

SAS Solutions for the Web: Static and Dynamic Alternatives

Matthew Grover, S-Street Consulting, Inc.

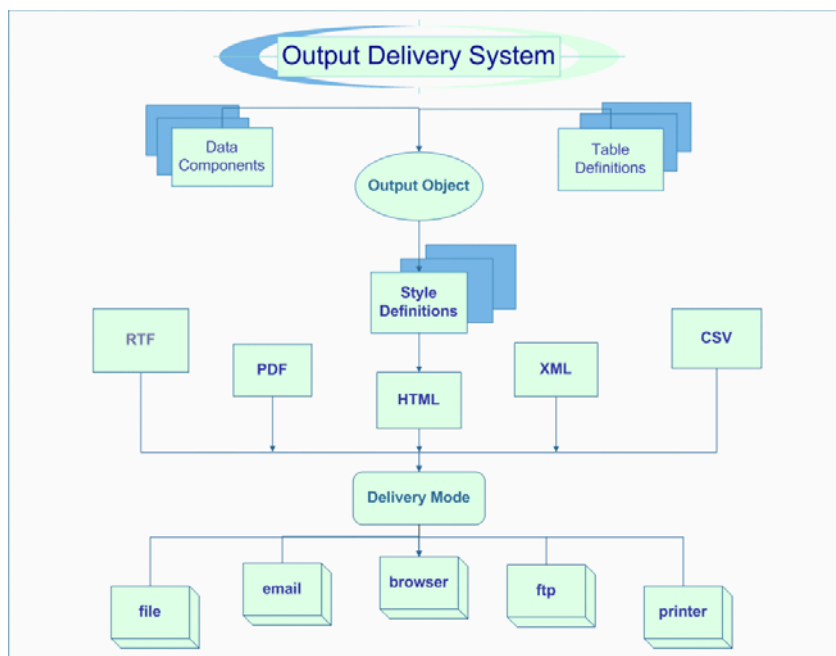
Abstract

This paper provides a detailed analysis of creating static and dynamic web content using the SAS System. The objective is to provide synthesis of the SAS technologies that enable communication with SAS applications via the web. Instead of looking at the dynamic solutions from a product perspective, this paper attempts to look at the software components that are involved with both remote data and remote computing services from a technology perspective.

Web Content Generation

HTML content primarily can be created within SAS in one of two ways

- Write out html tag statements (either in the data step or SCL)
- Output Delivery System (ODS)



Benefits of Using ODS

- Provides an easy way to markup an output file with very little code.
- Content is separated from presentation format within the code by using pre-defined SAS supplied template definitions.
- Provides the ability to isolate distinct pieces of information (content) created by a procedure into separate tables

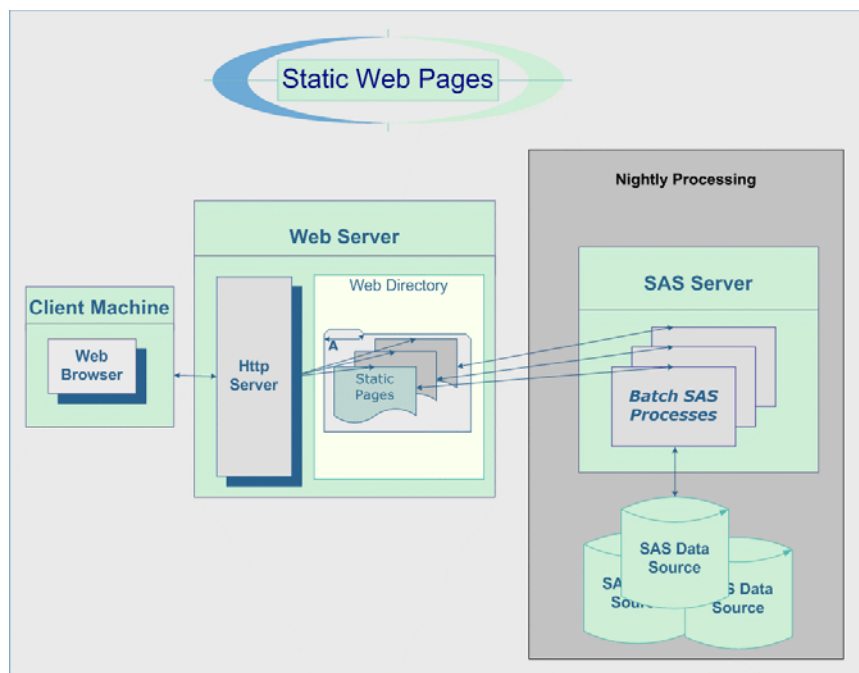
- Provides the ability manage the structure of the content (column order, data formats, additional columns) through the use of custom table definitions
- Provides a variety of choices for formatting the content (HTML, RTF, MARKUP, XML, PCL).
- Provides a way to apply style (color, font, size) to the formatted content via the style definitions
- Provides via table definitions a way to restrict the tabular output for a procedure

Benefits of using Literal Tagsets

- Total control over the exact tags that are written to the file
- Does not require understanding of Proc Template, a very complex procedure
- WISWIG, what you see is what you get, the literal tags are what appears in the file, with ods and proc template the developer gets extra html tags in the code without an easy to remove the tags, other rewriting the default html template.
- A Developer can create macros to build custom tagsets (xhtml, html4.0, etc)

Static Content

A Static page refers to page that physically exists on the directory of a web server. Static web pages get created prior to a user requesting the page, and only need to be recreated when the content of page needs to change. Static web page production usually occurs nightly, pages are sent to the web server directory and are available for the users the following morning.



Advantages:

- Report execution occurs nightly and does not impact the hardware/network during the business day
- Single execution: a single report can be share among all users, does not require each user to query the database to get the information, which would normally occur if the reports were using a dynamic approach
- Speed: the reports are pre-generated as such all that is occurring is the web server is downloading a file to the client browser. There is NO processing time.
- Availability: the report are static and available throughout the day, the individual users do not have to wait for the same report to execute again.
- Archive: the process for archival and retrieval would be straightforward,
- Easy of Setup: only requires mapping an http server to a file and then copying files to those files nightly.

Disadvantages:

- Point-in- time reports: If data is updated, this is not reflected in the report. The report only reflects the data at the point and time the report was created
- Disk Space: storing large amounts of static reports requires disk space.
- Maintenance: multiple files need to be managed and edited
- Limited Functionality: a user for instance could not query a database for new records.

Dynamic Content

In contrast to a static website, a dynamic website is one whose content is regenerated every time a user visits or reloads a page. Dynamic web page content is generated based on some information usually stored in a database. The pages are called "dynamic" because the content is generated each time a page is requested.

Advantages of Dynamic Content

- Interactivity: dynamic content allows the pages user to utilize HTML forms to submit requests to the server. The results of the request can be tailored based on user input.
- Maintenance: because the data used for web page creation is stored in a database, adding and removing content can be a simple as adding or deleting a record in a database.
- Real-Time Access: pages are created based on the data in the database at the point and time of the request. Should an update occur to the database for the information in that page, the user can see the update immediately after it occurs by requesting the page again.

Disadvantages of Dynamic Content

- Complexity: the functionality provided by the dynamic content requires knowledge of a server side programming language required to parse the user request, as well as, setup and maintenance of a database.

- **Software Cost:** costs include; developer time to write the scripts, software licenses, database licensing and maintenance.
- **Hardware Cost:** additional hardware maybe necessary to support resource requirements of a dynamic solution, the hardware requirements are predicated on the software technology selected for creating the dynamic content.

Types of Dynamic Content

Common Gateway Interface (CGI)

The Common Gateway Interface (CGI) is a language-independent interface standard used developing software programs for interfacing external applications with information servers (HTTP, Web servers.) CGI functionality allows an information server to start an external process which gets information about a request through environment variables passed by the server, manipulates the variables within the process and then writes response data to its standard output stream. Each request for the CGI Program starts a separate process in which runs a separate instance of the CGI script (program).

The term CGI request refers to the specific connection from the client browser to that separate CGI process. A CGI program is considered dynamic, because every request results in the real-time execution of the CGI Script.

Risks of CGI

- **Scalability**

Every time a request for the CGI script is submit, on the server side, the CGI script is loaded into memory and a new process is created to run the script. Because HTTP is a stateless protocol, the script is initialized and executed from scratch for each request and stays running just long enough to produce a response to the request. The server has limited number of resources, memory, cpu, and disk space, and there is a limit to how many requests a single server can handle. Scalability is not a problem in an environment where the number of possible requests is known and the hardware for the system is well defined. The real problem is that the path to scalability for CGI is achieved only through hardware, more physical resources must be allocated to the problem.

- **Security**

The CGI protocol itself is not inherently insecure. The security issues are most often a result of following poor coding practices and not being aware of known security issues with the scripting language selected to implement CGI and the operating system the on which script is executed.

- **Portability**

CGI is a specification that can be implemented in many different programming languages. Creating CGI scripts in one programming language on a single operating system in no way assures that the those scripts will run another operating system. In fact, the scripts are limited to use only on the operating systems that support script language.

Java Servlets

Java Servlets are a component-based technology designed for creating dynamic web content. Servlets exist and run inside a web container, often referred to as a servlet engine. The web container is part of an application server that provides the network connections on which requests are received and responses are sent. Unlike CGI, which is a language independent protocol, servlets are only written in Java. However, Java servlets address many of the risks presented by a CGI solution.

Once a web container creates and initializes an instance of the servlet, servlet is available to service requests. There exists only one instance of the servlet with in the web container regardless of the number of request to the servlet.

Every time the server receives a new request for the servlet, the servlet engine spawns a new thread. The servlet engine creates a request object and a response object. The servlet engine invokes the servlet service () method, passing the request and response objects.

The service() method gets information about the request from the request object, processes the request, and uses methods of the response object to create the client response.

Advantages:

- Scalability: The servlet is a single instance, running in a single process, multiple requests to the servlet result in multiple threads getting created, one thread per request.
- Portability: java code can run on any platform that supports a Java Virtual Machine
- Error Handling: The exception handling capabilities of java force programmers to catch errors otherwise code will not compile.
- Component Based: the object-oriented design of java allows developers to create classes that can be reused to build other servlets.

Disadvantages:

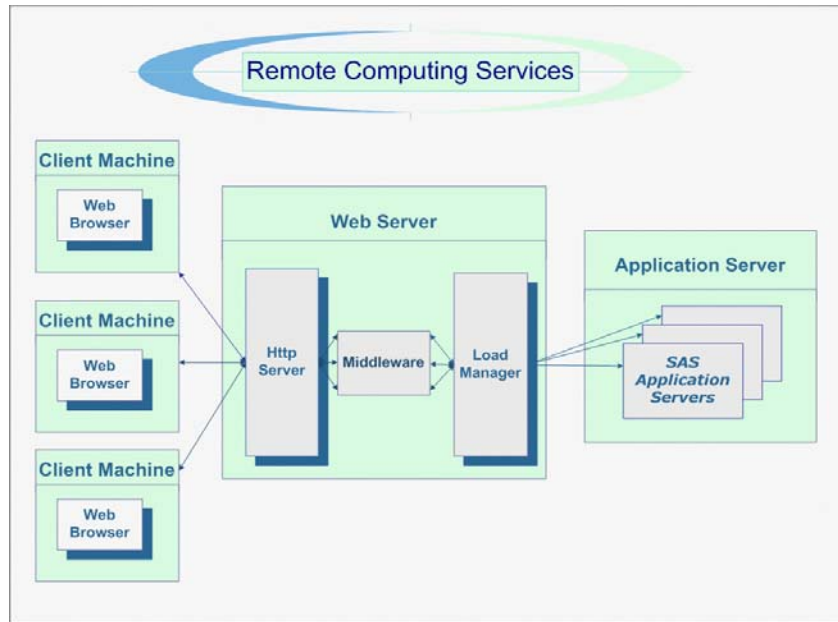
- Complexity: a component based design, while providing code reuse, is inherently more complex than a single CGI script
- Servlet Engine: a servlet runs inside a servlet engine, there is a limited choice for production worthy servlet engines (Web Sphere or Web Logic)
- Cost: life-cycle development costs and software licensing costs are greater than CGI solutions

Functionality of Dynamic Solutions

Regardless of whether the type of dynamic solution is CGI or JAVA, dynamic web page creation with SAS generally can be separated into two functional categories;

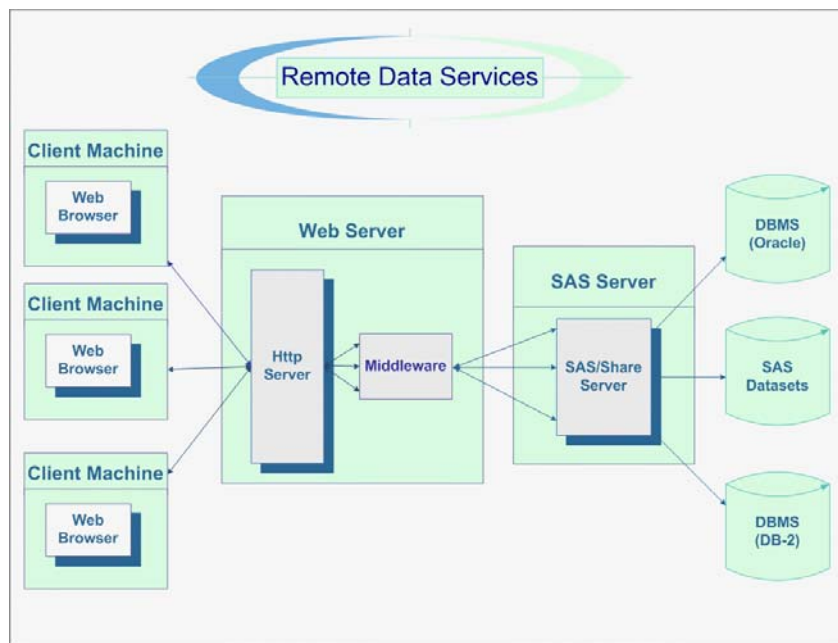
Remote Computing Services:

Provides the ability for a client to leverage computing services provided by a SAS Application Server, by letting the user submit requests to execute SAS programs on a remote SAS server and returning the results of request delivered back to the client.



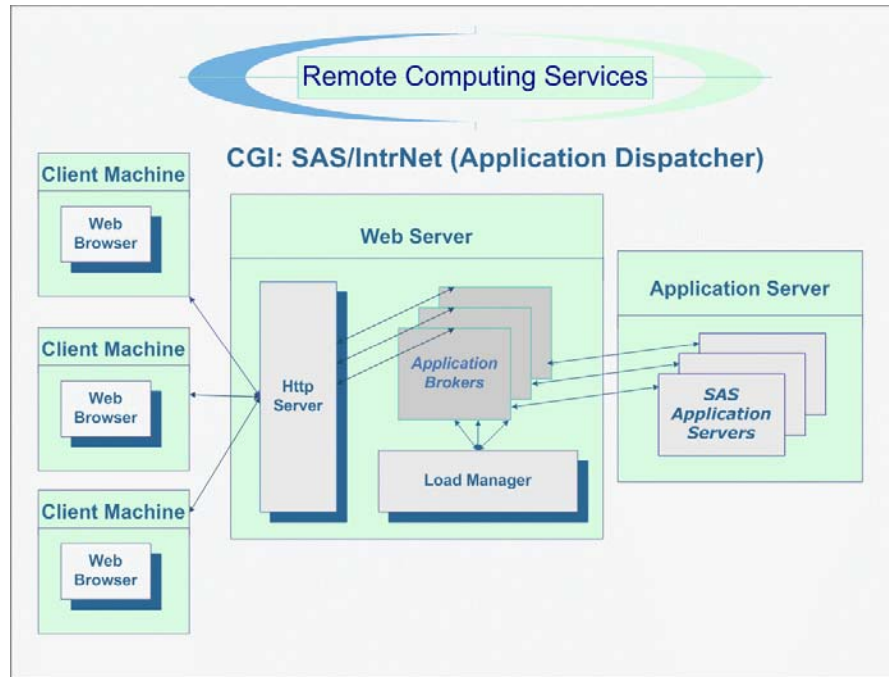
Remote Data Services

Provides the ability for a client communicating through a remote SAS Data Server to query a SAS dataset and any other relational database that SAS can access.



Remote Computing Services: SAS/Intrnet (Application Dispatcher)

The Application Dispatcher allows users to execute SAS programs on remote a SAS Server by clicking on link in their browser and have results returned to them in the browser.



Components:

Input Component: (HTML Form or URL)

Name-value pairs submitted by the client via an html form or explicit URL reference with name-value pairs as part of the URL String. The name-value pairs include a specific name `_program` that specifies what SAS program to execute on the remote SAS server.

Application Broker: (CGI Script)

This is just a single CGI script exists in the cgi-bin directory of a Web server. The Application Broker passes the received name-pair values to a waiting Application Server defined in the broker configuration file and waits for the results to stream back from the Application Server.

Application Server (SAS process started with PROC APPSRV)

The Application Server is a SAS session that accepts as input the name-value pairs from the broker and translates those into SAS macro variables. The Application Server is started by running a batch process that executes PROC APPSRV procedure.

Program Component:

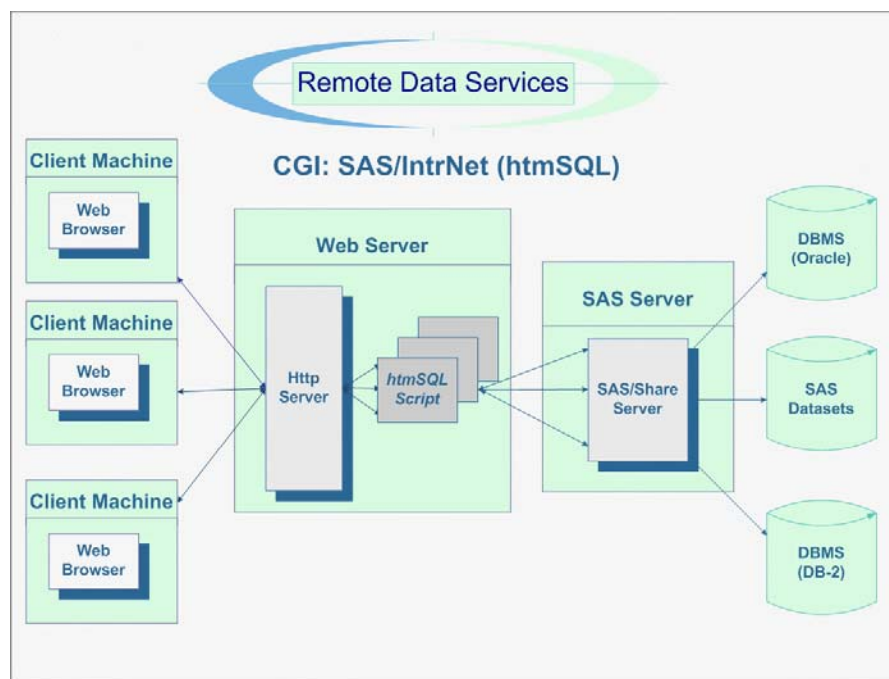
The program component is just a SAS program that is executed by the Application Server. The program processes the name-value pairs as macro variables and streams the resulting output back to the Application broker using a socket connection defined with a reserved file reference (_webout)

Load Manager:

The Application Load Manager is a separate optional process that manages the request between an application broker and a pool of available application servers ready to accept requests. When the broker requests is received the Load Manager gives the request to first available Application Server in the pool, and then removes that application server from the pool. When the request is complete the Application Server is returned to the dynamic pool of available application servers.

Remote Data Services: SAS/IntrNet (HtmSQL)

htmSQL is a single CGI script with the ability to query (SQL syntax) remote data sources served by a SAS/Share Server. The htmSQL script requires an input file to process. The input file contains a mixture of html tags as well as special directives similar to server side includes, these directives allow variable substitution as well as query strings. The record set return from the query is iteratively processed record by record and results are streamed to standard out.



Components

htmSQL Script:

The CGI script processes the directives inside the input file. Literal html tags are streamed to standard out and the directives are processed and results are then streamed back to standard out.

Input file

The file contains html syntax, a query, as well as special directives designed to retrieve values returned from the result set from the query. The directives can retrieve individual column value as well as entire record sets.

SAS Share Server

Provides access from simultaneous access to SAS data from multiple sources.

Advantages SAS CGI Solutions:

- Simplicity: the CGI components are simple to install and configure and can be setup by an experience individual in minutes
- Skill Sets: does not require that programmer understand the complexity of java based application
- Learning curve: existing SAS programmers only need to learn to use the output delivery system.
- Development time: a very quick way to get applications running on the web.
- Maintenance: no software installed on the client machine.

Disadvantages of SAS CGI Solutions:

- Scalability:

Scalability with htmSQL is a greater concern because the script actually receives and processes records sets for formatting on the web server. The Application broker script opens a connection to remote SAS Server and waits to receive the results so that is can stream them back to the waiting client. The processing for the Application dispatcher occurs on the remote SAS server. However, both still create a new process on the web server for each request.

- Security:

The application broker script provides an administrative screen to let an administrator shutdown remote application servers. This screen requires a password, however, a user could potentially guess the password and shutdown the servers.

- Portability:

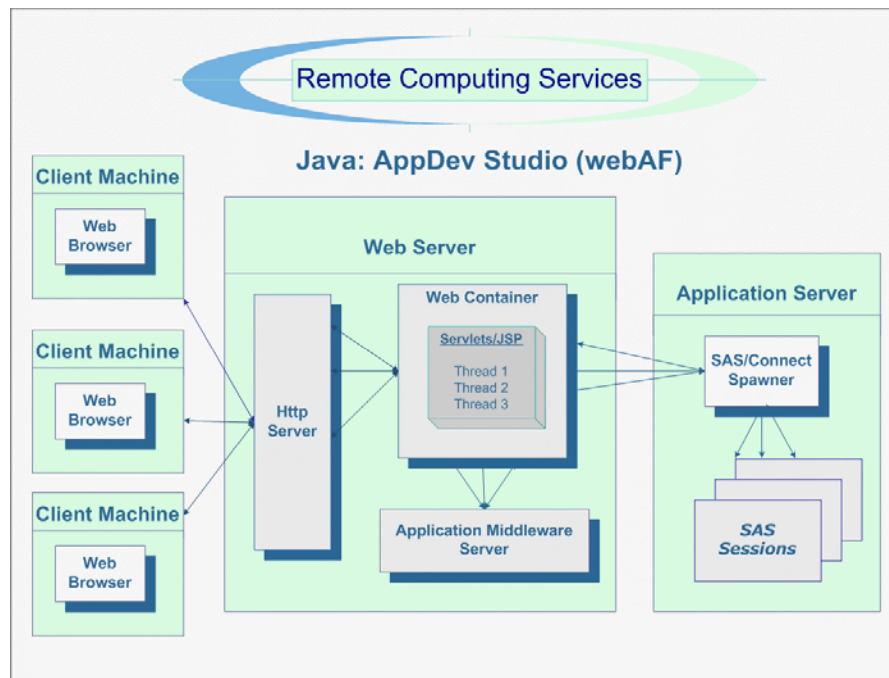
This often a problem when a CGI script is written in a software language that is only available on one platform. The SAS supplied CGI scripts are written in C and have been compiled on

a number of different hardware platforms. The program component for the application broker is just a SAS program that can be ported to any platform running SAS. Portability is not really an issue.

SAS Java Solutions

Remote Computing Services with Java Servlets: (AppDevStudio, WebAF Product)

SAS Institute provides a set of Java packages designed for communicating with remote SAS sessions. These java classes come as part of the AppDevStudio webAF product. WebAF is an integrated development environment (IDE) that allows developers with some training to build Java based Web Applications that connect to the SAS System.



Components:

Servlet (`com.sas.sasserver.submit.SubmitInterface`):

The `SubmitInterface` class is used inside a servlet to create a connection to a remote session and submit code to that session.

SAS/Connect Spawner

The SAS/Connect Spawner is a process that runs on the remote application server machine and manages startup and connections to remote SAS Sessions. The Spawner is designed to allow a client to reuse SAS Sessions and returns subsequent requests from a previous client to a specific SAS session.

LoadManager: AppDev Studio Middleware Server (MWS).

The AppDev Studio Middleware Server is a server-side Java application designed to manage Java client requests to remote SAS Servers. MWS is needed most often in cases where an application received more requests than there are available SAS servers.

Advantages:

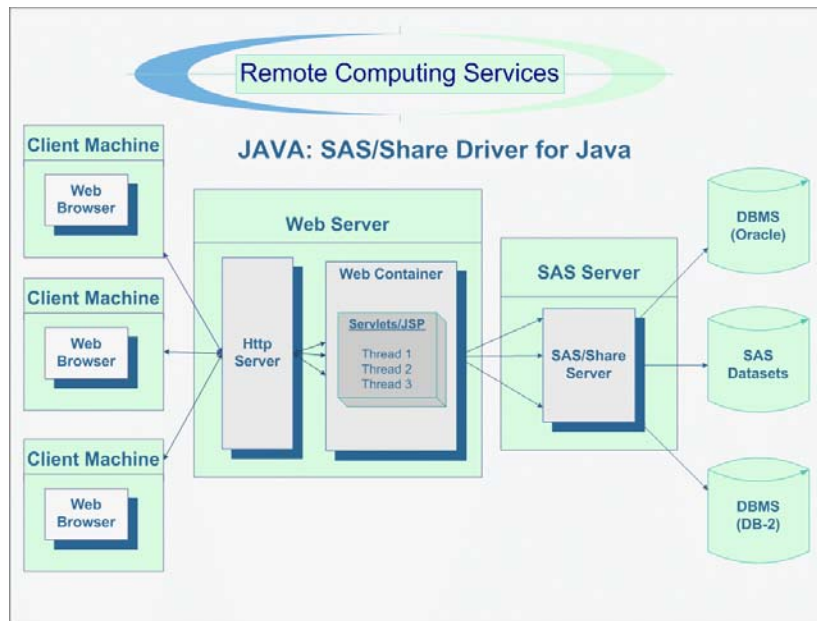
- **Java Packages:**
The java packages are provided with the product provide for remote computing object communication with SCL Classes and remote submission of SAS programs.
- **SAS supplied Tagsets:**
For JSP developers a whole set of custom tag sets are supplied.
- **J2EE:** this product is J2EE Compliant

Disadvantages:

- **Application Middleware Server:**
MWS is a Java application through which multiple Java clients can share a single SAS Server session. The stability and robustness of this product historically has been quite questionable.
- **Proprietary Protocol:**
The protocol to communicate from the Java Client to the remote SAS session is proprietary.

Remote Data Services with Java Servlets (SAS/IntrNet: SAS/Share driver for Java)

The SAS/Share driver for JDBC provides Structured Query Language (SQL) access for both read and update to the any data defined to a SAS/SHARE server. The SAS/Share server provides simultaneous access to SAS datasets, as well as, access to other database products through the use of SAS/ACCESS.



Advantages:

- JDBC Compliance:
the driver is JDBC 2.0 complaint
- Data Access:
Queries can be sent to any data source server by a remote SAS session including RBMS' accessed via SAS/Access

Disadvantages:

- Performance:
The performance of this is solution is slow when compared with alternative CGI based solutions.

References:

DAS SAS System:

http://www.degenring.de/Diplomarbeit_Kapitel4.html

SAS On-line documentation:

<http://support.sas.com/web/index.html>

"Overview of Java Components and Applets in SAS/IntrNet Software", Barbara Walters, Don Chapman, SUGI 23.

"Comparison of CGI Technology Provided in SAS/IntraNet", Barbara Walters, Don Chapman, SUGI25

"Client Server Setup and Implementation: Web and non-web Environments", Sue Herr, SUGI25