República Federativa do Brasil
Ministério da Indústria, Comércio Exterior
e Serviços
Instituto Nacional da Propriedade Industrial

(11) **PI 0620001-0 B1**

(22) Data do Depósito: 18/10/2006

(45) Data de Concessão: 18/09/2018

(54) Título: MÉTODO E SISTEMA PARA SERIALIZAR O ESTADO DE NAVEGAÇÃO EM UM PORTAL

(51) Int.Cl.: G06F 17/30

(30) Prioridade Unionista: 14/12/2005 EP 05112112.7

(73) Titular(es): INTERNATIONAL BUSINESS MACHINES CORPORATION

(72) Inventor(es): STEFAN BEHL; CARSTEN LEUE; FALK POSCH

(85) Data do Início da Fase Nacional: 17/06/2008

"MÉTODO E SISTEMA PARA SERIALIZAR O ESTADO DE NAVEGAÇÃO EM UM PORTAL"

Objeto da presente invenção

[0001] A presente invenção se refere a um método, sistemas e produto de programa de computador para realizar de forma eficiente a serialização do estado de navegação em um aplicativo de portal e, em particular para reduzir o tamanho de marcação das páginas do portal, reduzir o comprimento de URL, e reduzir o tempo de processamento necessário para gerar a URLs sendo parte da página do Portal.

Campo da presente invenção

[0002] O estado de navegação, como usado pela presente invenção descreve "a visualização atual do portal que é o resultado de todas as interações de navegação de um cliente em particular". O cliente pode solicitar (query) diferentes visualizações, interagindo com a página do Portal, por exemplo, navegando até uma nova página. Este tipo de interação do usuário não muda o estado do lado do servidor, mas apenas solicita uma nova visualização do servidor; é, portanto, uma operação "segura" em termos de HTTP. A natureza desta interação do usuário é tal que o cliente pode navegar para trás e para frente através de suas visualizações recentes usando o botão voltar e avançar do seu navegador e que os clientes podem registrar visualizações como favoritas e voltar a elas em um momento posterior, invocando um marcador do navegador.

[0003] Uma das principais características do HTTP é que ele é um protocolo sem estado, ou seja, a noção de uma sessão que abrange múltiplas interações de solicitação/

resposta não existe em HTTP. Mas, como quase todos os cenários de aplicativo requerem algum mecanismo para salvar seu estado em solicitações, alguns mecanismos que surgiram permitem criar (lógica) sessões de estado completo e que podem certamente ser consideradas como estado da técnica na atualidade. Os dois mecanismos de salvamento de estado mais populares são Cookies (RFC 2109) e campos de entrada ocultos.

[0004] No entanto, ambas as abordagens têm algumas desvantagens importantes em relação à capacidade de marcação como favoritos, cachê, botão de Voltar/ Avançar e indexação pelos mecanismos de busca ("crawlability"). Portanto, uma nova abordagem para salvar o estado de navegação emergiu, que codifica o estado de navegação à URL. Diferentes estados de navegação resultam em URLs diferentes. Sendo antes aplicativos baseados em forma somente, aplicações da web atuais tornam-se mais e mais complexas, em particular no ambiente do portal, onde muitos componentes (portlets) são combinados em uma aplicação de portal maior.

[0005] Isso leva ao problema de que o estado de navegação "descrevendo" uma certa visualização de um portal torna-se bastante volumoso, porque o portal deve agregar o estado de navegação de todos os portlets com os quais o usuário interagiu. A capacidade de realizar eficientemente a serialização (e deserialização, respectivamente) do estado de navegação em uma URL deve ser considerado um dos principais recursos para atender às exigências de desempenho atuais sendo forçado para cima de versão para versão (“mais rápido!”, “Menos memória!”).

Estado da Técnica

[0006] Em aplicativos do portal, o estado de navegação é representado como um documento do tipo árvore hierárquica que contém toda a informação que descreve o estado de navegação atual. A estrutura válida do documento hierárquico é definida em um modelo de estado (geralmente uma definição do tipo de documento (DTD) ou definição de esquema XML (XSD)). Para ser capaz de codificar o estado de navegação em URL ou no cabeçalho de uma nova página do portal (por exemplo, na etiqueta de base HTML), é necessário serializar este documento de estado hierárquico.

[0007] As técnicas do estado da técnica para a serialização de estruturas de objetos hierárquicos são normalmente baseadas em técnicas de serialização de XML ou serialização de objeto Java. No campo de portais estas técnicas não são suficientes, porque elas não consideram as características do estado de navegação específicas do portal. Portais combinam várias aplicações, chamadas *portlets*, em uma aplicação de portal maior. Assim, o portal tem de gerenciar o estado de navegação de todos os portlets também. Em outras palavras, o estado de navegação descrevendo uma visualização particular do portal precisa incluir os estados de navegação de todos os portlets.

[0008] O estado de navegação de um portlet é normalmente expresso por meio dos chamados parâmetros de renderização que são definidos pelo programador de portlet. Assim, o portal não é capaz de controlar o estado de navegação dos portlets. Programadores de portlets são livres para definir arbitrariamente os parâmetros de renderização tantos e tão complexos como eles querem. Negligenciar esse

fato durante a serialização do estado de navegação irá provavelmente resultar em URLs muito longas que até mesmo ultrapassem o limite de comprimento de URL de HTTP específica de 2KB. Além do estado de portlet específico, o estado de navegação de um portal tem de incluir também o estado dos controles de navegação de página, barras de ferramentas administrativas, e quaisquer outros elementos da interface de usuário do portal. Assim, o estado de navegação de uma certa visualização de portal torna-se extraordinariamente complexo.

Objetivo da presente invenção

[0009] É objetivo da presente invenção fornecer um método, sistema e produto de programa de computador para realizar a serialização de forma eficiente do estado de navegação de um portal que evita os inconvenientes do estado da técnica.

Sumário da presente invenção

[0010] A presente invenção utiliza uma serialização completamente baseada em fluxo, que transforma a representação do objeto hierárquico do estado de navegação (entrada) em uma representação plana baseada em caracteres de comprimento mínimo (saída). Basicamente, a presente invenção divide a serialização baseada em fluxo em dois subprocessos.

[0011] O primeiro subprocesso de serialização baseada em fluxo que é orientado por hierarquia usa a representação de objeto hierárquico do estado de navegação e transforma o mesmo em uma série de eventos. Os eventos servem como entrada para várias estratégias para a

compactação de informação que está associada com estes eventos. No final do subprocesso, a informação de estado de navegação compactada conduzida pelos eventos recebidos é transformada em uma representação baseada em caracteres e a estrutura hierárquica do estado de navegação é derivada da ordem dos eventos recebidos e transformada em uma representação baseada em caracteres adicional sendo ambas diretamente transmitidas para o segundo subprocesso.

[0012] O segundo subprocesso de serialização baseado em fluxo que é independente de hierarquia, utiliza o resultado do primeiro subprocesso e aplica estratégias de compressão e codificação de caracteres adicionais antes de transmitir a informação codificada em caracteres e comprimida em uma URL ou o cabeçalho da referida nova página do Portal. Ambos os subprocessos são interligados entre si sem interrupção.

[0013] A serialização baseada em fluxo inventiva é eficiente porque minimiza o tempo de processamento total necessário para gerar uma URL e ainda eficiente, pois garante que o resultado de serialização seja o mais curto possível para atender às exigências quanto ao comprimento da URL e tamanho de marcação.

[0014] Em uma concretização preferida da presente invenção, o primeiro subprocesso orientado por hierarquia é baseado em uma cadeia de filtro baseada em eventos que é responsável pela referida compactação das informações de estado de navegação. Cada um destes filtros aplica uma certa estratégia sobre os eventos recebidos gerados no início deste subprocesso. Após o processamento de um evento, o filtro

passa o evento incluindo as informações de estado associado (agora compactadas) para o próximo filtro sendo parte da cadeia de filtro.

[0015] O segundo subprocesso independente de hierarquia é realizado, de preferência, usando uma cadeia de registradores baseada em caracteres. Isto permite transmitir diretamente as informações de estado que foram compactadas no âmbito do primeiro subprocesso de serialização para aquela cadeia de registradores baseada em caracteres, a fim de comprimir imediatamente as informações recebidas.

[0016] O método de serialização inventivo permite adicionar estratégias arbitrárias, ou filtros que estão incluídos na cadeia de filtro baseada em eventos ou registradores que estão incluídos na cadeia de registradores baseada em caracteres.

[0017] Em uma concretização preferida da invenção, a cadeia de filtro baseada em eventos consiste em dois filtros que implementam as seguintes estratégias:

[0018] A chamada estratégia de mapeamento de parâmetro de renderização tem foco sobre o estado de navegação dos portlets, em particular sobre os parâmetros de renderização dos portlets. A estratégia de mapeamento dos parâmetros de renderização monitora a complexidade dos parâmetros de renderização de um portlet particular em termos do comprimento dos nomes e valores de parâmetro. Se um nome ou valor de parâmetro excede um certo limite predefinido, a estratégia irá mapear o parâmetro completo para uma chave curta, composta por um único caractere. Esta estratégia é posta em prática por meio de filtragem dos eventos que

representam os parâmetros de renderização e processamento dos mesmos em conformidade.

[0019] A chamada estratégia de mapeamento de token é responsável pelo tratamento de tokens de estado específico de portal predefinidos (nomes, valores de estado) que podem ser diretamente derivados do modelo de estado de navegação (o DTD ou XSD mencionado). Ele mapeia todos os nomes e valores para uma representação de um caractere curta. Esta estratégia é posta em prática através da filtragem de cada evento e, posteriormente, o processamento das informações associadas (os nomes e os valores mencionados acima) em conformidade.

Breve descrição das várias vistas dos desenhos

[0020] O acima exposto, bem como objetivos, características e vantagens adicionais da presente invenção serão expressos na seguinte descrição detalhada por escrito.

[0021] As novas características da invenção são definidas nas reivindicações anexas. A invenção em si, porém, assim como um modo de uso preferencial, outros objetivos e vantagens serão melhor compreendidos por referência à seguinte descrição detalhada de uma concretização ilustrativa, quando lidos em conjunto com os desenhos anexos, em que:

A figura 1 mostra o método inventivo de serialização baseado em fluxo, de maneira simplificada e, em particular, ilustra a abordagem de duas etapas preferenciais de transformar o documento do estado de navegação hierárquico em uma representação plana baseada em caracteres e,

posteriormente, ainda comprimindo a representação baseada em caracteres;

A figura 2 mostra a transformação de um determinado documento de estado de navegação hierárquica em uma série de eventos por meio de um documento de estado de navegação exemplar de acordo com a presente invenção;

A figura 3 mostra uma concretização preferida da abordagem de serialização em duas etapas, em particular a concretização concreta da cadeia de filtro baseada em eventos e cadeia de registrador baseada em caracteres de acordo com a presente invenção;

A figura 4 mostra o processo inventivo de serialização sendo a contrapartida do processo de serialização, como mostrado na figura 1 e em particular ilustra a abordagem preferida em duas etapas de descompressão da representação plana baseada em caracteres e, posteriormente, transformação desta representação plana de volta para a representação de objeto hierárquico original;

A figura 5 mostra a estrutura interna do filtro do parâmetro de renderização incluindo as interações internas realizadas durante a serialização do estado de preferência usada pela presente invenção;

A figura 6 mostra a estrutura interna do filtro do parâmetro de renderização incluindo as interações internas realizadas durante a deserialização do estado preferencialmente utilizada pela presente invenção;

A figura 7 mostra uma definição de um modelo de estado exemplar usando uma sintaxe DTD e a tabela de mapeamento que

pode ser derivada a partir da mesma, de preferência usada pela presente invenção;

A figura 8 mostra o filtro de mapeamento de token usando a tabela de mapeamento para mapear todos os tokens predefinidos para representações de um caractere curto durante a serialização do estado de preferência usada pela presente invenção;

A figura 9 mostra a estrutura do Portal que executa a presente invenção.

[0022] O método inventivo de serialização baseado em fluxo é mostrado na figura 1.

[0023] O método de serialização é dividido em dois subprocessos que são ambos baseados em fluxo. O primeiro subprocesso de serialização baseado em fluxo que é orientado por hierarquia usa a representação de objeto hierárquico do estado de navegação e transforma o mesmo em uma série de eventos, onde os eventos servem como entrada para várias estratégias para a compactação da informação que está associada com estes eventos, em que no final do referido subprocesso a informação de estado de navegação compactada conduzida pelos eventos recebidos é transformada em uma representação baseada em caracteres e a estrutura hierárquica do referido estado de navegação é derivada da ordem dos eventos recebidos e transformados em uma representação baseada em caracteres adicionais sendo ambas diretamente transmitidas para o referido segundo subprocesso.

[0024] O segundo subprocesso de serialização baseado em fluxo que é independente de hierarquia utiliza o resultado

do dito primeiro subprocesso e aplica estratégias de compressão e codificação de caracteres adicionais antes de colocar em sequência a informação codificada em caracteres e comprimida em uma URL ou cabeçalho da dita nova página de Portal.

[0025] Ambos os subprocessos estão interligados entre si sem interrupção.

[0026] Em uma concretização preferida, o primeiro subprocesso orientado por hierarquia é realizado utilizando uma cadeia de filtro baseada em eventos que consiste em um conjunto de filtros F1 ... Fn sendo responsável por compactar as informações de estado contidas na representação de objeto hierárquico a ser serializada. Cada filtro F1 implementa uma certa estratégia que contribui para alcançar este objetivo.

[0027] A cadeia de filtros obtém uma série de eventos que reflete a estrutura hierárquica do estado de navegação de entrada. Cada evento traz as informações de estado concreta de um nó particular da estrutura hierárquica, ou seja, o nome, o valor e atributos do nó. Assim, os filtros são capazes de analisar as informações do estado recuperado do respectivo evento e reagir em conformidade. Normalmente, um filtro concentra-se em um aspecto específico de informações do estado, por exemplo, estado de navegação de portlet específico. Em outras palavras, o filtro intercepta os eventos que está interessado (por exemplo, através de seu nome), processa estes eventos nesse sentido, e então passa os mesmos para o próximo filtro na cadeia. Todos os outros eventos que não correspondem aos critérios exigidos são delegados inalterados até o próximo filtro.

[0028] Antes de executar a cadeia de filtros, o componente de serialização tem que gerar a série de eventos que corresponde à determinada representação de objeto hierárquico a ser serializada. O componente que está sendo responsável por esta etapa de tradução é o chamado mediador de serialização (ver figura 1).

[0029] Como mostrado na figura 2, o mediador de serialização atravessa o documento de estado hierárquico (usando um algoritmo de busca em largura, como ilustrado por meio das setas pontilhadas), a fim de dar início ao fluxo de eventos. Para cada nó visitado, o mediador de serialização gera um evento *startNode* preenchido com o nome de nó, bem como os atributos do nó (se houver), um evento *nodeValue* preenchido com o valor do nó (se houver), e um evento *endNode* para indicar que o processamento do nó está concluído. Note que a ordem dos eventos gerados, em especial a nidificação de eventos *startNode*, reflete a hierarquia do documento de entrada.

[0030] Após sua criação, o mediador de serialização envia o evento para o primeiro filtro de processamento de evento F1 cadeia. A figura 2 também mostra o fluxo de eventos que correspondem ao documento do estado de navegação exemplar representado. Os nomes que são usados para denotar os vários eventos voltam a termos que são relacionados a técnicas de processamento de XML (por exemplo, o Java API SAX).

[0031] Por meio das informações que estão associadas com um evento (por exemplo, o nome do nó que é transmitido juntamente com um evento *startNode*), cada filtro é capaz de interceptar o tipo de evento que está interessado e modificar

a informação transportada (por exemplo, o valor do nó ou mesmo o nome do nó) antes de passar o evento para o próximo filtro.

[0032] Depois que um evento já passou por todos os filtros, ele atinge o chamado manipulador de serialização (ver figura 2) que é responsável por transformar os eventos recebidos conduzindo as informações de estado compactadas em uma representação de caractere plana. A fim de conseguir que o manipulador de serialização gradualmente acrescente os nomes, valores e atributos para uma representação baseada em caracteres e em seguida, passa a sequência de caracteres resultante para o segundo subprocesso de serialização para compressão e codificação de caracteres adicional. Depois de todos os eventos terem sido processados, o manipulador de serialização manipula uma parte adicional de informação que reflete a hierarquia do documento de estado de navegação. Essa parte de informação é necessária para restaurar a hierarquia do documento de estado durante a deserialização mais tarde. Para codificar as informações da hierarquia, o manipulador de serialização simplesmente codifica uma representação de bit que corresponde ao assentamento dos eventos *startNode* anteriormente processados.

[0033] O segundo subprocesso, que também é completamente baseado em fluxo, processa as cadeias de caracteres recebidas por meio da referida cadeia de registrador baseada em caracteres Wi ... Wn (ver figura 1). Cada registrador aplica uma estratégia de compressão ou de codificação de caracteres específica para as cadeias de caracteres recebidas antes de passá-las para o próximo

registrador da cadeia. Note que esses registradores não podem fazer suposições sobre as cadeias de caracteres recebidas, nem sobre a semântica das informações contidas nem sobre sua estrutura interna. Portanto, eles só podem aplicar técnicas genéricas de compressão e codificação de caracteres, por exemplo, técnicas de compressão do estado da técnica como o GZIP. Depois de vários registradores terem aplicado a compressão (por exemplo, GZIP) e estratégias de codificação de caracteres (por exemplo, UTF-8), a representação de caracteres resultante é gravada em um determinado dissipador de dados (*data sink*). O *data sink* é normalmente uma URL que conduz o formulário serializado do estado de navegação. Ao clicar em uma URL como a parte do estado de navegação tem que ser recuperada a partir dessa URL para a deserialização de estado. Durante o processo de deserialização, todas as estratégias de serialização têm que ser revogadas, conforme explicado na seção seguinte.

[0034] A concretização preferida da cadeia de filtro baseada em eventos e da cadeia de registrador baseada em caracteres é mostrada na figura 2. O gradiente de cor realça a transição sem interrupção da cadeia de filtro baseada em eventos à esquerda para a cadeia de registrador baseada em caracteres à direita.

[0035] Conforme ilustrado, a cadeia de filtro baseada em eventos consiste em dois filtros (além do mediador serialização e manipulador de serialização de sistema específico) que foram descritos no sumário da invenção. O primeiro filtro de processamento de evento implementa a estratégia de mapeamento de parâmetro de renderização de

portlet, enquanto o segundo filtro aplica um mapeamento genérico de todos os tokens predefinidos (os nomes e valores) que constam do documento do estado de navegação. Para mais detalhes sobre esses filtros concretos, por favor, consulte a seção seguinte na seção de serialização. A cadeia de registrador baseado em caracteres consiste (além do registrador de dissipação, que só transmite o resultado para um *data sink* em particular) de um registrador que realiza uma compressão GZIP e outro registrador que executa uma codificação com base em caracteres 64.

[0036] O processo de serialização em contrapartida é responsável por transformar uma determinada representação de caractere plana de estado de navegação de volta para a representação do objeto do tipo árvore hierárquica original, do estado de navegação. O processo de serialização é mostrado na figura 4.

[0037] Conforme ilustrado, os dados de entrada processados pelo método de deserialização da presente invenção é a representação de caracteres plana de comprimento mínimo. O resultado de deserialização é a representação do objeto hierárquico do estado de navegação que é equivalente ao estado de navegação antes da serialização do mesmo. Como a contrapartida da serialização, o processo de deserialização também é dividido em dois subprocessos. Em primeiro lugar, uma cadeia de leitor baseada em caracteres revoga as estratégias de compressão e codificação de caracteres aplicadas durante a serialização, aplicando em contrapartida técnicas de descompressão e de decodificação de caracteres R'_n ... R'_1. Após esse subprocesso, a

hierarquia do documento do estado de navegação é restaurada antes de aplicar novamente uma cadeia de filtro baseada em eventos F'_n ... F'_1, a fim de revogar as estratégias de compactação de informações feitas durante a serialização. Note-se que F'i denota o filtro que se aplica a estratégia inversa que corresponde com filtro Fi. Os parágrafos seguintes descrevem a deserialização em detalhes, como mostrado na figura 4.

[0038] O primeiro componente que está envolvido na deserialização é o chamado leitor fonte. O leitor fonte é um leitor de caracteres que funciona em uma dada fonte de dados. No nosso caso, ele lê o estado de navegação serializado a partir de uma URL e envia as sequências de caracteres lidas para a cadeia de leitor R'_n ... r'_1. A cadeia de leitor revoga a codificação de caractere e compressão realizada durante a serialização, aplicando a decodificação de caractere correspondente e técnicas de descompressão. Atuando como o adaptador entre a cadeia de leitor baseado em caracteres e a cadeia de filtro baseada em eventos, o analisador de serialização armazena temporariamente as cadeias de caracteres recebidas para finalmente transformá-las de volta para uma série de eventos que é equivalente ao fluxo de eventos durante a serialização. A fim de conseguir que ele leve as informações da hierarquia codificada em conta para gerar os eventos na ordem certa, com a correta nidificação. Note que o analisador de serialização pode facilmente executar essa tarefa, pois é o componente de contrapartida do referido manipulador de serialização, assim, tendo o

conhecimento necessário sobre como interpretar as informações da hierarquia serializada.

[0039] Em seguida, o fluxo de eventos passa a cadeia de filtro baseada em eventos F'_n ... F'_1 que reverte a compactação de informação feita durante a serialização. No final da cadeia de filtro baseada em eventos, os chamados mediadores de deserialização (o componente de contrapartida do mediador de serialização) mapeia diretamente os eventos recebidos para operações de controle a fim de restaurar a representação do objeto hierárquico, criando os nós e conectando uns aos outros. Afinal, o documento de estado de navegação original inteiro foi restaurado. Esta etapa conclui o processo de serialização.

O filtro de mapeamento de parâmetro de renderização

[0040] O filtro de mapeamento dos parâmetros de renderização tem foco no estado de navegação dos portlets, em particular sobre os parâmetros de renderização de portlets. A estratégia implementada por este filtro monitora a complexidade dos parâmetros de renderização de um portlet particular em termos do comprimento dos nomes e valores de parâmetros. Se um nome de parâmetro ou valor excede um certo limite predefinido, a estratégia irá mapear o evento representando o parâmetro completo para um evento conduzindo uma chave curta, composta por um único caractere enquanto os nomes de parâmetro concretos e os valores são serializados para a sessão de HTTP do usuário.

[0041] A figura 5 mostra a estrutura interna do filtro de parâmetro de renderização que é usado durante a

serialização do estado (o filtro inverso necessário durante de deserialização é discutido mais tarde).

[0042] Primeiro, o filtro verifica se o evento *startNode* recebido indica um parâmetro de renderização de portlet (através do nome do nó "parâmetro"). Se o filtro detecta um evento como esse (o ramo "sim"), ele passa para um componente mapeador de parâmetro interno que realiza a verificação de complexidade. Caso contrário (o ramo "não"), apenas delega o evento para o próximo filtro na cadeia de filtro.

[0043] O mapeador de parâmetro executa a verificação de complexidade, avaliando o nome e o valor do parâmetro de renderização em termos de seu comprimento. Se o limite for de comprimento configurado é excedido, ele mapeia o parâmetro para uma chave de curta (um caractere) e envia um evento conduzindo essa chave somente para o próximo filtro. O parâmetro de renderização atual, bem como a chave gerada é serializado na sessão de HTTP. Note que este procedimento corre o risco de sessões de HTTP muito grandes. Portanto, os mapeamentos de parâmetros de chaves têm de ser gerenciados através de uma política de expiração que expulsa os mapeamentos armazenados em uma estratégia de menos utilizado recentemente (LRU).

[0044] A figura 6 mostra a estrutura interna do filtro de contrapartida que revoga esse mapeamento no decorrer da deserialização. Nesse caso, o filtro intercepta eventos representando tais chaves somente. Se tal evento foi detectado, o filtro re-encaminha o evento chave para o componente mapeador de parâmetro interno que utiliza a chave

recebida para pesquisar o nome do parâmetro concreto e valor na sessão de HTTP. Assim que o evento levando o parâmetro de renderização atual é restaurado, ele é passado para o próximo filtro na cadeia.

Filtro de mapemento de Token

[0045] O filtro de mapeamento de token implementa uma estratégia genérica que visa compactar tantos tokens quanto possíveis. Por isso, não intercepta eventos que correspondem a determinados critérios (por exemplo, um nome de nó específico), mas tenta mapear todos os tokens, ou seja, o nome, valor e atributos de cada evento recebido. Para realizar esse mapeamento, ele reutiliza a informação obtida a partir da definição do modelo de estado. Como já mencionado anteriormente, a definição do modelo de estado especifica a estrutura válida (isto é, as relações entre os nomes de nó, bem como os tipos de dados dos valores de nó e valores de atributo) de um documento do estado de navegação e é tipicamente descrita em um arquivo DTD ou XSD. Baseado nessa definição, o filtro de mapeamento de token deriva uma tabela de mapeamento que atribui um caracter substituto a cada token predefinido contido. Os tokens predefinidos incluem os nomes de nós e atributos (que são completamente especificados na definição de modelo de estado), bem como todos os nós e valores de atributo que são explicitamente especificados na definição do modelo de estado.

[0046] A figura 7 mostra uma definição do modelo de estado de amostra utilizando a sintaxe de DTD semelhante, incluindo a tabela de mapeamento de token que é derivada dessa definição. A tabela de mapeamento representada é, por

exemplo, plana, ou seja, não reflete a hierarquia do modelo de estado. Em cenários típicos onde o modelo de estado é muito mais complexo, recomenda-se também explorar a hierarquia do modelo de estado. Nesse caso, uma chamada árvore de mapeamento é derivada da DTD em vez de uma tabela de mapeamento plana. Em uma árvore de mapeamento, os mapeamentos têm de ser únicos no âmbito de um certo nível de árvore somente.

[0047] A figura 8 mostra como a tabela de mapeamento plano é aplicada em uma amostra de fluxo de eventos. Note que o filtro de mapeamento de token deve ser o último filtro na cadeia de filtro baseada em eventos, pois modifica os nomes dos nós do documento normalmente servindo como critério de filtragem dos outros filtros (caso contrário, todos os outros filtros que são executados após o filtro de mapeamento de token teriam que conhecer a tabela de mapeamento).

[0048] Todos os valores de nó e atributos que não são especificados na definição do modelo de estado não podem ser mapeados em uma representação curta de um caractere. Isto aplica-se, em particular, aos nomes e aos valores dos parâmetros de renderização de portlet.

[0049] A figura 9 mostra a estrutura do Portal implementando a presente invenção.

[0050] A estrutura do portal, de preferência, compreende componentes de função que já fazem parte de cada portal do estado da técnica e os componentes que foram recentemente adicionados a fim de proporcionar a funcionalidade inventiva.

[0051] Os componentes de função do estado da técnica que estão fazendo parte de cada Portal são os seguintes:

Servlet 10

[0052] O Servlet 10 é o controlador frontal que recebe as solicitações de HTTP de entrada. Ele geralmente prepara o mecanismo de processamento de solicitação 20 para processar a solicitação de HTTP recebida para inicializar os componentes envolvidos. Depois disso, o servlet 10 delega o processamento para o mecanismo de processamento de solicitação 20.

O mecanismo de processamento de solicitação 20

[0053] O mecanismo de processamento de solicitação 20 também faz parte do controlador frontal sendo responsável por controlar o processamento das solicitações recebidas. Normalmente define um ciclo de processamento de solicitação que é composto de várias fases do processamento da solicitação. Em um Portal, uma solicitação tem que caminhar por quatro fases. Primeiro, a fase inicial executa tarefas de inicialização de solicitação específica seguida pela fase de ação que está sendo responsável pela autenticação e execução de ação (ações Portlet, bem como comandos). Após a fase de ação, a fase de renderização é executada invocando o processo de agregação. A fase terminal conclui o processamento da solicitação, realizando tarefas de limpeza de solicitação específica.

Componente de Agregação 60

[0054] O componente de agregação 60 é invocado durante a fase de renderização do ciclo de processamento da solicitação. É responsável por transformar o modelo de layout

(descreve como os portlets são organizados na página) da página solicitada em uma árvore de apresentação, bem como escrever a marcação que corresponde a esta árvore de apresentação para a resposta.

Componente de Autenticação 50

[0055] O componente de autenticação 50 é responsável pela verificação da identidade do usuário. Cada solicitação de entrada tem que passar por autenticação. O Portal 2 usa a identidade do usuário para determinar o conteúdo que o usuário está autorizado a acessar, assim como os comandos para executar.

Portlet Container 30

[0056] O *Portlet Container* 30 fornece acesso unificado ao Portlets. Em especial, permite reunir a marcação de um certo Portlet ou executar uma ação de Portlet. O Portlet container invoca Portlets por meio da API do Portlet.

API de Comando 40

[0057] A API (*Application Programming Interface*) de comando 40 fornece uma camada de abstração para comandos específicos de portal. Em especial, permite a execução de comandos através de uma interface unificada. Os comandos podem ser usados para executar tarefas administrativas, como a criação e/ ou exclusão de páginas do portal, adição e/ ou remoção de portlets e/ ou através das páginas do portal, organizar portlets em páginas já existentes e assim por diante.

Componente de gerenciamento do estado de navegação 70

[0058] O componente de gerenciamento de estado de navegação 70 tem as seguintes atribuições:

- definição de um ciclo de vida para o estado de navegação, bem como proporcionar uma interface que permite que o mecanismo de processamento de solicitação 20 incorpore as tarefas de processamento de estado definidas no ciclo global de processamento de solicitação;

- definição de um modelo de objeto para representar o estado de navegação, bem como fornecendo uma interface de programação de aplicativo (API) que permite a leitura e escrita/ modificação do estado de navegação;

- fornecer um quadro que permite a serialização, de forma eficiente, da representação do objeto de estado de navegação em uma URL, bem como deserialização do estado de navegação a partir de uma URL (entrada) para restaurar a representação do objeto interno. O quadro deve incluir interfaces que podem ser invocadas pelo componente de geração de URL para criar estado de navegação carregando URLs.

API de geração de URL 80

[0059] A API de geração de URL 80 prevê, como o nome indica, uma API que permite criar URLs para uma variedade de casos de uso. Ele permite ao programador associar estado de navegação à URL criada, bem como escrever a URL em um fluxo de destino determinado. Esta operação de escrita envolve serializar o estado de navegação que tem sido associado com a URL criada. Além de criar URLs por programação, a API de geração de URL tipicamente oferece algumas etiquetas de URL para criar URLs dentro de JSPs.

[0060] Por padrão, uma URL criada é inicializada com o estado de navegação de solicitação específica para se certificar de que o estado de navegação de interações

anteriores não seja perdido. Para determinar a semântica específica da URL, esse estado de navegação pode ser mudado para esta URL em particular.

[0061] O componente de função recentemente acrescentado que fornece a funcionalidade inventiva refina a componente de gerenciamento de estado de navegação 70:

Quadro de serialização de estado 70 A

[0062] O quadro de serialização de estado 70 A é um subcomponente do componente de navegação de gerenciamento de estado 70, que coloca o método de serialização baseado em fluxo inventado. É responsável, em particular por fornecer:

- um primeiro subprocesso de serialização baseado em fluxo sendo orientado por hierarquia que usa a representação de objeto hierárquico de estado de navegação e transforma a mesma em uma série de eventos, em que os referidos eventos servem de entrada para várias estratégias para a compactação da informação que está associada com estes eventos, em que no final do referido subprocesso a informação de estado de navegação compactada realizada pelos referidos eventos recebidos é transformada em uma representação baseada em caracteres e a estrutura hierárquica do referido estado de navegação é derivada da ordem dos ditos eventos recebidos e transformada em uma representação baseada em caráter adicional, ambas sendo diretamente transmitidas ao referido segundo subprocesso;

- um segundo subprocesso de serialização com base em fluxo sendo independente de hierarquia que utiliza o resultado do dito primeiro subprocesso e aplica estratégias de compressão e codificação de caracteres adicionais e,

finalmente, transmite a informação codificada em caractere e comprimida em uma URL ou cabeçalho da referida nova página do Portal, onde os dois subprocessos são interligados entre si sem interrupção, montando estratégias de serialização múltiplas (bem como a contrapartida de estratégias de deserialização) para uma cadeia de deserialização;

- oferecer pontos de ligação para a ligação de novas estratégias no futuro;

- fornecer uma interface de programação de aplicativo (API) que permite a execução da cadeia de serialização montada durante a geração de URL, bem como a execução da cadeia de serialização durante decodificação de URL.

[0063] Os dados de entrada do componente constituem um documento de estado de navegação hierárquico para serialização ou uma representação de caractere serializado do estado de navegação para deserialização.

REIVINDICAÇÕES

1. Método para serializar o estado de navegação em um Portal, em que o referido Portal está em execução num sistema de servidor, em que o referido sistema de servidor inclui um componente de comunicação que permite a comunicação entre o referido Portal e o navegador do cliente através de um canal de comunicação, em que o referido Portal determina o layout da página de Portal solicitada, invoca a renderização dos vários elementos de página pertencentes à referida página do Portal, e transmite a referida página do Portal ao navegador do cliente para exibição, em que pelo menos um elemento de página da referida página do Portal fornece uma funcionalidade de link para iniciar a renderização de uma nova página ou novo elemento de página pelo referido Portal, em que cada interação de usuário ao clicar no referido link no referido elemento de página gera um novo estado de navegação no referido lado do Portal, em que o referido estado de navegação descreve a visualização atual do referido Portal como o resultado de todas as interações de navegação anteriores de um cliente particular, em que, em resposta a uma solicitação do cliente para uma nova página de Portal, o referido estado de navegação é codificado em uma forma serializada em cada URL e/ ou no cabeçalho da nova página do Portal, o referido método **caracterizado pelo** fato de que compreende os dois subprocessos a seguir:

um primeiro subprocesso de serialização baseado em fluxo que usa a representação de objeto hierárquico do referido estado de navegação e o transforma em uma série de eventos, cada evento conduz a informação de estado concreta

de um nó particular na estrutura hierárquica, em que os referidos eventos servem como entrada para compactar a informação que está associada com estes eventos, em que no final do referido subprocesso a informação de estado de navegação compactada conduzida pelos eventos recebidos é transformada numa representação baseada em caracteres e a estrutura hierárquica do referido estado de navegação é derivada da ordem dos referidos eventos recebidos e transformada em uma representação adicional baseada em caracteres,

um segundo subprocesso de serialização baseado em fluxo que usa o resultado do primeiro subprocesso e aplica compressão e codificação de caracteres adicionais e finalmente transmite as informações comprimidas e codificadas em caracteres em uma URL ou cabeçalho da referida nova página do Portal,

em que ambos os subprocessos estão interligados entre si.

2. Método, de acordo com a reivindicação 1, **caracterizado pelo** fato de que o referido primeiro subprocesso é baseado em uma cadeia de filtro baseada em evento, em que cada filtro opera em um subconjunto de eventos especificamente atribuído e aplica compactação da informação de estado de navegação que está associada com os referidos eventos.

3. Método, de acordo com a reivindicação 2, **caracterizado pelo** fato de que o primeiro subprocesso gera o fluxo de eventos para a cadeia de filtros baseada em eventos pela transposição da representação de objeto

hierárquico do estado de navegação e gera para cada nó visitado da referida representação de objeto um evento *startNode* preenchido com o nome do nó, bem como os atributos do nó, um evento *nodeValue* preenchido com o valor do nó e um evento *endNode* para indicar o fim do nó.

4. Método, de acordo com a reivindicação 3, **caracterizado pelo** fato de que os eventos *startNode*, *nodeValue* e *endNode* gerados são gerados em uma ordem que reflete a estrutura do tipo árvore hierárquica da representação do objeto de estado de navegação, em que o referido evento *endNode* não é gerado até antes que a subárvore do respectivo nó seja processada.

5. Método, de acordo com a reivindicação 4, **caracterizado pelo** fato de que no final da cadeia de filtros baseada em eventos, um filtro é usado para codificar a estrutura hierárquica que é estabelecida na ordem dos eventos recebidos e na nidificação dos eventos *startNode* recebidos.

6. Método, de acordo com a reivindicação 2, **caracterizado pelo** fato de que o dito segundo subprocesso utiliza uma cadeia de escrita baseada em caracteres, em que cada registrador executa uma compressão ou codificação de caracteres na informação de caracteres recebida gerada no final do referido primeiro subprocesso.

7. Método, de acordo com a reivindicação 6, **caracterizado pelo** fato de que no final da referida cadeia de escrita baseada em caracteres é utilizado um registrador que transmite diretamente a informação de caracteres comprimida para a URL atribuída ou para o cabeçalho de uma página de Portal.

8. Sistema para serializar o estado de navegação num Portal, em que o referido Portal está em execução no referido sistema, em que o referido sistema inclui um componente de comunicação que permite a comunicação entre o referido Portal e o navegador do cliente através de um canal de comunicação, em que o referido Portal determina o layout da página de Portal solicitada, invoca a renderização dos vários elementos de página pertencentes à referida página do Portal, e transmite a referida página do Portal ao navegador do cliente para exibição, em que pelo menos um elemento de página da referida página do Portal fornece uma funcionalidade de link para iniciar a renderização de uma nova página ou novo elemento de página pelo referido Portal, em que o referido Portal inclui um componente de gerenciamento de estado de navegação que gera uma nova navegação para cada interação de usuário, se ao clicar no referido elemento de página, em que o referido estado de navegação descreve a visualização atual do referido Portal como resultado de todas as interações de navegação anteriores de um cliente em particular, em que o referido componente de gerenciamento de estado de navegação codifica o referido estado de navegação em uma forma serializada em cada URL e/ ou no cabeçalho da nova página do Portal em resposta a uma solicitação do cliente para uma nova página do Portal, o referido sistema **caracterizado pelo** fato de que compreende os seguintes componentes de função:

meios que fornecem um primeiro subprocesso de serialização baseado em fluxo que usa a representação de

objeto hierárquico do referido estado de navegação e o transforma em uma série de eventos,

cada evento transporta a informação de estado concreta de um nó particular na estrutura hierárquica, em que os ditos eventos servem como entrada para compactação da informação que está associada a estes eventos, em que no fim do referido subprocesso a informação de estado de navegação compactada conduzida pelos referidos eventos recebidos é transformada numa representação baseada em caracteres e a estrutura hierárquica do referido estado de navegação é derivada da ordem dos referidos eventos recebidos e transformada em uma representação adicional baseada em caracteres sendo transmitida diretamente para o segundo subprocesso, e

meios que fornecem um segundo subprocesso de serialização baseado em fluxo sendo independente de hierarquia que usa o resultado do primeiro subprocesso e aplica compressão e codificação de caracteres adicionais e finalmente envia as informações compactadas e de caracteres codificados em uma URL ou cabeçalho da nova página do Portal,

em que ambos os subprocessos fornecidos pelos ditos meios estão interligados entre si.

9. Sistema, de acordo com a reivindicação 8, **caracterizado pelo** fato de que os referidos meios são parte integrante do dito componente de gerenciamento de estado de navegação.

10. Sistema, de acordo com a reivindicação 9, **caracterizado pelo** fato de que o referido componente de gerenciamento de estado de navegação é parte do referido Portal.

11. Sistema, de acordo com a reivindicação 8, **caracterizado pelo** fato de que os referidos meios que fornecem o referido primeiro subprocesso gera o fluxo de eventos para a cadeia de filtros baseada em eventos, pela transposição da representação de objeto hierárquico do estado de navegação e gera para cada nó visitado da representação do objeto um evento *startNode* preenchido com o nome do nó, bem como os atributos do nó, um evento *nodeValue* preenchido com o valor do nó e um evento *endNode* para indicar o final do nó, em que os referidos eventos *startNode*, *nodeValue* e *endNode* gerados são gerados em uma ordem que reflete a estrutura tipo árvore hierárquica da representação do objeto de estado de navegação, em que o referido evento *endNode* não é gerado até antes da sub-árvore completa do respectivo nó ser processada, em que no final da cadeia de filtros baseada em eventos um filtro especial é usado para codificar a estrutura hierárquica que é colocada na ordem dos eventos recebidos e na nidificação dos eventos *startNode* recebidos.

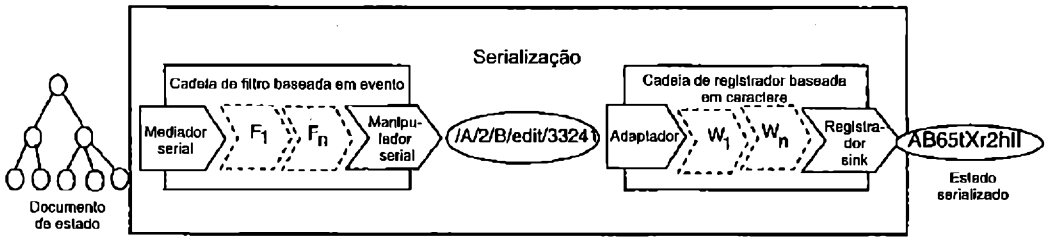
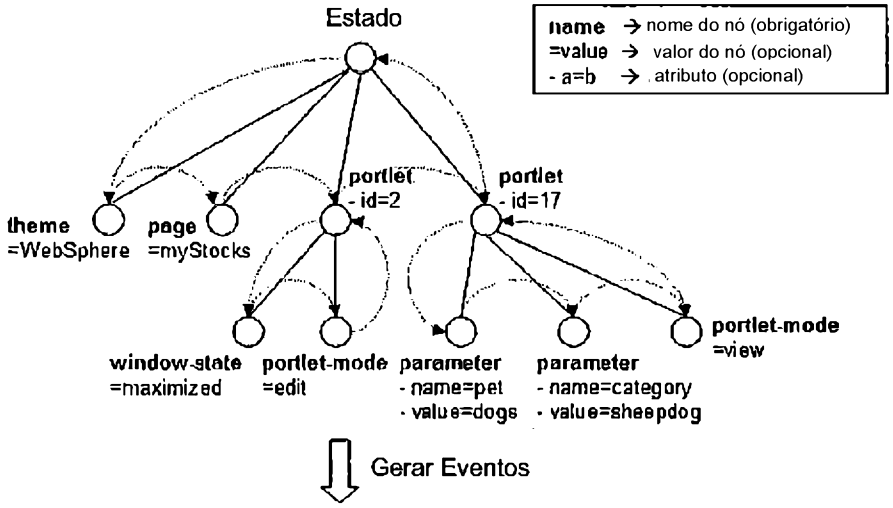


FIGURA 1




```
startNode(state, -)
startNode(theme, -)
nodeValue(WebSphere)
endNode(theme)
startNode(page, -)
nodeValue(myStocks)
endNode(page)
startNode(portlet, {id=2})
startNode(window-state)
nodeValue(maximized)
endNode(window-state)
startNode(portlet-mode)
nodeValue(edit)
endNode(portlet-mode)
endNode(portlet)
etc.
```

FIGURA 2

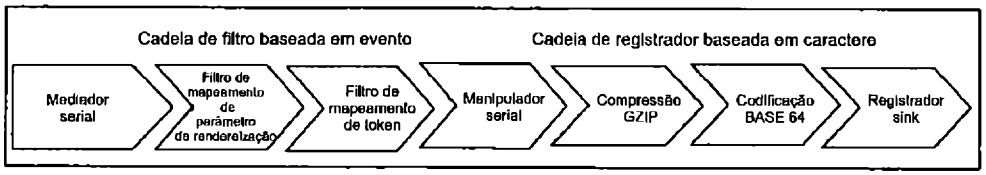


FIGURA 3

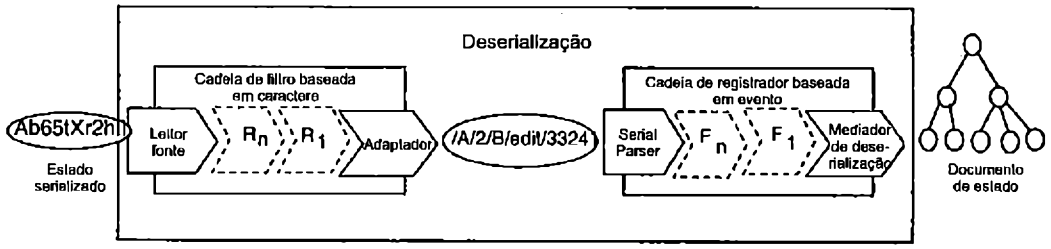


FIGURA 4

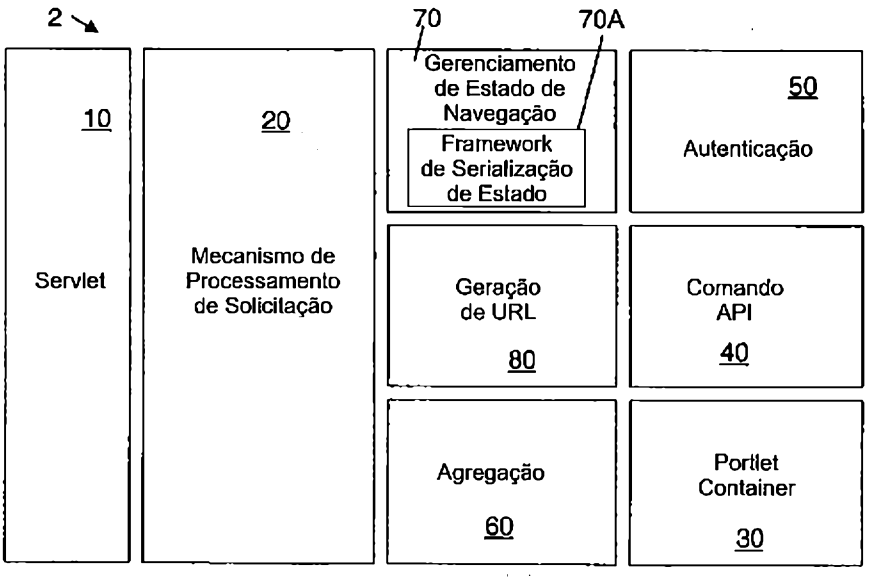


FIGURA 9

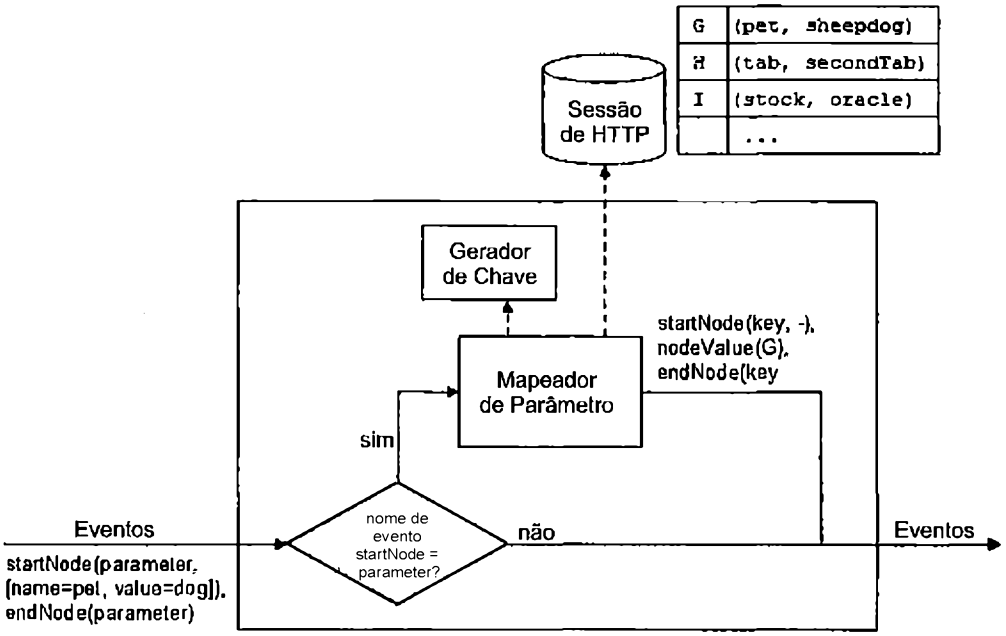


FIGURA 5

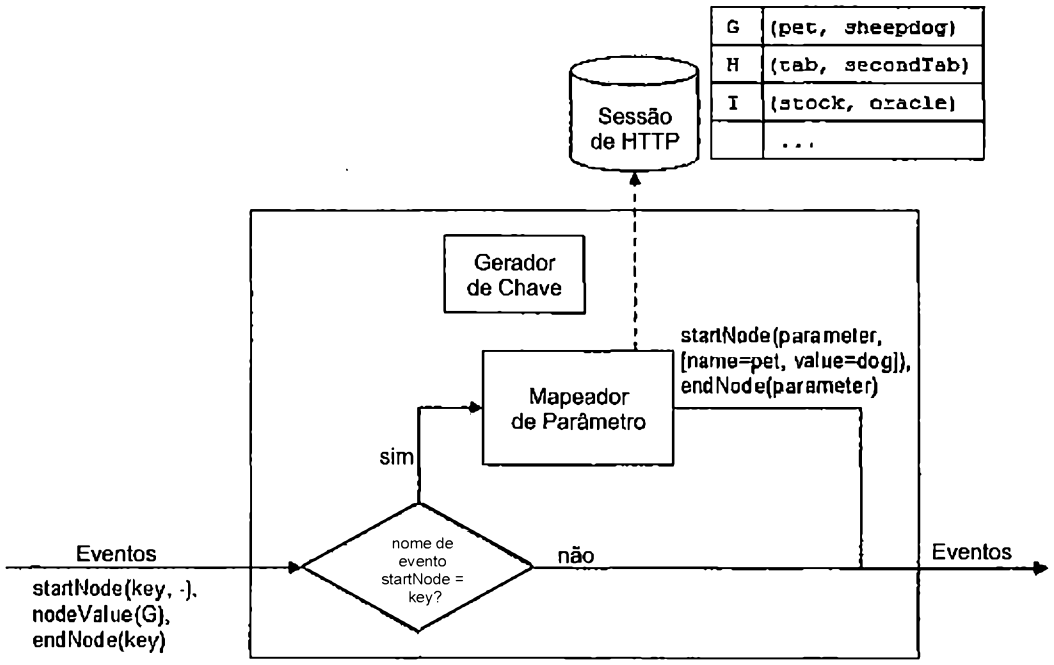


FIGURA 6

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT state (theme?, page?, portlet*)>
<!ELEMENT portlet (portlet-mode?, window-state?, parameter*)>
<!ATTLIST portlet
    id CDATA #REQUIRED
>
<!ELEMENT portlet-mode (view|edit|help)>
<!ELEMENT window-state (normal|max|min)>
<!ELEMENT parameter EMPTY>
<!ATTLIST parameter
            name CDATA #REQUIRED
      value CDATA #IMPLIED
>
<!ELEMENT page (#PCDATA)>
<!ELEMENT theme (#PCDATA)>
...
```

Mapping table (flat)

State → A, theme → B, page → C, portlet → D, id → E, portlet-mode → F, view → G, edit → H, help → I, window-state → J, normal → K, max → L, min → M, parameter → N, name → O, value → P, etc.

FIGURA 7

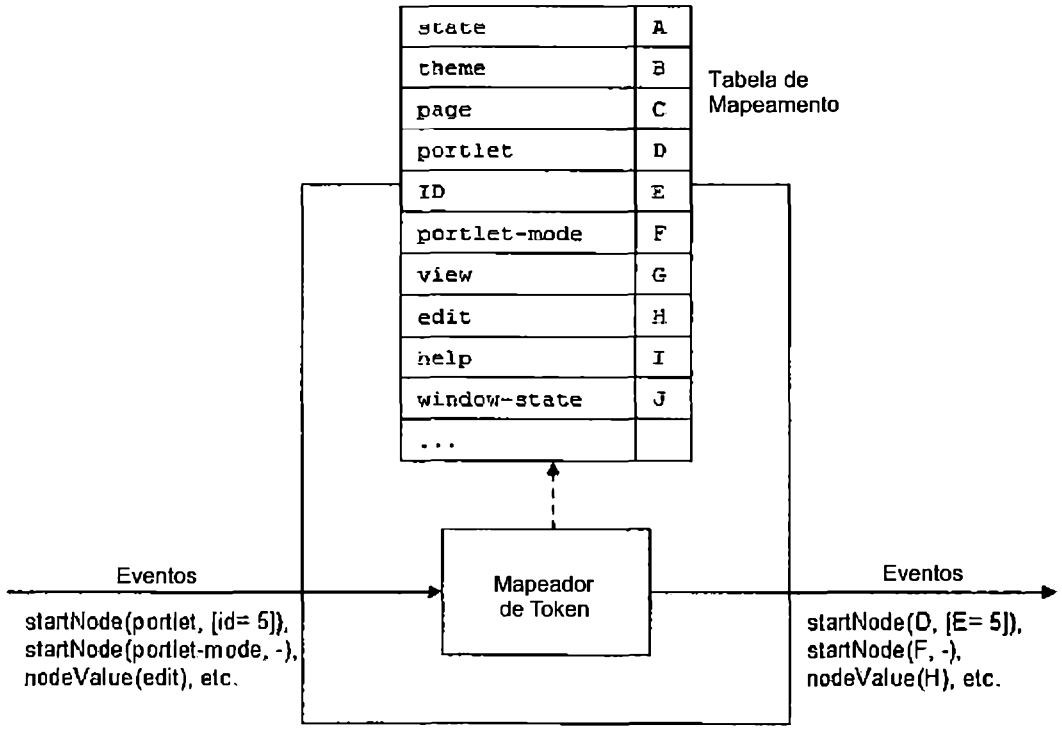


FIGURA 8