# Java8 und diese Sache mit den Applets

*Seit dem letzten Java Upgrade auf die Version SE 8u25 verweigern viele Applets die Arbeit. Im folgenden sollen hierzu die Hintergründe erläutert als auch eine Lösung für das Problem gezeigt werden.*

Eigentlich ist es für Applet-Betreiber bereits ein gewohntes Bild - mit jedem Upgrade der Java-Software wird die Ausführung von Applets ein Stückchen umständlicher. So verwunderte es auch nicht mehr sonderlich, dass nach dem Umstieg auf die Version SE 8u25 anstelle vertrauter Oberflächen befremdliche Fenster mit Fehlermeldungen erschienen (Abb. 1).

[caption id="attachment_9074" align="alignleft" width="300"]

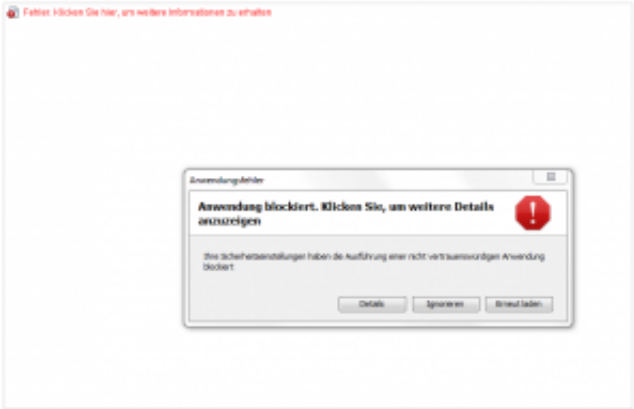


Abb. 1: "*Gauss*"-*Applet mit Fehlermeldung*[/caption]

Um das zu verstehen lohnt es sich, einen Blick auf die Ursprünge und Hintergründe der Applet-Programmierung zu werfen.

Die Boom-Zeit der Applets begann in den 90er-Jahren des letzten Jahrhunderts, als durch das Aufkommen dieser Java-Mini-Programme bislang ungeahnte interaktive Möglichkeiten in das World Wide Web gebracht wurden. Die Programmiersprache Java erfuhr allein aufgrund der Applet-Möglichkeiten eine wahnwitzige Verbreitung. Der kleine Duke, das erste Wahrzeichen der Java-Entwickler, hatte im wahrsten Sinne des Wortes Bewegung ins starre HTML-Coding gebracht.

[caption id="attachment_9075" align="alignright" width="165"]

Abb. 2: *Duke*[/caption]

Die technische Basis, die hinter dem Applet steht, ist dabei eher schlicht. Über den -Tag wird das Applet in der HTML-Seite verankert. Über den Seitenaufruf wird der Applet-Byte-Code dann auf den rufenden Rechner übertragen und über die dort vorhandene Java-Maschine ausgeführt. Die eigentliche Programmausführung findet also auf dem Rechner des Anwenders statt. Diese Technik besaß in den damaligen Zeiten der begrenzten Server-Ressourcen durchaus Charme.

Gleichzeitig wurde den Applets mit diesem scheinbaren Vorteil auch ein Geburtsmakel angeheftet, der sich im Laufe der Zeit immer stärker manifestieren sollte und dessen vorerst letzte Ausprägung sich in Fehlermeldungen wie der oben aufgeführten niederschlägt. Denn, ein Programm, welches auf dem eigenen Rechner ausgeführt wird, birgt immer auch ein gewisses Sicherheitsrisiko - insbesonders wenn man gar nicht erst abschätzen kann aus welcher Quelle es stammt.

Die erste Sicherungsmaßnahme bestand in der Einrichtung der sogenannten Applet-*Sandbox*. Der Grundgedanke der "Sandkiste" besteht darin, dass die Java-Maschine, die das Applet ausführt, auch für dessen Sicherung Sorge zu tragen hat. Dieses tut sie, indem sie dem Applet sämtliche Zugriffe auf die Ressourcen des gastgebenden Rechners untersagt. Ein Applet darf dadurch bspw. nicht Dateien des lokalen File-Systems öffnen oder gar Einträge in der Registry ändern. Eine der ganz wenigen Ausnahmen, die Sandbox zu verlassen, besteht darin, eine Netzwerkverbindung zu dem Server herzustellen, von dem das Applet ursprünglich geladen worden ist.

Im weiteren Verlauf wurden,  quasi parallel mit der Einführung neuerer Technologien (bspw. JavaScript in Verbindung mit AJAX, Flash, ...), die Sicherheitsauflagen für die verbleibenden Applets immer weiter verschärft. Waren es ursprünglich lediglich Sicherheitshinweise, die vor dem Applet-Start bestätigt werden mussten ("Ja, ich habe zur Kenntnis genommen, dass gleich Fremd-Code auf meinem Gerät ausgeführt wird"), so mussten die Anwender wenig später bereits Ausnahmeregeln in der eigenen Java-Maschine hinterlegen, damit bestimmte Applets überhaupt anliefen. Der vorläufige Höhepunkt dieser Kette war dann mit der oben aufgeführten Version SE 8u25 erreicht, welche (ob bewusst oder als Bug) auch die o.a. Ausgänge aus der Sandbox nicht mehr zuläßt.

**Sicher mit Signatur**

Für diejenigen, die aus welchen Gründen auch immer, nicht von den Applets lassen wollen (oder können) bietet sich der Ausweg über *signierte Applets* an. Die Signierung erfolgt dabei mit Hilfe von Zertifikaten, die von einer vertrauenswürdigen Zertifikats-Instanz (bspw. der DFN-PKI) ausgestellt wurden. Applets werden dadurch, im Vergleich mit ihren unsignierten Kollegen, natürlich nicht sicherer, sie bekommen durch die Signierung aber bildlich gesprochen jemanden an die Seite gestellt (nämlich den Zertifikatsinhaber), der für ihr Verhalten bürgt. Dieses "bürgen" reicht indes der Java-Maschine verschlossene Wege, raus aus der Sandbox, wieder zu öffnen.

Was wird ergo benötigt, um aus einem unsignierten/unsicheren Applet ein sicheres/signiertes zu erstellen?

1. Der Applet-Code, der normalerweise in einer .jar-Datei abgelegt ist.
2. Ein signaturfähiges Zertifikat, welches von einer vertrauenswürdigen Instanz ausgestellt worden ist. FernUni-MitarbeiterInnen können sowas kostenlos über unsere hauseigene Certification Authority erhalten.
3. Eine detaillierte Anleitung wie die Signierung vorgenommen werden kann. Diese befindet sich, um alle Nicht-Techniker an dieser Stelle nicht zu langweilen, in unserem Wiki passenderweise unter der Artikelüberschrift Signieren von Applets.

Auch nach erfolgreicher Signierung kommt die Java-Maschine natürlich nicht *ohne* einen Sicherheitshinweis aus (s. Abb. 3) - nach dessen Bestätigung stehen der Applet-Ausführung allerdings tatsächlich keine weiteren Barrieren im Wege.

[caption id="attachment_9083" align="alignleft" width="300"]

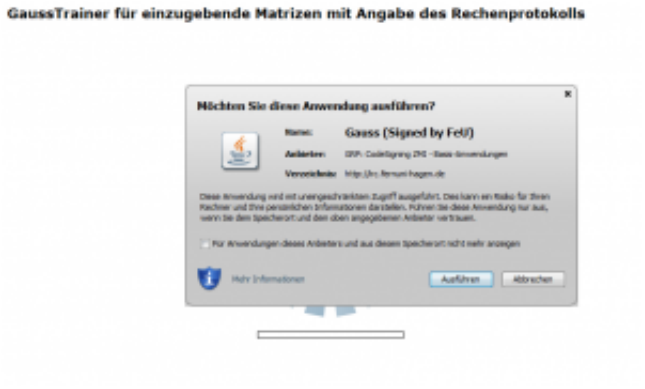


Abb. 3: *Das signierte Gauss-Applet.*[/caption]