami-ke

Damit ajaxWM lauffähig ist, ist lediglich Python<sup>2</sup> nötig. Pyco, ein JIT-Compiler<sup>3</sup> für Python-Code, der optional mit ajaxWM verwendet werden kann, beschleunigt den Server enorm. Da Linux die Hauptentwicklungsplattform ist, läuft der ajaxWM-Server darauf relativ stabil. Jedes andere Betriebssystem, unter dem Python lauffähig ist und das Threading und PTTYs unterstützt<sup>4</sup>, sollte auch ohne beziehungsweise mit geringem Arbeitsaufwand nutzbar sein.

Beim Ausführen des Server-Programms, das im Normalfall als `ajaxwm.py` zur Verfügung steht, hat man eine Auswahl an übergebbaren POSIX-konformen Parametern zu treffen. Ein `-h` dient zur Anzeige des folgenden Hilfetexts:

Listing 3.1: Hilfetext

```
1 r0q@kana /home/r0q $ ./ajaxwm.py
2 usage: ajaxwm.py [options]
3
4 options:
5  -h, --help            show this help message and exit
6  -d, --daemon          run in background
7  -p PORT, --port=PORT  set the TCP port (default: 8022)
8  -l, --log             log requests to stderr (default: quiet mode)
9  -P PIDFILE, --pidfile=PIDFILE
10                      set the pidfile (default: /var/run/ajaxwm.pid)
11  -U UID, --uid=UID     set the user id
12  -u, --uncompressed    responds not compressed (lower system load, higher
13                      network load)
14  -s SSHCMD, --sshc=SSHCMD
15                      ssh client to use
16 r0q@kana /home/r0q $
```

Das ganze auf Deutsch:

`-h` oder `--help`: Hilfetext anzeigen

`-d` oder `--daemon`: Im Hintergrund laufen: Der Server läuft, man kann jedoch im Terminal weiterarbeiten.

`-p PORT` oder `--port=PORT`: Den Port, auf dem der Socket lauscht, festlegen. Standardwert ist 8022.

---

<sup>2</sup>Getestet mit Python 2.4 und 2.5

<sup>3</sup>JIT steht für Just in Time

<sup>4</sup>\*BSD, Cygwin unter Windows, ...

-l oder --log: Anfragen auf der Standardfehlerausgabe ausgeben.<sup>5</sup> Andernfalls werden Anfragen nicht geloggt.

-P PIDFILE oder --pidfile=PIDFILE: Die Datei, in der die Process-ID des Programms geschrieben werden soll. Diese PID ist dazu notwendig dem Programm, wenn es bereits läuft, Signale, wie einen SIGTERM<sup>6</sup>, zu senden.

-U UID oder --uid=UID: Festlegen des Users, als der das Programm laufen soll, mittels der User-ID. Nur möglich, sofern es als root ausgeführt wurde. Dadurch gibt das Programm seine Rechte ab und die Nutzung ist sicherer, da eine Sicherheitslücke im Programm nur die Dateien des aktuellen Users gefährdet, nicht wie sonst das gesamte System. Standardmäßig behält das Programm die UID, unter der es ausgeführt wurde.

-u oder --uncompressed: Im Normalfall werden die Antworten auf Anfragen mit gzip komprimiert. Hierdurch kann dieses Verhalten deaktiviert werden, was insbesondere in lokalen Netzwerken von Vorteil ist. Unkomprimierte Antworten benötigen nämlich weniger Rechenleistung, dafür steigt die Netzwerklast.

-s SSHCMD oder --sshcmt=SSHCMD: Auszuführender SSH Clients, im Normalfall ssh.

Eine Manpage<sup>7</sup> wurde noch nicht geschrieben, ist jedoch für die Zukunft geplant.

Beim Starten im Vordergrund (nicht als Daemon) wird nur die Adresse ausgegeben, auf die man im Webbrowser zugreifen kann. Beim Starten im Hintergrund wird zusätzlich noch die Process-ID ausgegeben, damit man den Prozess wieder beenden kann.<sup>8</sup> Ist der Port, der verwendet werden soll, bereits belegt, so wird dies ausgegeben und der Server beendet.

### Listing 3.2: Testlauf

```
1 r0q@kana /home/r0q $ ./ajaxwm.py -d
2 http://localhost:8022/ PID: 6910
3 r0q@kana /home/r0q $ ./ajaxwm.py
4 http://localhost:8022/
5 port 8022 already in use
6 r0q@kana /home/r0q $ kill 6910
7 r0q@kana /home/r0q $ ./ajaxwm.py
8 http://localhost:8022/
```

Wenn der Server gestartet wird, werden lediglich zwei Hauptaufgaben erledigt, bevor er in einen passiven Zustand fällt und auf den Verbindungsaufbau von Seiten eines Clients wartet: einen Socket öffnen und auf Anfragen lauschen und die Dateien,

---

<sup>5</sup>Im Normalfall stellt der Monitor die Standardfehlerausgabe dar. Mit `./ajaxwm -l 2> anfragen.txt` kann man sie beispielsweise auf die Datei `anfragen.txt` umleiten.

<sup>6</sup>Der Begriff SIGTERM kommt von SIG für Signal und TERM für Terminieren und beendet den Programmprozess.

<sup>7</sup>Als Manpage (Manual Page) bezeichnet man die in unixoiden Betriebssystemen eingesetzten Hilfeseiten.

<sup>8</sup>Weitere Informationen über das Beenden des Servers unter Beenden des Programms.

die Clients zur Verfügung gestellt werden sollen, in den Arbeitsspeicher laden und dabei selektieren und eventuell die Inhalte der Dateien verändern.

### 3.2.1 SOCKET ÖFFNEN UND LAUSCHEN

AjaxWM verwendet das von Anthony Lesuisse<sup>9</sup> stammende QWeb-Framework. Es stellt einen WSGI<sup>10</sup>-Server zur Verfügung. Möglicherweise wird das Framework jedoch in nächster Zeit durch ein anderes ersetzt. Performance-Tests sind zur Auffindung eines geeigneteren Frameworks geplant.

Der WSGI-Server sorgt dafür, dass alle gültigen Anfragen, die am geöffneten Socket eingehen, an die Server-Klasse weitergeleitet werden. Diese Klasse hat dann die Aufgabe eine Antwort zu generieren, dazu aber später unter Beantworten von Anfragen mehr.

### 3.2.2 DATEIEN SELEKTIEREN UND LADEN

Die Files-Klasse ist für das Auswählen und Laden der Dateien zuständig. Sie wird später durch ein File-Modul ersetzt werden, das weitaus mehr Fähigkeiten bieten wird. Bisher werden lediglich rekursiv alle Dateien im Ordner *htdocs* und alle Themes, die mit der verwendeten Version von ajaxWM kompatibel sind, zur Verfügung gestellt. Letzteres funktioniert dadurch, dass die Themes, die im Ordner *themes* liegen, in weitere Ordner mit den Versionsnummern unterteilt werden. Beispielsweise der Ordner des Themes *lim* (less is more) liegt unter *themes/0.2.0*. Der Server weiß, dass der von ihm zur Verfügung gestellte Client mit allen Themes, deren Versionsnummer zwischen 0.1.0 und 0.3.0 liegt, lauffähig ist und lädt *lim*. Würde sich aber in Version 0.4.0 der Aufbau von Themes ändern, so wird ein zu dieser Version gehörender Theme nicht geladen, da er nicht zwischen 0.1.0 und 0.3.0 liegt. Beim Laden wird weiterhin eine Datei *themes.xml* on-the-fly im Arbeitsspeicher erstellt. Dadurch kann der Client herausfinden, welche Themes ihm zur Verfügung gestellt werden und dem User eine Auswahlmöglichkeit bieten. Die Datei sieht dann beispielsweise so aus:

Listing 3.3: Verfügbare Themes

```
1 <?xml version="1.0" ?>
2 <ajaxwm>
3   <theme author="Andreas Waidler" email="andreaswaidler@arcor.de" name="lim">
4     <path>
5       /themes/lim/lim.xml
6     </path>
7     <description>
8       Less Is More. Extremely minimalistic theme.
```

---

<sup>9</sup>Anthony Lesuisse ist – nebenbei erwähnt – auch der Autor von AjaxTerm.

<sup>10</sup>Web Server Gateway Interface



```
9         </description>
10     </theme>
11     <theme author="Anderer Autor" email="irgendeine.email@adres.se" name="bla">
12         <path>
13             /themes/bla/bla.xml
14         </path>
15         <description>
16             Ein Beispieltheme.
17         </description>
18     </theme>
19 </ajaxwm>
```

Das File-Modul wird zusätzlich über eine Kompressionsfunktion verfügen. Dabei können wahlweise JavaScript-Dateien platzsparender übertragen werden, indem Kommentare und nutzlose Zeilen entfernt werden. Möglich ist auch, alle JavaScript-Dateien in einer zusammenzufassen, wodurch ein enormer Geschwindigkeitsschub beim Initialisieren des Clients erreicht werden würde, da die Übertragung von vielen einzelnen, kleinen Dateien stets mehr Zeit in Anspruch nimmt als die Übertragung einer großen Datei. Um nicht bei jedem Neustarten der Serveranwendung die Kompressionsalgorithmen wiederholen zu müssen, werden die komprimierten Dateien gespeichert werden. Bei einem erneuten Starten mit aktivierter Kompression kann überprüft werden, ob sich die unkomprimierten JavaScript-Dateien seit der letzten Kompression verändert haben. Ist dies nicht der Fall, entfällt eine erneute Kompression.

Diese Kompressionsmethoden haben jedoch einen entscheidenden Nachteil, weshalb sie unbedingt optional angeboten werden müssen: Auf Seiten des Clients kommt ein für Menschen schwer lesbarer und unkommentierter Code an. Dies erschwert die Fehlersuche.

Eine weitere Funktion, die das Debuggen des JavaScript-Codes erleichtern wird, werden die DEBUG-Blöcke, die im JavaScript-Code in Form von speziellen Kommentaren angegeben werden können und vom File-Modul dann, entsprechend dem gewählten Parameter, im Code beibehalten bleiben oder entfernt werden. Somit ist eine Abwägung zwischen Performance (normaler Betrieb) und erweitertem Debugging (Testbetrieb) möglich.

### 3.3 SICHERE VERBINDUNG

*Ich glaube, die meisten Menschen  
wissen gar nicht was ein Rootkit ist,  
warum sollen sie sich also darum kümmern?*

Thomas Hesse, Präsident Global Digital  
Business-Abteilung bei Sony BMG

Standardmäßig lässt sich ajaxWM nur lokal nutzen. Dieses Verhalten ist beabsichtigt, da der Server über keine eigene Verschlüsselung bei der Kommunikation verfügt. Um ajaxWM nach außen hin anzubieten, ist es nötig einen lokalen Proxy laufen zu lassen. In diesem Fall den Apache WebServer, der eine Proxy-Weiterleitung mit HTTPS bietet. HTTPS ist mit jedem aktuellen Browser nutzbar und verwendet SSL/TLS zur Verschlüsselung. Dadurch ist, im Gegensatz zu HTTP, asynchronen Schlüsseln sei Dank, ein Mitlesen der Kommunikation im Klartext durch alle Stationen, die zwischen Client und Server liegen, nicht möglich.

#### 3.3.1 APACHE-KONFIGURATION

Eine VHost-Konfigurationsdatei für Apache kann wie folgt aussehen:

Listing 3.4: /etc/apache2/vhosts.d/ajaxwm.conf

```
1 Listen 443
2 NameVirtualHost *:443
3
4 <VirtualHost *:443>
5     ServerName localhost
6     SSLEngine On
7     SSLCertificateKeyFile ssl/apache.pem
8     SSLCertificateFile ssl/apache.pem
9
10    ProxyRequests Off
11    <Proxy *>
12        AuthType Basic
13        AuthName "ajaxWM access"
14        AuthUserFile /etc/apache2/htpasswd
15        Require user r0q
16        Order deny,allow
17        Allow from all
18    </Proxy>
19    ProxyPass /ajaxwm/ http://localhost:8022/
20    ProxyPassReverse /ajaxwm/ http://localhost:8022/
21 </VirtualHost>
```

### 3.4 BEANTWORTEN VON ANFRAGEN

*Konata: Skills you learn from playing  
video games have no use later in life.*

*Kagami: You're living proof of that.*

Lucky Star

Sowohl der Inhalt eines Terminals in Form einer XML-Datei als auch reguläre Dateien werden, sofern nicht per Parameter deaktiviert, mit gzip komprimiert übertragen.

Die Bedeutungen der jeweiligen Antworten sind dem Kapitel Kommunikationsprotokoll zu entnehmen.

#### 3.4.1 ZUR VERFÜGUNG STELLEN VON DATEIEN

Wie bereits unter Dateien selektieren und laden beschrieben, werden die Dateien in den Arbeitsspeicher geladen. Das *htdocs*-Verzeichnis dient als root, was bedeutet, dass die Datei *htdocs/index.html* als */index.html* verfügbar gemacht wird. Intern werden die Dateien in einem Wörterbuch (assoziatives Array) gespeichert. Wird versucht auf nicht vorhandene Dateien zuzugreifen, wird der HTTP-Statuscode 404 (Not Found) zurückgegeben.

#### 3.4.2 LOGIN

Bei einer Login-Anfrage werden der übergebene Benutzername und das Passwort auf Gültigkeit überprüft. Der Benutzername muss dem Muster eines UNIX-Usernames entsprechen, darf also nur aus alphanumerischen Zeichen, Unterstrichen, Punkten, und Dollarzeichen bestehen. Beim Passwort wird lediglich geprüft, ob kein leerer String übergeben wurde.

Anschließend wird eine zufällige, noch nicht belegte, Session-ID generiert und die Korrektheit von Benutzername und Passwort durch einen Login-Versuch sichergestellt. Das Festlegen des Zielrechners ist noch nicht möglich, da der Client keine derartige Information überträgt.

Es wird nicht nur eine Session erstellt, sondern auch eine Instanz der Loop-Klasse. Diese überprüft, ob innerhalb der letzten 10 Minuten kein Zugriff auf die soeben erstellte Session mehr hergestellt wurde. Dies würde bedeuten, dass die Verbindung zum Client unterbrochen wurde und dieser sie nicht wieder aufbauen konnte. In diesem Fall wird die Session beendet.

### 3.4.3 LOGOUT

Der Logout ist das Gegenstück zum Login. Er beendet die angegebene Session, sofern sie existiert. Da dies zur Zeit für jeden möglich ist, ist es wichtig, dass beim Login eine zufällige Session-ID zwischen 0 und der maximalen Größe eines Integers auf dem jeweiligen System<sup>11</sup> zugewiesen wird.

### 3.4.4 FENSTER ERZEUGEN

Um die Performance zu erhöhen, wird die Fenster-ID immer um eins hochgezählt bis sie 64 erreicht hat, auch wenn vorige Fenster bereits beendet wurden. Sind bereits 64 Fenster erstellt worden, wird stattdessen nach nicht mehr geöffneten Fenstern gesucht und das erste gefundene neu initialisiert.

### 3.4.5 FENSTER SCHLIESSEN

Das Schließen von Fenstern ist das Gegenstück zum Erzeugen. Das im Fenster laufende Programm wird lediglich gebeten, nicht gezwungen, sich zu beenden.

### 3.4.6 FENSTER ZERSTÖREN

Das Zerstören von Fenstern ist noch nicht integriert. Es wird vermutlich mit SIGKILL arbeiten, während das Schließen des Fensters ein SIGTERM bewirkt.

### 3.4.7 SYNCHRONISATION UND TASTENDRUCK

Diese beiden, nahezu identischen Events erreichen den Server am häufigsten. Eine Synchronisation ermöglicht es die Größe eines Fensters zu verändern. Dabei wird dem Fenster, der dem Fenster zugehörigen Instanz der Terminal-Klasse, und dem Terminal selbst die Veränderung mitgeteilt.

Durch das Tastendruck-Event werden die übermittelten Tasten in das Terminal geschrieben. Anschließend wird für einen kurzen Zeitraum gewartet, damit das im Terminal laufende Programm diese verarbeiten kann. Dieser Zeitraum ist so gewählt, dass er einer Shell die Verarbeitung erlaubt.

Bei beiden Events wird jetzt der Inhalt des Terminals erneut ausgelesen und die Terminal-Klasse generiert das zurückzugebende XML-Dokument. Ist dieses jedoch

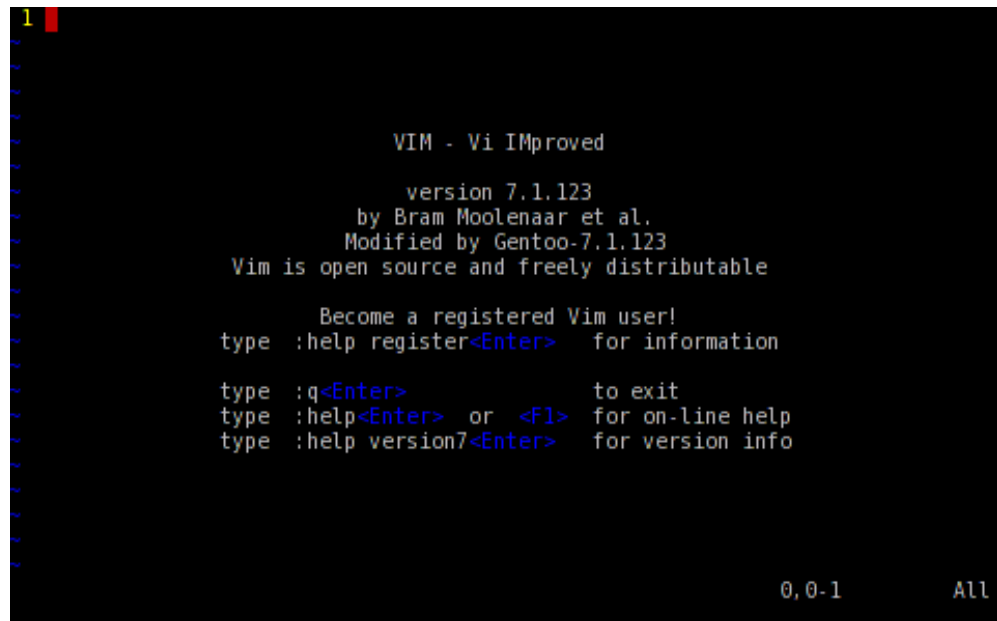
---

<sup>11</sup>Auf einem x86-PC ist dies 2147483647, also  $2^{31} - 1$ .

identisch mit dem letzten übertragenem, so wird mit einem *idem* signalisiert, dass das Terminal sich nicht verändert hat.

### Listing 3.5: Rückgabe bei verändertem Inhalt

```
1 <?xml version="1.0" encoding="UTF-8"?><pre class="term"><span class="f11 b0"> 1 </  
  span><span class="f7 b1"> </span><span class="f7 b0">  
2  
3 </span><span class="f12 b0">~  
4  
5 ~  
6 ~  
7 ~  
8 ~  
  </span><span class="f7 b0">VIM - Vi IMproved</span><  
  span class="f12 b0">  
9 ~  
10 ~  
  </span><span class="f7 b0">version 7.1.123</span><  
  span class="f12 b0">  
11 ~  
  </span><span class="f7 b0">by Bram Moolenaar et al.</span>  
  <span class="f12 b0">  
12 ~  
  </span><span class="f7 b0">Modified by Gentoo-7.1.123</  
  span><span class="f12 b0">  
13 ~  
  </span><span class="f7 b0">Vim is open source and freely  
  distributable</span><span class="f12 b0">  
14 ~  
15 ~  
  </span><span class="f7 b0">Become a registered Vim user!</  
  span><span class="f12 b0">  
16 ~  
  </span><span class="f7 b0">type :help register</span><span class="  
f12 b0">&lt;Enter>;</span><span class="f7 b0"> for information </span><span  
class="f12 b0">  
17 ~  
18 ~  
  </span><span class="f7 b0">type :q</span><span class="f12 b0">&lt;Enter>;</span><span class="f7 b0">  
to exit </span><span  
class="f12 b0">  
19 ~  
  </span><span class="f7 b0">type :help</span><span class="f12 b0">&  
lt;Enter>;</span><span class="f7 b0"> or </span><span class="f12 b0">&lt;F1>;</span><span class="f7 b0">  
for on-line help</span><span class="f12 b0">  
20 ~  
  </span><span class="f7 b0">type :help version7</span><span class="  
f12 b0">&lt;Enter>;</span><span class="f7 b0"> for version info</span><span  
class="f12 b0">  
21 ~  
22 ~  
23 ~  
24 ~  
25 ~  
26 </span><span class="f7 b0">  
0,0-1 All  
27 </span></pre>
```

A screenshot of the Vim editor's startup screen. The background is black with white text. At the top left, there is a yellow '1' and a red cursor bar. The text in the center reads: 'VIM - Vi IMproved', 'version 7.1.123', 'by Bram Moolenaar et al.', 'Modified by Gentoo-7.1.123', and 'Vim is open source and freely distributable'. Below this, it says 'Become a registered Vim user!' followed by instructions: 'type :help register<Enter> for information', 'type :q<Enter> to exit', 'type :help<Enter> or <F1> for on-line help', and 'type :help version7<Enter> for version info'. In the bottom right corner, it shows '0,0-1' and 'All'.

```
1
VIM - Vi IMproved
      version 7.1.123
      by Bram Moolenaar et al.
      Modified by Gentoo-7.1.123
Vim is open source and freely distributable

  Become a registered Vim user!
type :help register<Enter>  for information

type :q<Enter>              to exit
type :help<Enter> or <F1>   for on-line help
type :help version7<Enter> for version info

                                0,0-1      All
```

Listing 3.6: Rückgabe bei unverändertem Inhalt

```
1 <?xml version="1.0" encoding="UTF-8"?><idem></idem>
```

## 3.5 SSH

*Come the Sun King blade ablur  
Hammer down eclipse the Sun  
And Puff, the land secured  
The new King Barbarian!*  
Ty Semaka  
„Puff the Barbarian“

Die SSH-Klasse bietet dynamische und erweiterbare Unterstützung für SSH-Client-Programme. OpenSSH und DropBear sind bereits integriert und sollten für die meisten Anwender genügen. Die Klasse kann selbstständig erkennen, um welchen Client es sich handelt und für diesen die Kommandozeilen zum Überprüfen der Logindaten und zum Einloggen zur Verfügung stellen.

## 3.6 KOMMUNIKATION MIT TERMINALS

*Unicode: everyone wants it,  
until they get it.*

Barry Warsaw, 2000-05-16

Für die Kommunikation mit dem virtuellen Terminal ist die Window-Klasse zuständig. Der Inhalt des virtuellen Terminals wird in einer Instanz der Terminal-Klasse gespeichert. Diese Klasse ist besonders aufwändig, da sich in Terminals nicht nur ausgebenbare Zeichen befinden können. ASCII- und ANSI-Steuersequenzen sind ebenfalls ein wichtiger Bestandteil eines jeden Terminals.

<i>Wert</i>	<i>Aktion</i>
0x08	Bewegt den Cursor um eine Position nach links. Falls sich der Cursor bereits am Zeilenanfang befindet, bleibt dieses Steuerzeichen wirkungslos.
0x09	Positioniert den Cursor auf die nächste Tabulatormarke oder an den rechten Bildschirmrand, falls keine weiteren Tabulatormarken vorhanden sind.
0x0A	Setzt den Cursor an den Anfang der nächsten Zeile.
0x0D	Verschiebt den Cursor an den Anfang der aktuellen Zeile.

Tabelle 3.1: Unterstützte ASCII-Steuersequenzen

<i>Wert</i>	<i>Aktion</i>
CSI n A	Bewegt den Cursor um n (Standardwert 1) Zeilen nach oben.
CSI n B	Bewegt den Cursor um n (Standardwert 1) Zeilen nach unten.
CSI n C	Bewegt den Cursor um n (Standardwert 1) Spalten nach rechts.
CSI n D	Bewegt den Cursor um n (Standardwert 1) Spalten nach links.
CSI n E	Bewegt den Cursor an den Anfang der um n (Standardwert 1) Zeilen weiter unten liegenden Zeile.
CSI n F	Bewegt den Cursor an den Anfang der um n (Standardwert 1) Zeilen weiter oben liegenden Zeile.
CSI n G	Bewegt den Cursor zu Zeile n.
CSI n H	Bewegt den Cursor zu Zeile n, Spalte m (Standardwerte 1).



CSI n J	Leert einen Teil des Terminals. Wenn n null ist (oder fehlt), wird vom Cursor aus bis zum Ende des Terminals geleert. Wenn n eins ist, wird vom Cursor aus bis zum Anfang des Terminals geleert. Wenn n zwei ist, wird das gesamte Terminal geleert.
CSI n K	Leert einen Teil der Zeile. Wenn n null ist (oder fehlt), wird vom Cursor aus bis zum Ende der Zeile geleert. Wenn n eins ist, wird vom Cursor aus bis zum Anfang der Zeile geleert. Wenn n zwei ist, wird die gesamte Zeile geleert. Die Cursorposition bleibt unverändert.
CSI n n;m f	Das selbe wie CSI n H.
CSI n n[;m] m	Setzt den SGR (Select Graphic Rendition) Parameter. Nach CSI können null oder mehr Parameter durch ; getrennt, folgen. Ohne Parameter wird CSI m wie CSI 0 m (reset / normal) behandelt, was für die meisten ANSI-Steuersequenzen typisch ist.
CSI s	Speichert die Position des Cursors.
CSI u	Stellt die Position des Cursors wieder her.
CSI ?25l	Macht den Cursor unsichtbar.
CSI ?25h	Macht den Cursor sichtbar.

Tabelle 3.2: Auszug aus unterstützten ANSI-Steuersequenzen

Code	Effekt
0, 27, 39, 49	Reset / Normal (Alle Attribute zurücksetzen)
1	Fettschrift
7	Hintergrund- und Vordergrundfarben vertauschen
30 31 32 33 34 35 36 37	Vordergrundfarbe setzen
40 41 42 43 44 45 46 47	Hintergrundfarbe setzen

Tabelle 3.3: Unterstützte SGR-Parameter

Die Terminal-Klasse speichert die Zeichen (in Unicode), die Vorder-, die Hintergrundfarben, und die Cursorposition und kann diese auf Abruf zurückgeben.

### 3.7 BEENDEN DES PROGRAMMS

*I have to stop now.*

*I've already told you more than I know.*

Wolf Logan, 1999-01-14

Bei einem Keyboard-Interrupt – in den meisten Betriebssystemen durch **STRG + C** hervorzurufen – wird der Server beendet. Hat man ihn jedoch als Daemon gestartet, ist dies nicht möglich. Es gibt mehrere Möglichkeiten den Hintergrundprozess zu beenden: Es wird beim Starten als Daemon auch die PID ausgegeben. Lautet diese beispielsweise *1234*, lässt sich `ajaxWM` mit einem `kill 1234` beenden. Hat man die PID nicht mehr zur Verfügung, lässt sich diese so herausfinden:

Listing 3.7: Instanz beenden

```
1 r0q@kana /home/r0q $ ps -ef | grep python | grep ajaxwm.py | grep -v grep | \
2   tr -s ' ' | cut -d' ' -f2,8-
3 9640 python src/ajaxwm.py -d
4 9685 python src/ajaxwm.py -d -p 8023
5 r0q@kana /home/r0q $ kill 9640
6 r0q@kana /home/r0q $ ps -ef | grep python | grep ajaxwm.py | grep -v grep | \
7   tr -s ' ' | cut -d' ' -f2,8-
8 9685 python src/ajaxwm.py -d -p 8023
9 r0q@kana /home/r0q $
```

Eine kurze Erklärung der Befehlszeile<sup>12</sup>: Das `ps -ef` liefert eine Liste mit allen laufenden Prozessen und Informationen zu diesen. Durch eine Pipe (`|`) erhält `grep` dessen Ausgabe und behält nur die Zeilen bei, die das Wort *python* enthalten, also alle Prozesse, die zu Python-Programmen gehören. Anschließend wird nach `ajaxwm.py` gefiltert und `grep` von der Liste durch ein `-v` ausgeschlossen. Jetzt befinden sich zwischen den Informationen zu jedem Prozess mehrere Leerzeichen, die mit einem `tr -s ' '` durch ein einzelnes ersetzt werden. `cut` unterteilt nun mit `-d' '` nach diesen Leerzeichen in Blöcke und gibt dabei Block zwei, die PID, und alles ab Block acht, also das Programm und die Parameter, mit denen es ausgeführt wurde, aus.

Wir können sehen, dass bereits zwei Instanzen von `ajaxWM` im Hintergrund (`-d`) laufen. Die eine auf dem Port 8022 (Standard), die andere auf 8023 (`-p`). Wie schon zuvor lassen sich nun die Instanzen mit `kill` beenden. Es ist ebenfalls möglich alle Instanzen von `ajaxWM` zu beenden:

Listing 3.8: Alle Instanzen beenden

```
1 r0q@kana /home/r0q $ ps -ef | grep python | grep ajaxwm.py | grep -v grep | \
2   tr -s ' ' | cut -d' ' -f2,8-
3 10516 python src/ajaxwm.py -d -p 1234
4 10529 python src/ajaxwm.py -d -p 1235
5 10535 python src/ajaxwm.py -d -p 1236
```

---

<sup>12</sup>Die Befehlsanreihung ist aus ästhetischen Gründen in zwei Zeilen unterteilt.

```
6 10541 python src/ajaxwm.py -d -p 1237
7 10547 python src/ajaxwm.py -d -p 1238
8 r0q@kana /home/r0q $ kill `ps -ef | grep python | grep ajaxwm.py | \
9   grep -v grep | tr -s ' ' | cut -d' ' -f2`
10 r0q@kana /home/r0q $ ps -ef | grep python | grep ajaxwm.py | grep -v grep | \
11   tr -s ' ' | cut -d' ' -f2,8-
12 r0q@kana /home/r0q $
```

Man kann sehen, dass wir im Vergleich zur bisherigen Befehlszeile nicht viel verändert haben. Das `kill` beendet direkt den oder die Prozesse, die in den ``` stehen. Durch das `-f2` gibt `cut` nur die PIDs, nicht aber den Programmnamen und die Parameter, aus.

Um zu verhindern, dass man die Process-ID nicht weiß und durch obige Methode herauszufinden hat, kann man beim Start die PID in eine Datei speichern mit `./ajaxwm.py -d -P ajaxwm.pid`. Zum beenden dieser Instanz genügt nun ein

```
kill `cat ajaxwm.pid`
```

# KAPITEL 4

## CLIENT

### 4.1 VERWENDETE TECHNOLOGIEN

*Web 2.0 is of course a piece of jargon,  
nobody even knows what it means.*

Tim Berners-Lee, Begründer des World Wide Webs

#### 4.1.1 SPRACHEN UND WERKZEUGE

Um die gewünschte Plattformunabhängigkeit zu erreichen und dabei die Voraussetzung zu erfüllen, dass keine Software installiert werden muss um ajaxWM zu verwenden, fiel die Wahl auf Ajax. Ajax ist ein Akronym aus „Asynchronous Javascript and XML“ und beschreibt eine Vorgehensweise der Internetseitengestaltung, bei der der Browser nicht wie bei herkömmlichen Internetangeboten für jede Anfrage an den Server eine komplette Internetseite zurück bekommt und diese dann darstellt, sondern dass JavaScript auf die Interaktion des Benutzers eingeht, Anfragen an den Server im Hintergrund startet und als Antwort nur die nötigen Informationen bekommt. Diese Informationen liegen meistens in XML vor, können jedoch auch aus normalem Text oder HTML-Fragmenten bestehen. JavaScript kann die erhaltenen Informationen zum Beispiel dadurch darstellen, dass das Design der Seite verändert wird, die Informationen an einer ausgesuchten Stelle dargestellt werden oder vorhandene Informationen aktualisiert werden. Ajax erfreut sich im Zeitalter von „Web 2.0“<sup>1</sup> immer größerer Beliebtheit. Obwohl JavaScript durch die Ecma International<sup>2</sup> standardisiert wurde, ist der Browserkrieg zwischen Netscape und Microsoft zwar teilweise gelöst, dennoch unterscheiden sich die verschiedenen Implementationen in manchen Punkten. Vor allem Microsofts Internet Explorer besitzt bis Version 7 noch keine XMLHttpRequest-API<sup>3</sup>

---

<sup>1</sup>Schwammige Bezeichnung für die Weiterentwicklung des WWW, Websites werden interaktiver, sind Nutzerbezogener (z.B. Youtube) es wird vermehrt „soziale Software“ verwendet, Informationen werden nicht durch zentrale Organisationen bereitgestellt oder verbreitet sondern z.B. auch über diverse Weblogs (s.g. Blogs).

<sup>2</sup>European Computer Manufacturers Association

<sup>3</sup>Das XMLHttpRequest-Objekt ist das Kernobjekt zur Verwendung von Ajax, es stellt Funktionen bereit um asynchrone Anfragen abzusetzen, welche normalerweise als XML beantwortet werden.

sondern benutzt Microsofts hauseigenes ActiveX-Objekt. Um Anwendungen, die auf Ajax bzw. JavaScript basieren dennoch ohne großen Aufwand so zu programmieren, dass sie in allen Browsern lauffähig sind, wurden im Laufe der letzten Jahre mehrere Cross-Browser-Libraries geschaffen, die dynamisch den Browser ermitteln und plattformunabhängige Schnittstellen bereitstellen. Solche Libraries sind zum Beispiel Sarissa oder prototype.js, die auch in ajaxWM verwendet werden<sup>4</sup>.

#### 4.1.2 ARCHITEKTURMODELL

Für den Client wurde ein modifiziertes MVC-Modell<sup>5</sup> verwendet, wobei das Model den Server darstellt und damit nicht im Client vorhanden ist. Die Views<sup>6</sup> bekommen ihre Daten über Controller<sup>7</sup>, vom Model und stellen sie dar. Der Controller regelt den kompletten Programmfluss. Er ist die Schnittstelle zwischen Model und den Views, fängt Tastendrücke ab, überprüft sie auf Steuerungssequenzen, kontrolliert die Session, erstellt Views und holt auf Anfrage Daten vom Model und übergibt sie den Views.

#### 4.1.3 GRUNDLEGENDE FUNKTIONSWEISE

Der grundlegende Ablauf des ajaxWM-Clients ist ähnlich anderer Ajax-Anwendungen. Der Benutzer geht mit dem Browser auf die URL, auf der der ajaxWM-Server läuft und bekommt als Antwort auf diese Anfrage eine index.html-Datei die Pfade zu den verschiedenen JavaScript-Dateien enthält. Der Browser lädt diese Dateien und fährt danach `window.onload()` aus, welche die Methode zum Starten des ajaxWM-Clients aufruft. Diese Methode behandelt die Initialisierung des Window Managers und damit auch die der grafischen Oberfläche. Danach wird ein Loginbildschirm angezeigt, welcher bestehen bleibt bis der Login erfolgreich war. Es folgt der Desktopmodus, welcher der Kern von ajaxWM ist. Hier kann der Benutzer neue Fenster erstellen und mit ihnen und mit ajaxWM allgemein interagieren. Dieser Desktop-Bildschirm bleibt bis zum Logout bestehen. Der Logout kann entweder vom Benutzer selbst initiiert werden oder zwangsweise nach einem bestimmten Zeitintervall erfolgen, in dem der Benutzer mit keinen Objekten des Clients interagiert hat und auch keine Fenster geöffnet waren. Nach einem Logout wird wieder der Loginbildschirm dargestellt<sup>8</sup>. Alle Events, die im

---

<sup>4</sup>Prototype.js ist derzeit noch nicht eingebaut.

<sup>5</sup>Model-View-Controller, Architekturmuster bei dem die Software in drei Teile aufgeteilt wird. Das Model enthält die Daten, die Views Präsentieren sie und der Controller regelt den Programmfluss.

<sup>6</sup>Die Fenster und der Loginbildschirm. Virtuelle Desktops zählen auch dazu, sind jedoch noch nicht vollständig implementiert.

<sup>7</sup>Der Manager, globale Instanz der Manager-Klasse.

<sup>8</sup>Der Zwangslogout ist derzeit clientseitig noch nicht vollständig implementiert, es wird nur die Session und die GUI zerstört.

Desktopmodus anfallen, werden vom Manager in eine Warteschleife gesteckt, welche von einer anderen Methode des Managers abgearbeitet wird. Läuft diese Methode nicht, wird sie direkt nachdem ein Event eingereicht wurde, gestartet. Die Methode läuft durch, bis die Warteliste komplett leer ist. Events werden als asynchrone Requests abgesetzt. Typisch für eine ajaxbasierte Anwendung kommt zu keinem Reload oder Aufruf einer anderen Seite.

## 4.2 FEHLERBEHEBUNG

*javascript coders always get their punishment*

Zero, <http://german-bash.org/12562>

Um den ordnungsgemäßen Ablauf des Clients zu überprüfen, wurde ein Debugger entwickelt. Nachdem dieser initialisiert wurde, kann an jeder Stelle des Clients ein Aufruf des Debuggers getätigt und ihm dabei Klassen- und Funktionsname sowie die Nachricht, die angezeigt werden soll, übergeben werden. Der Debugger zeigt diese Informationen in der rechten oberen Ecke des Fensters an. Alle Nachrichten werden unten an die Liste angehängt. Würde durch das Anhängen einer neuen Nachricht das Limit überschritten werden, werden oben am Anfang der Liste so lange alte Nachrichten entfernt, bis die neue Nachricht hinzugefügt werden kann ohne dass das Limit dabei überschritten wird. Die Konfiguration des Debuggers läuft über Variablen des Managers. So können die Dauer, wie lange die Nachrichtenliste nach dem hinzufügen einer Nachricht angezeigt werden soll, und die Anzahl der Zeichen, die maximal in der Liste enthalten sein dürfen, eingestellt werden. Momentan wird ein neuer Debugger geschrieben, der mit Call-Traces<sup>9</sup> funktioniert. Dieser Debugger ersetzt jede Funktion durch einen Wrapper, der, zusätzlich zu den selben Parametern, wie sie auch die Originalfunktion erwartet, ein weiteres Objekt - die Call-Trace - erwartet. Wurde kein Trace-Objekt übergeben, nimmt der Wrapper an, dass die Funktion direkt und nicht über eine andere, registrierte Funktion aufgerufen wurde und erstellt ein neues Trace-Objekt. Nach dieser Prüfung liegt immer ein Trace-Objekt vor. Der Wrapper ruft eine Methode des Trace-Objektes auf, die den jeweiligen Funktions- und Klassennamen speichert, sodass er später ausgegeben werden kann. Danach ruft der Wrapper die Funktion auf die er ersetzt hat und übergibt ihr alle Parameter in der selben Reihenfolge, wie er sie selbst bekommen hat - ausgenommen vom Trace-Objekt. Da die Originalfunktion innerhalb der Wrapperfunktion definiert wird, ist das Trace-Objekt trotzdem verfügbar. Der Rückgabewert der Originalfunktion wird temporär gespeichert und als Rückgabewert des Wrappers zurückgegeben, nachdem nach Beendigung der Originalfunktion das Trace-Objekt aufgefordert wird, den Eintrag dieser Funktion wieder zu entfernen. Außerdem wird in jeder registrierten Funktion bei jedem Aufruf einer anderen (oder der selben) registrierten Funktion das Trace-Objekt als letzter Parameter angehängt. Dies geschieht auch wenn eine Ausnahme geworfen wird. Der

---

<sup>9</sup>Call-Traces verfolgen, welche Funktion momentan läuft, durch welche Funktion sie aufgerufen wurde, durch welche Funktion diese wiederum aufgerufen wurde, u.s.w.. Tritt ein Fehler auf, kann so über die Call-Trace genau erkannt werden, durch welche Funktionen die Funktion aufgerufen wurde, die den Fehler verursachte. Desweiteren können die übergebenen Parameter angezeigt werden. So ist der komplette Programmablauf bei weitem leichter nachvollziehbar und Fehler können leichter gefunden werden.

catch-Block, der die Ausnahme auffängt, kann so die das Trace-Objekt der Ausnahme an den Debugger weitergeben. Dieser kann eine formatierte Zeichenkette mit der Call-Trace ausgeben.

Listing 4.1: Originalfunktion

```
1 function toString() {  
2     return '[object Object]';  
3 }
```

Listing 4.2: Modifizierte Funktion, die Call-Traces unterstützt

```
1 function toString(__trace) {  
2     if (null == __trace) {  
3         __trace = new Trace();  
4     }  
5     __trace.enter(this._CLASS_NAME, 'toString');  
6     _this = this;  
7     __trace_toString = function() {  
8         return '[object Object]';  
9     }  
10    var result = __trace_toString();  
11    __trace.leave();  
12    return result;  
13 }
```

So können Call-Traces in JavaScript verwendet werden, obwohl diese Sprache dies eigentlich nicht unterstützt.



### 4.3 STARTVORGANG UND ANMELDUNG

*Thunder is good, thunder is impressive;  
but it is lightning that does the work.*

Mark Twain

#### 4.3.1 GRAFISCHE OBERFLÄCHE

Bevor die grafische Oberfläche initialisiert wird, wird zuerst der Debugger initialisiert. Es folgt eine Überprüfung des vom Benutzer ausgewählten Themes<sup>10</sup>. Wurde kein Theme ausgewählt, wird der Standard-Theme<sup>11</sup> verwendet. Es wird ein Objekt der Klasse, die für die Themes zuständig ist, erstellt und ihm dabei der Parameter des zu verwendenden Themes übergeben. Dieses Objekt fügt der dargestellten HTML-Seite im Head-Bereich das Theme-eigene Stylesheet hinzu, wodurch der Browser automatisch veranlasst wird, selbstständig diese Datei herunterzuladen. Des weiteren erstellt das Theme-Objekt eine Anfrage nach der XML-Datei des Themes, in der das Grundgerüst des Themes beschrieben ist. Ist diese XML-Datei fertig heruntergeladen, wird sie vom Theme-Objekt unter Berücksichtigung der Attribute, denen besondere Funktionen zugeordnet sind<sup>12</sup>, in HTML umgesetzt. Am Ende dieser Prozedur ist das HTML-Grundgerüst und das dazugehörige Stylesheet vorhanden, spezielle Attribute der XML-Datei wurden in internen Datenstrukturen abgespeichert, sodass die jeweiligen Elemente, denen besondere Funktionen zugeordnet wurden, wiedergefunden werden können.

#### 4.3.2 ANMELDUNG

Nachdem der Benutzer die Seite aufgerufen und der Client die Oberfläche initialisiert hat, wird der Loginbildschirm angezeigt. Die Arbeit des Managers ist nun vorerst getan. Jede Interaktion wird nun von der Loginbildschirm-Klasse behandelt. Wenn der Benutzer sich einloggen möchte, wird der Loginbildschirm verriegelt und eine Methode Managers aufgerufen, welche das Login-Event mit den angegebenen Daten absetzt. Der Manager überprüft ob der Server die Anfrage mit einem Fehlercode beantwortet hat oder ob der Login erfolgreich war. War der Login erfolgreich wird eine Session<sup>13</sup> mit der ID, die der Server als Antwort gesendet hat, erstellt. In beiden Fällen übergibt der Manager dem Loginbildschirm eine Rückmeldung. Bei einem erfolgreichen Login schließt sich der Loginbildschirm daraufhin selbst, schlug er fehl wird das eingegebene

---

<sup>10</sup>Derzeit kann der Benutzer noch kein Theme auswählen.

<sup>11</sup>lim - Less Is More. Minimalistischer Theme, der nur Fensterinhalt und Titelleiste anzeigt.

<sup>12</sup>move, resize, position, size, onclick, onmousedown, content

<sup>13</sup>Hier nur clientseitig um sich auszuweisen, der Server überprüft die Gültigkeit der Sessions.

Passwort gelöscht und der Loginbildschirm entriegelt. Der Loginbildschirm bleibt so lange aktiv, bis der Login erfolgreich war.

#### 4.3.3 ÜBERGANG IN DEN NORMALBETRIEB

Nachdem nach einem erfolgreichen Login der Loginbildschirm geschlossen wurde betritt der Manager den Desktopmodus. Es werden acht virtuelle Desktops erstellt und zum ersten gewechselt. Der Manager setzt Keyboardhooks<sup>14</sup> auf, die jede gedrückte Taste(nkombination) abfangen und daraufhin überprüfen, ob es sich um einen Hotkey handelt. Falls es sich um keinen Hotkey handelt und ein Fenster fokussiert ist werden die gedrückten Tasten UTF-8-kodiert an den Server gesendet. Hotkeys werden direkt behandelt, ohne dass dem Server irgendeine Meldung darüber zukommen gelassen wird. Nachdem diese Initialisierungen abgeschlossen sind, ist es möglich verschiedene Fenster zu erstellen, Befehle in ihnen einzugeben, sie zu verschieben und ihre Größe zu ändern.

---

<sup>14</sup>Hierbei handelt es sich um lokale Keyboardhooks, das heißt es werden nur dann Tasten abgefangen, wenn nicht nur das Browserfenster/-tab aktiv ist und Fokus hat, sondern auch die Seite, die ajaxWM darstellt, angewählt ist und Tasten direkt an sie geleitet werden.

## 4.4 DER DESKTOP

*Very funny, Scotty.  
Now beam down my clothes!*  
Author unbekannt

### 4.4.1 VIRTUELLE DESKTOPS

Es sind acht virtuelle Desktops verfügbar, man kann zwischen ihnen mit den Hotkeys STRG+ALT+1 bis STRG+ALT+8 hin- und herschalten. Alle Fenster, die erstellt werden, werden - soweit nicht anders angegeben - auf dem virtuellen Desktop erstellt, der gerade aktiv ist und in allen anderen virtuellen Desktops nicht angezeigt.

### 4.4.2 FENSTER

#### **Fenster erstellen**

Das Erstellen der Fenster läuft immer über den Manager<sup>15</sup>. Soll ein neues Fenster erstellt werden, wird zuerst immer eine Anfrage an den Server gesendet. Wenn kein Fehler beim Server aufgetreten ist, wird ein neues Fenster-Objekt erstellt und ihm dabei direkt die vom Server gelieferte ID übergeben. Das Fenster-Objekt bezieht seine HTML-Elemente durch klonen der Schablone, die das Theme-Objekt aus der XML-Datei erstellt hat. Da jedoch die Datenstrukturen, die Elemente mit speziellen Funktionen beinhalten sollen, keine Referenz auf das Element beinhalten können<sup>16</sup> wird hier immer nur ein relativer Pfad ausgehend vom Hauptelement zum gewünschten Element angegeben. So kann, nachdem die Hierarchie der HTML-Elemente ausgehend vom Hauptelement geklont wurde, jeder Pfad anhand des geklonten Hauptelementes durch eine Referenz des Elementes, auf das der Pfad zeigt, ersetzt werden<sup>17</sup>. Elemente, die Event-Handler<sup>18</sup> zugewiesen bekommen sollen, werden auf die selbe Art erst

---

<sup>15</sup>Derzeit können Fenster nur durch den Hotkey STRG+ALT+W erstellt und durch STRG+ALT+C geschlossen werden.

<sup>16</sup>Würden die Datenstrukturen Referenzen auf HTML-Elemente beinhalten, würde das Klonen neue Elemente erstellen und das Kopieren der Referenzen auf die Elemente der Schablone verweisen.

<sup>17</sup>Durch dieses Verfahren ist gesichert, dass es zu keinen überflüssigen Klonungen kommt. Es wird nur das Hauptelement (und dadurch die ihm untergeordneten Elemente) geklont und alle Pfade durch Referenzen auf das geklonte Hauptelement oder ihm untergeordnete Elemente ersetzt. Das Ersetzen der Pfade durch Referenzen beschleunigt den Client enorm, da bei jedem Zugriff auf ein Element nicht erst das Element in der kompletten Hierarchie ausgehend vom Hauptelement dieses Fensters gesucht werden muss, sondern als Referenz direkt vorliegt und dadurch auch kaum zusätzlichen Arbeitsspeicher verbraucht.

<sup>18</sup>Ein Event-Handler wird immer dann ausgeführt, wenn ein Ereignis eintritt. Solche Ereignisse sind in JavaScript zum Beispiel onclick (tritt bei einem Mausklick auf), onkeydown (tritt auf, wenn

durch relative Pfade dargestellt, die beim Erstellen des Fensters durch Referenzen ersetzt werden. Obwohl das Fenster-Objekt seine Elemente erstellt und alle Pfade ersetzt hat, wird es noch nicht angezeigt. Zuerst ruft das Fenster-Objekt noch den Manager auf, der dem Server ein Synchronisations-Event sendet. Während dieser Event noch abgearbeitet wird, beginnt das Fenster-Objekt mit der Darstellung des Fensters. Damit ist die Erstellung des Fensters abgeschlossen und der Manager schiebt es in den Vordergrund.

### Synchronisation und Tastendruck

Ajax hat ein so genanntes „Polling Problem“. Dieses Problem liegt darin, dass der Server dem Client keine Anfragen senden kann. Jegliche Anfragen können nur vom Client ausgehen<sup>19</sup>. Dadurch muss der Client immer wieder beim Server anfragen, ob neue Informationen vorliegen<sup>20</sup>. Das ist für einen Window Manager bzw. einen Terminal Emulator ein Problem, da das Terminal in Wirklichkeit auf dem Server läuft und der Client nicht nur bei einer Änderung des serverseitigen Terminals aktualisiert werden kann, sondern immer wieder eine Anfrage an den Server senden muss, welche mit dem Inhalt des Terminals beantwortet wird, und danach überprüfen muss, ob sich der Inhalt geändert hat. AjaxWM löst dies im Stil von Ajaxterm: Auf eine Synchronisationsanfrage des Clients antwortet der Server entweder mit dem Terminalinhalt oder mit `<idem></idem>` wenn sich nichts geändert hat. Dies führt dazu, dass im Falle eines gleichbleibenden Inhaltes einerseits die Übertragung schneller abgeschlossen ist, andererseits der Client nur bei geändertem Inhalt ein Fenster aktualisieren muss. Diese Anfrage wird in einem bestimmten Zeitintervall gesendet, dass sich dynamisch daran anpasst, ob sich der Inhalt verändert oder nicht. AjaxWM geht aber weiter als Ajaxterm. Bei ajaxWM gibt es nicht nur ein einziges zentrales Terminal-Objekt, das selbst die Serveranfragen sendet, sondern einen Window Manager der eine große Anzahl an Terminals unterstützt. Jedes Terminal ist eigenhändig für seinen Inhalt zuständig und regelt seine dynamischen Aktualisierungen. Ist der Zeitpunkt einer Aktualisierung gekommen, wird der Manager aufgerufen, der ein Synchronisationsevent sendet. Die Antwort des Events wird dem Fenster-Objekt übergeben. Das Fenster-Objekt überprüft daraufhin, ob sich der Inhalt geändert hat oder nicht. Bei geändertem Inhalt wird das Fenster aktualisiert und die Dauer bis zur nächsten Aktualisierung auf das Minimum gesetzt. Hat sich der Inhalt nicht geändert, wird die Dauer bis zur nächsten Aktualisierung erhöht, vorausgesetzt sie hat ein bestimmtes Maximum noch nicht

---

eine Taste gedrückt wird) u.a.

<sup>19</sup>Auch bei Ajax handelt es sich nur um das HTTP-Protokoll, welches ein zustandsloses Protokoll ist und nicht für Anwendungen, bei denen der Server dem Client eigenständig Daten senden kann, vorgesehen ist.

<sup>20</sup>S.g. polling

überschritten. Ein Tastendruck läuft ähnlich ab wie die Synchronisation: Der Manager überprüft, ob die gedrückte Taste kein Hotkey ist und sendet dann dem Server dann ein Keypress-Event, dass die ID des betroffenen Fensters und das Zeichen (UTF-8-kodiert) enthält. Dieses Event wird mit dem Terminalinhalt, nachdem das Zeichen eingegeben wurde, beantwortet. Ab hier läuft es wieder wie bei einer Synchronisation ab: Der Manager übergibt dem jeweiligen Fenster-Objekt den Terminalinhalt und das Fenster stellt ihn dar. Dadurch wird bei jedem Tastendruck, bei dem sich der Inhalt des Terminals ändert, die Dauer bis zur nächsten Aktualisierung auf das Minimum gesetzt und mögliche weitere Auswirkungen des Tastendrucks werden innerhalb eines kurzen Intervalles gepollt. So können Prozessorzeit und Bandbreite bei inaktiven Terminals gespart und für aktivere Terminals verwendet werden.

### Fenster verschieben und vergrößern

AjaxWM ist ein Window Manager, der sich in der Handhabung wenig von anderen Window Managern unterscheidet<sup>21</sup>. Fenster lassen sich durch klicken und ziehen an der Titelleiste verschieben, vergrößern oder verkleinern geht durch klicken und ziehen an den Fensterrahmen<sup>22</sup>. Wenn das Fenster verschoben oder seine Größe verändert wird, wirkt sich jede Änderung sofort auf das Fenster aus<sup>23</sup>. Elemente, die beim Verschieben bzw. beim Vergrößern ihr Aussehen ändern sollen, werden so lange anders dargestellt, wie das Fenster verschoben oder vergrößert wird. Fenster können durch jedes Element vergrößert oder verschoben werden<sup>24</sup>. Beim gedrückt halten der Maustaste auf ein solches Element betritt das Fenster den Verschiebe- bzw. Vergrößerungsmodus, welche Event-Handler aufsetzen, die ausgeführt werden, wenn der Benutzer die Maustaste loslässt oder die Maus bewegt. Es wird der Stylesheet-Klassename eines jeden Elementes, welches beim Verschieben oder Vergrößern seine Klasse ändern soll<sup>25</sup>, geändert<sup>26</sup>. Beim Bewegen der Maus wird je nach Modus das Fenster verschoben oder seine Größe geändert. Beim Loslassen werden die Event-Handler und die Stylesheet-Klassennamen

---

<sup>21</sup>Window Manager haben meist keinen Taskpanel o.ä., diese Aufgaben erledigen separate Programme. AjaxWM wird dies jedoch noch hinzugefügt bekommen. Tiling wird auch noch implementiert.

<sup>22</sup>Da der Standardtheme lim keine Fensterrahmen besitzt - und auch niemals besitzen wird - ist auch keine Vergrößerung des Fensters möglich. Daher ist der Vergrößerungsmodus nicht ausführlich getestet und funktioniert möglicherweise noch nicht.

<sup>23</sup>Eine Alternative wäre, das echte Fenster unberührt zu lassen, nur eine Vorschau zu verschieben oder ihre Größe ändern und danach die Größe der Alternative auf die der Vorschau anzupassen. Wird möglicherweise auch noch hinzugefügt, aber nicht in der nächsten Zeit.

<sup>24</sup>onmousedown="move" bzw. onmousedown="resize" in der Theme-XML bei jedem Element, das das verschieben oder vergrößern soll.

<sup>25</sup>move="change" bzw. reszie="change" in ihren XML-Tags in der XML-Datei des Themes.

<sup>26</sup>An den Stylesheet-Klassennamen wird ein `_moving` respektive `_resizing` angehängt. Das Aussehen beim Verschieben oder Vergrößern kann so einfach per CSS definiert werden.

zurückgesetzt und dadurch der jeweilige Modus verlassen.

### **Fenster schließen**

Fenster können entweder durch Klicken auf eine „Schließen“-Schaltfläche<sup>27</sup> oder durch das Drücken von STRG+ALT+C geschlossen werden. Wenn ein Fenster geschlossen werden soll, bereitet es sich erst auf das Schließen vor<sup>28</sup> und weist den Manager an, dieses Fenster zu schließen. Der Manager sendet dem Server die Anfrage, das ausgewählte Fenster serverseitig zu schließen. Meldet der Server Erfolg, schließt der Manager das Fenster und zerstört dann das zugehörige Objekt<sup>29</sup>. Der Fokus wird dem vorherigen Fenster übergeben.

---

<sup>27</sup>Lim hat keine Schaltflächen und wird auch keine bekommen. Daher ist noch keine Funktionsweise für Schaltflächen implementiert.

<sup>28</sup>Derzeit wird nur ein Flag gesetzt, das weitere Synchronisationen verhindert.

<sup>29</sup>Derzeit ist noch keine Fehlerbehandlung eingebaut, die einen serverseitigen Fehlschlag beim Schließen behandeln könnte.

## KAPITEL 5

### THEMES

#### 5.1 DESIGN

##### 5.1.1 PLANUNG

Entworfen werden sollte eine Website auf welchen Fenster geöffnet werden können, in denen Programme des Rechners auf welchen man übers Internet zugreift ausgeführt werden. Daher auch der Projektname Ajaxwm, welcher kurz für Ajax Window Manager steht. Es bot sich an das ganze im Stil von Windows XP zu entwerfen, weiter Ziele waren, ein Minimalstil für Menschen mit einer Abneigung gegenüber Microsoft sowie ein Vista und Mac OS Stile.

Als weiteres Ziel sollte ich noch einen Login Screen entwerfen.

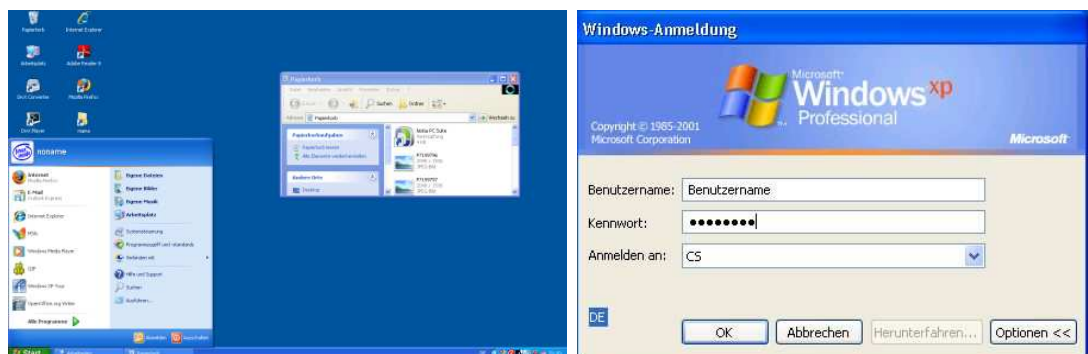


Abbildung 5.1: So sieht's bei Original Microsoft Windows XP aus

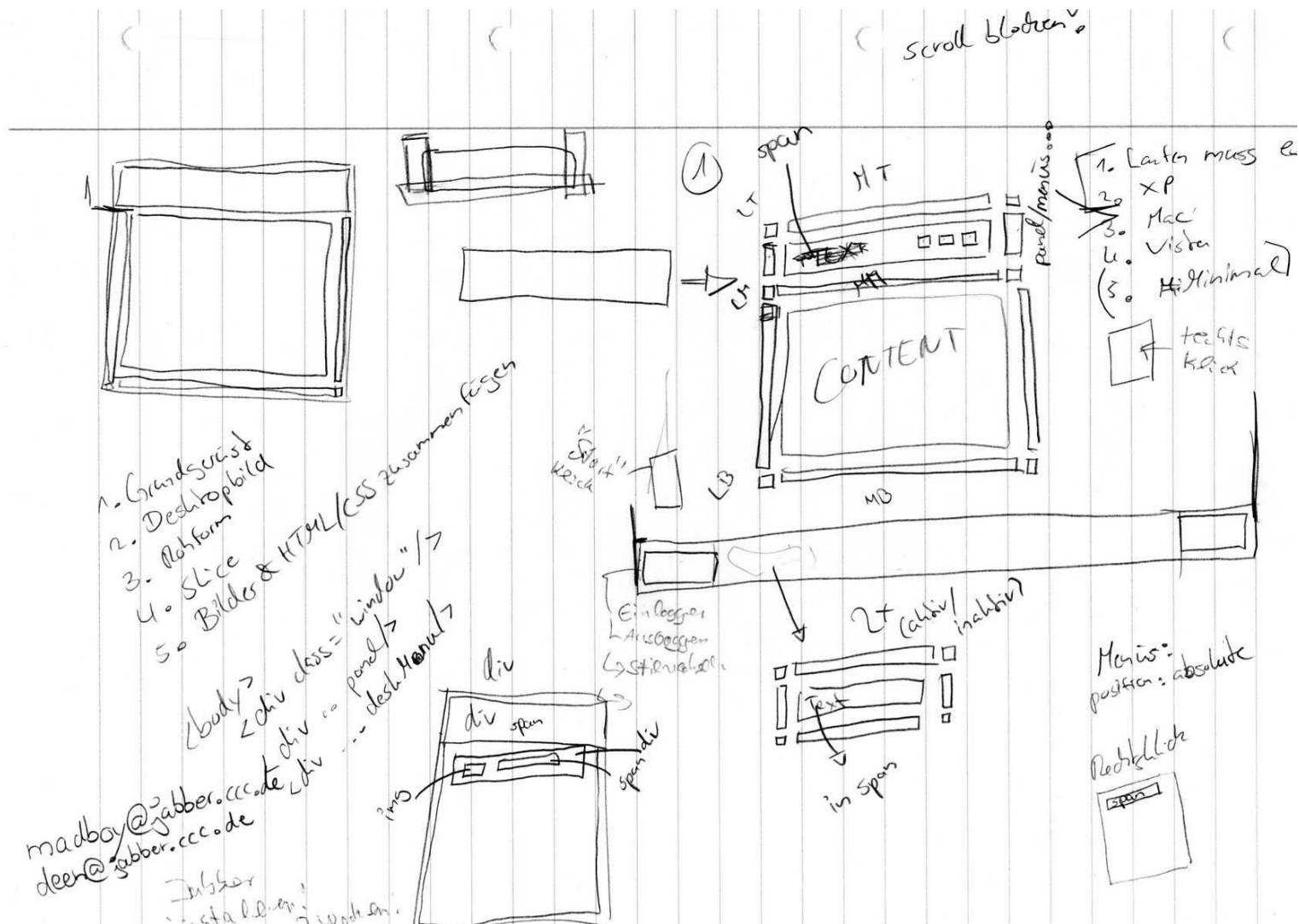
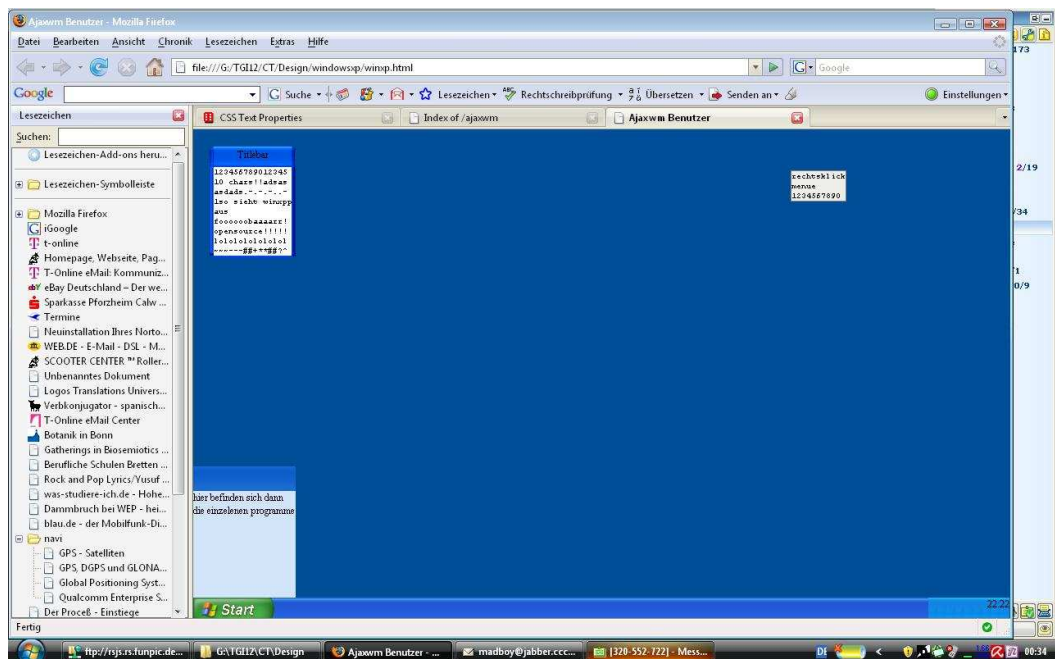


Abbildung 5.2: Handgezeichneter Entwurf des Windows XP Stils



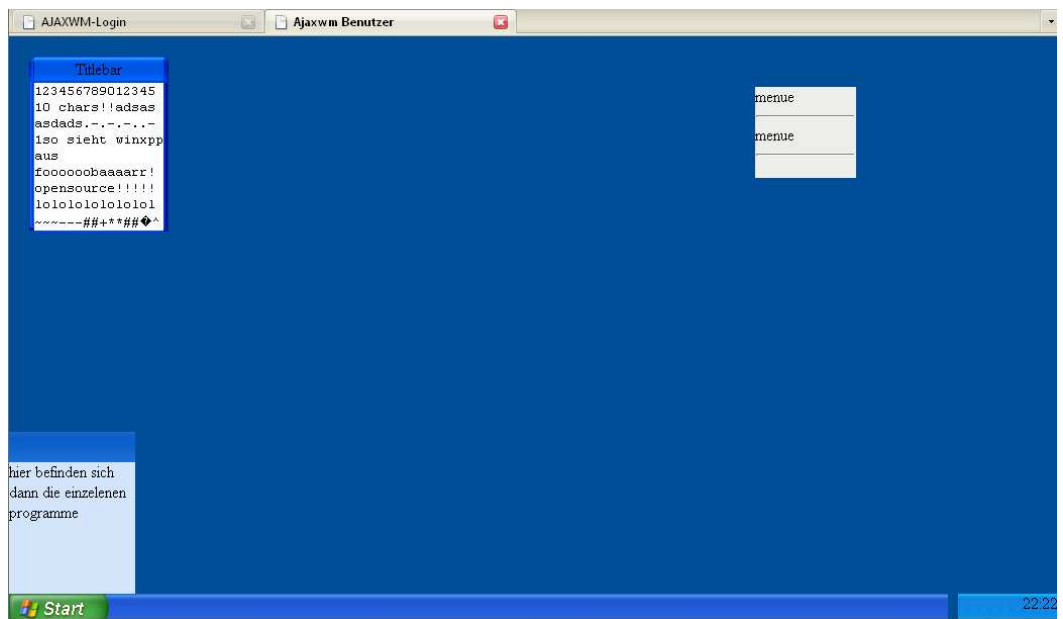
### 5.1.2 STATISCHER AUFBAU

Die Design Webseiten sind statisch geschrieben, d.h. sie sehen bei jedem Aufruf genau identisch aus und lassen sich auch nicht verändern. Wäre ein Besucherzähler enthalten, dann wären Sie bereits nicht mehr statisch, denn dieser Zähler verändert sich mit der Zahl der Seitenaufrufe. Der Seite Leben einzuhauchen ist Andys Aufgabe. Sein geschriebenes JavaScript wird sich dann darum kümmern die geöffneten Fenster an der richtigen Stelle anzuzeigen, oder wenn sie verschoben werden, sie mit den neuen Koordinaten frisch aufzubauen.



### 5.1.3 STAND

Der Windows XP Stil, steht demnächst vor seiner Fertigstellung, allerdings muss die Desktop Ansicht aufgrund eines Fehlers bei der Taskleiste noch einmal redesignn werden. Der Anmeldebildschirm muss noch um die zugehörigen Bilder erweitert werden.



## 5.2 HTML

Im HTML Teil der Webseiten stehen keine Angaben zur Formatierung, zum Stil und zur Ausrichtung. Siehe winxp.html für die HTML-Datei.

```
22:22
Titlebar

123456789012345
10 chars!!adsas
asdads.-.-.-.-
1so sieht winxpp
aus
foooooobaaaarr!
opensource!!!!
lololololololol
~~~---##+*##◆^

menue
_____
menue
_____

hier befinden sich dann die einzelnen programme
```

Abbildung 5.3: Aussehen der Seite ohne CSS

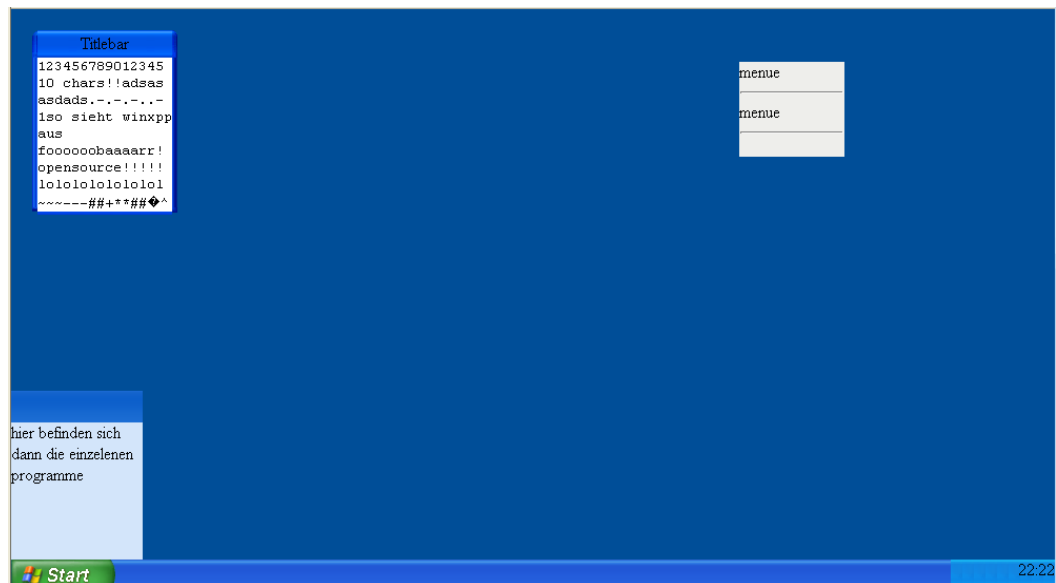


Abbildung 5.4: Jetziger Stand mit CSS

### 5.3 CSS

Die CSS-Datei steht unter `style1.css` zur Verfügung.

Der Name des zu definierenden Elements wird hingeschrieben, falls es sich innerhalb eines anderen Elements befinden wird der Name des Elternelements davor geschrieben. Die verwendeten Anweisungen sind in diese Klammern zu schreiben, jede Anweisung muss mit einem `;` geschlossen werden sonst übersieht der Browser sie. Der Internet Explorer von Microsoft stellt die meisten obwohl richtig geschriebenen Anweisungen trotzdem falsch dar, da er sich nur selten an Standards hält. Zur steht's korrekten Anzeige sollte daher der Mozilla Firefox verwendet werden.

<i>CSS-Befehl</i>	<i>Funktion (Angaben ohne Gewähr)</i>
<b>Width</b>	ist für die Breite des zu erstellenden Elements, die Angabe kann absolut also in Pixel erfolgen, z.B. 100px oder in Prozenten relativ zur Größe des Browserfensters oder Elternelements z.B. 20%
<b>Height</b>	ist für die Höhe des zu erstellenden Elements, die Angabe kann absolut also in Pixel erfolgen, z.B. 100px oder in Prozenten relativ zur Größe des Browserfensters oder Elternelements z.B. 20%

Background-color	damit lässt sich die Farbe des Hintergrundes entweder von der ganzen Seite oder nur vom betreffenden Element einstellen. Angaben wie z.B. black oder #004e98 sind zulässig.
Background-image	Mit der Angabe von url(bild.jpg) kann man ein Hintergrundbild definieren.
Background-repeat	Die Angabe no-repeat verhindert das als Hintergrund angegeben Bild nicht wiederholt wird.
Margin	Abstand zu einem anderen Element.
Padding	Definiert den Innenabstand zwischen Elementrand und Inhalt.
Position	Mit diesem Befehl kann man die gewünschte Ausrichtung bestimmen.
Top	Gibt an, an welcher Stelle das Element erscheinen wird. Von oben ausgehend in Pixel oder Prozent.
Left	Wie top nur von links ausgehend in Pixel oder Prozent. Right und Bottom (unten) sind auch möglich.
Text-align	Hiermit lässt sich die Textausrichtung festlegen. Text-align} center; richtet den Text mittig aus.

Tabelle 5.1: Die verwendeten Befehle

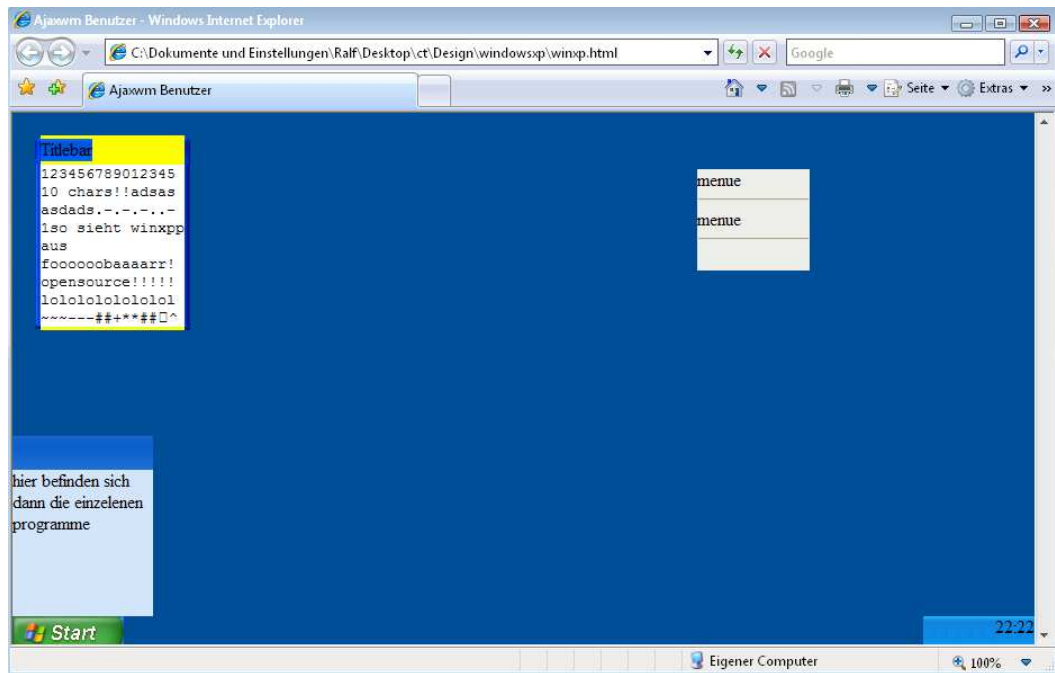
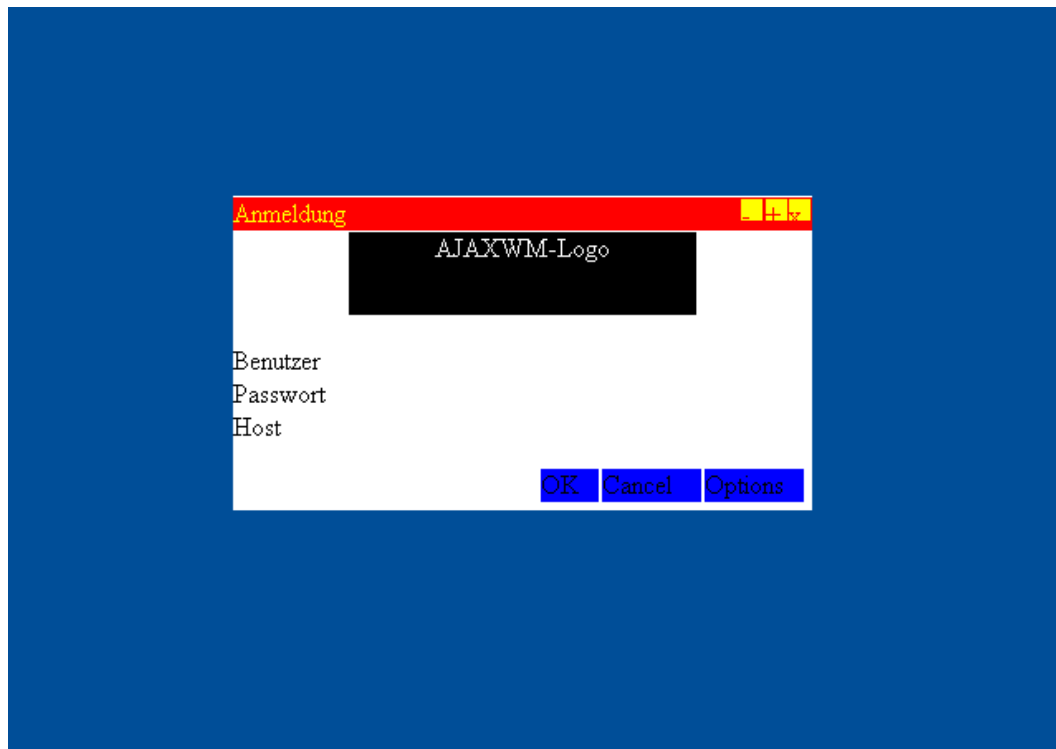


Abbildung 5.5: So stellt der Internet Explorer die unter Punkt 2(HTML)- Stand gezeigte Website (Desktop Seite) dar:

## 5.4 UMSETZUNG

### 5.4.1 LOGINSEITE

Auf der Loginseite trägt man seinen Benutzernamen, das Passwort und den Hostpc auf welchen man zugreifen möchte ein. Hier fehlen noch die Formulare zur Eingabe der Daten.



#### 5.4.2 DESKTOPSEITE

Die Desktopseite wird verwendet um wie im Vorbild zu funktionieren, so wurde eine Taskleiste verwirklicht, in welchen später die geöffneten Programme erscheinen sowie ein Info Teil mit Uhrzeit und Verbindungsstatus. Was bei Windows XP Start heißt wird hier die Funktion erhalten sich auszuloggen oder einzelne Programme zu starten.



## KAPITEL 6

### QUELLTEXT

*Meine Codes sind wie moderne Kunst:  
Keiner kann sie einordnen,  
und wenn man sie zu lange betrachtet,  
wird einem schlecht.  
Author unbekannt*

#### 6.1 SERVER

##### 6.1.1 AJAXWM.PY

```
1 #!/usr/bin/env python
2 """
3 ajaxWM – the free web based window manager for remote terminals
4 Copyright (C) 2007 Dennis Felsing, Andreas Waidler
5
6 This program is free software: you can redistribute it and/or modify
7 it under the terms of the GNU General Public License as published by
8 the Free Software Foundation, either version 3 of the License, or
9 (at your option) any later version.
10
11 This program is distributed in the hope that it will be useful,
12 but WITHOUT ANY WARRANTY; without even the implied warranty of
13 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14 GNU General Public License for more details.
15
16 You should have received a copy of the GNU General Public License
17 along with this program. If not, see <http://www.gnu.org/licenses/>.
18
19 $Id: ajaxwm.py 76 2007-12-15 11:26:24Z r0q $
20 """
21
22 #{{{ Imports
23 # Psycopy really speeds up ajaxWM (well, any script written in Python), but it
24 # is only available for x86 and will likely _not_ get ported.
25 try:
26     import psyco
27     psyco.full()
28 except ImportError:
29     pass
30
31 import sys
```



```
32 import optparse
33 import os
34 import commands
35
36 from Ssh import Ssh
37 import qweb
38 #}}}
39 ENCODING = 'UTF-8'
40 XMLSTRING = u'<?xml version="1.0" encoding="%s"?>' % ENCODING
41 # All themes between these two versions (including them) are valid for this
42 # version.
43 VALID_THEMES = ([0,1,0], [0,3,0])
44 def read(fd):#{{{
45     try:
46         content = os.read(fd, 65536)
47         return unicode(content, ENCODING)
48     except OSError:
49         return u''
50 #}}}
51 def main(argv=None):#{{{
52     # filling the default dynamically, because the default argument of main()
53     # is calculated right at the definition and won't get updated
54     if argv is None:
55         argv = sys.argv
56
57     parser = optparse.OptionParser()
58     parser.add_option('-d',
59                     '--daemon',
60                     dest = 'daemon',
61                     action = 'store_true',
62                     help = 'run in background')
63     parser.add_option('-p',
64                     '--port',
65                     dest = 'port',
66                     default = '8022',
67                     help = 'set the TCP port (default: 8022)')
68     parser.add_option('-l',
69                     '--log',
70                     action = 'store_true',
71                     dest = 'log',
72                     default = 0,
73                     help = 'log requests to stderr (default: quiet mode)')
74     parser.add_option('-P',
75                     '--pidfile',
76                     dest = 'pidfile',
77                     default = '/var/run/ajaxwm.pid',
78                     help = 'set the pidfile (default: /var/run/ajaxwm.pid)')
79     parser.add_option('-U',
80                     '--uid',
81                     dest = 'uid',
82                     help = 'set the user id')
83     parser.add_option('-u',
84                     '--uncompressed',
85                     action = 'store_true',
86                     dest = 'gzip',
87                     help = 'responds not compressed ' + \
88                          '(lower system load, higher network load)')
89     parser.add_option('-s',
90                     '--sshcmd',
```

```
91         dest = 'sshCmd',
92         default = 'ssh',
93         help = 'ssh client to use')
94 (options, arguments) = parser.parse_args()
95
96 ssh = Ssh(options.sshCmd)
97 if not ssh.tryCmd():
98     return
99
100 if options.daemon:
101     # Fork a child process, so the parent (this process) can exit.
102     pid = os.fork()
103
104     # Child process
105     if pid == 0:
106         # Make the child process a session leader. (own group)
107         os.setpgrp()
108
109         # don't output and input anything as a daemon
110         io_null = file('/dev/null', 'rw')
111         os.dup2(io_null.fileno(), sys.stdin.fileno())
112         os.dup2(io_null.fileno(), sys.stdout.fileno())
113         os.dup2(io_null.fileno(), sys.stderr.fileno())
114
115         # Running as root, but not supposed to.
116         if os.getuid() == 0 and options.uid:
117             try:
118                 # User id really is an id.
119                 os.setuid(int(options.uid))
120             except:
121                 # User name instead of id.
122                 os.setuid(pwd.getpwnam(options.uid).pw_uid)
123
124         # Parent process
125     else:
126         print 'http://localhost:%s/ PID: %s' % (options.port, pid)
127
128         try:
129             # Try to write pid to file.
130             file(options.pidfile, 'w+').write(str(pid)+'\n')
131         except:
132             # But just ignore if it's not working (permissions, space, ...)
133             pass
134
135         # Exit the parent process, child stays alive as daemon.
136         return 0
137
138 print 'http://localhost:%s/' % options.port
139
140 from Server import Server
141 ajaxwm = Server(not options.gzip)
142
143 try:
144     qweb.QWebWSGIServer(ajaxwm, ip = 'localhost',
145                         port = int(options.port), threaded = 0,
146                         log = options.log).serve_forever()
147 except KeyboardInterrupt, e:
148     print 'keyboard interrupt'
149 except qweb.SocketServer.socket.error:
```

```
150         print 'port %s already in use' % options.port
151
152     #}}}
153     if __name__ == '__main__':#{#{
154         # sys.exit here and the return in main allows running main() in the
155         # interactive prompt without having the prompt killed with main by an
156         # exit-call, useful for testing and debugging.
157         sys.exit(main())
158     #}}}
159
160     #
161     # vim: fdm=marker ts=4 sw=4 expandtab
```

---

### 6.1.2 SERVER.PY

```
1  """
2  ajaxWM – the free web based window manager for remote terminals
3  Copyright (C) 2007 Dennis Felsing, Andreas Waidler
4
5  This program is free software: you can redistribute it and/or modify
6  it under the terms of the GNU General Public License as published by
7  the Free Software Foundation, either version 3 of the License, or
8  (at your option) any later version.
9
10 This program is distributed in the hope that it will be useful,
11 but WITHOUT ANY WARRANTY; without even the implied warranty of
12 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 GNU General Public License for more details.
14
15 You should have received a copy of the GNU General Public License
16 along with this program. If not, see <http://www.gnu.org/licenses/>.
17
18 $Id: Server.py 76 2007-12-15 11:26:24Z r0q $
19 """
20
21 #{#{ Imports
22 import os
23 import mimetypes
24 import re
25 import random
26 import time
27 import sys
28
29 import Files
30 import Singleton
31 import Loop
32 import ajaxwm
33 from Session import Session
34 from Error import ID_Error, Login_Error
35 import qweb
36 #}}}
37 class Server(Singleton.Singleton):#{#{
38     """
39     Server handles the initialization of a new client and responses to
40     requests. There can only be one instance, if you want multiple instances of
41     ajaxWM running on one and the same computer, just run it with defferent port as
42     paremeters.
43     """
44
```

---

```

45     sessions = {}
46     # Loop-instances, one for every session.
47     loops = {}
48
49     MIME = mimetypes.types_map.copy()
50     MIME['.html'] = '%s; charset=%s' % (MIME['.html'], ajaxwm.ENCODING)
51     # Descriptors, keys and types of keys.
52     KEYS = [('session', 's', int), ('event', 'e', str), #{#{
53         ('window', 'W', int), ('key', 'k', unicode),
54         ('width', 'w', int), ('height', 'h', int),
55         ('user', 'u', str), ('password', 'p', str)]
56     #}}}
57     # The mimetypes for every return to an event.
58     EVENT_MIMETYPES = {'sy': 'text/xml', 'kp': 'text/xml', #{#{
59         'nw': 'text/plain', 'cw': 'text/plain',
60         'li': 'text/plain', 'lo': 'text/plain'}
61     #}}}
62     # The maximum of sessions is set to the max of an integer on your system.
63     # That's a compromise about space and security.
64     SESSION_MAX = sys.maxint
65
66     def __init__(self, gzip):#{#{
67         self.gzip = gzip
68
69         self.sessionExists = lambda sid: \
70             type(sid) == int and \
71             self.sessions.has_key(sid) and \
72             self.sessions[sid]
73         self.windowExists = lambda sid, wid: \
74             type(wid) == int and \
75             wid < len(self.sessions[sid].windows) and \
76             self.sessions[sid].windows[wid]
77
78         self.files = Files.Files()._content
79         # Read in all files we need to run.
80         for i in ['themes', 'javascript']:
81             os.path.walk(os.path.abspath(i), self.__addFile, None)
82             self.__addFile(None, os.getcwd(), ('index.html',))
83     #}}}
84     # def __addFile(self, arg, directory, names):#{#{
85     #     """
86     # Adds a File to the self.files-list, so it can be accessed later
87     # when a client connects.
88     #     """
89     #     for name in names:
90     #         # os.getcwd() gets the current working directory, which is the
91     #         # virtual root of the webserver
92     #         path = os.path.join(directory, name)
93     #         if not os.path.isdir(path):
94     #             # Save all found files and set the current working directory
95     #             # as / for requests to them.
96     #             self.files[path[len(os.getcwd()):]] = file(path).read()
97     #             extension = os.path.splitext(name)[1]
98     #             # Use default mimetype if available, else octet-stream.
99     #             self.MIME[extension] = '%s; charset=%s' % \
100                 (self.MIME.get(extension.lower(), |
101                 'application/octet-stream'), |
102                 ajaxwm.ENCODING)
103     #     #}}}

```

---

```

104     def __call__(self, environment, startResponse):#{#{
105         """
106 Gets called by the webserver of the qweb-framework on each request.
107
108 Format of the call:
109 key
110     value1
111     value2
112     ...
113
114 actions describe what a event should do someday.
115 cw and kw for example can do the same while in development phase.
116
117
118 s # session id
119 e # event that happened or should be executed
120     li # login
121         # response: session id or -1 if login fails
122     lo # logout
123         # response: none?
124     nw # new window
125         # parameters: width and height (w, h)
126         # response is the new windows id or -1 on failure
127     cw # close window
128         # response: 1 or 0 (true/false)
129     sy # synchronize window. parameters: W [w, h]
130         # response: new contents/<idem></idem>
131     kp # key press, needs additional parameter (see below)
132         # see e=sy
133
134 # some commands need a target, by now only a window (desktop may come
135     later)
136 W # window (target of the event (e.g. keypress))
137     [0-9]* # value is a integer from 0 (first window) until buffer
138         overflow :P
139
140 # some additional parameters that are used by some commands:
141 k # key, only needed by e=kp
142     !!! see ajaxterm for values
143 w # width, used (optional) by e=sy
144     [0-9]* # chars
145 h # height, used (optional) by e=sy
146     [0-9]* # lines
147 u # user login name, needed by e=li
148 p # password needed for login
149     """
150
151     # Parameters and values from request from client.
152     request = qweb.QWebRequest(environment, startResponse, session=None)
153
154     # query (POST or GET, both work)
155     # GET should be faster than POST, because POST always has to send at
156     # at least 2 packets, while for GET 1 is enough (reduces the traffic
157     # to the server by 50%) According to w3.org GET-Requests also are
158     # encrypted when using SSL, so there should be no security problems
159     #
160     # can be tested by using wget: (POST)
161     # wget --post-data="s=123456&e=kp&k=x" localhost:8022/event
162     # or: (GET)

```

```

163 # wget localhost:8022/event?s=123456|&e=kp|&k=x
164 if request.PATH_INFO.endswith('/event'):
165     keys = {}
166     for key, ident, typ in Server.KEYS:
167         try:
168             if typ == unicode:
169                 keys[key] = typ(request.REQUEST[ident],
170                                ajaxwm.ENCODING)
171             else:
172                 keys[key] = typ(request.REQUEST[ident])
173         except ValueError:
174             # Seems like the key was not valid. Will happen often
175             # because every key is tried to read in while some keys
176             # have no use on some events.
177             if typ == int:
178                 keys[key] = None
179             else:
180                 keys[key] = typ('')
181
182     try:
183         request.response_headers['Content-Type'] = \
184             Server.EVENT_MIMETYPES[keys['event']]
185     except KeyError:
186         # Don't do anything on an invalid key (for now).
187         pass
188
189     if self.sessionExists(keys['session']):
190         ssn = self.sessions[keys['session']]
191         loop = self.loops[keys['session']]
192         if self.windowExists(keys['session'], keys['window']):
193             window = ssn.windows[keys['window']]
194         else:
195             window = None
196     else:
197         ssn = None
198
199     #{{{ Sync and Keypress
200     if keys['event'] in ('sy', 'kp'):
201         try:
202             if not ssn:
203                 raise ID_Error, ID_Error.SESSION
204             if not window:
205                 raise ID_Error, ID_Error.WINDOW
206
207             if keys['event'] == 'sy':
208                 if keys['window'] and keys['height']:
209                     window.resize(keys['width'], keys['height'])
210             else: # keys['event'] == 'kp'
211                 window.write(keys['key'])
212                 # Because the virtual terminal needs some time to
213                 # handle the key(s) we have to wait for that time.
214                 # Else the client would receive nothing at all and
215                 # would have to send another sync just to get thes
216                 # results of his keypresses.
217                 time.sleep(0.001)
218             ssn.sync(keys['window'])
219             # Reset killing time.
220             loop.newTime()
221             r = window.html()

```

```

222         except ID_Error, e:
223             r = e.value
224             # Python 2.5+
225             #finally:
226             #    request.write((u''.join((XMLSTRING, r)))\
227                               #    .encode(ajaxwm.ENCODING))
228             request.write((u''.join((ajaxwm.XMLSTRING, r)))\
229                           .encode(ajaxwm.ENCODING))
230     #}}}
231     #{{{ New window
232     elif keys['event'] == 'nw':
233         if ssn:
234             # Return the window id, -1 on error.
235             request.write(str(ssn.newWindow(keys['width'],
236                                           keys['height'])))
237         else:
238             request.write('-1')
239     #}}}
240     #{{{ Close window
241     elif keys['event'] == 'cw':
242         if ssn and window:
243             ssn.deleteWindow(keys['window'])
244             request.write('1')
245         else:
246             request.write('0')
247     #}}}
248     #{{{ Login
249     elif keys['event'] == 'li':
250         try:
251             # Check for valid UNIX username and non-empty password.
252             if not (re.match('^[0-9A-Za-z-_. ]+$', keys['user']) \
253                   and keys['password']):
254                 raise Login_Error, -1
255
256             sessionId = self.SESSION_MAX
257             while sessionId != -1 \
258                   and (sessionId == self.SESSION_MAX \
259                       or not self.sessionExists(sessionId)):
260                 sessionId = random.randint(0, self.SESSION_MAX - 1)
261                 if self.loops.has_key(sessionId):
262                     # Wait for the loop to finish (max: one second).
263                     self.loops[sessionId].join()
264                 self.sessions[sessionId] = \
265                     Session(u'localhost', keys['user'],
266                           keys['password'])
267             login = self.sessions[sessionId].tryLogin()
268             if login < 0:
269                 del self.sessions[sessionId]
270                 raise Login_Error, login
271         except Login_Error, e:
272             sessionId = e.value
273         else:
274             # Also create a new loop, which is assigned to the session.
275             self.loops[sessionId] = Loop.Loop(sessionId)
276             self.loops[sessionId].start()
277             request.write(str(sessionId))
278     #}}}
279     #{{{ Logout
280     elif keys['event'] == 'lo':

```

```
281         if ssn:
282             # Both have to be deleted in order to call the
283             # __del__-method of the Session object.
284             del ssn
285             del self.sessions[keys['session']]
286             # We already killed the session, so tell the loop to abort.
287             loop.abort()
288             # Not deleting the loop, so at the login can check if the
289             # one second the loop needs already is over.
290
291             request.write('1')
292         else:
293             request.write('0')
294     #}}}
295
296     else:
297         file = ""
298         if self.files.has_key(request.PATH_INFO):
299             file = request.PATH_INFO
300
301         elif request.PATH_INFO == '/':
302             file = '/index.html'
303
304         else:
305             # Redirect anything else to the 404-error.
306             request.http_404()
307
308         if file:
309             request.response_headers['Content-Type'] = \
310                 self.MIME[os.path.splitext(file)[1]]
311
312             request.write(self.files[file])
313
314         if not 'text/plain' in request.response_headers.get('Content-Type'):
315             request.response_gzencode = self.gzip
316
317     return request
318 #}}}
319 #}}}
320
321 #
322 # vim: fdm=marker ts=4 sw=4 expandtab
```

---

### 6.1.3 FILES.PY

```
1 """
2 ajaxWM - the free web based window manager for remote terminals
3 Copyright (C) 2007 Dennis Felsing, Andreas Waidler
4
5 This program is free software: you can redistribute it and/or modify
6 it under the terms of the GNU General Public License as published by
7 the Free Software Foundation, either version 3 of the License, or
8 (at your option) any later version.
9
10 This program is distributed in the hope that it will be useful,
11 but WITHOUT ANY WARRANTY; without even the implied warranty of
12 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 GNU General Public License for more details.
14
```



```
15 You should have received a copy of the GNU General Public License
16 along with this program. If not, see <http://www.gnu.org/licenses/>.
17
18 $Id: Files.py 76 2007-12-15 11:26:24Z r0q $
19 """
20
21 #{#{ Imports
22 import os
23 import re
24 import xml.dom.minidom
25
26 import ajaxwm
27 #{}}
28 class Files:#{#{
29     """
30 Reads in all files the clients will have access to. It does not read in
31 files beginning with a dot.
32     """
33     _content = {}
34     _themes = []
35
36     FILES_DIR = 'htdocs'
37     THEMES_DIR = 'themes'
38     BLACKLIST = re.compile(
39         """
40         \.* # No hidden files and files in hidden directories.
41         """
42         , re.DOTALL | re.VERBOSE)
43
44     def __init__(self):#{#{
45         for dir in (self.FILES_DIR, self.THEMES_DIR):
46             self._walk(dir)
47             self._themesToXml()
48     #{}}
49     def _parseTheme(self, path):#{#{
50         """
51 Reads in the xml file of a theme and returns its attributes.
52         """
53         sAttrs = xml.dom.minidom.parse(path).firstChild.attributes
54         dAttrs = {}
55         for attribute in sAttrs.keys():
56             dAttrs[attribute] = sAttrs[attribute].value
57         return dAttrs
58     #{}}
59     def _themesToXml(self):#{#{
60         # <?xml version="1.0" ?>
61         doc = xml.dom.minidom.Document()
62         # <ajaxwm>
63         eAjaxwm = doc.createElement('ajaxwm')
64         doc.appendChild(eAjaxwm)
65         for theme in self._themes:
66             values = self._parseTheme(theme['realPath'])
67             # <theme name="NAME" author="AUTHOR" email="EMAIL">
68             eTheme = doc.createElement('theme')
69             eTheme.setAttribute('name', values['name'])
70             eTheme.setAttribute('author', values['author'])
71             eTheme.setAttribute('email', values['email'])
72             eAjaxwm.appendChild(eTheme)
73
```

```

74         # <path>
75         ePath = doc.createElement('path')
76         eTheme.appendChild(ePath)
77         tPath = doc.createTextNode(theme['userPath'])
78         ePath.appendChild(tPath)
79         # </path>
80
81         # <description>
82         eDesc = doc.createElement('description')
83         eTheme.appendChild(eDesc)
84         tDesc = doc.createTextNode(values['description'])
85         eDesc.appendChild(tDesc)
86         # </description>
87         # </theme>
88     # </ajaxwm>
89
90     self._content['/' + self.THEMES_DIR + '/themes.xml'] = \
91     doc.toprettyxml(indent="    ")
92     #}}}
93     def _walk(self, path):#{{{
94         """
95     Walks through the paths given as parameters and writes the valid ones to the
96     content variable of the class.
97     """
98     if not os.path.isdir(path):
99         return
100    for root, dirs, files in os.walk(path):
101        self._validate(dirs)
102        self._validate(files)
103        if root == self.THEMES_DIR:
104            self._validateThemes(dirs)
105        for file in files:
106            self._read('/'.join((root, file)))
107    #}}}
108    def _validate(self, objects):#{{{
109        """
110    Deletes all objects that do not match the BLACKLIST.
111    """
112    for object in objects:
113        if self.BLACKLIST.search(object).group():
114            objects.remove(object)
115    #}}}
116    def _validateThemes(self, dirs):#{{{
117        """
118    Deletes all theme directories that are not valid for this version.
119    """
120    for dir in dirs:
121        if not ajaxwm.VALID_THEMES[0] \
122        <= map(int, dir.split('.')) \
123        <= ajaxwm.VALID_THEMES[1]:
124            dirs.remove(dir)
125    #}}}
126    def _destination(self, path):#{{{
127        pathElements = path.split('/')
128        for pos, pathElement in enumerate(pathElements):
129            if pathElement == self.THEMES_DIR:
130                del pathElements[pos + 1]
131                themePath = '/'.join([''] + pathElements)
132                if themePath.endswith('.xml'):

```

```
133             self._themes.append({'realPath': path,
134                                   'userPath': themePath})
135         return themePath
136     return path[len(self.FILES_DIR):]
137     #}}}
138     def _read(self, path):#{#{
139         """
140 Reads in a file to the content.
141         """
142         self._content[self._destination(path)] = open(path, 'r').read()
143     #}}}
144 #}}}
145
146 #-----
147 # vim: fdm=marker ts=4 sw=4 expandtab
```

#### 6.1.4 SESSION.PY

```
1  """
2  ajaxWM – the free web based window manager for remote terminals
3  Copyright (C) 2007 Dennis Felsing, Andreas Waidler
4
5  This program is free software: you can redistribute it and/or modify
6  it under the terms of the GNU General Public License as published by
7  the Free Software Foundation, either version 3 of the License, or
8  (at your option) any later version.
9
10 This program is distributed in the hope that it will be useful,
11 but WITHOUT ANY WARRANTY; without even the implied warranty of
12 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 GNU General Public License for more details.
14
15 You should have received a copy of the GNU General Public License
16 along with this program. If not, see <http://www.gnu.org/licenses/>.
17
18 $Id: Session.py 76 2007-12-15 11:26:24Z r0q $
19 """
20
21 #{#{ Imports
22 import time
23 import pty
24 import fcntl
25 import os
26
27 from Window import Window
28 from Error import Login_Error
29 from Ssh import Ssh
30 import ajaxwm
31 #}}}
32 class Session:#{#{
33     """
34 Each client connects to exactly one Session object. This object manages
35 the windows of the client.
36     """
37     WINDOW_MAX = 2 ** 6
38     TIMEOUT = 10 * 60 # in seconds
39     LOGIN_SLEEP = 0.005
40
41     def __init__(self, host, username, password):#{#{
```

```
42         # Do not permit spaces!
43         self.host = host.translate({32: None})
44         if self.host.find(':') != -1:
45             self.host, self.port = host.split(':')
46         else:
47             self.port = u'22'
48         self.username = username
49         self.password = password
50         self.windows = []
51         self.time = time.time()
52     #}}}
53     def tryLogin(self):#{#{
54         """
55         Checks if logging in with the entered data is possible. Works
56         nearly the same way as the method run of the Window class, look there
57         for more details.
58         """
59         pid, fd = pty.fork()
60         if pid != 0:
61             content = ''
62             timeout = time.time()
63             if self.host in (u'localhost', u'127.0.0.1'):
64                 # On localhost delays normally aren't that big.
65                 timeout += 1
66             else:
67                 # But they might be if we are working as a gateway.
68                 timeout += 5
69
70             fcntl.fcntl(fd, fcntl.F_SETFL, os.O_NONBLOCK)
71             while 1:
72                 try:
73                     content = ajaxwm.read(fd)
74                 except OSError:
75                     pass
76
77                 try:
78                     if 'assword:' in content:
79                         os.write(fd, self.password + '\n')
80                         while 1:
81                             try:
82                                 content = ajaxwm.read(fd)
83                             except OSError:
84                                 pass
85
86                             if ('Connection to %s closed.'
87                                 % self.host.encode(ajaxwm.ENCODING)) in content:
88                                 # Login worked.
89                                 raise Login_Error, 0
90                             if timeout < time.time():
91                                 # Wrong username or password.
92                                 raise Login_Error, -1
93                             time.sleep(self.LOGIN_SLEEP)
94                             if timeout < time.time():
95                                 # Wrong host or sshd not running.
96                                 raise Login_Error, -2
97
98             except Login_Error, e:
99                 os.close(fd)
100             return e.value
```

```
101
102         time.sleep(self.LOGIN_SLEEP)
103
104     ssh = Ssh()
105     cmd = ssh.tryLogin(self.username, self.host, self.port)
106     os.execvpe(cmd[0], cmd, None)
107     #}}}
108     def newWindow(self, width=80, height=25):#{{{
109         """Create a new Window and return its ID."""
110         win = Window(self.username, self.host, self.port, self.password,
111                     width, height)
112         # To improve performance append windows normally.
113         if len(self.windows) < Session.WINDOW_MAX:
114             self.windows.append(win)
115             self.windows[-1].start()
116             self.windows[-1].join()
117             return len(self.windows) - 1
118         # Else search for free window from 0 to limit (no need in most cases).
119         else:
120             for windowId, wind in enumerate(self.windows):
121                 if wind == None:
122                     wind = win
123                     wind.start()
124                     return windowId
125         # All windows are used.
126         return -1
127     #}}}
128     def deleteWindow(self, window):#{{{
129         """Delete an existing window and return 1 on success, 0 on failure."""
130         self.windows[window] = None
131     #}}}
132     def sync(self, window):#{{{
133         """
134 Reads the contents of a specified pty to a Terminal object.
135 The time for the loop is set to the time.
136         """
137         if window < len(self.windows) and self.windows[window]:
138             self.time = time.time()
139             self.windows[window].read()
140     #}}}
141 #}}}
142
143 #
144 # vim: fdm=marker ts=4 sw=4 expandtab
```

---

### 6.1.5 LOOP.PY

```
1 """
2 ajaxWM - the free web based window manager for remote terminals
3 Copyright (C) 2007 Dennis Felsing, Andreas Waidler
4
5 This program is free software: you can redistribute it and/or modify
6 it under the terms of the GNU General Public License as published by
7 the Free Software Foundation, either version 3 of the License, or
8 (at your option) any later version.
9
10 This program is distributed in the hope that it will be useful,
11 but WITHOUT ANY WARRANTY; without even the implied warranty of
12 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
```

```
13 GNU General Public License for more details.
14
15 You should have received a copy of the GNU General Public License
16 along with this program. If not, see <http://www.gnu.org/licenses/>.
17
18 $Id: Loop.py 76 2007-12-15 11:26:24Z r0q $
19 """
20
21 #{{{ Imports
22 import time
23 import threading
24
25 import Server
26 import Session
27 #}}}
28 class Loop(threading.Thread):#{{{
29     """
30 Loop class for checking every TIMEOUT seconds if the session
31 it was called for did do something. If the session was idling all the
32 time, which in most cases means the client holding the session has been
33 disconnected, it is killed.
34
35 If the session already got killed somewhere else the object of this class
36 is destroyed so it doesn't delete another session that took the place of
37 the old one.
38     """
39     def __init__(self, sessionId):#{{{
40         threading.Thread.__init__(self)
41
42         self.sessionId = sessionId
43         self._wantAbort = False
44
45         self.newTime()
46     #}}}
47     def run(self):#{{{
48         while True:
49             oldTime = time.time()
50             for i in range(Session.Session.TIMEOUT):
51                 # Check every second if we have been aborted.
52                 time.sleep(1)
53                 if self._wantAbort:
54                     # abort-Method has been run. That means someone else
55                     # already killed the session this object is watching over.
56                     return
57             if time.time() - self._time > Session.Session.TIMEOUT:
58                 # Commit suicide
59                 Server.Server.sessions[self.sessionId] = None
60             return
61     #}}}
62     def newTime(self):#{{{
63         """Save the current time."""
64         self._time = time.time()
65     #}}}
66     def abort(self):#{{{
67         """Make the loop leave and the thread die."""
68         self._wantAbort = True
69     #}}}
70 #}}}
71
```

```
72 #
73 # vim: fdm=marker ts=4 sw=4 expandtab
```

### 6.1.6 WINDOW.PY

```
1 """
2 ajaxWM – the free web based window manager for remote terminals
3 Copyright (C) 2007 Dennis Felsing, Andreas Waidler
4
5 This program is free software: you can redistribute it and/or modify
6 it under the terms of the GNU General Public License as published by
7 the Free Software Foundation, either version 3 of the License, or
8 (at your option) any later version.
9
10 This program is distributed in the hope that it will be useful,
11 but WITHOUT ANY WARRANTY; without even the implied warranty of
12 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 GNU General Public License for more details.
14
15 You should have received a copy of the GNU General Public License
16 along with this program. If not, see <http://www.gnu.org/licenses/>.
17
18 $Id: Window.py 76 2007-12-15 11:26:24Z r0q $
19 """
20
21 #{#{ Imports
22 import threading
23 import os
24 import pty
25 import fcntl
26 import time
27
28 from Terminal import Terminal
29 from Ssh import Ssh
30 import ajaxwm
31 #{}}
32 class Window(threading.Thread):#{{{
33     """
34     For every window a new thread is spawned. Objects of the window class
35     control one Terminal each. So they also have to open and close the pty for
36     the Terminal.
37
38     A programm, which is specified by cmd, is assigned to the Terminal object
39     and set to ssh localhost by default. By this the client can for example
40     login on the server. But of course this is not the only way of using an
41     ajaxWM-Window. Another useful thing is assigning a screen session
42     read-only, so one interact with that screen session, while others can watch
43     him from their PC.
44
45     Windows can only be spawned by and accessed via an object of the Session
46     class.
47     """
48     def __init__(self, username, host, port, password, width, height):#{{{
49         threading.Thread.__init__(self)
50
51         self.username = username
52         self.host = host
53         self.port = port
54         self.password = password
```

---

```

55         self.width = width
56         self.height = height
57     #}}}
58     def run(self):#{
59         """
60         Forks a virtual terminal and there runs the specified command
61         gets called when the thread is started.
62         """
63         pid, fd = pty.fork()
64         if pid == 0:
65             # The process of the virtual terminal
66             try:
67                 # A list of the existing file descriptors.
68                 fdl = map(int, os.listdir('/proc/self/fd'))
69             except OSError:
70                 fdl = range(256)
71             for i in filter(lambda i: i > 2, fdl):
72                 # We only need stdin(i == 0), stdout(1) and stderr(2).
73                 try:
74                     os.close(i)
75                 except OSError:
76                     pass
77             # SSH instead of /bin/login even for login on localhost because
78             # /bin/login requires root access and I myself wouldn't try to use
79             # this program as root. (paranoia is healthy!) If you intended to
80             # do so, better use another user with as little rights as
81             # possible. The only disadvantage is the longer time to login
82             # (or to realize that you can't) when using SSH.
83             ssh = Ssh()
84             cmd = ssh.login(self.username, self.host, self.port)
85             env = {'COLUMNS': str(self.width),
86                  'LINES': str(self.height),
87                  'TERM': 'linux',
88                  'PATH': os.environ['PATH']}
89             # See the POSIX execvp (which gets called) for information.
90             os.execvp(cmd[0], cmd, env)
91         else:
92             # The main/parent process
93             self.pid = pid
94             self.fd = fd
95
96             # sets the file status flags for the open file descriptor fd to
97             # non blocking mode (just continue if nothing can be read)
98             fcntl.fcntl(fd, fcntl.F_SETFL, os.O_NONBLOCK)
99             self.terminal = Terminal(self.width, self.height)
100             self.resize(self.width, self.height)
101             self.writePassword()
102     #}}}
103     def __del__(self):#{
104         # Kill all base classes.
105         for base in self.__class__.__bases__:
106             # Avoid problems with diamond inheritance.
107             basekey = 'del_' + str(base)
108             if not hasattr(self, basekey):
109                 setattr(self, basekey, 1)
110             else:
111                 continue
112             # Call this base class' destructor if it has one.
113             if hasattr(base, '__del__'):

```



```
114         base.__del__(self)
115     try:
116         # Closing the pty also kills all processes on it.
117         os.close(self.fd)
118     except (IOError, OSError):
119         pass
120     #}}}
121     def resize(self, width=80, height=25):#{#{
122         """
123     Client has resized his window, so now the virtual terminal and the
124     object of the Terminal class have to be informed and actualised.
125     """
126     if not self.width is width and not self.height is height:
127         self.width, self.height = width, height
128         self.terminal.resize(width, height)
129         fcntl.ioctl(fd, termios.TIOCSWINSZ,
130                     struct.pack('HHHH', height, width, 0, 0))
131     #}}}
132     def read(self):#{#{
133         """Makes the Terminal object read the content of the pty."""
134     try:
135         self.terminal.write(ajaxwm.read(self.fd))
136     except (KeyError, IOError, OSError):
137         del self
138     #}}}
139     def write(self, s): #{#{
140         """Writes new keypresses to the pty."""
141     try:
142         os.write(self.fd, s.encode(ajaxwm.ENCODING))
143     except UnicodeEncodeError:
144         os.write(self.fd, s)
145     except (IOError, OSError):
146         del self
147     #}}}
148     def html(self, color=1):#{#{
149         """Returns the html version of the content of the window."""
150         self.read()
151         self.join()
152     try:
153         return self.terminal.dumphtml(color)
154     except KeyError:
155         return False
156     #}}}
157     def writePassword(self):#{#{
158         while 1:
159             try:
160                 if 'Password:' in ajaxwm.read(self.fd):
161                     self.write(self.password + u'\n')
162                     return
163             except OSError:
164                 pass
165             time.sleep(0.2)
166     #}}}
167 #}}}
168
169 #
170 # vim: fdm=marker ts=4 sw=4 expandtab
```

## 6.1.7 TERMINAL.PY

```
1 """
2 ajaxWM – the free web based window manager for remote terminals
3 Copyright (C) 2007 Dennis Felsing, Andreas Waidler
4
5 This program is free software: you can redistribute it and/or modify
6 it under the terms of the GNU General Public License as published by
7 the Free Software Foundation, either version 3 of the License, or
8 (at your option) any later version.
9
10 This program is distributed in the hope that it will be useful,
11 but WITHOUT ANY WARRANTY; without even the implied warranty of
12 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 GNU General Public License for more details.
14
15 You should have received a copy of the GNU General Public License
16 along with this program. If not, see <http://www.gnu.org/licenses/>.
17
18 $Id: Terminal.py 76 2007-12-15 11:26:24Z r0q $
19 """
20
21 #{#{ Imports
22 import re
23 import array
24 import cgi
25
26 import ajaxwm
27 #{}}
28 class Terminal:#{#{
29     """
30     Represents the contents of a terminal, with wich it can be populated
31     with. Printing the contents can either be done as pure unicode and included
32     in a pre-tag for being directly used in HTML.
33
34     Up to now there are still some problems with encoding, if you don't use
35     Unicode (Unicode should work without problems, _should_, doesn't at some
36     extended characters). A class for handling the process of encoding and
37     decoding is already planned.
38     """
39     def __init__(self, width=80, height=24):#{#{
40         self.width = width
41         self.height = height
42         self.init()
43         self.reset()
44     #{}}
45     def init(self):#{#{
46         """
47         Initialization of the escape-, the csi-sequences, the regex and the
48         to-html-translation table.
49         """
50         self.esc_seq = {u'\x00': None,
51                         u'\x05': self.esc_da,
52                         u'\x07': None,
53                         u'\x08': self.esc_0x08,
54                         u'\x09': self.esc_0x09,
55                         u'\x0a': self.esc_0x0a,
56                         u'\x0b': self.esc_0x0a,
57                         u'\x0c': self.esc_0x0a,
```

---

```

58         u'\x0d': self.esc_0x0d,
59         u'\x0e': None,
60         u'\x0f': None,
61         u'\x1b#8': None,
62         u'\x1b=': None,
63         u'\x1b>': None,
64         u'\x1b(0': None,
65         u'\x1b(A': None,
66         u'\x1b(B': None,
67         u'\x1b[c': self.esc_da,
68         u'\x1b[0c': self.esc_da,
69         u'\x1b]R': None,
70         u'\x1b7': self.esc_save,
71         u'\x1b8': self.esc_restore,
72         u'\x1bD': None,
73         u'\x1bE': None,
74         u'\x1bH': None,
75         u'\x1bM': self.esc_ri,
76         u'\x1bN': None,
77         u'\x1bO': None,
78         u'\x1bZ': self.esc_da,
79         u'\x1ba': None,
80         u'\x1bc': self.reset,
81         u'\x1bn': None,
82         u'\x1bo': None}
83     for k, v in self.esc_seq.items():
84         if v == None:
85             self.esc_seq[k] = self.esc_ignore
86     # regex
87     d = {r'\[\??([0-9;]*)(@ABCDEFGHJKLMPXacdefghlmnrstu')':
88         self.csi_dispatch, r'\|([^\x07+)]\x07': self.esc_ignore}
89     self.esc_re = []
90     for k,v in d.items():
91         self.esc_re.append((re.compile(''.join((u'\x1b', k))), v))
92     # define csi sequences
93     self.csi_seq = {'@': (self.csi_at, [1]),
94                    ' ': (self.csi_G, [1]),
95                    'J': (self.csi_J, [0]),
96                    'K': (self.csi_K, [0])}
97     for i in map(lambda i: i[4], filter(lambda i: i.startswith('csi_') and
98                                     len(i) == 5, dir(self))):
99         if not self.csi_seq.has_key(i):
100             self.csi_seq[i] = (getattr(self, ''.join(('csi_', i))), [1])
101     # First 33 characters are not printable. (^Z excluded)
102     # Tuple instead of List to increase performance.
103     validChars = tuple(range(0x20, 0xffff))
104     self.trUnicode = (0x20,) * 0x20 + validChars
105     self.trHTML = [0xa0] * 0x20
106     self.trHTML[0x0a] = 0x0a
107     self.trHTML = tuple(self.trHTML) + validChars
108     #}}}
109     def resize(self, width, height):#{{{
110         """Resizes the Terminal. (Just says where to break line and how many
111         lines there are.)
112         """
113         self.width = width
114         self.height = height
115     #}}}
116     def reset(self, s=''):#{{{

```

```

117         """Resets all values."""
118         # self.scr and self.sgr:
119         # 0 x 0 0 0 0 0 0
120         #   V V \   /
121         #   bg fg   char
122         #   0-9 0-9 0-0xfffd
123         self.scr = array.array('i', [0x070000] * (self.width * self.height))
124         self.st = 0
125         self.sb = self.height - 1
126         self.cx_bak = self.cx = 0
127         self.cy_bak = self.cy = 0
128         self.cl = 0
129         self.sgr = 0x070000
130         self.buf = []
131         self.outbuf = []
132         self.last_html = u''
133     #}}}
134     def peek(self, y1, x1, y2, x2):#{{{
135         return self.scr[self.width * y1 + x1:self.width * y2 + x2]
136     #}}}
137     def poke(self, y, x, s):#{{{
138         pos = self.width * y + x
139         self.scr[pos:pos + len(s)] = s
140     #}}}
141     def zero(self, y1, x1, y2, x2):#{{{
142         w = self.width * (y2 - y1) + x2 - x1 + 1
143         z = array.array('i', [0x070000] * w)
144         self.scr[self.width * y1 + x1:self.width * y2 + x2 + 1] = z
145     #}}}
146     def scroll_up(self, y1, y2):#{{{
147         self.poke(y1, 0, self.peek(y1 + 1, 0, y2, self.width))
148         self.zero(y2, 0, y2, self.width - 1)
149     #}}}
150     def scroll_down(self, y1, y2):#{{{
151         self.poke(y1 + 1, 0, self.peek(y1, 0, y2 - 1, self.width))
152         self.zero(y1, 0, y1, self.width - 1)
153     #}}}
154     def scroll_right(self, y, x):#{{{
155         self.poke(y, x + 1, self.peek(y, x, y, self.width))
156         self.zero(y, x, y, x)
157     #}}}
158     def cursor_down(self):#{{{
159         if self.cy >= self.st and self.cy <= self.sb:
160             self.cl = 0
161             q, r = divmod(self.cy + 1, self.sb + 1)
162             if q:
163                 self.scroll_up(self.st, self.sb)
164                 self.cy = self.sb
165             else:
166                 self.cy = r
167     #}}}
168     def cursor_right(self):#{{{
169         q, r = divmod(self.cx + 1, self.width)
170         if q:
171             self.cl = 1
172         else:
173             self.cx = r
174     #}}}
175     def echo(self, c):#{{{

```

```
176         if self.cl:
177             self.cursor_down()
178             self.cx = 0
179             #print hex(self.sgr | ord(c))
180             self.scr[(self.cy * self.width) + self.cx] = self.sgr | ord(c)
181             self.cursor_right()
182         #}}}
183     def esc_0x08(self, s):#{{{
184         self.cx = max(0, self.cx - 1)
185     #}}}
186     def esc_0x09(self, s):#{{{
187         x = self.cx + 8
188         q, r = divmod(x, 8)
189         self.cx = (q * 8) % self.width
190     #}}}
191     def esc_0x0a(self, s):#{{{
192         self.cursor_down()
193     #}}}
194     def esc_0x0d(self, s):#{{{
195         self.cl = 0
196         self.cx = 0
197     #}}}
198     def esc_save(self, s):#{{{
199         self.cx_bak = self.cx
200         self.cy_bak = self.cy
201     #}}}
202     def esc_restore(self, s):#{{{
203         self.cx = self.cx_bak
204         self.cy = self.cy_bak
205         self.cl = 0
206     #}}}
207     def esc_da(self, s):#{{{
208         self.outbuf = [i for i in u'\x1b[?6c']
209     #}}}
210     def esc_ri(self, s):#{{{
211         self.cy = max(self.st, self.cy - 1)
212         if self.cy == self.st:
213             self.scroll_down(self.st, self.sb)
214     #}}}
215     def esc_ignore(self, *s):#{{{
216         pass
217         #print 'term:ignore: %s' % repr(s)
218     #}}}
219     def csi_dispatch(self, seq, mo):#{{{
220         # CSI sequences
221         s = mo.group(1)
222         c = mo.group(2)
223         f = self.csi_seq.get(c, None)
224         if f:
225             try:
226                 l = map(lambda i: min(int(i), 1024),
227                        filter(lambda i: len(i) < 4, s.split(';')))
228             except ValueError:
229                 l = []
230             if len(l) == 0:
231                 l = f[1]
232             f[0](l)
233         #else:
234         # print 'csi ignore', c, l
```

```
235     #}}}
236     def csi_at(self, l):#{#{
237         for i in range(l[0]):
238             self.scroll_right(self.cy, self.cx)
239     #}}}
240     def csi_A(self, l):#{#{
241         self.cy = max(self.st, self.cy - l[0])
242     #}}}
243     def csi_B(self, l):#{#{
244         self.cy = min(self.sb, self.cy + l[0])
245     #}}}
246     def csi_C(self, l):#{#{
247         self.cx = min(self.width - 1, self.cx + l[0])
248         self.cl = 0
249     #}}}
250     def csi_D(self, l):#{#{
251         self.cx = max(0, self.cx - l[0])
252         self.cl = 0
253     #}}}
254     def csi_E(self, l):#{#{
255         self.csi_B(l)
256         self.cx = 0
257         self.cl = 0
258     #}}}
259     def csi_F(self, l):#{#{
260         self.csi_A(l)
261         self.cx = 0
262         self.cl = 0
263     #}}}
264     def csi_G(self, l):#{#{
265         self.cx = min(self.width, l[0]) - 1
266     #}}}
267     def csi_H(self, l):#{#{
268         if len(l) < 2:
269             l = [1, 1]
270         self.cx = min(self.width, l[1]) - 1
271         self.cy = min(self.height, l[0]) - 1
272         self.cl = 0
273     #}}}
274     def csi_J(self, l):#{#{
275         if l[0] == 0:
276             self.zero(self.cy, self.cx, self.height - 1, self.width - 1)
277         elif l[0] == 1:
278             self.zero(0, 0, self.cy, self.cx)
279         elif l[0] == 2:
280             self.zero(0, 0, self.height - 1, self.width - 1)
281     #}}}
282     def csi_K(self, l):#{#{
283         if l[0] == 0:
284             self.zero(self.cy, self.cx, self.cy, self.width - 1)
285         elif l[0] == 1:
286             self.zero(self.cy, 0, self.cy, self.cx)
287         elif l[0] == 2:
288             self.zero(self.cy, 0, self.cy, self.width - 1)
289     #}}}
290     def csi_L(self, l):#{#{
291         for i in range(l[0]):
292             if self.cy < self.sb:
293                 self.scroll_down(self.cy, self.sb)
```

```

294     #}}}
295     def csi_M(self, l):#{#{
296         if self.cy >= self.st and self.cy <= self.sb:
297             for i in range(l[0]):
298                 self.scroll_up(self.cy, self.sb)
299     #}}}
300     def csi_P(self, l):#{#{
301         w = self.width
302         cx = self.cx
303         cy = self.cy
304         end = self.peek(cy, cx, cy, w)
305         self.csi_K([0])
306         self.poke(cy, cx, end[l[0]:])
307     #}}}
308     def csi_X(self, l):#{#{
309         self.zero(self.cy, self.cx, self.cy, self.cx + l[0])
310     #}}}
311     def csi_a(self, l):#{#{
312         self.csi_C(1)
313     #}}}
314     def csi_c(self, l):#{#{
315         #u'\x1b[?0c' 0-8 cursor size
316         pass
317     #}}}
318     def csi_d(self, l):#{#{
319         self.cy = min(self.height, l[0]) - 1
320     #}}}
321     def csi_e(self, l):#{#{
322         self.csi_B(1)
323     #}}}
324     def csi_f(self, l):#{#{
325         self.csi_H(1)
326     #}}}
327     def csi_h(self, l):#{#{
328         if l[0] == 4:
329             pass
330             #print 'insert on'
331     #}}}
332     def csi_l(self, l):#{#{
333         if l[0] == 4:
334             pass
335             #print 'insert off'
336     #}}}
337     def csi_m(self, l):#{#{
338         # Select Graphic Rendition
339         for i in l:
340             if i in (0, 27, 39, 49):
341                 self.sgr = 0x070000
342             elif i == 1:
343                 self.sgr = (self.sgr | 0x080000)
344             elif i == 7:
345                 self.sgr = 0x700000
346             elif i >= 30 and i <= 37:
347                 c = i - 30
348                 self.sgr = (self.sgr & 0xf8ffff) | (c << 16)
349             elif i >= 40 and i <= 47:
350                 c = i - 40
351                 self.sgr = (self.sgr & 0x8fffff) | (c << 20)
352         # else:

```

---

```

353         #          print 'CSI sgr ignore', l, i
354         #print 'sgr: %r %x' % (l, self.sgr)
355     #}}}
356     def csi_r(self, l):#{{{
357         if len(l) < 2: l = [0, self.height]
358         self.st = min(self.height - 1, l[0] - 1)
359         self.sb = min(self.height - 1, l[1] - 1)
360         self.sb = max(self.st, self.sb)
361     #}}}
362     def csi_s(self, l):#{{{
363         self.esc_save(0)
364     #}}}
365     def csi_u(self, l):#{{{
366         self.esc_restore(0)
367     #}}}
368     def escape(self):#{{{
369         e = u''.join(self.buf)
370         if len(e) > 32:
371             #print 'error %r' % e
372             self.buf = []
373         elif e in self.esc_seq:
374             self.esc_seq[e](e)
375             self.buf = []
376         else:
377             for r, f in self.esc_re:
378                 mo = r.match(e)
379                 if mo:
380                     f(e, mo)
381                     self.buf = []
382                     break
383             #if self.buf == []: print 'ESC %r\n' % e
384     #}}}
385     def write(self, s):#{{{
386         for i in s:
387             #print hex(ord(i)),
388             if len(self.buf) or (i in self.esc_seq):
389                 #print '1'
390                 self.buf.append(i)
391                 self.escape()
392             elif i == '\x1b':
393                 #print '2'
394                 self.buf.append(i)
395             else:
396                 #print '3'
397                 self.echo(i)#}}}
398     def read(self):#{{{
399         b = u''.join(self.outbuf)
400         self.outbuf = []
401         return b
402     #}}}
403     def dumpunicode(self):#{{{
404         """
405         Returns all characters of the terminal, non-printable ones excluded
406         in plain text.
407         """
408         return (''.join(map(lambda c: unichr(c & 0x00ffff), self.scr)))\
409             .translate(self.trUnicode)
410     #}}}
411     def dumphtml(self, color=1):#{{{

```



---

```

412     """Returns the terminal in (or not in) color as html."""
413     h = self.height
414     w = self.width
415     r = []
416     span = []
417     span_bg = -1
418     span_fg = -1
419     for i in range(h * w):
420         q, c = divmod(self.scr[i], 0x10000)
421         if color:
422             bg, fg = divmod(q, 0x10)
423         else:
424             bg = 0 # black
425             fg = 7 # white
426         if i == self.cy * w + self.cx:
427             # cursor
428             bg = 1 # red
429             fg = 7 # white
430         if (bg != span_bg or fg != span_fg or i == h * w - 1):
431             # new color
432             if len(span):
433                 r.append(u'<span class="f%d b%d">%s</span>' % \
434                     (span_fg, span_bg, cgi.escape(u''.join(span))\
435                     .translate(self.trHTML)))
436                 span = []
437                 span_bg = bg
438                 span_fg = fg
439             span.append(unichr(c))
440             if i % w == w - 1:
441                 span.append(u'\n')
442     r = u'<pre class="term">%s</pre>' % u''.join(r)
443     if self.last_html == r:
444         # Same result as on last run, no need to write again.
445         return u'<idem></idem>'
446     else:
447         # Contents changed, return it all.
448         self.last_html = r
449         return r
450     #}}}
451     def __repr__(self):#{#{
452         d = self.dumpunicode()
453         r = []
454         for i in range(self.height):
455             r.append(u'|%s|\n' % d[self.width * i:self.width * (i + 1)])
456         return ''.join(r).encode(ajaxwm.ENCODING)
457     #}}}
458 #}}}
459
460 #-----
461 # vim: fdm=marker ts=4 sw=4 expandtab

```

---

### 6.1.8 SINGLETON.PY

```

1  """
2  ajaxWM - the free web based window manager for remote terminals
3  Copyright (C) 2007 Dennis Felsing, Andreas Waidler
4
5  This program is free software: you can redistribute it and/or modify
6  it under the terms of the GNU General Public License as published by

```

```

7 the Free Software Foundation, either version 3 of the License, or
8 (at your option) any later version.
9
10 This program is distributed in the hope that it will be useful,
11 but WITHOUT ANY WARRANTY; without even the implied warranty of
12 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 GNU General Public License for more details.
14
15 You should have received a copy of the GNU General Public License
16 along with this program. If not, see <http://www.gnu.org/licenses/>.
17
18 $Id: Singleton.py 76 2007-12-15 11:26:24Z r0q $
19 """
20
21 class Singleton(object):#{#{
22     """A simple singleton class."""
23     def __new__(cls, *p, **k):#{#{
24         if not 'instance' in cls.__dict__:
25             cls._instance = object.__new__(cls)
26         return cls._instance
27     #}}}
28 #}}}
29
30 #-----
31 # vim: fdm=marker ts=4 sw=4 expandtab

```

### 6.1.9 ERROR.PY

```

1 """
2 ajaxWM – the free web based window manager for remote terminals
3 Copyright (C) 2007 Dennis Felsing, Andreas Waidler
4
5 This program is free software: you can redistribute it and/or modify
6 it under the terms of the GNU General Public License as published by
7 the Free Software Foundation, either version 3 of the License, or
8 (at your option) any later version.
9
10 This program is distributed in the hope that it will be useful,
11 but WITHOUT ANY WARRANTY; without even the implied warranty of
12 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 GNU General Public License for more details.
14
15 You should have received a copy of the GNU General Public License
16 along with this program. If not, see <http://www.gnu.org/licenses/>.
17
18 $Id: Error.py 76 2007-12-15 11:26:24Z r0q $
19 """
20
21 class Login_Error(Exception):#{#{
22     """Exception class for errors when logging in."""
23     def __init__(self, value=None):#{#{
24         self.value = value
25     #}}}
26 #}}}
27
28 class ID_Error(Exception):#{#{
29     SESSION = 'session'
30     WINDOW = 'window'
31     UNKNOWN = 'unkown'

```

```
32     """Exception class for errors about session IDs and window IDs."""
33     def __init__(self, value=None):
34         if value in (self.SESSION, self.WINDOW):
35             self.value = value
36         else:
37             self.value = self.UNKNOWN
38         self.value = '<error>%s</error>' % self.value
39     #}}}
40
41 #-----
42 # vim: fdm=marker ts=4 sw=4 expandtab
```

## 6.2 CLIENT

### 6.2.1 AJAXWM\_MANAGER.JS

```
1  /**
2  * ajaxWM – the free web based window manager for remote terminals
3  * Copyright (C) 2007 Dennis Felsing, Andreas Waidler
4  *
5  * This program is free software: you can redistribute it and/or modify
6  * it under the terms of the GNU General Public License as published by
7  * the Free Software Foundation, either version 3 of the License, or
8  * (at your option) any later version.
9  *
10 * This program is distributed in the hope that it will be useful,
11 * but WITHOUT ANY WARRANTY; without even the implied warranty of
12 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 * GNU General Public License for more details.
14 *
15 * You should have received a copy of the GNU General Public License
16 * along with this program. If not, see <http://www.gnu.org/licenses/>.
17 *
18 * $Id: AjaxWM_Manager.js 64 2007-11-30 19:10:59Z pwnd $
19 *
20 * * * * *
21 * AjaxWM_Manager
22 *
23 * The window manager class that holds the windows, configuration, handles
24 * events, communicates with the server and manages the virtual desktops.
25 * Performs requests to the server only when its by one of the other classes
26 * or a key is pressed. Mouse interaction and dynamic synchronization timeouts
27 * are handled by the windows or other elements themselves.
28 */
29
30 //-----
31
32 // Object.clone()//{{{
33
34 Object.prototype.clone = function ()
35 {
36     var clone = new Object();
37     for (var property in this) {
38         if ( 'object' == typeof(this[property]))
39             clone[property] = this[property].clone();
40         else
41             clone[property] = this[property];
42     }
43     return clone;
44 }
45
46 //}}}
47 // Array.clone()//{{{
48
49 Array.prototype.clone = function ()
50 {
51     var clone = new Array();
52     for (var property in this) {
53         if ( 'object' == typeof (this[property]))
54             clone[property] = this[property].clone();
```

```
55         else
56             clone[property] = this[property];
57     }
58     return clone;
59 }
60
61 //}}}
62 // function traceNode(path, node)//{{{
63
64 /**
65  * Follows a path and returns the node the path points to.
66  * Path has to be relative to node.
67  */
68
69 function traceNode(path, node)
70 {
71 // if (node.nodeType != Node.ELEMENT_NODE)
72 //     traceNode(path[0], node.nextSibling);
73
74     var trace = path.clone();
75     trace.shift(); // this position
76
77     if (trace.length == 0)
78         return node;
79
80     var next = trace[0];
81     return traceNode(trace, node.childNodes[next]);
82 }
83
84 //}}}
85 //function _replaceTrace(path, node)//{{{
86 //
87 //
88 // /**
89 //  * Replaces the passed position with a node.
90 //  */
91 //
92 // function _replaceTrace(path, node)
93 // {
94 // }
95 //
96 //}}}
97
98
99 ajaxWM = new function AjaxWM_Manager()
100 {
101 // {{{ properties
102
103     // version of the whole project
104     var _version = "0.1.1";
105     var _class = "AjaxWM_Manager";
106     // session id that will be generated by the server on login
107     var _sid;
108     var _sessionTimeout;
109     var _maxIdleTime = 600000; // ms = 10min // TODO value in minutes
110     // current scree, values can be "login", "desktop", undefined
111     var _screenType;
112     var _theme = 'lim';
113     var _templates = new Object();
```

```
114 // html nodes of virtual desktops. Amount can be changed
115 var _desktops = new Array(8);
116 // array holding instances of the windows (AjaxWM_Window)
117 var _windows = new Object();
118 // order of the windows. Key is position, value is id
119 var _windowOrder = new Array();
120 // ids of the currently focused/active window or desktop
121 var _activeDeskId;
122 var _focusedWinId;
123 // queue of event objects (associative arrays (Object))
124 var _queuedEvents = new Array();
125 // flag that indicates if currently performing a request
126 var _isSending = false;
127 // timeout object for calling the method _processQueue
128 var _processQueueTimeout;
129 // timeout object for displaying that a request failed
130 var _connectionTimeout;
131 // indicates if the javascript part should print debug messages
132 var _enableDebugging = true;
133 // div node where debug messages should be displayed
134 var _debugNode;
135 // maximum characters in the debug message area
136 var _debugMaxLength = 500;
137 // how long should the debug area be displayed (seconds)
138 // -1 disables auto-hiding
139 var _debugDisplayDuration = -1;
140 // before the debug message area is hidden again
141 var _debugHideTimeout;
142
143 // }}}
144 // {{{ this.init = function()
145
146 /**
147  * Initializes the window manager by setting up the GUI and
148  * creating the login screen.
149  *
150  * @see AjaxWM_Manager::_initGui();
151  * @see AjaxWM_Login_Screen
152  */
153
154 this.init = function()
155 {
156     _initGui();
157
158     // start displaying a login screen
159     ajaxWM_LoginScreen.create();
160
161     _screenType = 'login';
162 };
163
164 // }}}
165 // {{{ this.login = function(name, password)
166
167 /**
168  * Tries to log in with the given login name and password.
169  *
170  * Does only create a login event and send it to the server _directly_
171  * via _communicate(). The queue is skipped since this method should
172  * only be called on when the user is not logged in and can't create
```

```
173      * any other events, neither directly nor indirectly.
174      * _communicate() will handle the response (session id or failure).
175      */
176
177      this.login = function(name, password)
178      {
179          var event_li = { 'e': 'li', 'u': name, 'p': password };
180          _communicate(event_li);
181      }
182
183      //}}}
184      //{{{ this.newWindow = function(w, h, x, y)
185
186          /**
187           * Creates a new window.
188           *
189           * Creates the event and enqueues it. The window ID is generated
190           * by the server and will be the response to the request. Despite
191           * x and y not being sent to the server they have to be put in the
192           * event so that _handleResponse() can handle them. They have to
193           * be the last two keys in the event object so they can be cut off
194           * easily. x has to come before y!
195           */
196
197      this.newWindow = function(w, h, x, y)
198      {
199          if (null == w) var w = 80;
200          if (null == h) var h = 25;
201          if (null == x) var x = 0;
202          if (null == y) var y = 0;
203
204          var event_nw = { 'e': 'nw', 'w': w, 'h': h, 'x': x, 'y': y };
205          _enqueueEvent(event_nw);
206      }
207
208      //}}}
209      //{{{ this.closeWindow = function(win)
210
211          /**
212           * Sends a close-window-event to the server
213           *
214           * The response is handled by _communicate(). When the server reports
215           * success, the function destroy() of that window should be called
216           * which finally removes it from the interface.
217           *
218           * @see AjaxWM_Window::close()
219           * @see AjaxWM_Window::destroy()
220           */
221
222      this.closeWindow = function(win)
223      {
224          if (win.getStatus() != "closing") {
225              ajaxWM.debugMessage(_class, 'closeWindow',
226                  'Window not closed. ' +
227                  'Call _windows[' + win.getId() + '].close() before. ');
228              return;
229          }
230
231          var event_cw = { 'e': 'cw', 'W': win.getId() };
```

```
232         _enqueueEvent(event_cw);
233     }
234
235 //}}}
236 //{{{ this.switchToWindow = function(win)
237
238 /**
239  * Focuses the window passed as parameter.
240  *
241  * When the window is hidden its previous state will also be
242  * restored. The order of the windows will also be updated.
243  */
244
245 this.switchToWindow = function(win)
246 {
247     var id = win.getId();
248
249     // do nothing if window is already focused
250     if (_focusedWinId == id)
251         return;
252
253     ajaxWM.debugMessage(_class, "switchToWindow",
254         _focusedWinId + "->" + id);
255
256     // update order
257
258     // clear this window from list (order)
259     for (var i = 0; i < _windowOrder.length; ++i) {
260         if (_windowOrder[i] == id)
261             _windowOrder.splice(i, 1);
262     }
263     // and add it to its top
264     _windowOrder.push(id)
265     // set flag for focused window
266     _focusedWinId = id;
267
268     // gui part: move to foreground and show it
269     _desktops[_activeDeskId].appendChild(win.getNode());
270     win.show();
271 }
272
273 //}}}
274 //{{{ this.switchToDesktop = function(deskId)
275
276 /**
277  * Switch to the virtual desktop specified.
278  *
279  * Switches to the desktop with the passed desktop id by hiding
280  * the node of the desktop currently active and displaying
281  * the one from the desktop of which the id has been passed.
282  */
283
284 this.switchToDesktop = function(deskId)
285 {
286     // check if that desktop exists
287     if (_desktops[deskId] == undefined) {
288         ajaxWM.debugMessage(_class, 'switchToDesktop',
289             'Desktop ' + deskId + ' does not exist. ');
290         return false;
291     }
292 }
```



```
291     }
292
293     // desktop exists, continue switching to it
294
295     var debug = 'Switching ';
296
297     if (_activeDeskId != undefined) {
298         debug += 'from ' + _activeDeskId + ' ';
299         _desktops[_activeDeskId].style.display = "none";
300     }
301
302     debug += 'to ' + deskId;
303     ajaxWM.debugMessage(_class, 'switchToDesktop', debug);
304
305     _desktops[deskId].style.display = "block";
306     _activeDeskId = deskId;
307
308     return true;
309 }
310
311 //}}}
312 {{{ this.syncWindow = function(win, isResized)
313
314     /**
315      * Enqueues a synchronization event to be sent to the server to poll
316      * the current contents of the window which is passed as parameter.
317      *
318      * @param win object the window to be synchronized
319      * @param isResized bool optional: when true resend width and height
320      * @see AjaxWM_Manager::_enqueueEvent()
321      * @see AjaxWM_Manager::_processQueue()
322      * @see AjaxWM_Window::update()
323      */
324
325     this.syncWindow = function(win, isResized)
326     {
327         // create the event and add it to the queue
328         var event_sy = { 'e': 'sy', 'W': win.getId() };
329
330         if (isResized) {
331             event_sy['w'] = win.getWidth();
332             event_sy['h'] = win.getHeight();
333         }
334
335         _enqueueEvent(event_sy);
336     }
337
338 //}}}
339 {{{ this.debugMessage = function(message)
340
341     /**
342      * Prints a debugging message.
343      *
344      * Should always be called directly without checking if debug mode
345      * is enabled before. This method will check if a message should
346      * be printed itself!
347      */
348
349     this.debugMessage = function(className, functionName, message)
```

```
350     {
351         // when debug mode is disabled do nothing
352         if (!_enableDebugging)
353             return;
354
355         // alert when this method is called the wrong way
356         if (arguments.length != 3) {
357             alert("AjaxWM_Manager.debugMessage()" +
358                 "called with missing parameters!");
359         }
360
361         // create div layer of new message
362         var div_message = document.createElement('div');
363         var p_caller     = document.createElement('p');
364         var p_text       = document.createElement('p');
365         p_caller.innerHTML = className + "::<" + functionName + "()";
366         p_text.innerHTML   = message;
367         // set their classes
368         div_message.className = 'message';
369         p_caller.className    = 'caller';
370         p_text.className      = 'text';
371
372         var length = p_caller.innerHTML.length + p_text.innerHTML.length;
373
374         // when the message itself is longer than or equal to the allowed
375         // maximum simply clear the other messages and cut a bit off of
376         // this message in case that it is to long
377         switch (true) {
378             case (length > _debugMaxLength):
379                 if (p_caller.innerHTML.length >= _debugMaxLength) {
380                     p_text.innerHTML = "";
381                     p_caller.innerHTML.substr(0, _debugMaxLength);
382                 } else {
383                     var charsLeft = _debugMaxLength - p_caller.innerHTML.length;
384                     p_text.innerHTML = p_text.innerHTML.substr(0, charsLeft);
385                 }
386
387             case (length == _debugMaxLength):
388                 _debugNode.innerHTML = "";
389                 break;
390
391             default:
392                 // current length of all debug messages
393                 var cLength = _calcDebugLength();
394
395                 // if the message plus the current length exceeds the limit
396                 // remove the oldest message until it fits
397                 while((cLength + length) > _debugMaxLength) {
398                     _debugNode.removeChild(_debugNode.firstChild);
399                     cLength = _calcDebugLength();
400                 }
401         }
402
403         // current debug message area and possibly the message
404         // have been edited to fit into the limit.
405         // Add the message to the area
406         div_message.appendChild(p_caller);
407         div_message.appendChild(p_text);
408         _debugNode.appendChild(div_message);
```

```
409
410     // display debugging area
411     _displayDebugMessages();
412 }
413
414 //}}}
415 this.getVersion      = function() { return _version; }
416 this.getTheme        = function() { return _theme; }
417 this.getTemplate     = function() { return _templates[_theme]; }
418 this.getDesktopById  = function(id) { return _desktops[id]; }
419 this.getActiveDesktop = function() { return _desktops[_activeDeskId]; }
420 this.getActiveDeskId = function() { return _activeDeskId; }
421 //{{{function _initGui()
422
423 /**
424  * Initializes the graphical user interface.
425  *
426  * Takes care that everything concerning the look and usability is set up
427  * correctly.
428  */
429
430 function _initGui()
431 {
432     document.title += "-" + _version;
433
434     // if debug mode is enabled create the debug message node
435     if (_enableDebugging) {
436         _debugNode = document.createElement('div');
437         _debugNode.className = 'debug';
438         document.getElementsByTagName('body')[0].appendChild(_debugNode);
439     }
440
441     _templates[_theme] = new AjaxWM_Template(_theme);
442     _templates[_theme].loadCss();
443     _templates[_theme].parse();
444 }
445
446 //}}}
447 //{{{function _initDesktopUsage()
448
449 /**
450  * Initializes the desktops.
451  *
452  * Prepares the body for desktop usage, creates the virtual desktops,
453  * appends them to the body, switches to the first one of them and
454  * starts the keyboard hook.
455  */
456
457 function _initDesktopUsage()
458 {
459     // get body and initialize it for desktop usage
460     var body = document.getElementsByTagName('body')[0];
461     body.className = "desktop";
462
463     // create desktops
464     for (var i = 0; i < _desktops.length; ++i) {
465         _desktops[i] = document.createElement('div');
466         _desktops[i].className = 'desktop';
467         body.appendChild(_desktops[i]);
```

```
468     }
469
470     // switch to first desktop
471     ajaxWM.switchToDesktop(0);
472
473     // set up the keyboard hook
474     document.onkeypress = _onKeyPress;
475     document.onkeydown = _onKeyDown;
476
477     _screenType = 'desktop';
478 }
479
480 //}}}
481 //{{{function _onKeyDown(window_event)
482
483 /**
484  * Event handler for the onkeydown method.
485  */
486
487 function _onKeyDown(window_event)
488 {
489     if (!_SARISSA_IS_IE)
490         return;
491
492     if (!window_event)
493         var window_event = window.event;
494
495     o = { 9:1, 8:1, 27:1, 33:1, 34:1, 35:1, 36:1, 37:1, 38:1, 39:1,
496          40:1, 45:1, 46:1, 112:1, 113:1, 114:1, 115:1, 116:1, 117:1,
497          118:1, 119:1, 120:1, 121:1, 122:1, 123:1 };
498
499     if (o[window_event.keyCode]
500         || window_event.ctrlKey || window_event.altKey) {
501         window_event.which = 0;
502         return _onKeyPress(window_event);
503     }
504 }
505
506 //}}}
507 //{{{function _onKeyPress(window_event)
508
509 /**
510  * Event handler for the onkeypress method.
511  *
512  * Checks which keys were pressed, convert their key codes back
513  * into strings regarding special key combinations suchs as e.g.
514  * [CTRL][<KEY>]. Hotkeys are handled by
515  * AjaxWM_Manager::_handleHotkey(). Returns false so that the
516  * action which would normally be executed on a key press
517  * will be blocked.
518  */
519
520 function _onKeyPress(window_event)
521 {
522     var keyCode;
523     var keyString = "";
524
525     // determine the key that was pressed and handle hotkeys//{{{
526
```

```

527         if (!window_event)
528             var window_event = window.event;
529
530         if (window_event.keyCode)
531             keyCode = window_event.keyCode;
532
533         if (window_event.which)
534             keyCode = window_event.which;
535
536         // check if ALT was pressed
537         if (window_event.altKey) {///<{{{
538             // when the ctrl key is pressed additionally to the alt key
539             // then it should be a hotkey
540             if (window_event.ctrlKey && _handleHotkey(window_event)) {
541                 // when a hotkey is found and could be executed
542                 // cancel this event
543                 return false;
544             } else {
545                 if (keyCode >= 65 && keyCode <= 90)
546                     keyCode += 32;
547                 if (keyCode >= 97 && keyCode <= 122) {
548                     keyString = String.fromCharCode(27)
549                         + String.fromCharCode(keyCode);
550                 }
551             }///<}}}
552         // check if CTRL was pressed
553         } else if (window_event.ctrlKey) {///<{{{
554             if (keyCode >= 65 && keyCode <= 90)
555                 keyString = String.fromCharCode(keyCode-64); // Ctrl-A..Z
556             else if (keyCode >= 97 && keyCode <= 122)
557                 keyString = String.fromCharCode(keyCode-96); // Ctrl-A..Z
558             else if (keyCode == 54)
559                 keyString = String.fromCharCode(30); // Ctrl-^
560             else if (keyCode == 109)
561                 keyString = String.fromCharCode(31); // Ctrl-_
562             else if (keyCode == 219)
563                 keyString = String.fromCharCode(27); // Ctrl-[
564             else if (keyCode == 220)
565                 keyString = String.fromCharCode(28); // Ctrl-\
566             else if (keyCode == 221)
567                 keyString = String.fromCharCode(29); // Ctrl-]
568             else if (keyCode == 219) // TODO: duplicate, remove?
569                 keyString = String.fromCharCode(29); // Ctrl-]
570             else if (keyCode == 219) // TODO: duplicate, remove?
571                 keyString = String.fromCharCode(0); // Ctrl-@
572             }///<}}}
573         // when window.event.which is usable but its value is 0
574         // we check the window.event.keyCode here. This gives us a special key
575         // like Esc, Tab...
576         } else if (window_event.which == 0) {///<{{{
577             if (keyCode == 9)
578                 keyString = String.fromCharCode(9); // Tab
579             else if (keyCode == 8)
580                 keyString = String.fromCharCode(127); // Backspace
581             else if (keyCode == 27)
582                 keyString = String.fromCharCode(27); // Escape
583             // pressed key has no ASCII value. Enter the string directly,
584             // prefixed with escape
585             else {

```

---

```

586         if (keyCode === 33) keyString = "[5~"; // PgUp
587         else if (keyCode === 34) keyString = "[6~"; // PgDn
588         else if (keyCode === 35) keyString = "[4~"; // End
589         else if (keyCode === 36) keyString = "[1~"; // Home
590         else if (keyCode === 37) keyString = "[D"; // Left
591         else if (keyCode === 38) keyString = "[A"; // Up
592         else if (keyCode === 39) keyString = "[C"; // Right
593         else if (keyCode === 40) keyString = "[B"; // Down
594         else if (keyCode === 45) keyString = "[2~"; // Ins
595         else if (keyCode === 46) keyString = "[3~"; // Del
596         else if (keyCode === 112) keyString = "[[A"; // F1
597         else if (keyCode === 113) keyString = "[[B"; // F2
598         else if (keyCode === 114) keyString = "[[C"; // F3
599         else if (keyCode === 115) keyString = "[[D"; // F4
600         else if (keyCode === 116) keyString = "[[E"; // F5
601         else if (keyCode === 117) keyString = "[17~"; // F6
602         else if (keyCode === 118) keyString = "[18~"; // F7
603         else if (keyCode === 119) keyString = "[19~"; // F8
604         else if (keyCode === 120) keyString = "[20~"; // F9
605         else if (keyCode === 121) keyString = "[21~"; // F10
606         else if (keyCode === 122) keyString = "[23~"; // F11
607         else if (keyCode === 123) keyString = "[24~"; // F12
608         if (keyString.length) {
609             keyString = String.fromCharCode(27)+keyString;
610         }
611     }
612 } else {
613     if (keyCode === 8)
614         keyString = String.fromCharCode(127); // Backspace
615     else
616         keyString = String.fromCharCode(keyCode);
617 }}}}
618
619 //}}}
620 // everything that reaches this point is a key press event that
621 // should be passed to a window
622 try {
623     if (!keyString.length || isNaN(_focusedWinId)) {
624         var e = 'keyString="' + keyString + '"';
625         e += ', isNaN(_focusedWinId)= ' + isNaN(_focusedWinId);
626         throw e;
627     }
628     // encode keys to UTF-8
629     switch (keyString) {
630         case '+':
631             keyString = '%2B';
632             break;
633         default:
634             keyString = encodeURIComponent(keyString);
635     }
636     _enqueueEvent({ 'e': 'kp', 'W': _focusedWinId, 'k': keyString });
637 } catch (e) {
638     var error = "Error: " + e;
639     ajaxWM.debugMessage(_class, "_onKeyPress", error);
640 } finally {
641     window_event.cancelBubble = true;
642     if (window_event.stopPropagation) window_event.stopPropagation();
643     if (window_event.preventDefault) window_event.preventDefault();
644     return false;

```

```
645     }
646   }
647
648   ///}}}
649   ///{{{function _handleHotkey(window_event)
650
651   /**
652    * Checks if the onkeypress event contains an hotkey. If so, its
653    * corresponding action will be executed.
654    *
655    * Since this method is only called when _onKeyPress() finds alt
656    * and ctrl pressed it will not check for them a second time.
657    * Returns true if the pressed combination is a registered hotkey,
658    * false if not.
659    */
660
661   function _handleHotkey(window_event)
662   {
663     var key = String.fromCharCode(window_event.which);
664     switch (key) {
665       case 'w':
666         ajaxWM.newWindow(80,25,0,0);
667         return true;
668
669       case 'c':
670         if (_focusedWinId == undefined)
671           return true;
672
673         if (_windows[_focusedWinId] == undefined)
674           return true;
675
676         _windows[_focusedWinId].close();
677         return true;
678
679       case 'd':
680         var tmp = "";
681         for (var i = 0; i < _windowOrder.length; ++i) {
682           tmp += "i=" + i + " _winO[i]=" + _windowOrder[i] + "\n";
683         }
684         alert (tmp);
685         return true;
686
687       default:
688         if (!isNaN(key)) {
689           // debug for IE
690           ajaxWM.debugMessage(_class, '_handleHotkey',
691             'which=' + window_event.which + ' key=' + key);
692           // key is a number - change desktop
693           // subtract 1 because desktop ids start at 0, not at 1
694           ajaxWM.switchToDesktop(key - 1);
695           return true;
696         } else {
697           return false;
698         }
699     }
700   }
701
702   ///}}}
703   ///{{{function _enqueueEvent(ajaxWM_event)
```

```
704
705  /**
706   * Adds an event to the queue for sending it to the server.
707   *
708   * The event is added to the top of the queue. When no request is
709   * performed a timeout of one millisecond for the _processQueue()
710   * method is set which will send everything in the queue. If that
711   * method is already running it won't be called twice as it will
712   * run until the whole queue is empty.
713   */
714
715  function _enqueueEvent (ajaxWM_event)
716  {
717      _queuedEvents.push (ajaxWM_event);
718      if ( _isSending )
719          return;
720      window.clearTimeout ( _processQueueTimeout ); // bugfix
721      _processQueueTimeout = window.setTimeout ( _processQueue , 1 );
722  }
723
724  }}}}
725  function _isEventValid (ajaxWM_event)
726
727  /**
728   * Checks whether an event is valid or not.
729   *
730   * An event is invalid, when
731   * * it concerns a window that is not existing
732   *
733   * @return bool true if event is valid, false if not
734   */
735
736  function _isEventValid (ajaxWM_event)
737  {
738      // TODO code cleaning
739      var result = true;
740      try {
741          if ( undefined != ajaxWM_event[ 'W' ] ) {
742              if ( undefined == _windows[ ajaxWM_event[ 'W' ] ] ) {
743                  var e = 'Window ' + ajaxWM_event[ 'W' ]
744                      + ' does not exist. '
745                      + 'Event: ' + ajaxWM_event[ 'e' ];
746                  throw e;
747              }
748          }
749      } catch ( e ) {
750          ajaxWM.debugMessage ( _class , '_isEventValid' , e );
751          result = false;
752      } finally {
753          return result;
754      }
755  }
756
757  }}}}
758  function _processQueue ()
759
760  /**
761   * Processes the queue until it is empty.
762   *
```



```
763      * Calls the window managers _communicate() method which handles the
764      * response according to the request that has been sent.
765      *
766      * FIXME: add flag that indicates if all requests have completed
767      */
768
769  function _processQueue()
770  {
771      // check if this method is already running.
772      // Since this method will always run until the queue is empty there is
773      // no need to call it twice.
774      if (_isSending !== 0)
775          return;
776      _isSending = 1;
777
778      if (!_screenType)
779          return;
780
781      window.clearTimeout(_sessionTimeout);
782      _sessionTimeout = window.setTimeout(_onSessionTimeout, _maxIdleTime);
783
784      // start the loop which will run until all events are processed
785      while (_queuedEvents.length > 0) {
786          // get the next event in list. It will be on bottom of the stack
787          // (index = 0, new events will be inserted on top)
788          var e = _queuedEvents.shift();
789
790          // when the event is not valid, continue with next event
791          if (!_isEventValid(e))
792              continue;
793
794          // send the request for this event and handle its response
795          _communicate(e);
796      }
797
798      _isSending = 0;
799  }
800
801  //}}}
802  //{{{function _communicate(ajaxWM_event)
803
804  /**
805   * Performs a request for the passed event and handles the servers
806   * response.
807   *
808   * Contains the method onreadystatechange of the request that was end
809   * to the server. That function will check if everything went fine
810   * and then call AjaxWM_Manager::_handleResponse().
811   */
812
813  function _communicate(ajaxWM_event)
814  {
815      var query = (_sid !== undefined ? "s=" + _sid : '');
816      for (key in ajaxWM_event) {
817          // cut off x and y since they are only needed for internal use
818          if (ajaxWM_event['e'] === 'nw' && key === 'x')
819              break;
820          query += "&" + key + "=" + ajaxWM_event[key];
821      }
```

```
822
823     // create request object
824     var request = new XMLHttpRequest();
825
826     // before sending the request we have to set the response handling up
827     {{{ request.onreadystatechange = function()
828
829         /**
830          * Handles the response of the XMLHttpRequest.
831          *
832          * Validates request state and calls
833          * AjaxWM::_handleResponse() on the request passing that
834          * checks.
835          */
836
837         request.onreadystatechange = function ()
838         {
839             // state 4 means that the request is completed
840             if (request.readyState != 4)
841                 return;
842
843             window.clearTimeout(_connectionTimeout);
844
845             // check the HTML status code if it is 200 (OK)
846             if (request.status != 200) {
847                 ajaxWM.debugMessage(_class, "request.onreadystatechange",
848                     "Request got bad HTML status code: " + request.status);
849                 return;
850             }
851
852             // every request that reaches this point is valid
853
854             _handleResponse(request, ajaxWM_event);
855         }
856
857     }}}
858 //ajaxWM.debugMessage(_class, "_communicate", query);
859 request.open("POST", "event", true);
860 request.setRequestHeader('Content-Type',
861     'application/x-www-form-urlencoded');
862 try {
863     window.clearTimeout(_connectionTimeout); // bugfix
864     _connectionTimeout = window.setTimeout(_timeout, 20000);
865     request.send(query);
866 } catch (exception) {
867     window.clearTimeout(_connectionTimeout);
868     var error = "Could not send request.";
869     error += " Query: " + query;
870     error += " Exception: " + exception.message;
871     ajaxWM.debugMessage(_class, "_communicate", error);
872 }
873 }
874
875 }}}
876 {{{ function _handleResponse(r, e)
877
878     /**
879     * Handles the passed response for the given ajaxWM event.
880     *
```

```
881 * @param XMLHttpRequest r Completed request (only response is needed)
882 * @param Object e AjaxWM-event for wich the request has been launched
883 */
884
885 function _handleResponse(r, e)
886 {
887     switch (e['e']) {
888         case 'li':
889             // response is session id or negative error code
890             var response = r.responseText;
891             if (response < 0) { // error codes are negative
892                 ajaxWM.debugMessage(_class, '_communicate',
893                                     'Login failed. Response is ' + response);
894                 ajaxWM_LoginScreen.setResult(false);
895                 break;
896             }
897             _sid = response;
898             ajaxWM.debugMessage(_class, '_communicate',
899                                 'Login successful, session id is ' + _sid);
900             ajaxWM_LoginScreen.setResult(true);
901             _initDesktopUsage();
902             _sessionTimeout = window.setTimeout(_onSessionTimeout, _maxIdleTime);
903             break;
904
905         case 'nw':
906             // response is id of new window or -1 on failure
907             var id = r.responseText;
908             if (id == -1) {
909                 ajaxWM.debugMessage(_class, '_communicate',
910                                     'Could not create new window. ' +
911                                     'Response was "' + id + '"');
912                 break;
913             }
914             _windows[id] = new AjaxWM_Window(
915                 _activeDeskId, id, e['w'], e['h'], e['x'], e['y']);
916             ajaxWM.switchToWindow(_windows[id]);
917             break;
918
919         case 'cw':
920             // response can be handled the same way as in e=kw (by now)
921
922         case 'kw':
923             // response is result (0 = false, 1 = true)
924
925             var id = e['W'];
926             var result = (r.responseText == 1);
927
928             if (!result) {
929                 ajaxWM.debugMessage(_class, '_communicate',
930                                     'Server returned ' + result + ' for event ' +
931                                     e['e'] + ' of id ' + e['W']);
932             } else if (_windows[id].destroy()) {
933                 ajaxWM.debugMessage(_class, '_communicate',
934                                     'Removing window ' + id);
935
936                 delete(_windows[id]);
937                 // remove window id from order
938                 for (var i = 0; i < _windowOrder.length; ++i) {
939                     if (_windowOrder[i] == id)
```

```
940         _windowOrder.splice(i, 1);
941     }
942     var previousWindow = _getRecentWindow();
943     // when there is another window
944     if (previousWindow !== -1) {
945         // switch to it
946         ajaxWM.switchToWindow(previousWindow);
947         // if not
948     } else {
949         // set the flag to indicate that no window is focused anymore
950         _focusedWinId = undefined;
951     }
952 } else {
953     ajaxWM.debugMessage(_class, '_communicate',
954         'destroy() failed.');
```

```
955 }
956 break;
957
958 case 'kp':
959     // response can be handled the same way as in e=sy
960
961 case 'sy':
962     // response is either new content or <idem />;
963     // AjaxWM_Window::update() handles this
964     var id = e['W'];
965     var response = r.responseXML.documentElement;
966     _windows[id].update(response);
967     if (e['w'] && e['h'])
968         _windows[id].calcCharSize();
969     break;
970 }
971 }
972
973 //}}}
974 //{{{function _timeout()
975
976 /**
977  * Called by _communicate() when the client did not receive a response
978  * after a timeout of some seconds.
979 */
980
981 function _timeout()
982 {
983     ajaxWM.debugMessage(_class, "_communicate", "Timeout.");
984 }
985
986 //}}}
987 //{{{ function _calcDebugLength([div_message])
988
989 /**
990  * Calculates the length of debugging messages and returns the
991  * amount of characters in total.
992  * 
993  * If the optional parameter div_message is passed, this method
994  * will only calculate the total chars of that message. If not
995  * the whole length of all currently existing debugging messages
996  * is calculated.
997 */
998
```

```
999     function _calcDebugLength(div_message)
1000     {
1001         // the total length
1002         var length = 0;
1003         // array containing all message div-layers
1004         var messages = new Array();
1005         // temporary array that holds the two paragraphs of a message div-layer
1006         var p = new Array();
1007
1008         if (arguments.length != 0) {
1009             messages[0] = div_message;
1010         } else {
1011             messages = _debugNode.getElementsByTagName('div');
1012         }
1013
1014         for (i = 0; i < messages.length; ++i) {
1015             if (messages[i].className != 'message')
1016                 continue;
1017
1018             p = messages[i].getElementsByTagName('p');
1019             for (j = 0; j < p.length; ++j) {
1020                 if (p[j].className != 'caller' && p[j].className != 'text')
1021                     continue;
1022
1023                 length += p[j].innerHTML.length;
1024             }
1025         }
1026         return length;
1027     }
1028
1029 //}}}
1030 //{{{function _displayDebugMessages()
1031
1032 /**
1033  * Displays the debug message area.
1034  *
1035  * Sets a timeout after which the area will be hidden again
1036  * and clear the existing timeout before. The message area won't
1037  * be hidden if -1 was specified as duration.
1038  */
1039
1040 function _displayDebugMessages()
1041 {
1042     if (_debugNode.style.display != "block")
1043         _debugNode.style.display = "block";
1044
1045     // clear timeout, if existing
1046     if (_debugHideTimeout)
1047         window.clearTimeout(_debugHideTimeout);
1048
1049     // duration of -1 means forever
1050     if (_debugDisplayDuration == -1)
1051         return;
1052
1053     // set timeout
1054     var time = _debugDisplayDuration * 1000;
1055     _debugHideTimeout = window.setTimeout(_hideDebugMessages, time);
1056 }
1057
```

```
1058 //}}}
1059 //{{{ function _hideDebugMessages()
1060
1061 /**
1062  * Hides the debug message area.
1063  *
1064  * Since this method is normally called by a timeout set by
1065  * _displayDebugMessages() the timeout will be removed.
1066  */
1067
1068 function _hideDebugMessages()
1069 {
1070     _debugNode.style.display = "none";
1071     window.clearTimeout(_debugHideTimeout);
1072 }
1073
1074 //}}}
1075 //{{{ function _getRecentWinId()
1076
1077 /**
1078  * Returns ID of the most recent window
1079  * or -1 when there has no activity been yet.
1080  *
1081  * @return int ID of most recent window or -1
1082  */
1083
1084 function _getRecentWinId()
1085 {
1086     var result = -1;
1087     while (_windowOrder.length > 0) {
1088         var top = _windowOrder.pop();
1089         if (top !== undefined) {
1090             _windowOrder.push(top);
1091             result = top;
1092             break;
1093         }
1094     }
1095     return result;
1096 }
1097
1098 //}}}
1099 //{{{ function _getRecentWindow()
1100
1101 /**
1102  * Returns the most recent window or -1 when there has no
1103  * activity been yet.
1104  *
1105  * @return AjaxWM_Window The most recent window, -1 on failure.
1106  */
1107
1108 function _getRecentWindow()
1109 {
1110     var result = _getRecentWinId();
1111     if (result !== -1)
1112         result = _windows[result];
1113     return result;
1114 }
1115
1116 //}}}
```

```
1117 //function _onSessionTimeout()//{{{
1118
1119 /**
1120  * Called when the user was idle for a specified timeout.
1121  * Kills the session and removes the GUI.
1122  */
1123
1124 function _onSessionTimeout ()
1125 {
1126     window.setTimeout(_kill, 1);
1127     alert("ajaxWM session timed out.");
1128 }
1129
1130 //}}}
1131 //function _removeWindows()//{{{
1132
1133 /**
1134  * Destroys all windows without sending any events to the server.
1135  * Should only be called when the session has already been closed.
1136  */
1137
1138 function _removeWindows()
1139 {
1140     //FIXME handle case when no windows is existing
1141     for (key in _windows)
1142         _windows[key].destroy();
1143     _windows = new Object();
1144     _windowOrder = new Array();
1145 }
1146
1147 //}}}
1148 //function _removeDesktops()//{{{
1149
1150 /**
1151  * Removes all virtual desktops.
1152  *
1153  * Since the windows are on the desktops they will be removed from
1154  * display, too. But they won't be deleted, they may be added to
1155  * a desktop again.
1156  */
1157
1158 function _removeDesktops()
1159 {
1160     for (var i = 0; i < _desktops; ++i) {
1161         document.getElementsByTagName( 'body' )[0].removeChild(
1162             _desktops[i]);
1163         _desktops[i] = undefined;
1164     }
1165 }
1166
1167 //}}}
1168 //function _kill()//{{{
1169
1170 /**
1171  * Kills the graphical user interface.
1172  *
1173  * Removes any nodes created and resets event handlers.
1174  * No server request will be launched by any action taken
1175  * in this method.
```

```
1176     */
1177
1178     function _kill()
1179     {
1180         _screenType = undefined;
1181         _sid = undefined;
1182         _removeWindows();
1183         _removeDesktops();
1184         if (_enableDebugging)
1185             document.getElementsByTagName('body')[0].removeChild(_debugNode);
1186         document.getElementsByTagName('body')[0].className = '';
1187         document.onkeypress = null;
1188         document.onkeydown = null;
1189     }
1190
1191 //}}}
1192 };
1193
1194 //-----
1195 // Ensure that the code does not exceed 79 characters per line using an indent
1196 // of 4 spaces/level. Use NO TABS at all. Leave the mode line as it is. For
1197 // further information see file docs/CodingConvention.
1198 // vim: et ts=4 sw=4 fdm=marker
```

### 6.2.2 AJAXWM\_LOGIN\_SCREEN.JS

```
1 /**
2  * ajaxWM - the free web based window manager for remote terminals
3  * Copyright (C) 2007 Dennis Felsing, Andreas Waidler
4  *
5  * This program is free software: you can redistribute it and/or modify
6  * it under the terms of the GNU General Public License as published by
7  * the Free Software Foundation, either version 3 of the License, or
8  * (at your option) any later version.
9  *
10 * This program is distributed in the hope that it will be useful,
11 * but WITHOUT ANY WARRANTY; without even the implied warranty of
12 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 * GNU General Public License for more details.
14 *
15 * You should have received a copy of the GNU General Public License
16 * along with this program. If not, see <http://www.gnu.org/licenses/>.
17 *
18 * $Id: AjaxWM_Login_Screen.js 55 2007-11-30 13:40:43Z pund $
19 *
20 * * * * *
21 * AjaxWM_Login_Screen
22 *
23 * The login screen object displays the screen where the user has to enter name
24 * and password to proceed. It will implement some additional options like
25 * specifying the host on which the user wants to be logged in, debug verbosity
26 * and theme selection later.
27 *
28 * Locks the screen, calls the window manager which will send a request to the
29 * server and handle the response. After that, the method setResult() of this
30 * object is called to display if the login was successful or not and removing
31 * the login screen or its lock.
32 */
33 //-----
```



```
34
35 ajaxWM_LoginScreen = new function AjaxWM_Login_Screen ()
36 {
37   //{{{properties
38
39   // self-referencing properties
40   var _this = this;
41   var _class = "AjaxWM_Login_Screen";
42
43   // flag that indicates if the screen is locked or not
44   var _isLocked = false;
45   // {{{ HTML nodes
46
47   var _background;
48   var _div;
49   var _form;
50   var _input_name;
51   var _input_pass;
52   var _input_submit;
53
54   //}}}}
55 //}}}}
56 //{{{this.create = function ()
57
58   /**
59    * Creates the login screen.
60    *
61    * Creates the html nodes and appends them to the body.
62    * Sets the login screen class for the body.
63    */
64
65   this.create = function ()
66   {
67     //{{{create HTML nodes
68
69     _background = document.getElementsByTagName( 'body' )[0];
70     _background.className = 'login';
71
72     _div = document.createElement( 'div' );
73     _div.className = 'login';
74
75     _form = document.createElement( 'form' );
76     _form.className = 'login';
77     _form.onsubmit = _submit;
78
79     _input_name = document.createElement( 'input' )
80     _input_name.name = 'name';
81     _input_name.className = 'name';
82
83     _input_pass = document.createElement( 'input' )
84     _input_pass.name = 'pass';
85     _input_pass.className = 'pass';
86     _input_pass.type = 'password';
87
88     _input_submit = document.createElement( 'input' );
89     _input_submit.name = 'submit';
90     _input_submit.className = 'submit';
91     _input_submit.type = 'image';
92     _input_submit.src = 'themes/' + ajaxWM.getTheme() + '/img/login.png';
```

```
93 //      _input_submit.value = 'login';
94 //      _input_submit.onclick = _submit;
95 //      _input_submit.onsubmit = _submit;
96
97 //}}}
98 //{{{append HTML nodes
99
100     _form.appendChild(_input_name);
101     _form.appendChild(_input_pass);
102     _form.appendChild(_input_submit);
103     _div.appendChild(_form);
104     _background.appendChild(_div);
105
106 //}}}
107 }
108
109 //}}}
110 //{{{this.setResult = function(result)
111
112 /**
113  * Sets the login result.
114  *
115  * Displays if the login was successful. If so, the login screen
116  * will be closed by _close(). If not the screen will be unlocked.
117  * Will be called by AjaxWM_Manager on handling the response of the
118  * login event.
119  */
120
121 this.setResult = function(result)
122 {
123     ajaxWM.debugMessage(_class, "setResult", "result=" + result);
124
125     switch (result) {
126     case true:
127         _close();
128         break;
129
130     case false:
131         _clear('password');
132         _setLockState(false);
133         break;
134
135     default:
136         ajaxWM.debugMessage(_class, "setResult",
137                             "result has a bad value");
138     }
139 }
140
141 //}}}
142 //{{{function _submit(window_event)
143
144 /**
145  * Submits the login form if it not empty.
146  *
147  * Calls the window managers login method which will perform
148  * the request. After the request is finished the function
149  * setResult() will be called which handles screen unlocking
150  * and closing.
151  *
```

```
152      * @return false Prevents form from being submitted normally
153      */
154
155  function _submit()
156  {
157      try {
158          if (_isLocked)
159              throw 'locked';
160
161          var name = _input_name.value;
162          var pass = _input_pass.value;
163
164          if (" " == name || " " == pass)
165              throw 'empty';
166
167          _setLockState(true);
168          ajaxWM.login(name, pass);
169      } catch (e) {
170          var error;
171          switch (e) {
172              case 'locked':
173                  error = 'Screen is locked, not submitting.';
174                  break;
175              case 'empty':
176                  error = 'At least one field is empty, not submitting.';
177                  break;
178              default:
179                  throw e;
180          }
181          ajaxWM.debugMessage(_class, '_submit', error);
182      } finally {
183          return false;
184      }
185  }
186
187  //}}}
188  //{{{ function _setLockState(state)
189
190      /**
191       * Locks or unlocks the screen if it is not in the state that was
192       * passed as parameter.
193       *
194       * Sets a flag that indicates if the screen is locked or not. If
195       * that flag is already set to the one passed as parameter this
196       * method immediately returns.
197       */
198
199  function _setLockState(state)
200  {
201      if (state == _isLocked) {
202          ajaxWM.debugMessage(_class, "_setLockState", "State " +
203                              state + " already enabled");
204          return;
205      }
206
207      switch (state) {
208          case false:
209          case true:
210              _input_name.disabled = state;
```

```
211         _input_pass.disabled = state;
212         _input_submit.disabled = state;
213
214         if (state == true) {
215             _background.className = "login_locked";
216             _input_submit.className = "submit_locked";
217         } else {
218             _background.className = "login";
219             _input_submit.className = "submit";
220         }
221
222         _isLocked = state;
223
224         break;
225
226     default :
227         ajaxWM.debugMessage(_class, "_setLockState",
228                             "Wrong state: " + state);
229     }
230 }
231
232 //}}}
233 //{{{function _clear()
234
235 /**
236  * Clears input of the login form.
237  *
238  * If no parameter is passed or if the parameter is "both" both name
239  * and password are cleared. If it is "name" or "password" the corresponding
240  * input will be cleared.
241  *
242  * @param string input [optional] Input to be cleared.
243  *                                     Valid values are "name", "password" and "both".
244  * @return void
245  */
246
247 function _clear(input)
248 {
249     if (null == input)
250         input = 'both';
251
252     switch (input) {
253     case 'both':
254         _input_pass.value = "";
255         // no break here to clear name in "case 'name'"
256     case 'name':
257         _input_name.value = "";
258         break;
259     case 'password':
260         _input_pass.value = "";
261         break;
262     default :
263         throw 'Invalid argument: ' + input;
264     }
265 }
266
267 //}}}
268 //{{{function _close()
269
```

```
270  /**
271   * Closes the login screen.
272   *
273   * Removes the nodes from the display and deletes their contents
274   * to free resources.
275   */
276
277  function _close()
278  {
279      _background.removeChild(_div);
280      delete(_input_name);
281      delete(_input_pass);
282      delete(_input_submit);
283      delete(_form);
284  }
285
286  ///}}}
287  }
288
289  //-----
290  // Ensure that the code does not exceed 79 characters per line using an indent
291  // of 4 spaces/level. Use NO TABS at all. Leave the mode line as it is. For
292  // further information see file docs/CodingConvention.
293  // vim: et ts=4 sw=4 fdm=marker
```

### 6.2.3 AJAXWM\_WINDOW.JS

```
1  /**
2   * ajaxWM – the free web based window manager for remote terminals
3   * Copyright (C) 2007 Dennis Felsing, Andreas Waidler
4   *
5   * This program is free software: you can redistribute it and/or modify
6   * it under the terms of the GNU General Public License as published by
7   * the Free Software Foundation, either version 3 of the License, or
8   * (at your option) any later version.
9   *
10  * This program is distributed in the hope that it will be useful,
11  * but WITHOUT ANY WARRANTY; without even the implied warranty of
12  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13  * GNU General Public License for more details.
14  *
15  * You should have received a copy of the GNU General Public License
16  * along with this program. If not, see <http://www.gnu.org/licenses/>.
17  *
18  * $Id: AjaxWM_Window.js 51 2007-11-29 14:33:13Z pwnd $
19  *
20  * * * * *
21  * AjaxWM_Window
22  *
23  * This class is responsible for the correct display of a window, it creates
24  * its events, handles user interaction and executes corresponding actions.
25  * Each window sets up a dynamic timeout after which the synchronization
26  * method of the window manager should be called to poll the current contents
27  * of this window from the server. After that, the manager feeds the update
28  * method of that window.
29  *
30  * Note that incoming keys are not captured by the window itself but by the
31  * window manager to handle hotkeys first. When no AjaxWM hotkey is detected
32  * the keys are sent directly from the manager to the server. The manager will
```

```
33  * then feed the corresponding window with the response of the server.
34  */
35  //-----
36
37  AjaxWM_Window = function(deskId, id, width, height, pos_x, pos_y)
38  {
39  //{{{ properties
40
41      // reference of this object
42      var _this = this;
43      var _class = 'AjaxWM_Window';
44
45      var _deskId = deskId;
46      var _id = id;
47      var _width = width;
48      var _height = height;
49      var _charWidth;
50      var _lineHeight;
51
52      // the status of this window
53      // values: initializing, default, hidden, maximized, closing
54      var _status = 'initializing';
55
56      // timeout value (in milliseconds) after which the window
57      // will check if something happened.
58      // default value (minimum)
59      var _defaultTimeout = 30;
60      // maximum value
61      var _maxTimeout = 2400;
62      // dynamically increasing value, generated by AjaxWM_Window.update()
63      var _dynTimeout = _defaultTimeout;
64      // timeout object
65      var _timeout;
66
67      // position of the mouse cursor (set by some event handlers)
68      var _mouseX; // x axis
69      var _mouseY; // y axis
70
71      // contains direction to which the window is resized
72      var _direction;
73
74      // HTML nodes
75
76      var _node = new Object();
77
78  //}}}}
79  //{{{ this.focus = function()
80
81      /**
82       * Focuses this window.
83       * Callback function.
84       */
85
86      this.focus = function()
87      {
88          ajaxWM.switchToWindow(_this);
89      }
90
91  //}}}}
```

```
92 //{{{ this.show = function()
93
94     /**
95      * Displays this window.
96      */
97
98     this.show = function()
99     {
100         _node['main'].style.display = "block";
101         _status = "default";
102     }
103
104 //}}}
105 //{{{ this.hide = function()
106
107     /**
108      * Hides the window.
109      */
110
111     this.hide = function()
112     {
113         _node['main'].style.display = "none";
114         _status = "hidden";
115     }
116
117 //}}}
118 //{{{ this.update = function(contents)
119
120     /**
121      * Updates the window with the contents passed.
122      *
123      * Checks what kind of response has been passed. When the response
124      * is <idem></idem> nothing changed and the content is left untouched
125      * and the timeout until the next synchronization will be increased.
126      * If not, the content will be updated and the timeout set back to
127      * its default value.
128      */
129
130     this.update = function(contents)
131     {
132         //      ajaxWM.debugMessage(_class, "update",
133         //      "id=" + _id + ", tagName=" + contents.tagName);
134
135         window.clearTimeout(_timeout);
136
137         if (contents.tagName != "pre") {
138             _dynTimeout *= 2;
139             if (_dynTimeout > _maxTimeout)
140                 _dynTimeout = _maxTimeout;
141         } else {
142             Sarissa.updateContentFromNode(contents, _node['content']);
143             _dynTimeout = _defaultTimeout;
144         }
145
146         if (_status != 'closing')
147             _timeout = window.setTimeout(_sync, _dynTimeout);
148     }
149
150 //}}}
```

```
151 //{{{ this.moveToDeskId = function(id)
152
153 /**
154  * Moves this window to the desktop with the passed ID.
155  */
156
157 this.moveToDeskId = function(id)
158 {
159 //      ajaxWM.getDesktopById(_deskId).removeChild(_node['main']);
160      ajaxWM.getDesktopById(id).appendChild(_node['main']);
161      this._deskId = id;
162 }
163
164 //}}}
165 //this.moveTo = function(x, y)//{{{
166
167 /**
168  * Moves window to specified coordinates.
169  *
170  * Coordinates have to be integers.
171  *
172  * @param x int Position on x-axis
173  * @param y int Position on y-axis
174  * @return bool True on moving being (partially) successful
175  */
176
177 this.moveTo = function(x, y)
178 {
179     x = parseInt(x);
180     if (isNaN(x))
181         return false;
182     y = parseInt(y);
183     if (isNaN(y))
184         return false;
185
186     // TODO: move all classes in _node['setpos']
187     _node['main'].style.left = x + "px";
188     _node['main'].style.top = y + "px";
189
190     return true;
191 }
192
193 //}}}
194 //this.resizeTo = function(x, y)//{{{
195
196 /**
197  * Resizes a window to the amount of characters/lines passed.
198  *
199  * @param x int New width [characters]
200  * @param y int New height [lines]
201  * @return bool True on resizing being (partially) successful
202  */
203
204 this.resizeTo = function(x, y)
205 {
206     x = parseInt(x);
207     if (isNaN(x))
208         return false;
209     y = parseInt(y);
```



```
210         if (isNaN(y))
211             return false;
212
213         _width += x;
214         _height += y;
215
216
217         // FIXME resize all elements in _node['setsize']
218
219         _node['main'].style.width = _getProperty(_node['main'], "width")
220                                     + x + "px";
221         _node['main'].style.height = _getProperty(_node['main'], "height")
222                                     + y + "px";
223
224         return true;
225     }
226
227 //}}}
228 //{{{ this.close = function()
229
230 /**
231  * Starts the closing procedure of this window.
232  *
233  * Sets a flag that this window is currently closing and calls
234  * the window manager to close it on the server. Window
235  * is destroyed by the manager if the server reports success.
236  */
237
238 this.close = function()
239 {
240     _status = 'closing';
241     ajaxWM.closeWindow(_this);
242 }
243
244 //}}}
245 //{{{ this.destroy = function()
246
247 /**
248  * Destroy this window: Remove nodes, free resources.
249  */
250
251 this.destroy = function()
252 {
253     if (_status != 'closing') {
254         ajaxWM.debugMessage(_class, 'destroy',
255                             'Warning. Status was ' + _status);
256         _status = 'closing';
257     }
258
259     ajaxWM.getDesktopById(_deskId).removeChild(_node['main']);
260     _node['main'].innerHTML = '';
261     delete(_node['main']);
262
263     return true;
264 }
265
266 //}}}
267 this.getDeskId = function(){return _deskId}
268 this.getId      = function(){return _id}
```

```
269     this.getWidth = function(){return _getProperty(_node[ 'main' ], 'width')}
270     this.getHeight = function(){return _getProperty(_node[ 'main' ], 'height')}
271     this.getStatus = function(){return _status}
272     this.getNode = function(){return _node[ 'main' ]}
273 //{{{ function _init()
274
275 /**
276  * Initializes properties of this window and creates its nodes.
277  *
278  * This method will create every HTML node and their events without
279  * noticing the user. The objects method _draw() should be called
280  * afterwards which will finally display it.
281  * It also calls the method _sync() which synchronizes its contents
282  * with the server and sets up a timeout for further synchronizations.
283  */
284
285 function _init()
286 {
287     ajaxWM.debugMessage(_class, "_init",
288         "id=" + _id + ', w=' + _width + ', h=' + _height);
289     var template = ajaxWM.getTemplate();
290     _node = template.cloneNodes('window');
291     ajaxWM.getDesktopById(_deskId).appendChild(_node[ 'main' ]);
292     _registerEvents();
293 }
294
295 //}}}
296 //{{{ function _draw()
297
298 /**
299  * Starts displaying the window and its contents.
300  *
301  * Should be called only once after init().
302  * At this moment, this is only a "wrapper" for display() but it could get
303  * some fancy features in the future.
304  */
305
306 function _draw()
307 {
308     _node[ 'main' ].style.display = "block";
309     _status = "default";
310 }
311
312 //}}}
313 //{{{ function _sync()
314
315 /**
316  * Synchronizes the contents of this window with the server.
317  *
318  * Calls the AjaxWM_Manager to send a synchronization event for
319  * this window to the server. AjaxWM_Manager then calls
320  * AjaxWM_Window.update().
321  * Wont be executed when window is already closing to prevent a
322  * synchronization event to be enqueued after the close- or
323  * kill-window-event.
324  */
325
326 function _sync()
327 {
```

```
328         if ( _status == 'closing' )
329             return;
330
331         _node[ 'title' ].innerHTML = 'ajaxWM-' + ajaxWM.getVersion();
332         ajaxWM.syncWindow( _this );
333     }
334
335     //}}}
336     //function _calcCharSize()//{{{
337
338     /**
339      * Calculates width and height of a character
340      * (in pre.content) in pixels.
341      * FIXME: add calculation of height
342      */
343
344     function _calcCharSize()
345     {
346         var width = _getProperty( _node[ 'content' ], 'width' );
347         _charWidth = width / _width;
348         ajaxWM.debugMessage( _class, 'calcCharSize',
349             _charWidth + ' = ' + width + ' / ' + _width );
350     }
351
352     //}}}
353     //{{{ function _getProperty(node, property, withUnit)
354
355     /**
356      * Gives a property (size) of a document node.
357      *
358      * @param node      the node to get information about
359      * @param property the information to get
360      * @param withUnit optional. Return the value with its unit
361      * @return int/float size of the property passed without unit
362      * @return string   size of the property passed without unit
363      */
364
365     function _getProperty( node, property, withUnit )
366     {
367         var prpty = node.style[ property ];
368         if ( true == withUnit )
369             var result = prpty;
370         else
371             var result = prpty.substr( 0, prpty.length - 2 );
372
373         if ( result == "" || isNaN( result ) ) {
374             // fallback determination of property,
375             // all values are returned WITHOUT unit, unit would be px
376             switch ( property ) {
377                 case "width":
378                     result = node.clientWidth;
379                     break;
380
381                 case "height":
382                     result = node.clientHeight;
383                     break;
384
385                 case "left":
386                     result = node.offsetLeft;
```

```
387         break;
388
389     case "top":
390         result = node.offsetTop;
391         break;
392
393     default :
394         ajaxWM.debugMessage(_class , "_getProperty",
395                             "wrong parameter: " + property);
396     }
397
398     if (true == withUnit)
399         result += 'px';
400 }
401 if (false == withUnit)
402     result = parseFloat(result);
403
404 return result;
405 }
406
407 //}}}
408 //function _registerEvents()//{{{
409
410 /**
411  * Set up all events of this window.
412  *
413  * TODO: add recursion?
414  */
415
416 function _registerEvents()
417 {
418     var events = _node['events'];
419     var length = events.length;
420     for (var i = 0; i < length; ++i) { // event type
421         for (e in events[i]) { // event
422             if (typeof(events[i][e]) != 'object')
423                 continue;
424             var action = events[i][e]['action'];
425             switch (action) {
426                 case 'focus':
427                     var fn = _this.focus;
428                     break;
429                 case 'move':
430                     var fn = _startMove;
431                     break;
432                 case 'resize':
433                     var fn = _startResize;
434             }
435             var path = events[i][e]['path'];
436             var node = traceNode(path, _node['main']);
437             node[e] = fn;
438         }
439     }
440     delete(_node['events']);
441 }
442
443 //}}}
444 //function _setClassForMode(mode)//{{{
445
```

```
446
447  /**
448   * Enables move- or resize mode for all nodes.
449   *
450   * Changes the class names of all nodes of this window that have
451   * the attribute move="change" or resize="change" to the
452   * appropriate class name.
453   *
454   * @param mode string Mode that should be set. "move" or "resize"
455   * @return void
456   */
457
458 function _setClassForMode(mode)
459 {
460     switch (mode) {
461         case 'move':
462             var key = 'chmove';
463             var suffix = '_moving';
464             break;
465
466         case 'resize':
467             var key = 'chresize';
468             var suffix = '_resizing';
469             break;
470
471         default:
472             ajaxWM.debugMessage(_class, '_setClassForMode', 'Wrong mode '+mode);
473             return;
474     }
475     // for each element that should change its class
476     var length = _node[key].length;
477     for (var i = 0; i < length; ++i) {
478         var node = _node[key][i];
479
480         // continue if the current element has the suffix already added
481         if (-1 != node.className.indexOf(suffix))
482             continue;
483
484         // does this node has the size set manually?
485         var setSize = false;
486         if (-1 != _node['size'][ 'set' ].indexOf(node)) {
487             var setSize = true;
488             var width = _getProperty(node, 'width', true);
489             var height = _getProperty(node, 'height', true);
490         }
491
492         node.className += suffix;
493
494         if (setSize) {
495             node.style.width = width;
496             node.style.height = height;
497         }
498     }
499 }
500
501 //}}}
502 //function _unsetClassForMode(mode){//{{{
503
504     /**
```

```
505      * Disables move- or resize mode for all nodes.
506      *
507      * @see AjaxWM_Window::_setClassForMode()
508      * @param mode string Mode that should be unset. "move" or "resize"
509      * @return void
510      */
511
512 function _unsetClassForMode(mode)
513 {
514     switch (mode) {
515         case 'move':
516             var key = 'chmove';
517             var suffix = /_moving/;
518             break;
519
520         case 'resize':
521             var key = 'chresize';
522             var suffix = /_resizing/;
523             break;
524
525         default:
526             ajaxWM.debugMessage(_class, '_unsetClassForMode', 'Wrong mode '+mode);
527             return;
528     }
529
530     var length = _node[key].length;
531     for (var i = 0; i < length; ++i) {
532         _node[key][i].className =
533         _node[key][i].className.replace(suffix, '');
534     }
535 }
536
537 //}}}
538 //{{{ function _onMove(e)
539
540 /**
541  * Moves the window to the current mouse position.
542  * Event handler for onmousemove if move mode is enabled.
543  *
544  * @param e onmousemove event that contains new mouse position
545  * @return false to finish this event
546  */
547
548 function _onMove(e)
549 {
550     // subtract difference between window border and mouse pointer
551     // which was in the window while entering move mode
552     _this.moveTo(e.pageX - _mouseX, e.pageY - _mouseY);
553
554     return false;
555 }
556
557 //}}}
558 //{{{function _startMove(e)
559
560 /**
561  * Enters move-mode to move this window around.
562  *
563  * Stores the current position of the cursor in this window,
```

```
564      * sets up event handlers for displaying new window state and
565      * fixing window on that position.
566      *
567      * @param e onmousedown onmousedown-event that initiated move mode
568      * @return false to finish this event
569      */
570
571  function _startMove(e)
572  {
573      ajaxWM.debugMessage(_class, "_startMove",
574                          "Window " + _id + ": Move mode entered.");
575
576      _this.focus();
577
578      // calculate distance between window border and
579      // mouse pointer (in window)
580      _mouseX = e.pageX - _getProperty(_node[ 'main' ], "left");
581      _mouseY = e.pageY - _getProperty(_node[ 'main' ], "top");
582
583      _setClassForMode('move');
584
585      // event handlers
586      document.onmousemove = _onMove;
587      document.onmouseup   = _stopMove;
588
589      // block event from further execution:
590      return false;
591  }
592
593  // }}}
594  //{{{function _stopMove(e)
595
596      /**
597      * Fixes the window on the new position.
598      *
599      * Leaves move mode by resetting onmouse*-handlers.
600      *
601      * @param e onmouseup event that should stop window from being moved
602      * @return false to finish this event
603      */
604
605  function _stopMove(e)
606  {
607      ajaxWM.debugMessage(_class, "_startMove",
608                          "Window " + _id + " stopped moving.");
609
610      _unsetClassForMode('move');
611
612      // reset handlers
613      document.onmousemove = null;
614      document.onmouseup   = null;
615
616      // block event:
617      return false;
618  }
619
620  //}}}
621  //{{{function _onResize(e)
622
```

```
623  /**
624   * Resizes the window.
625   *
626   * Resizing is only done step by step, every in-/decrease is equal
627   * to the width/height of a character (terminal-like ;).
628   * Changes window position if window should be resized to the north
629   * or west so that the other borders will remain on their current
630   * positions.
631   *
632   * @param e onmousemove Onmousemove-event that launched this function
633   * @return true To not block the event from further effects
634   */
635
636  function _onResize(e)
637  {
638      var pixelsX = 0;
639      var pixelsY = 0;
640      var chars   = 0;
641      var lines   = 0;
642      // get amount of pixels the window should be resized by/{}{
643
644      switch (_direction) {
645      case 'ne':
646          pixelsX = e.pageX - _mouseX;
647      case 'n':
648          pixelsY = _mouseY - e.pageY;
649          break;
650
651      case 'se':
652          pixelsY = e.pageY - _mouseY;
653      case 'e':
654          pixelsX = e.pageX - _mouseX;
655          break;
656
657      case 'sw':
658          pixelsX = _mouseX - e.pageX;
659      case 's':
660          pixelsY = e.pageY - _mouseY;
661          break;
662
663      case 'nw':
664          pixelsY = _mouseY - e.pageY;
665      case 'w':
666          pixelsX = _mouseX - e.pageX;
667          break;
668      }
669
670      //}}{
671      // calculate amount of chars/lines the window should be resized by/{}{
672
673      if (isNaN(_charWidth) || isNaN(_lineHeight))
674          _calcCharSize();
675
676      if (pixelsX < 0) {
677          // TODO: remove Math.abs() and add Math.ceil()?
678          chars = - Math.floor(Math.abs(pixelsX)/_charWidth);
679      } else if (pixelsX > 0) {
680          chars = Math.floor(pixelsX/_charWidth);
681      }
```



```

682
683         if (pixelsY < 0) {
684             // TODO: remove Math.abs() and add Math.ceil()?
685             lines = - Math.floor(Math.abs(pixelsY)/_lineHeight);
686         } else if (pixelsY > 0) {
687             lines = Math.floor(pixelsY/_lineHeight);
688         }
689
690     //}}}
691     // (possibly) move window//{{{
692
693         if (-1 != _direction.indexOf('w') || -1 != _direction.indexOf('n')) {
694             var x = _getProperty(_node['main'], 'left') - pixelsX;
695             var y = _getProperty(_node['main'], 'top') - pixelsY;
696             _this.moveTo(x, y);
697         }
698
699     //}}}
700
701     _this.resizeTo(chars * _charWidth, lines * _lineHeight);
702
703     return true;
704 }
705
706 //}}}
707 //{{{ function _startResize(e)
708
709 /**
710  * Starts resizing this window.
711  * Event handler for onmousedown on HTML nodes that should be able
712  * to resize this window.
713  *
714  * The mouse position and the direction to where the window should
715  * be resized are stored as object properties. The direction will
716  * be extracted from the className of the node that initiated this
717  * event. Resize-mode is active until an onmouseup-event is launched.
718  *
719  * @param e event Onmousedown event, launched by an HTML node
720  * @return false;
721  */
722
723 function _startResize(e)
724 {
725     try {
726         // store mouse position
727         _mouseX = e.pageX;
728         _mouseY = e.pageY;
729
730         _this.focus();
731
732         // get direction from class name, e.g. border_n or border_ne
733         var direction = this.className.substr(-2);
734         if ('_' == direction[0])
735             direction = direction[1];
736         else if ('n' != direction[0] && 's' != direction[0])
737             throw 'class';
738
739         _direction = direction;
740

```

```
741         // set up handler that finally resizes this window
742         document.onmouseup = _stopResize;
743         document.onmousemove = _onResize;
744
745         _setClassForMode('resize');
746     } catch (ex) {
747         switch (ex) {
748             case 'class':
749                 var error = 'Wrong class name: ' + this.className;
750                 break;
751
752             default:
753                 var error = 'Letting unhandled exception bubble up.';
754                 ajaxWM.debugMessage(_class, '_startResize', error);
755                 throw ex;
756         }
757         ajaxWM.debugMessage(_class, '_startResize', error);
758     } finally {
759         // block further execution
760         return false;
761     }
762 }
763
764 //}}}}
765 //{{{ function _stopResize(e)
766
767 /**
768  * Stops resizing mode and finally resizes the window.
769  * Event handler for onmouseup after _startResize() was called.
770  *
771  * @param e event Onmouseup event
772  * @return false;
773  */
774
775 function _stopResize(e)
776 {
777     document.onmouseup = null;
778     document.onmousemove = null;
779     ajaxWM.syncWindow(_this, true);
780     _unsetClassForMode('resize');
781     return false;
782 }
783
784 //}}}}
785
786 // set up properties, nodes, event handlers, timeouts
787 _init();
788 _sync();
789 // draw this window
790 _draw();
791 }
792
793 //
794 // Ensure that the code does not exceed 79 characters per line using an indent
795 // of 4 spaces/level. Use NO TABS at all. Leave the mode line as it is. For
796 // further information see file docs/CodingConvention.
797 // vim: et ts=4 sw=4 fdm=marker
```

## 6.2.4 AJAXWM\_TEMPLATE.JS

```
1 /**
2  * ajaxWM – the free web based window manager for remote terminals
3  * Copyright (C) 2007 Dennis Felsing, Andreas Waidler
4  *
5  * This program is free software: you can redistribute it and/or modify
6  * it under the terms of the GNU General Public License as published by
7  * the Free Software Foundation, either version 3 of the License, or
8  * (at your option) any later version.
9  *
10 * This program is distributed in the hope that it will be useful,
11 * but WITHOUT ANY WARRANTY; without even the implied warranty of
12 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 * GNU General Public License for more details.
14 *
15 * You should have received a copy of the GNU General Public License
16 * along with this program. If not, see <http://www.gnu.org/licenses/>.
17 *
18 * $Id: AjaxWM_Template.js 62 2007-11-30 19:10:35Z pwnd $
19 *
20 * * * * *
21 * AjaxWM_Template
22 *
23 * Handles the behaviour of the GUI according to the template parsed.
24 * Contains all the XML nodes of a theme, parses them to HTML, traces
25 * nodes that have special events like onmousedown.
26 * Classes get their very own nodes by calling AjaxWM_Template::cloneNodes().
27 * The only thing they have to do is additionally calling their own
28 * methods for registering the events (i.e. tracing positions of nodes
29 * in their main node).
30 *
31 * The node property contains the parsed HTML nodes and traces as follows:
32 * 'window' => associative array
33 *   'main' => main node
34 *   'title' => node that contains only the window title
35 *   'content' => node that contains only the terminal
36 *   'chmove' => array of nodes that change their class while moving
37 *   'chresize'=> array of nodes that change their class while resizing
38 *   'pos' => assoc. array of non-auto positioned nodes
39 *     'fixed' => array of nodes that move by the same amount as the window
40 *     'right' => array of n. placed directly right to the window content
41 *     'bottom' => array of n. placed directly under the window content
42 *   'size' => assoc. array of non-auto sized nodes
43 *   'set' => array o. nodes resized by the same amount as the window
44 *   'events' => array of event types
45 *     <type> => associative array of events
46 *     'path' => array. each value is part of path to this element
47 *     'action' => keyword of what should be done when that event occurs
48 */
49 //-----
50
51 AjaxWM_Template = function(name)
52 {
53   // properties//{{{
54
55   var _this = this;
56   var _class = 'AjaxWM_Template';
57
```

```
58     var _name;
59     var _path;
60     var _path_css;
61     var _path_xml;
62     var _xml;
63     var _info; // name (from xml), description, author, email
64     var _node; // window, panel, menu, startmenu
65     var _attrDefault; // element that has default attributes and values
66
67 //}}}
68 // this.parse = function()//{{{
69
70 /**
71  * Loads the XML file belonging to this theme, stores its
72  * contents as property and directly starts with parsing it.
73  */
74
75 this.parse = function()
76 {
77     ajaxWM.debugMessage(_class, 'parse', 'Begin parsing.');
```

```

117     _attrDefault.setAttribute('move',      'fixed');
118     _attrDefault.setAttribute('resize',     'fixed');
119     _attrDefault.setAttribute('position',   'fixed');
120     _attrDefault.setAttribute('size',       'set');
121 }
122
123 //}}}
124 //function _parse()//{{{
125
126 /**
127  * Parses the template.
128  */
129
130 function _parse()
131 {
132     ajaxWM.debugMessage(_class, '_parse', 'begin');
133     _node['window'] = new Object();
134     _node['window']['chmove'] = new Array();
135     _node['window']['chresize'] = new Array();
136     _node['window']['pos'] = new Object();
137     _node['window']['size'] = new Object();
138     _node['window']['pos']['fixed'] = new Array();
139     _node['window']['pos']['right'] = new Array();
140     _node['window']['pos']['bottom'] = new Array();
141     _node['window']['size']['set'] = new Array();
142     _node['window']['events'] = new Array();
143     var win = _xml.documentElement.getElementsByTagName('window')[0];
144     _node['window']['main'] = _parseNodeTree(win);
145 }
146
147 //}}}
148 //function _parseNodeTree(xml, pos)//{{{
149
150 /**
151  * Recursively parse a XML tree into HTML starting at node passed
152  * as first parameter. Will only parse specified node and its childs!
153  *
154  * On first call you should never pass the parameter pos. It will
155  * contain the path of the currently processed node in the tree,
156  * which can be accessed via childNodes of the root node and its
157  * other parents. Pos is needed to write an entry into the respective
158  * array if a node contains some special attributes. The element on
159  * the end of pos is the index of the current node.
160  *
161  * @param xml Node Root of XML node tree
162  * @param pos Array Position of currently processed node
163  * @param return Node HTML node parsed from passed XML one
164  * @param return false On all nodes/childs in root being invalid
165  */
166
167 function _parseNodeTree(xml, pos)
168 {
169     try {
170         // parameter checking//{{{
171
172         var html;
173         if (pos == undefined) {
174             var pos = new Array();
175             pos.push(0);

```

```

176         var isRoot = true;
177     } else if (!pos instanceof Array) {
178         throw 'Wrong parameter for pos: ' + pos;
179     }
180
181     if (!xml)
182         throw 'InvalidCall';
183
184     //}}}}
185     // parse this node//{{{
186
187     if (xml.nodeType != Node.ELEMENT_NODE)
188         throw 'InvalidElement';
189
190     html = _xml2html(xml, pos);
191
192     var debug = 'Node ' + xml.nodeName + '(';
193     debug += pos + ') parsed to ' + html.nodeName + '. ';
194
195     //}}}}
196     // parse childs and their trees, if existing//{{{
197
198     if (!xml.hasChildNodes())
199         throw 'IsLeaf';
200
201     var debug2 = debug;
202     debug2 += 'Continueing with child 1 of ';
203     debug2 += xml.childNodes.length + '. ';
204     debug2 += 'First child is ' + xml.firstChild.nodeName;
205     ajaxWM.debugMessage(_class, '_parseNodeTree', debug2);
206     delete(debug2);
207
208     var posClone = pos.clone();
209     var length = xml.childNodes.length;
210     var childNode;
211     var childPos = 0;
212     for (var i = 0; i < length; ++i) {
213         posClone.push(childPos);
214         childNode = _parseNodeTree(xml.childNodes.item(i), posClone);
215         if (childNode) {
216             html.appendChild(childNode);
217             ++childPos;
218         } else {
219             —childPos;
220         }
221         posClone.pop();
222     }
223     delete(i);
224
225     var debug = 'Childs of ' + xml.nodeName + '(';
226     debug += pos + ') parsed. ';
227
228     //}}}}
229 } catch (e) {
230     switch (e) {
231         // InvalidElement//{{{
232
233     case 'InvalidElement':
234         var debug = 'Node skipped: ' + xml.nodeName + ". ";

```

```

235         break;
236
237         //}}}
238         // IsLeaf//{{{
239
240         case 'IsLeaf':
241             //          debug += 'Node was a Leaf: ' + xml.nodeName + '. ';
242             debug += 'Node was a Leaf. ';
243             break;
244
245         //}}}
246         // let unexpected exceptions bubble up//{{{
247
248         default:
249             ajaxWM.debugMessage(_class, '_parseNodeTree',
250                 'Letting exception bubble up: ' + e);
251             throw e;
252
253         //}}}
254     }
255     } finally {
256         // return Node or false//{{{
257
258         if (!html) {
259             // TODO: throw exception?
260             var error = 'WARNING! ';
261             error += 'Invalid node <' + xml.nodeName + '>, ';
262             error += 'returning false!';
263             ajaxWM.debugMessage(_class, '_parseNodeTree', error);
264             return false;
265         }
266
267         debug += 'End of root, returning ' + html.nodeName;
268         debug += ' with ' + html.childNodes.length + ' childs';
269         ajaxWM.debugMessage(_class, '_parseNodeTree', debug);
270
271         return html;
272
273         //}}}
274     }
275 }
276
277 //}}}
278 // function _xml2html(xml, pos)//{{{
279
280 /**
281  * TODO update doc
282  *
283  * Parses the passed XML node into a HTML one and returns it.
284  *
285  * When the XML element contains some special attributes they will
286  * be written to the respective array in _node. For this reason
287  * the parameter indizes is needed. It will be added to the array
288  * to let the nodes in these arrays be reconstructed.
289  *
290  * @return Node
291  */
292
293 function _xml2html(xml, pos)

```

```
294     {
295         // create HTML node
296
297         switch (xml.nodeName) {
298             case 'window':
299             case 'panel':
300                 var name = 'div';
301                 xml.setAttribute('class', xml.nodeName);
302                 break;
303
304             case 'theme':
305                 throw '<theme> node cannot be parsed to html!';
306
307             default:
308                 var name = xml.nodeName;
309         }
310
311         //alert(pos);
312
313         var html = document.createElement(name);
314
315         // set attributes
316         _setAutoAttr(xml, html, pos);
317
318         return html;
319     }
320
321 //}}}
322 // function _setAutoAttr(xml, pos){{{
323
324 /**
325  * Automatically returns a NamedNodeMap of attributes by the node
326  * that was passed in consideration of all parent nodes and possibly
327  * inherited attributes.
328  * The parameter pos indicates the position of the node relative to
329  * the window node and thus what could be inherited.
330  */
331
332 function _setAutoAttr(xml, html, pos)
333 {
334     // get own attributes and the ones from the parent
335     if(xml.parentNode.nodeName != 'theme'
336     && xml.parentNode.nodeName != '#document') {
337         var parent = xml.parentNode;
338     } else {
339         var parent = _attrDefault;
340     }
341
342     // handle own XML attributes{{{
343
344     var length = xml.attributes.length;
345     for (var i = 0; i < length; ++i) {
346         var attr = xml.attributes.item(i);
347         var name = attr.name;
348         var value = attr.value;
349
350         // try to parse the attribute, if it is special
351         if (_parseSpecialAttr(name, value, pos))
352             continue;
```



```
353         // if not, write attribute to HTML node
354         html.setAttribute(name, value);
355     }
356
357     //}}}
358     // inherit attributes from parent//{{{
359
360     var length = parent.attributes.length;
361     for (var i = 0; i < length; ++i) {
362         var attr = parent.attributes.item(i);
363         var name = attr.name;
364         var value = attr.value;
365
366         // attributes not to be inherited
367         if ('content' == name || 'class' == name)
368             continue;
369         // do not inherit if already inherited
370         if (xml.hasAttribute(name) || html.hasAttribute(name))
371             continue;
372
373         // try to parse the attribute, if it is special
374         if (_parseSpecialAttr(name, value, pos))
375             continue;
376         // if not, write attribute to HTML node
377         html.setAttribute(name, value);
378     }
379
380     //}}}
381     return true;
382 }
383
384 //}}}
385 //function _isSpecialAttr(name)//{{{
386
387 /**
388  * Checks whether a given attribute is a special one which has
389  * nothing to do in a HTML document.
390  *
391  * @return true On name being something special
392  * @return false On name being some default HTML attribute
393  */
394
395 function _isSpecialAttr(name)
396 {
397     switch (name) {
398     case 'move':
399     case 'resize':
400     case 'position':
401     case 'size':
402     case 'onclick':
403     case 'onmousedown':
404     case 'content':
405     case 'name':
406     case 'description':
407     case 'author':
408     case 'email':
409         return true;
410
411     default:
```

```
412         return false;
413     }
414 }
415
416 //}}}
417 //function _parseSpecialAttr(name, value, pos){//{{{
418
419 /**
420  * Parses any special XML attribute that has nothing to do with
421  * HTML. Returns true when such an attribute was passed, false
422  * otherwise.
423  *
424  * Maps events to node positions which can be restored by calling
425  * traceNode() later.
426  *
427  * @return true On name being something special
428  * @return false On name being some default HTML attribute
429  */
430
431 function _parseSpecialAttr(name, value, pos)
432 {
433     switch (name) {
434     case 'move':
435         var key = 'chmove';
436     case 'resize':
437         if (!key)
438             var key = 'chresize';
439         if (value != 'change')
440             return true;
441         _node['window'][key].push(pos.clone());
442         return true;
443
444     case 'position':
445         var attrKey = 'pos';
446         var valueKey = value;
447         // position="auto" does not need to be handled
448         if ('auto' == value)
449             return true;
450         else if ('fixed' != value && 'right' != value && 'bottom' != value)
451             throw 'Invalid value for position: ' + value;
452     case 'size':
453         if (undefined == attrKey) {
454             var attrKey = 'size';
455             var valueKey = value;
456             // size="auto" does not need to be handled
457             if ('auto' == value)
458                 return true;
459             else if ('set' != value)
460                 throw 'Invalid value: "' + value + '"';
461         }
462
463     //alert('attribute attrKey=' + attrKey + ', valueKey=' + valueKey);
464     _node['window'][attrKey][valueKey].push(pos.clone());
465     //alert(
466     //_node['window'][attrKey][valueKey][
467     //_node['window'][attrKey][valueKey].length
468     //);
469     //);
470     //alert(_node['window'][attrKey] + ' ' + _node['window'][attrKey][valueKey]);
```

```
471         return true;
472
473     case 'onclick':
474     case 'onmousedown':
475         var e = new Object();
476         e[name] = new Object();
477         e[name]['path'] = pos.clone();
478         e[name]['action'] = value;
479         _node['window']['events'].push(e);
480         return true;
481
482     case 'content':
483         if (value == 'windowtitle') {
484             _node['window']['title'] = pos.clone();
485         } else if (value == 'windowcontent') {
486             _node['window']['content'] = pos.clone();
487         } else {
488             throw 'Wrong content: ' + value;
489         }
490         return true;
491
492     // TODO
493     case 'name':
494     case 'description':
495     case 'author':
496     case 'email':
497         return true;
498
499     default:
500         return false;
501     }
502 }
503
504 //}}}
505 //this.loadCss = function()//{{{
506
507 /**
508  * Loads the CSS file which belongs to this theme.
509  *
510  * @return void
511  */
512
513 this.loadCss = function()
514 {
515     var css = document.createElement('link');
516     css.rel = 'stylesheet';
517     css.type = 'text/css';
518     css.href = _path_css;
519     document.getElementsByTagName('head')[0].appendChild(css);
520 }
521
522 //}}}
523 //this.cloneNodes = function(root)//{{{
524
525 /**
526  * TODO: doc
527  */
528
529 this.cloneNodes = function(root)
```

```

530     {
531         switch (root) {
532             case 'window':
533                 return _cloneWindow();
534
535             default:
536                 throw "CloneError";
537         }
538     }
539
540 //}}}
541 //function _cloneWindow()//{{{
542
543 /**
544  * Clones window node, reconstructs references out of traces
545  * and returns that clone.
546  * Note that events are not reconstructed – the traces are simply
547  * cloned. You have to call registerEvents() first.
548  */
549
550 function _cloneWindow()
551 {
552     var clone = new Object();
553     clone['main'] = _node['window']['main'].cloneNode(true);
554     for (key in _node['window']) {
555         if (key == 'title' || key == 'content') {
556             var trace = _node['window'][key];
557             clone[key] = traceNode(trace, clone['main']);
558         } else if (key.substr(0,2) == 'ch') {
559             clone[key] = new Array();
560             var length = _node['window'][key].length;
561             for (var i = 0; i < length; ++i) {
562                 var trace = _node['window'][key][i];
563                 clone[key][i] = traceNode(trace, clone['main']);
564             }
565         } else if ('pos' == key || 'size' == key) {
566             clone[key] = new Object();
567             for (attrValue in _node['window'][key]) {
568                 if (attrValue == 'clone')
569                     continue;
570                 clone[key][attrValue] = new Array();
571                 var length = _node['window'][key][attrValue].length;
572                 for (var i = 0; i < length; ++i) {
573                     var trace = _node['window'][key][attrValue][i];
574                     clone[key][attrValue].push(traceNode(trace,
575                                                         clone['main']));
576                 }
577             }
578         } else if (key == 'events') {
579             clone['events'] = _node['window']['events'].clone();
580         }
581     }
582     return clone;
583 }
584
585 //}}}
586
587 _init(name);
588 }

```

```

589
590 //
591 // Ensure that the code does not exceed 79 characters per line using an indent
592 // of 4 spaces/level. Use NO TABS at all. Leave the mode line as it is. For
593 // further information see file docs/CodingConvention.
594 // vim: et ts=4 sw=4 fdm=marker

```

## 6.2.5 INDEX.HTML

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3
4 <html>
5 <head>
6     <title>ajaxWM</title>
7     <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
8     <!-- keep files in cache for 7 days -->
9     <meta http-equiv="expires" content="604800" />
10
11     <link rel="stylesheet" type="text/css" href="default.css" />
12
13     <script type="text/javascript" src="javascript/sarissa.js"></script>
14     <script type="text/javascript" src="javascript/sarissa_dhtml.js"></script>
15     <script type="text/javascript" src="javascript/AjaxWM_Manager.js"></script>
16     <script type="text/javascript" src="javascript/AjaxWM_Template.js"></script>
17     <script type="text/javascript" src="javascript/AjaxWM_Login_Screen.js"></script>
18     <script type="text/javascript" src="javascript/AjaxWM_Window.js"></script>
19     <script type="text/javascript">
20         window.onload = function ()
21         {
22             ajaxWM.init ();
23         };
24     </script>
25 </head>
26
27 <body>
28     <noscript>
29         <div class="fakeTerm">
30             <span class="login">madboy@T22</span>
31             <span class="directory">~</span>
32             <span class="prompt">$</span>
33             cowsay "ajaxWM depends on JavaScript"
34             <pre>
35
36 < ajaxWM depends on JavaScript >
37
38         \      ^__^
39         \    (oo)\_______
40            (__)\       )\/\
41               ||----w |
42               ||      ||</pre>
43             <span class="login">madboy@T22</span>
44             <span class="directory">~</span>
45             <span class="prompt">$</span>
46             <span class="cursor">x</span>
47             <pre>
48
49
50

```

```
51
52
53
54
55
56
57
58
59
60
61         </pre>
62     </div>
63 </noscript>
64 </body>
65 </html>
```

### 6.2.6 DEFAULT.CSS

```
1  /* GLOBALS */
2
3  body {
4      background-color: gray;
5  }
6
7
8  /* NOSCRIPT */
9
10 noscript {
11 }
12 noscript div.fakeTerm {
13     float: left;
14     color: lightgrey;
15     background-color: black;
16     border: 2px solid white;
17     font: 1.1em monospace;
18     position: absolute;
19     top: 20px;
20     left: 50px;
21     padding: 1px;
22 }
23 noscript div.fakeTerm span.login {
24     color: lime;
25 }
26 noscript div.fakeTerm span.directory ,
27 noscript div.fakeTerm span.prompt {
28     color: blue;
29 }
30 noscript div.fakeTerm span.cursor {
31     background-color: lightgrey;
32 }
33
34
35
36 /* DEBUG AREA */
37
38 div.debug {
39     background-color: white;
40     position: absolute;
41     right: 0px;
```

```
42     top: 0px;
43     width: 300px;
44     border-top: 2px dashed red;
45     border-left: 2px dashed red;
46     border-right: 2px dashed red;
47     display: none;
48 }
49 div.debug div.message p.caller,
50 div.debug div.message p.text {
51     margin: 0px;
52     padding: 3px;
53 }
54 div.debug div.message p.caller {
55     padding-bottom: 0px;
56     font-weight: bold;
57 }
58 div.debug div.message p.text {
59     border-bottom: 2px dashed red;
60     padding-top: 1px;
61 }
62
63
64
65 /* TERMINAL */
66
67 pre.term span.f0 { color: #000; }
68 pre.term span.f1 { color: #b00; }
69 pre.term span.f2 { color: #0b0; }
70 pre.term span.f3 { color: #bb0; }
71 pre.term span.f4 { color: #00b; }
72 pre.term span.f5 { color: #b0b; }
73 pre.term span.f6 { color: #0bb; }
74 pre.term span.f7 { color: #bbb; }
75 pre.term span.f8 { color: #666; }
76 pre.term span.f9 { color: #f00; }
77 pre.term span.f10 { color: #0f0; }
78 pre.term span.f11 { color: #ff0; }
79 pre.term span.f12 { color: #00f; }
80 pre.term span.f13 { color: #f0f; }
81 pre.term span.f14 { color: #0ff; }
82 pre.term span.f15 { color: #fff; }
83 pre.term span.b0 { background-color: #000; }
84 pre.term span.b1 { background-color: #b00; }
85 pre.term span.b2 { background-color: #0b0; }
86 pre.term span.b3 { background-color: #bb0; }
87 pre.term span.b4 { background-color: #00b; }
88 pre.term span.b5 { background-color: #b0b; }
89 pre.term span.b6 { background-color: #0bb; }
90 pre.term span.b7 { background-color: #bbb; }
```

## 6.3 THEMES

### 6.3.1 LIM

#### **lim.xml**

```
1 <theme
2   name="lim"
3   description="Less Is More. Extremely minimalistic theme."
4   author="Andreas Waidler"
5   email="andreaswaidler@arcor.de"
6 >
7   <window move="change" resize="change" size="set">
8     <div
9       class="titlebar" position="auto" size="auto"
10      onclick="focus" onmousedown="move"
11    >
12      <span class="text" content="windowtitle" />
13    </div>
14    <div
15      class="content" position="auto" size="auto"
16      content="windowcontent" onclick="focus"
17    />
18  </window>
19  <panel>
20  </panel>
21 </theme>
```

#### **lim.css**

```
1 /* GLOBALS */
2
3 body {
4   padding: 0px;
5   margin: 0px;
6 }
7
8 div, pre {
9   padding: 0px;
10  margin: 0px;
11 }
12
13
14
15 /* LOGIN */
16
17 body.login, body.login_locked {
18   cursor: default;
19   background-color: blue;
20 }
21 body.login_locked {
22   cursor: wait;
23 }
24 form.login {
25   position: absolute;
26   left: 300px;
27   top: 200px;
```



```
28     width: 200px;
29 }
30 form.login input.name {
31 }
32 form.login input.pass {
33     float: left;
34     clear: both;
35 }
36 form.login input.submit,
37 form.login input.submit_locked {
38     cursor: pointer;
39     float: right;
40     clear: both;
41 }
42 form.login input.submit_locked {
43     cursor: wait;
44 }
45
46
47
48
49 /* DESKTOP */
50
51 div.desktop {
52     display: none;
53 }
54 body.desktop {
55 }
56
57
58
59 /* WINDOW */
60
61 div.window,
62 div.window_moving {
63     display: none;
64     float: left;
65     position: absolute;
66 }
67 div.window_moving {
68     background-color: transparent;
69     border: 1px solid black;
70 }
71 div.window_moving div,
72 div.window_moving div pre {
73     display: none;
74 }
75 pre.term {
76     cursor: default;
77     font: 0.8em monospace;
78     color: white;
79     background-color: black;
80 }
81 div.window div.titlebar {
82     background-color: blue;
83     cursor: move;
84 }
85 div.window div.titlebar span.title {
86     font-size: 0.8em;
```

```
87     font-family: sans-serif;
88 }
```

### 6.3.2 WINXP

#### winxp.html

```
1
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de">
6     <head>
7         <!-- Content-Type und Charset uebermitteln,
8              damit der Client weiss, was hier ankommt und mit welchem
9              Charset er es dekodieren soll. -->
10        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
11        <!-- Sprache der Website -->
12        <meta http-equiv="content-language" content="de-DE" />
13        <!-- sofern es sich hier um statischen Inhalt handelt,
14              koennen wir ihn im Cache behalten.
15              Hier: 3 Tage. -->
16        <meta http-equiv="expires" content="259200" />
17
18        <!-- Seitentitel -->
19        <title>Ajaxwm Benutzer</title>
20        <!-- Author dieser Datei -->
21        <meta name="author" content="Andreas Waidler aka MAdbOY" />
22        <!-- Programm, mit dem die Datei erstellt wurde -->
23        <meta name="generator" content="MAdbOY himself (vim 7.0)" />
24        <!-- Cascading Style Sheet einbinden. -->
25        <link rel="stylesheet" type="text/css" href="style1.css" />
26
27    </head>
28
29    <body>
30
31    <div class="taskpanel">
32        <div class="start"></div>
33        <div class="content"></div>
34
35        <div class="info">22:22</div>
36    </div>
37
38
39    <div class="window">
40        <div class="left">
41            <div class="edge_nw"></div>
42            <div class="tborder_w"></div>
43            <div class="spacer_w"></div>
44            <div class="border_w"></div>
45
46            <div class="edge_sw"></div>
47        </div>
48
49        <div class="middle">
50            <div class="border_n"></div>
51            <div class="titlebar">
```

```

52             <span class="text">Titlebar</span>
53             <div class="minimize"></div>
54             <div class="maximize"></div>
55
56             <div class="close"></div>
57         </div>
58     <div class="spacer"></div>
59     <pre class="content">
60 123456789012345
61 10 chars!!adsas
62 asdads.-.-.-.-
63 Iso sieht winxpp
64 aus
65 foooooobaaaaarr!
66 opensource!!!!
67 lololololololol
68 ~~~---##+*#####
69 </pre>
70
71         <div class="border_s"></div>
72
73     <div class="right">
74
75         <div class="edge_ne"></div>
76         <div class="tborder_e"></div>
77         <div class="spacer_e"></div>
78         <div class="border_e"></div>
79         <div class="edge_se"></div>
80     </div>
81 </div>
82 </div>
83
84
85     <div class="rightclick">
86     <pre class="inhalt">rechtsklick
87
88     menue
89 1234567890</pre>
90
91     <div class="startmenue">
92         <div class="user"></div>
93         <div class="programs">hier befinden sich dann die einzelnen prog
94
95     </div>
96
97 </body>
98 </html>

```

### style1.css

```

1 body {
2     background-color: #004e98;
3 }
4
5 div, pre {
6     margin: 0px;
7     padding: 0px;
8 }
9

```

```
10 div.edge_ne,
11 div.edge_se,
12 div.edge_sw,
13 div.edge_nw,
14 div.spacer_e,
15 div.spacer_w {
16     background-color: black;
17     width: 5px;
18     height: 5px;
19 }
20 div.edge_nw {
21     background-image: url(nw.png);
22     background-repeat: no-repeat;
23 }
24 div.edge_ne {
25     background-image: url(ne.png);
26     background-repeat: no-repeat;
27 }
28 div.spacer_w {
29     background-image: url(spacerw.png);
30 }
31 div.spacer_e {
32     background-image: url(spacere.png);
33 }
34 div.edge_sw {
35     background-image: url(sw.png);
36     background-repeat: no-repeat;
37 }
38 div.edge_se {
39     background-image: url(se.png);
40     background-repeat: no-repeat;
41 }
42
43 div.border_n,
44 div.border_e,
45 div.border_s,
46 div.border_w,
47 div.tborder_e,
48 div.tborder_w,
49 div.spacer {
50     background-color: white;
51 }
52
53 div.border_n {
54     background-image: url(n.png);
55 }
56 div.tborder_w {
57     background-image: url(titlew.png);
58 }
59 div.tborder_e {
60     background-image: url(titleE.png);
61 }
62 div.spacer {
63     background-image: url(spacer.png);
64 }
65 div.border_w {
66     background-image: url(borderw.png);
67 }
68 div.border_s {
```

```
69         background-image: url(borderS.png);
70     }
71     div.border_e {
72         background-image: url(bordere.png);
73     }
74
75     div.window div.left div.tborder_w ,
76     div.window div.right div.tborder_e {
77         height: 16px;
78     }
79
80     div.window {
81         position: absolute;
82         top: 20px;
83         left: 20px;
84         background-color: green;
85     }
86
87
88     div.window div.left {
89         position: relative;
90         top: 0px;
91         left: 0px;
92     }
93     div.window div.middle {
94         position: absolute;
95         top: 0px;
96         left: 5px;
97     }
98     div.window div.right {
99         position: absolute;
100        top: 0px;
101        left: 100px; /* JS */
102    }
103
104     div.window div.left ,
105     div.window div.right ,
106     div.window div.left div.border_w ,
107     div.window div.right div.border_e {
108         width: 5px;
109     }
110
111     div.window div.middle {
112         background-color: yellow;
113     }
114     div.window div.middle div.titlebar {
115         background-color: blue;
116     }
117
118
119     div.window div.middle div.border_n ,
120     div.window div.middle div.spacer ,
121     div.window div.middle div.border_s {
122         height: 3px;
123     }
124
125     div.window div.middle div.titlebar {
126         height: 20px;
127     }
```

```
128 div.window div.middle pre.content {
129     white-space: pre;
130     display: block;
131     background-color: #ffffff;
132 }
133
134 div.window div.left div.border_w,
135 div.window div.right div.border_e {
136     height: 104px;
137 }
138
139 div.titlebar {
140     background-image: url(titlebar.png);
141     text-align: center;
142 }
143
144
145
146 div.taskpanel {
147     width: 100%;
148     height: 25px;
149     position: absolute;
150     bottom: 0px;
151     left: 0px;
152 }
153 div.taskpanel div.start {
154
155     width: 99px;
156     height: 25px;
157     background-color: #0057e9;
158     float: left;
159     background-image: url(start.png);
160     background-repeat: no-repeat;
161
162 }
163 div.taskpanel div.content {
164
165     width: 80%;
166     height: 25px;
167     background-color: #255edc;
168     background-image: url(task.png);
169     float: left;
170 }
171 div.taskpanel div.info {
172
173     width: 99px;
174     height: 25px;
175     background-color: #1186e3;
176     background-image: url(info.png);
177     text-align: right;
178     float: left;
179     /* ob man jetzt hier ein image fuer den status reinbringen soll oder ob das nich
180 }
181
182
183
184
185
186 div.rightclick {
```

```
187
188     position: absolute;
189     right: 200px;
190     top: 50px;
191     background-color: gray;
192
193 }
194 div.rightclick pre.inhalt {
195
196     background-color: #eeeeeb;
197     padding: 0px;
198     margin: 0px;
199     margin-left: 1px;
200     margin-right: 1px;
201     margin-bottom: 1px;
202     margin-top: 1px;
203 }
204
205
206
207
208
209
210 div.startmenue {
211     width: 125px;
212     height: 160px;
213     position: absolute;
214     left: 0px;
215     bottom: 25px;
216 }
217 div.startmenue div.programs {
218     width: 100%;
219     height: 130px;
220     background-color: #d3e5fa;
221
222 }
223
224 div.startmenue div.user {
225     width: 100%;
226     height: 30px;
227     background-color: #FFFF00;
228     background-image: url(user.png);
229
230 }
```

## KAPITEL 7

### AUTOREN

Datei	Autor
AjaxWM_Login_Screen.js	Andreas Waidler
AjaxWM_Manager.js	Andreas Waidler
AjaxWM_Template.js	Andreas Waidler
AjaxWM_Window.js	Andreas Waidler
default.css	Andreas Waidler
index.html	Andreas Waidler
lim.xml	Andreas Waidler
lim.css	Andreas Waidler
CodingConvention	Andreas Waidler
ToDo	Dennis Felsing and Andreas Waidler
ajaxwm.py	Dennis Felsing
Error.py	Dennis Felsing
Files.py	Dennis Felsing
Loop.py	Dennis Felsing
Server.py	Dennis Felsing
Session.py	Dennis Felsing
Singleton.py	Dennis Felsing
Ssh.py	Dennis Felsing
Terminal.py	Dennis Felsing
Window.py	Dennis Felsing
INSTALL	Dennis Felsing
README	Dennis Felsing
logo.svg	Dennis Felsing
bordere.png	Ralf Schaufelberger
borderS.png	Ralf Schaufelberger
borderw.png	Ralf Schaufelberger
info.png	Ralf Schaufelberger
ne.png	Ralf Schaufelberger
n.png	Ralf Schaufelberger



nw.png	Ralf Schaufelberger
se.png	Ralf Schaufelberger
spacere.png	Ralf Schaufelberger
spacer.png	Ralf Schaufelberger
spacerw.png	Ralf Schaufelberger
start.png	Ralf Schaufelberger
style1.css	Ralf Schaufelberger
sw.png	Ralf Schaufelberger
task.png	Ralf Schaufelberger
titlebar.png	Ralf Schaufelberger
titleE.png	Ralf Schaufelberger
titlew.png	Ralf Schaufelberger
user.png	Ralf Schaufelberger
winxp.html	Ralf Schaufelberger