Web Server

Краткое описание

Задача состоит в написании простого WEB сервера на Java который бы демонстрировал работу с сокетами, потоками и файловой системой. С помощью сервера можно просматривать HTML файлы, содержимое каталогов. Дополнительно с помощью сервера можно скачивать файлы.

Подробное описание

Разрабатываемый сервер должен максимально реализовывать круг задач возложенный на такого рода серверное ПО:

- интеграция с другим ПО, для расширения спектра выполняемых задач (например интерпретаторы);
- реализация механизма CGI;
- обработка событий возникаемых на сервере, например ошибка 404;
- обеспечение безопасности данных находящихся на серверной стороне;
- обеспечение гибкой конфигурации ядра и компонентов.

Требования

Звездочками * отмечены обязательные требования. Набор обязательных требований может быть изменен или дополнен куратором группы.

1. * Архитектура сервера должна быть реализована в соответствии с паттерном MVC.
2. * Сервер должен быть многопоточным и уметь работать с несколькими пользователями одновременно.
3. * Реализация должна быть выполнена с использованием сокетов.
4. * Сервер должен взаимодействовать с интерпретатором скриптового языка программирования (например PHP) посредством CGI.
5. Сервер должен обрабатывать OPTIONS, POST и GET * запросы.
6. Сервер должен реализовывать механизм *Server Side Includes* на уровне возможности включения файлов директивой: <!--#include file="file.inc" -->.
7. * Сервер должен иметь возможность хранения конфигурации во внешнем файле. Конфигурация должна как можно более гибко позволять настраивать сервер (порт, каталог с файлами, список индексных файлов, файл лога, и т.д.).
8. Конфигурационный файл должен быть в XML формате, и оснащен DTD схемой, работа с ним должна осуществляться с помощью jdom.
9. * Сервер должен уметь посылать основные коды ответов, например:
    a. 200 OK
    b. 403 Forbidden
    c. 404 File Not Found
    d. 405 Method Not Allowed
10. * Сервер должен автоматически определять наличие индексного файла в каталоге. Например:
    a. index.html
    b. index.htm
    c. index.php
11. * При отсутствии индексного файла в каталоге сервер должен предоставлять содержимое каталога.
12. Сервер должен поддерживать основные типы контента. Остальные типы можно принимать как application/octet-stream.
13. Сервер должен уметь обрабатывать .htaccess файлы для команд deny from и allow from.
14. * Сервер должен вестись учет (лог) запросов/ответов сервера.

1. * Программный код должен удовлетворять Java Code Conventions и быть снабженным JavaDoc.
2. * Логирование в коде организовать с помощью библиотеки Log4j.
3. * Для сборки приложения использовать Maven.
4. * Дизайн приложения. Дизайн приложения должен создаваться **до написания кода** и представляться группой куратору для обсуждения и утверждения. Дизайн должен состоять из:
    a. Диаграмма классов (Class diagram) приложения
    b. Диаграмма прецедентов (Use Case диаграмма)

Модификации

Окончательный набор требований к веб серверу должен быть согласован с куратором и зависит от состава группы. В качестве дополнительных не обязательных требований можно реализовать:

1) Главное окно интерфейса сервера должно быть реализовано с использованием библиотеки SWING.
2) Использование systray для сворачивания сервера.
3) Использование стороннего Look&Feel для графического интерфейса окна сервера.

Дополнительная информация

Изучаемые темы:

MVC, Сокеты, Потоки и их синхронизация, Работа с файловой системой

Ссылки:

1. http://systray.sourceforge.net/
2. http://www.javaportal.ru/java/articles/ClientServer.html
3. http://lib.juga.ru/article/articleview/163/1/68/
4. http://www.jdom.org/
5. http://logging.apache.org/log4j/
6. http://www.codenet.ru/cat/WEB-Development/CGI-Development/
7. http://maven.apache.org/guides/getting-started/index.html
8. http://maven.apache.org/guides/getting-started/index.html
9. http://www.info-system.ru/designing/methodology/uml/theory/class_diagram_theory.html
10. http://www.info-system.ru/designing/methodology/uml/theory/use_case_diagram_theory.html