5
Ferramenta de Geração de Interfaces Concretas

Este capítulo apresenta uma ferramenta de geração de interfaces concretas para aplicações hipermídia, gerando interfaces a partir de instâncias da ontologia de Widgets Abstratos, que por sua vez representam interfaces abstratas.

5.1.
Arquitetura

O sistema desenvolvido é composto por dois módulos:

- O módulo "*Abstract Interface Compiler* (AIC - Compilador de Interface Abstrata)" é responsável pela geração de um arquivo JSP composto por *Tag Libraries* (*taglibs*) e por declarações de *Java Beans* (*beans*). As *taglibs* representam os elementos de interface abstratos e os *beans* são objetos que disponibilizam as informações, que serão utilizadas por essas *taglibs*, visando a geração dos elementos concretos. Neste módulo, porém, as informações que estão contidas no *bean* não estarão disponíveis, pois é descrito no arquivo JSP apenas a declaração do uso de um ou mais *bean*s, não disponibilizando, desta forma, o seu objeto.
- O módulo "*Concrete Interface Renderer* (CIR - Gerador de Interface Concreta)" é responsável pela geração da página concreta, que é realizada através da interpretação do arquivo JSP, gerado pelo módulo AIC, em tempo de execução. No módulo CIR os objetos *beans* que contêm as informações necessárias, que são utilizadas pelas *taglibs* para a geração dos elementos concretos, são disponibilizados. Desta forma, a interpretação das *taglibs* é realizada e os elementos concretos que compõem a página são criados. A página final, gerada pelo módulo CIR do sistema, é constituída de código HTML e, em alguns casos, de Java Script. É pretensão futura desenvolver uma maneira de disponibilizar a geração da página concreta em outras linguagens.

A arquitetura do sistema pode ser visualizada na figura 50.

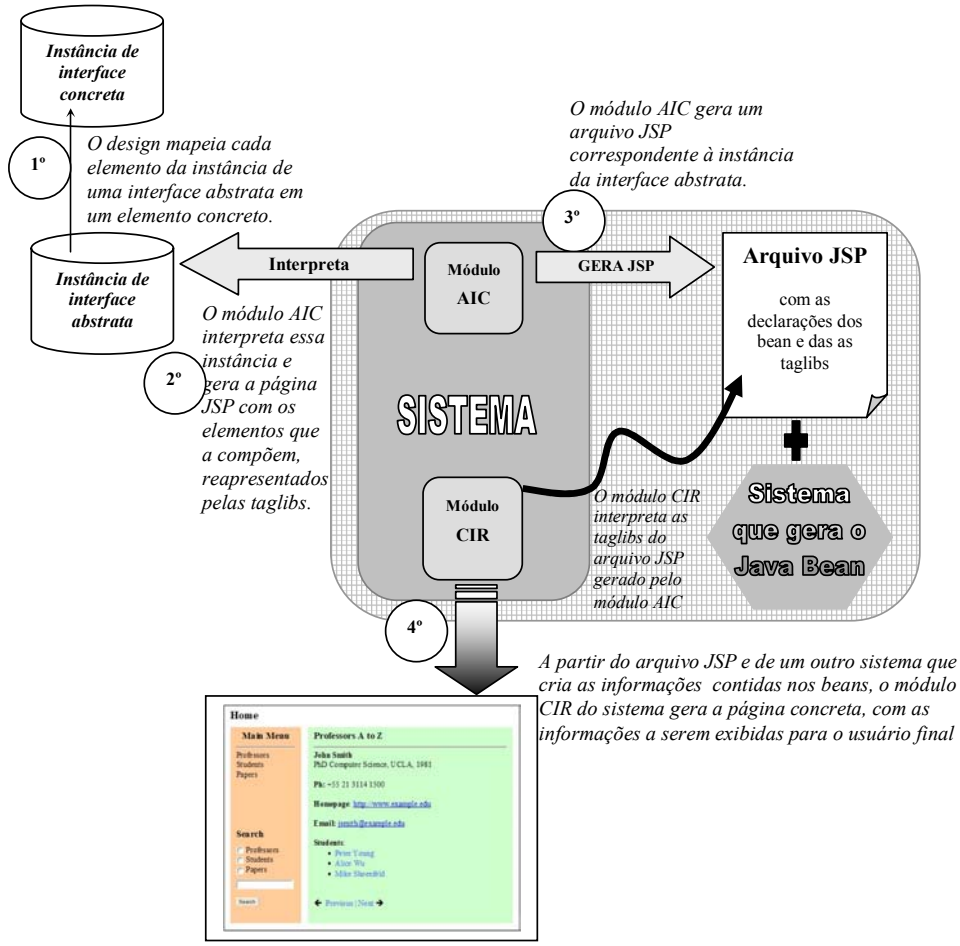


Figura 50 - Arquitetura do Sistema.

5.1.1. Abstract Interface Compiler (AIC - Compilador de Interface Abstrata)

O módulo AIC é responsável pela criação de um arquivo JSP a partir de uma instância da ontologia de *widgets* abstratos, que representa uma interface abstrata. Esse arquivo é composto por:

- uma descrição de todos os elementos abstratos que compõem a interface, através de *taglibs;*[1]
- uma declaração dos *beans* que serão utilizados para buscar as informações sobre esses elementos, com o objetivo de gerar a interface concreta.

Este módulo deve realizar duas funções para a geração do arquivo JSP: a geração da declaração dos *beans* e a descrição dos elementos da interface abstrata.

[1] http://java.sun.com/products/jsp/taglibraries/index.jsp

Na figura 51 pode-se observar a arquitetura que resume as funções desse módulo do sistema.

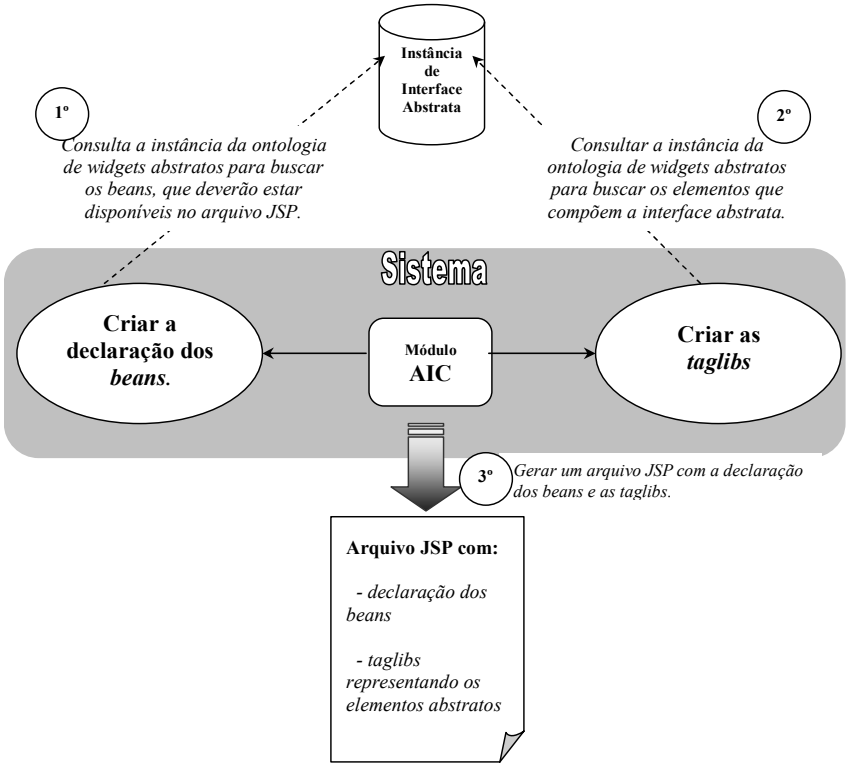


Figura 51 - Arquitetura do módulo AIC.

Para gerar a declaração de cada *bean*, o sistema precisa realiza uma consulta, na instância de interface abstrata, para buscar os nomes dos objetos *beans* que serão utilizados para a criação do elemento concreto. Um exemplo dessa declaração pode ser visualizado na figura 52.

```
1  <jsp:useBean id="idxMainMenu" class="shdm.data.Index" scope="request" />
2  <jsp:useBean id="ctxProfessorAlpha" class="shdm.data.Context" scope="request" />
```

Figura 52 - Declaração dos *beans* no arquivo JSP.

Na primeira linha da figura 52 é realizada a declaração do *bean* do tipo "*Index*" que está localizado no pacote "shdm.data.Index". A palavra "idxMainMenu" é o nome do objeto representado por esse *bean*, sendo ele o responsável pelas informações que serão utilizadas pelas *taglibs*. Existe ainda outra informação importante nessa declaração, a que indica que esse objeto estará disponível no "escopo" do arquivo via comando "*request*".

A segunda linha da figura 52 também representa uma declaração de um *bean*, mas neste caso o *bean* é do tipo "*Context*".

Para gerar as *taglibs* que representam os elementos abstratos que compõem a interface, o módulo do sistema AIC consulta a instância da interface abstrata

para buscar os elementos que deverão ser descritos por cada *taglib*. Na figura 53 pode-se observar o modelo de uma instância de um elemento abstrato em notação N3 [2]e a sua *taglib* correspondente.

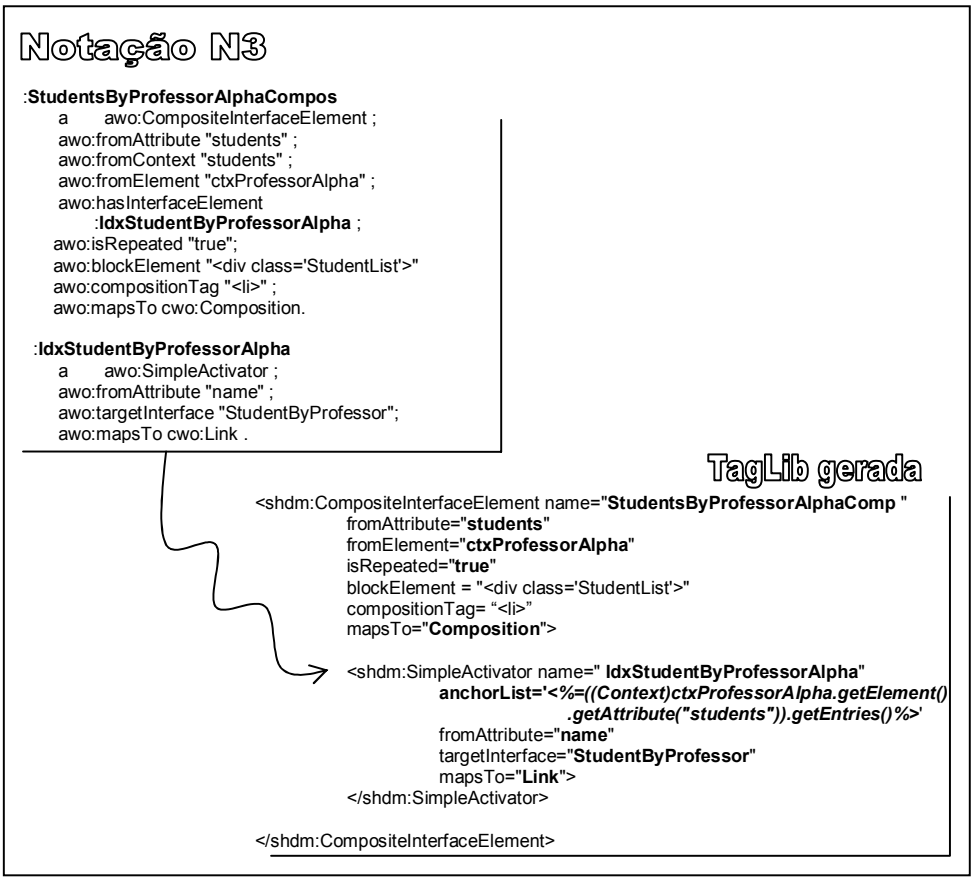


Figura 53 - Notação N3 de um elemento abstrato e a sua respectiva *taglib* gerada.

O sistema gerou a *taglib* "IdxStudentByProfessorAlpha" dentro da definição da *taglib* "StudentsByProfessorAlphaComp", como pode ser visto na figura 53, pois quando foram descritos esses elementos na modelagem abstrata, ilustrada pela notação N3 da figura acima, a propriedade "hasInterfaceElement", do elemento "StudentsByProfessorAlphaComp", indiciou que ele é composto pelo elemento "IdxStudentByProfessorAlpha".

As *taglibs* geradas descrevem os elementos com as suas propriedades, conforme a instância abstrata desses elementos. Entretanto, a *taglib* que descreve o elemento "idxStudentsByProfessorAlpha" possui uma propriedade que não está definida na sua descrição abstrata em N3, e que é denominada como "anchorList". Essa propriedade representa uma chamada ao *bean* "ctxProfessorAlpha", que por

[2] http://www.w3.org/2001/sw/RDFCore/ntriples/

sua vez contém a informação necessária para a criação do elemento concreto. Quando essa *taglib* é executada (o que ocorre apenas no módulo CIR), ela recebe um objeto que foi retornado pelo *bean* como valor da propriedade "anchorList". Esse objeto é passado para a classe *Java* correspondente a essa *taglib*, sendo manipulado por essa classe para a geração do elemento concreto com as informações a serem exibidas para o usuário.

A propriedade "anchorList" foi criada por esse módulo a partir das informações contidas nas propriedades "fromAttribute", "fromElement" e "isRepeated" da modelagem abstrata desse elemento, e da propriedade "fromContext" da modelagem abstrata do elemento "StudentsByProfessorAlphaCompos".

5.1.2. Concrete Interface Renderer (CIR - Gerador de Interface Concreta)

Este módulo é responsável por gerar a página concreta com as informações a serem exibida para o usuário em linguagem HTML. A geração ocorre em tempo de execução, ou seja, o usuário solicita a página concreta e o sistema interpreta o arquivo JSP correspondente à mesma, gerando-a. Primeiramente, o sistema executa o arquivo JSP, gerado pelo módulo AIC, interpretando cada *taglib* descrita nesse arquivo. Cada *taglib* possui uma classe *Java* correspondente, que gera o código concreto em HTML do elemento por ela representado.

Existe uma outra função necessária para a geração da página concreta que, no entanto, não é realizada pelo sistema descrito nessa dissertação: a geração dos objetos *beans*. Esses objetos possuem as informações que serão exibidas pelos elementos concretos e devem, portanto, ficar disponíveis na página JSP para que sejam utilizados pelas *taglibs*. Por esse motivo existe um outro sistema que interage com o módulo CIR, que gera esses objetos *beans* e os disponibiliza na página JSP. Este módulo não é parte do presente trabalho, pois faz parte de outra dissertação descrita em [25].

Na figura 54 pode-se observar a arquitetura que resume as funções deste módulo do sistema.

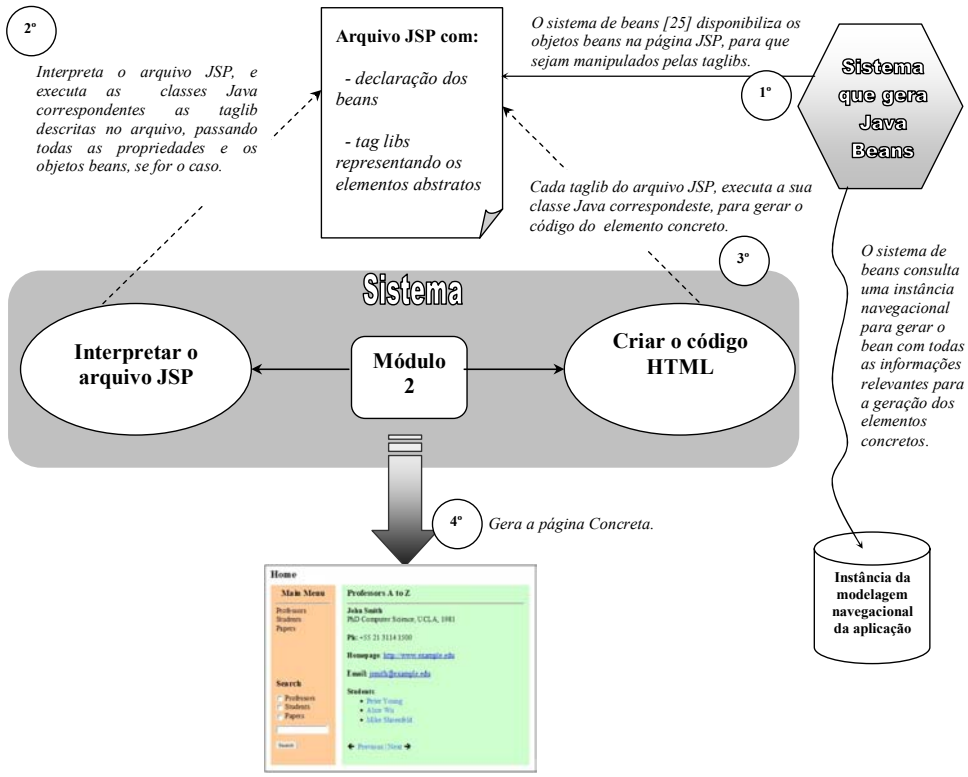


Figura 54 - Arquitetura do módulo do sistema CIR.

Para a geração do *bean,* o sistema descrito em [25] necessita buscar as informações descritas na instância da ontologia navegacional do método SHDM, referente à aplicação. Essa ontologia representa a fase de modelagem navegacional do método SHDM.

Para transformar cada *taglib* descrita no arquivo JSP, em um elemento concreto, com as informações a serem exibidas para o usuário final, este módulo necessita executar procedimentos adequados, que são realizados em tempo de execução. Esses procedimentos são:

- Disponibilizar os objetos *beans* no arquivo JSP;

  Esses objetos são gerados pelo sistema de *beans* e são compostos por informações que poderão ser solicitadas pelas propriedades das *taglibs*.

- Executar cada *taglib* descrita na página JSP;

  Cada modelo de *taglib* possui uma classe *Java* correspondente; desta forma, métodos específicos são executados para cada *taglib.* Esses métodos criam os elemento concretos em linguagem HTML representados pela *taglib*, recebendo o valor das propriedades da *taglib*, que são informações necessárias para a criação do elemento concreto. No caso do valor de uma dessas propriedades for um objeto retornado pelo *bean,* disponível na

página JSP, então a classe *Java* da *taglib* necessitará manipular esse objeto para obter as informações necessárias para a geração do elemento concreto. Por fim, esses métodos adicionam o código do elemento concreto gerado no lugar da descrição de sua *taglib* na página JSP.

- Apresentação da página concreta para o usuário final.

  Este módulo disponibiliza no *browser* a página concreta para o usuário final em linguagem HTML.

5.2. Exemplos

Esta seção apresenta exemplos de geração de elementos concretos. Para essa geração foram descritos 4 passos, em todos os exemplos:

- o elemento concreto a ser gerado;
- a modelagem abstrata em notação N3 desse elemento;
- a *taglib* gerada pelo sistema, correspondente à modelagem abstrata do elemento;
- o código HTML gerado pelo sistema a partir dos dados contidos na *taglib*.

A seguir, serão detalhados alguns dos elementos que compõem a página concreta de exemplo desta dissertação, ilustrada pela figura 19 no capítulo 4. Esses elementos são: o nome do professor, a sua formação e o elemento "Search". Um outro exemplo a ser apresentado é uma lista de nomes dos professores. A interface abstrata usada para descrever essa lista será utilizada para gerar três interfaces concretas diferentes, que representam essa mesma lista. O objetivo é ilustrar que pode-se ter várias interfaces concretas diferentes a partir de uma mesma instância abstrata.

Por fim, será apresentado um elemento mais complexo do que aqueles já explicados e descritos nos capítulos anteriores. Esse elemento é uma página da Web conhecida como "CSS Zen Garden" (http://www.csszengarden.com). Neste site são apresentados vários *layouts* para uma mesma página html. O objetivo é demonstrar que com a proposta dessa dissertação, e utilizando-se a tecnologia de CSS, consegue-se descrever interfaces complexas, de padrão industrial, geradas a partir de uma mesma modelagem abstrata. Esse terceiro exemplo comprova que essa proposta não é necessariamente dependente de um modelo de projeto, pois a

modelagem da interface desta aplicação foi feita sem utilizar um método particular de modelagem.

Para facilitar a leitura, reproduzimos aqui o exemplo de interface, descrito na figura 19 do capítulo 4.

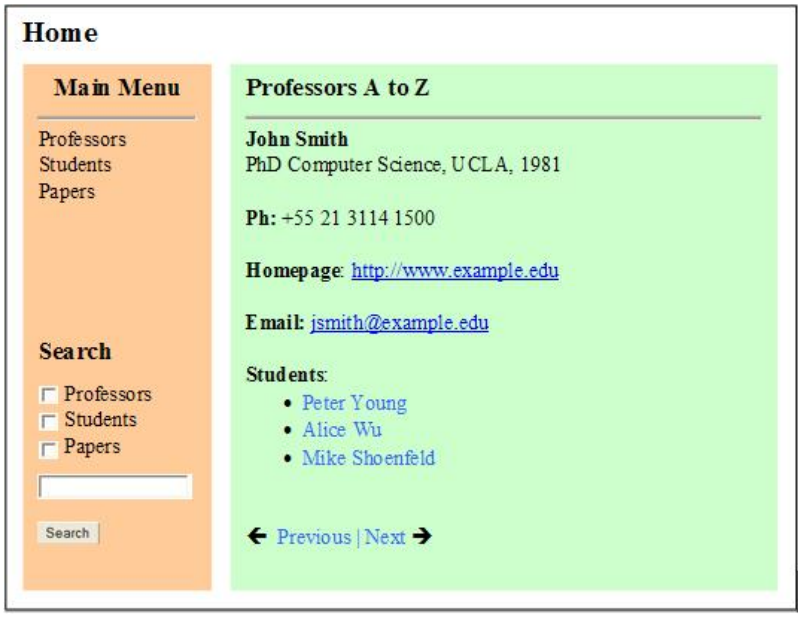


Figura 55 - (Figura 19 do capítulo 4) – Um exemplo de Interface Concreta.

5.2.1. Nome do Professor e sua Formação

Será apresentado, passo a passo, como se faz a geração de um elemento concreto, desde a sua modelagem abstrata até o seu código HTML. Os elementos que serão descritos apresentam, respectivamente, o nome e a formação do professor referente à interface de exemplo da figura 55. Esses elementos concretos são ilustrados na figura abaixo.

John Smith
PhD Computer Science, UCLA, 1981

Figura 56 - Elementos concretos: “Nome” e “Formação” do Professor.

A seqüência de passos, desde a modelagem abstrata até o elemento concreto da figura 56 é representada na figura 57. Primeiramente, a instância abstrata de dois elementos, do tipo “ElementExhibitor”, é descrita em notação N3; e logo após a *taglib*, gerada pelo módulo AIC do sistema, e por fim o código HTML desses elementos gerado pelo módulo CIR.

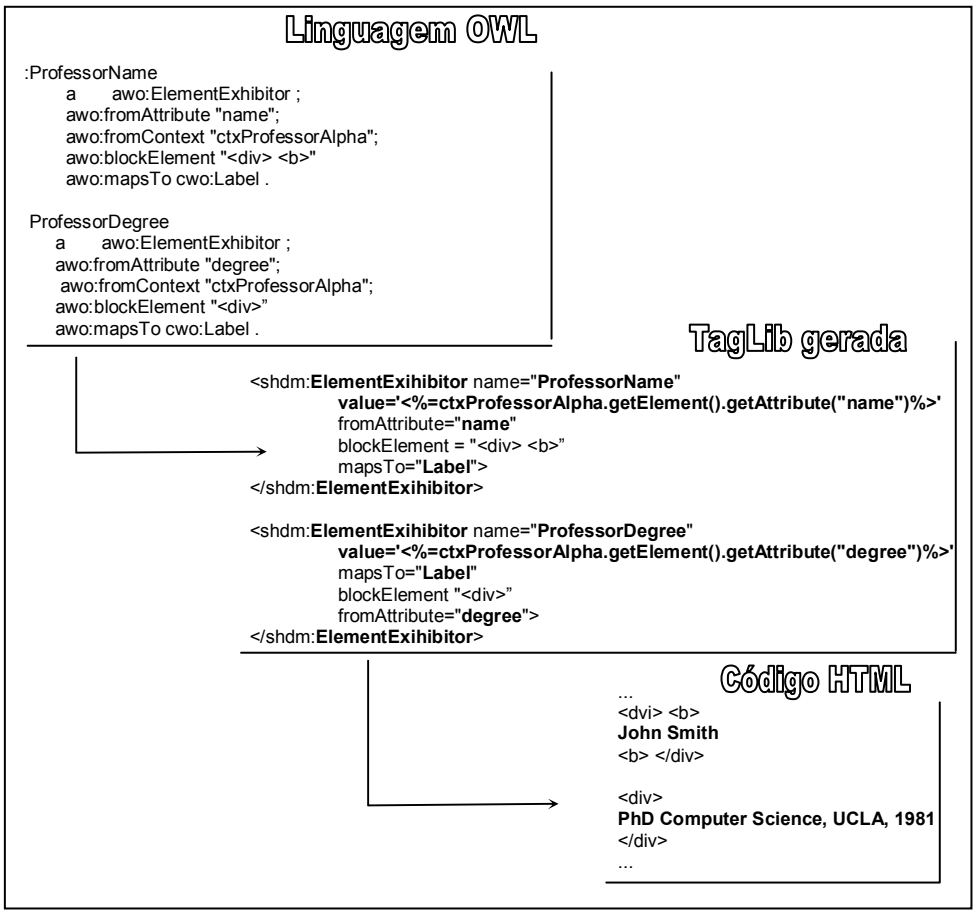


Figura 57 - Seqüência de Código para gerar os elemento concreto: “nome” e “formação” do professor.

No código da *taglib* dos elementos descritos na figura acima, o módulo AIC do sistema gerou uma outra propriedade que não estava descrita no código em N3. Essa propriedade recebeu o nome de “value” e foi gerada pelo sistema a partir das propriedades: “fromAttribute” e fromContex, descrita no código em N3 desses elementos. A propriedade “value” irá receber um objeto do *bean* “ctxProfessorAlpha”, que contém a informação a ser exibida por cada um desses elementos. Esta informação está descrita na ontologia navegacional [25] através dos atributos “nome” e “degree”.

5.2.2. Elemento “Search”

Esta seção apresentará a geração do elemento concreto “Search”, desde a sua modelagem abstrata até o seu código HTML. Esse elemento compõe a interface concreta do exemplo, ilustrada na figura 55. Ele representa um elemento de busca, ou seja, de pesquisa sobre “*Professors*”, “*Students*” e “*Papers*”. A Figura 58 apresentada esse elemento concreto.

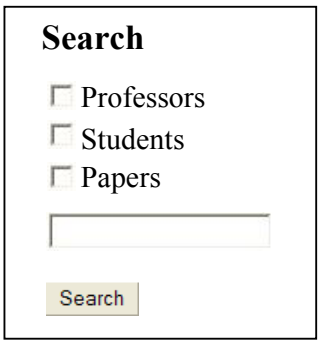


Figura 58 - Elemento Concreto “Search”.

Como esse elemento é composto por outros elementos - tornando assim a sua definição mais extensa - a sua instância abstrata, as *taglibs* e o código HTML final serão apresentados separadamente. Na figura 59 ilustra-se a instância abstrata desse elemento em notação N3.

```
:Search
   a       awo:CompositeInterfaceElement ;
   awo:fromIndex "idxSearch" ;
   awo:isRepeated "false" ;
   awo:hasInterfaceElement
         :TitleSearch, :SearchElements;
   awo:blockElement “<div class='search'>”;
   awo:mapsTo cwo:Composition .

 :TitleSearch
    a    awo:ElementExhibitor;
    awo:visualizationText “Search”;
    awo:blockElement “<div> <b>”;
    awo:mapsTo cwo:Label .

:SearchElements
   a       awo:CompositeInterfaceElement ;
   awo:fromIndex "idxSearch" ;
   awo:hasInterfaceElement
         :SearchProfessors, :SearchStudents, :SearchPapers, :SearchField;
   awo:isRepeated "false" ;
   awo:targetInterface "SearchResult";
    awo:blockElement “<div>”;
   awo:mapsTo cwo:Form .

:SearchProfessors
   a       awo:MultipleChoices ;
   awo:fromAttribute "section" ;
   awo:fromElement "SearchProfessors" ;
    awo:blockElement “<div>”;
   awo:mapsTo cwo:CheckBox.

:SearchStudents
   a       awo:MultipleChoices ;
   awo:fromAttribute "section" ;
   awo:fromElement "SearchProfessors" ;
    awo:blockElement “<div>”;
   awo:mapsTo cwo:CheckBox.

:SearchPapers
   a       awo:MultipleChoices ;
   awo:fromAttribute "section" ;
   awo:fromElement "SearchProfessors" ;
     awo:blockElement “<div>”;
   awo:mapsTo cwo:CheckBox .
```

Figura 59 - Instância abstrata do elemento concreto “Search”.

O módulo AIC, ao interpretar a instância da ontologia de *widgets* abstratos descrita na figura 59, gera as *taglibs* correspondente a essa instância. Essas *taglibs* geradas podem ser visualizadas na figura 60.

```
<shdm:CompositeInterfaceElement name="Search"
        isRepeated="false"   mapsTo="Composition"
        fromIndex="idxSearch"
        blockElement="<div class='search'>" >

        <shdm:ElementExihibitor name="TitleSearch"
                mapsTo="Label"
                visualizationText="Search"
                blockElement="<div> <b>">
        </shdm:ElementExihibitor>

        <shdm:CompositeInterfaceElement name="SearchElements"
                visualizationText="Search"   mapsTo="Form"
                isRepeated="false"   abstractInterface="SearchResult"
                blockElement="<div>">

                <shdm:MultipleChoices name="SearchProfessors"
                        element='<%=(Anchor)idxSearch.getEntry("SearchProfessors").getAttribute("section")%>'
                        fromElement="SearchProfessors"   fromAttribute="section"
                        mapsTo="CheckBox" blockElement="<div>">
                </shdm:MultipleChoices>

                <shdm:MultipleChoices name="SearchStudents"
                        element='<%=(Anchor)idxSearch.getEntry("SearchStudents").getAttribute("section")%>'
                        mapsTo="CheckBox"   fromElement="SearchStudents"
                        fromAttribute="section" blockElement="<div>">
                </shdm:MultipleChoices>

                <shdm:MultipleChoices name="SearchPapers"
                        element='<%=(Anchor)idxSearch.getEntry("SearchPapers").getAttribute("section")%>'
                        mapsTo="CheckBox"   fromElement="SearchPapers"
                        fromAttribute="section" blockElement="<div>">
                </shdm:MultipleChoices>

                <shdm:IndefiniteVariable name="SearchField"
                        mapsTo="TextBox"  blockElement="<div>">
                </shdm:IndefiniteVariable>

        </shdm:CompositeInterfaceElement>
</shdm:CompositeInterfaceElement>
```

Figura 60 - *Taglibs*, geradas pelo módulo AIC, que descrevem o elemento "Search".

O módulo CIR, do sistema, quando interpreta as *taglibs* descrita na figura 60, gera o código HTML correspondente a cada elemento descrito por elas. Esse código HTML gerado é ilustrado na figura 61.

```
...
1   <div class='search'>
2   <form method="GET" action="SearchResult.awo">
3
4       <div> <b>
5       <input type="checkbox" value="Professors" name="idxProfessors">Professors</p>
6       </b> </div>
7
8       <div>
9       <input type="checkbox" value="Students" name="idxStudents">Students</p>
10      </div>
11
12      <div>
13      <input type="checkbox" value="Papers" name="idxPapers">Papers</p>
14      </div>
15
16      <div>
17      <input type="text" name="SearchField" size="20">
18      </div>
19
20      <input type="submit" value="Search" name="Button">

21  </form>
22  </div>
```

Figura 61 - Código HTML, gerado pelo módulo CIR, do elemento "Search".

Para gerar o código HTML descrito na figura 61, o módulo CIR interpreta cada *taglib,* executa a sua classe *Java* correspondente, passando todas as

informações descritas nessa *taglib*. A classe *Java* de cada *taglib* manipula as informações recebidas e gera o código HTML do elemento no arquivo JSP.

5.2.3. Lista de Professores

Nesta seção será apresentada uma modelagem de interface abstrata que representa uma lista de nome de professores. Essa modelagem será utilizada para gerar três interfaces concretas diferentes, alterando apenas o valor da propriedade "mapsTo", que indica o elemento concreto a ser gerado, na modelagem abstrata. A primeira interface concreta a ser gerada será uma lista de *links*, onde cada *link* representa um nome de um professor. Na figura 62 pode-se visualizar essa interface concreta.

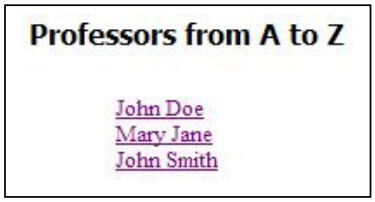


Figura 62 - Elemento concreto: Lista de l*inks*.

A interface abstrata, descrita em notação N3, do elemento concreto apresentado na figura 62 é ilustrada na figura 63. Nesta figura, o elemento de interface "IdxProfessorsEntries" do tipo "CompositeInterfaceElement" é composto pelo elemento "IdxProfessorsName", que é do tipo "SimpleActivator". Isto é indicado pela propriedade "hasInterfaceElement", que contém o nome do elemento que compõe o elemento "IdxProfessorsEntries". A propriedade "isRepeated" possui o valor *true* nessa definição, indicando que todos os elementos (neste caso apenas o "SimpleActivator") que compõem o elemento "IdxProfessorsEntries", poderão ser repetidos tantas vezes quanto necessário.

```
: IdxProfessorsEntries
    a   awo:CompositeInterfaceElement;
    awo:fromIndex "idxProfessors";
    awo:targetInterface "ProfessorAlpha";
    awo:isRepeated "true";
    awo:compositionTag "<br>"
    awo:mapsTo cwo;Composition;
    awo:hasInterfaceElement
            : IdxProfessorsName;
     awo:blockElement "<div>".

: IdxProfessorsName
    a   awo:SimpleActivator;
    awo:fromElement "idxProfessors";
    awo:fromAttribute "name";
    awo:mapsTo cwo;Link.
```

Figura 63 - Instância abstrata do elemento concreto "Lista de *links*".

A figura 64 apresenta as *taglibs* geradas pelo módulo AIC, correspondentes a modelagem ilustrada na figura 63. A *taglib* do tipo "SimpleActivator", descrita na figura 64, contém uma propriedade que não estava declarada na modelagem em notação N3. Essa propriedade representa uma chamada a um *bean* cujo valor será um objeto que será manipulado pela classe *Java* correspondente a essa *taglib*, para que seja gerada a lista de *links* com os nomes dos professores.

```
<shdm:CompositeInterfaceElement name="IdxProfessorsEntries"
        isRepeated="true"
        compositionTag="<br>"
        mapsTo="Composition"
        fromIndex="idxProfessors"
        targetInterface="ProfessorAlpha"
        blockElement="<div>">

        <shdm:SimpleActivator name="IdxProfessorsName"
              anchorList='<%=idxProfessors.getEntries()%>'
              mapsTo="Link"
              fromElement="idxProfessors"
              fromAttribute="name">
        </shdm:SimpleActivator>

</shdm:CompositeInterfaceElement>
```

Figura 64 - *Taglibs,* geradas pelo módulo AIC, do elemento concreto "Lista de *lnks*".

Na figura 65 pode ser visualizado o código HTML, gerado quando o módulo CIR do sistema executou as *taglibs* da figura 64. Como pode ser visto, a *taglib* do tipo "SimpleActivator" gerou uma lista de *links* composta por três elementos, que representam os nomes dos professores.

```
<div>

<br><a href="ProfessorAlpha.awo?ctxProfessorAlpha_ID=johndoe">John Doe</a>
<br><a href="ProfessorAlpha.awo?ctxProfessorAlpha_ID=maryjane">Mary Jane</a>
<br><a href="ProfessorAlpha.awo?ctxProfessorAlpha_ID=johnsmith">John Smith</a>

</div>
```

Figura 65 - Código HTML, gerado pelo módulo CIR, do elemento concreto da figura 62.

A segunda interface concreta a ser gerada é uma lista de "*Radio Button"*, onde cada "*Radio Button"* representa o nome de um professor. A figura 66 apresenta essa interface concreta.

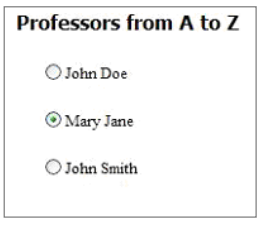


Figura 66 - Elemento concreto: Lista de *Radio Button*.

A figura 67 demonstra a instância abstrata, em notação N3, desse elemento concreto. Nela está descrito o elemento de interface composto "IdxProfessorsEntries", do tipo "CompositeInterfaceElement", do mesmo modo

como está descrito na figura 63. Entretanto, o elemento "IdxProfessorsName", que compõe o elemento composto, neste caso, é do tipo "SingleChoice" e não "SimpleActivator". O elemento "SingleChoice" possui as mesmas propriedades que o elemento "SimpleActivador" da figura 63, sendo a única diferença o elemento no qual eles estão mapeados; neste caso, para um *RadioButtonTarget*, e no caso do "SingleChoice", para um *link*. O *RadioButtonTarget* representa um ou mais elementos do tipo *Radio Button*, que, ao ser selecionado, dispara a execução de uma ação, neste caso, a ativação da interface correspondente ao professor selecionado.

```
: IdxProfessorsEntries
    a   awo:CompositeInterfaceElement;
    awo:fromIndex "idxProfessors";
    awo:targetInterface "ProfessorAlpha";
    awo:isRepeated "true";
    awo:compositionTag "<p>"
    awo:mapsTo cwo;Composition;
    awo:hasInterfaceElement
            : IdxProfessorsName;
     awo:blockElement "<div>".

 : IdxProfessorsName
    a   awo: SingleChoice;
    awo:fromElement "idxProfessors";
    awo:fromAttribute "name";
    awo:mapsTo cwo;RadioButtonTarget.
```

Figura 67 - Instância abstrata do elemento concreto "Lista de *Radio Button"*.

As *taglibs*, geradas pelo módulo AIC e correspondente ao código ilustrado na figura 67, podem ser visualizada pela figura abaixo. A *taglib* que descreve o elemento do tipo "SingleChoice" também possui uma propriedade que não esta declarada na sua definição em N3, assim como a *taglib* que descreve o elemento do tipo "SimpleActivator" da figura 64. Essa propriedade representa uma chamada a um *bean*, que já explicada anteriormente.

```
<shdm:CompositeInterfaceElement name="IdxProfessorsEntries"
        isRepeated="true"
        compositionTag="<p>"
        mapsTo="Composition"
        fromIndex="idxProfessors"
        targetInterface="ProfessorAlpha"
        blockElement="<div>">

        <shdm: SingleChoice name="IdxProfessorsName"
            elemList='<%=idxProfessors.getEntries()%>'
            mapsTo=" RadioButtonTarget"
            fromElement="idxProfessors"
            fromAttribute="name">
        </shdm: SingleChoice >

</shdm:CompositeInterfaceElement>
```

Figura 68 - *Taglibs*, geradas pelo módulo AIC, do elemento concreto: "Lista de *radio Button*".

O código HTML, gerado pelo módulo CIR e correspondente as *taglibs* descritas na figura 68, é ilustrado na figura 69. A *taglib* do tipo "SingleChoice"

gerou uma lista de "*radio button"*, composta por três elementos que representam os nomes dos professores. Cada um desses elementos possui uma função em *Java Script,* que executa uma ação específica no momento em que ele for selecionado. A *taglib* "CompositeInterfaceElement" gerou em um elemento concreto do tipo "Form".

```
<div>
<form method="GET" action="ProfessorAlpha.awo">

<p><input type="radio" value="johndoe" name="ctxProfessorAlpha_ID"
          onclick="window.location='johndoe.awo';">John Doe</p>
<p><input type="radio" value="maryjane" name="ctxProfessorAlpha_ID"
          onclick="window.location= 'maryjane.awo';">Mary Jane</p>
<p><input type="radio" value="johnsmith" name="ctxProfessorAlpha_ID"
          onclick="window.location= 'johnsmith.awo';">John Smith</p>
</form>
</div>
```

Figura 69 - Código HTML, gerado pelo módulo CIR, do elemento concreto da figura 66.

A terceira interface concreta a ser gerada é composta por uma lista de nomes de professores, representada por um elemento do tipo *Combo Box*. Na figura 70 ilustra-se essa interface concreta.

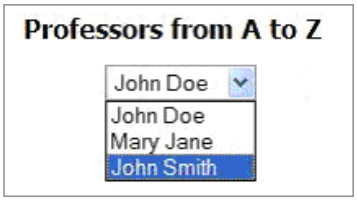


Figura 70 - Elemento concreto: Combo Box.

Na figura 71 tem-se a descrição da modelagem abstrata, em notação N3, do elemento concreto da figura 70. Nesta figura, o elemento "IdxProfessorsEntries", do tipo "CompositeInterfaceElement", está modelado da mesma forma como nos elementos anteriores. Entretanto, o elemento "IdxProfessorsName" que o compõe é do tipo "SingleChoice", como apresentado na figura 67. A única diferença entre a definição desse SingleChoice para a descrita na figura 67 é o valor da propriedade "mapsTo", pois neste último caso, o elemento foi mapeado para um "*ComboBoxTarget*" e na definição anterior para um "*RadioButtonTarget*".

```
: IdxProfessorsEntries
    a   awo:CompositeInterfaceElement;
    awo:fromIndex “idxProfessors”;
    awo:targetInterface “ProfessorAlpha”;
    awo:isRepeated “true”;
    awo:compositionTag "<br>"
    awo:mapsTo cwo;Composition;
    awo:hasInterfaceElement
            : IdxProfessorsName;
     awo:blockElement “<div>”.

: IdxProfessorsName
    a   awo: SingleChoice;
    awo:fromElement “idxProfessors”;
    awo:fromAttribute “name”;
    awo:mapsTo cwo; ComboBoxTarget.
```

Figura 71 - Instância abstrata do elemento concreto “Combo Box”.

As *taglibs* geradas pelo módulo AIC, correspondente ao código apresentado na figura 71, estão ilustradas na figura 72. A propriedade “elemList”, da *taglib* do tipo “SingleChoice”, descrita na figura 72, não foi definida na modelagem em N3 desse elemento, pois essa propriedade representa uma chamada a um *bean*.

```
<shdm:CompositeInterfaceElement name="IdxProfessorsEntries"
        isRepeated="true"
        mapsTo="Composition"
        fromIndex="idxProfessors"
        targetInterface="ProfessorAlpha"
        blockElement=”<div>”>

        <shdm: SingleChoice name="IdxProfessorsName"
            elemList='<%=idxProfessors.getEntries()%>'
            mapsTo="ComboBoxTarget"
            fromElement="idxProfessors"
            fromAttribute="name">
        </shdm: SingleChoice >

</shdm:CompositeInterfaceElement>
```

Figura 72 - *Taglibs*, geradas pelo módulo AIC, do elemento concreto: “Combo Box”.

O código HTML, gerado pelo módulo CIR e correspondente as *taglibs* descritas na figura 72, é apresentado na figura 73. A *taglib* “CompositeInterfaceElement” gerou o elemento concreto do tipo “*Form*” e a “SingleChoice” gerou o elemento concreto do tipo “*Combo Box*”. O “*Combo Box*” é composto por uma lista de nomes de professores, que ao selecionar um item desta lista, uma função em *Java Script* é executada

```
<div>
<form method="GET" action="ProfessorAlpha.awo">

<script>
  function redireciona(value)
  {
      window.location=value;
  }
</script>

<select size="1" name="ctxProfessorAlpha_ID" onchange="redireciona(this.value);">
            <option value="johndoe.awo">John Doe</option>
            <option value="maryjane.awo">Mary Jane</option>
            <option value="johnsmith.awo">John Smith</option>

</form>
</div>
```

Figura 73 - Código HTML, gerado pelo módulo CIR, do elemento concreto da figura 70.

5.2.4.
CSS Zen Garden

Esta seção descreve a interface abstrata de uma página da Web disponível na URL http://www.csszengarden.com, chamada de "CSS Zen Garden". Essa página representa uma interface de complexidade muito maior do que os exemplos apresentados anteriormente. O objetivo deste site é demonstrar o poder da tecnologia CSS, aplicando *designs* baseados neste, utilizando uma única página HTML. A página exemplo é típica daquelas encontradas em sites comerciais na WWW, e está incluída no Anexo.

Um dos objetivos de realizar a modelagem de interface e geração dessa página, utilizando o sistema desenvolvido nessa dissertação, é comprovar que com o uso da tecnologia CSS em conjunto com a modelagem abstrata, consegue-se obter diversos *layouts* concretos, de padrão industrial, partindo-se de uma mesma interface abstrata.

O outro objetivo é mostrar que, apesar desta dissertação apresentar a modelagem abstrata como parte do método SHDM, a descrição de interfaces proposta não dependente deste método, visto que a página em questão foi projetada de forma "ad hoc".

A figura 74 apresenta o código, em notação N3, do primeiro nível da modelagem abstrata dessa página. Esse código representa um elemento que é composto por todos os elementos que constituem essa página.

```
:csszengardenInterface
    a       awo:AbstractInterface ;
    awo:hasInterfaceElement
            :CssZenGardenTitle , :2oSubTitulo , :1oSubTitulo,
            :3oSubTitulo , :4oSubTitulo , :5oSubTitulo ,
            :6oSubTitulo , :lista_resources , :lista_estilos , :lista_archives ;
    awo:mapsTo cwo:Composition .
```

Figura 74 – Modelagem abstrata do primeiro nível do Css Zen Garden.

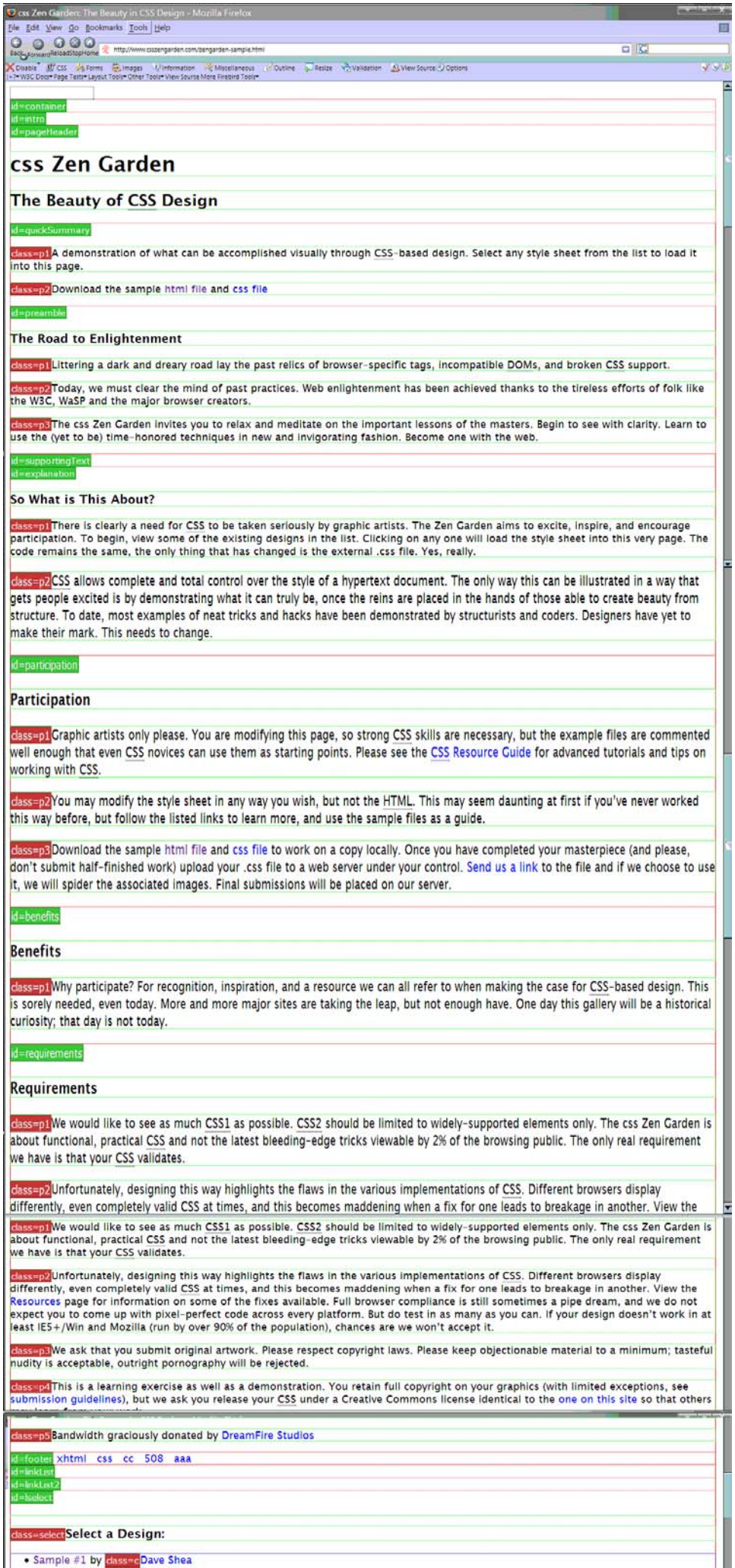
A figura 75 apresenta a modelagem abstrata do elemento "5oSubTitulo" em notação N3, que compõem o elemento "csszengardenInterface", descrito na figura 74.

```
:5oSubTitulo
    a       awo:ElementExihibitor ;
    awo:mapsTo cwo:Label ;
    awo:visualizationText
     """
              <h3 xmlns=\"http://www.inf.puc-rio.br/~sabrina/ontology/IA/csszengardenSimples.owl#\">
              <span>Benefits</span></h3>
              <p xmlns=\"http://www.inf.puc-rio.br/~sabrina/ontology/IA/csszengardenSimples.owl#\"
              class=\"p1\"><span>Why participate? For recognition, inspiration, and a resource we can
              all refer to when making the case for <acronym title=\"Cascading Style Sheets\">CSS
              </acronym>-based design. This is sorely needed, even today. More and more major sites
              are taking the leap, but not enough have. One day this gallery will be a historical
              curiosity; that day is not today.</span></p>
     """ rdf:XMLLiteral .
```

Figura 75 – Modelagem abstrata do quinto parágrafo do Css Zen Garden.

A modelagem abstrata completa, em notação N3, dessa página está disponível no item 1 do Anexo B. A partir dessa modelagem, a ferramenta proposta nessa dissertação gera a interface concreta. Primeiramente o módulo AIC gera o arquivo JSP com as *taglibs*; esse arquivo está descrito no item 2 e o código HTML gerado pelo módulo CIR está descrito no item 3 do Anexo B.

A figura 76 apresenta um trecho da página concreta do "CSS Zen Garden" sem *layout*, para possibilitar a visualização dos elementos que compõem essa interface, no qual foram ressaltados os elementos CSS. Em seguida, 3 exemplos dessa mesma interface são apresentados; esses exemplos foram gerados pela ferramenta dessa dissertação, utilizando *designs* diferenciados baseados em CSS, definidos em folhas de estilo obtidas naquele site. As figuras 77, 78 e 79, apresentam esses 3 exemplos gerados. Para a geração desses *layouts*, a mesma interface abstrata foi utilizada, apenas foi acrescentado um arquivo CSS que é responsável pelo *design* diferenciado das páginas.



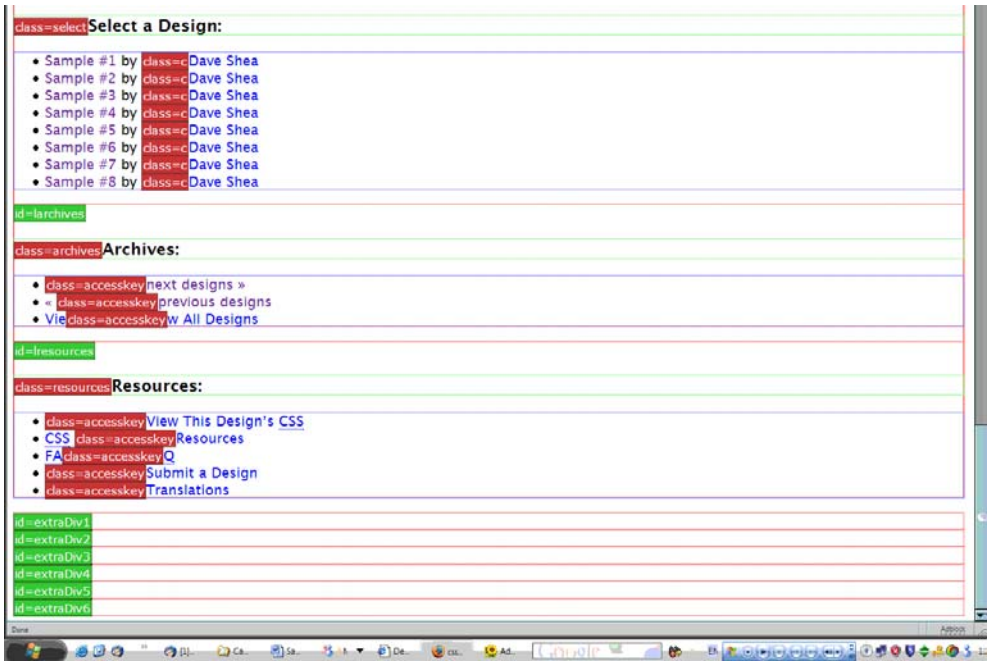


Figura 76 - Trecho da página concreta sem *layout* do site “CSS Zen Garden”, com os elementos CSS delineados.

A figura 77 ilustra a mesma página descrita pela figura 76, mas com uma folha de estilo CSS. O resultado dessa combinação pode ser visualizado na figura abaixo.

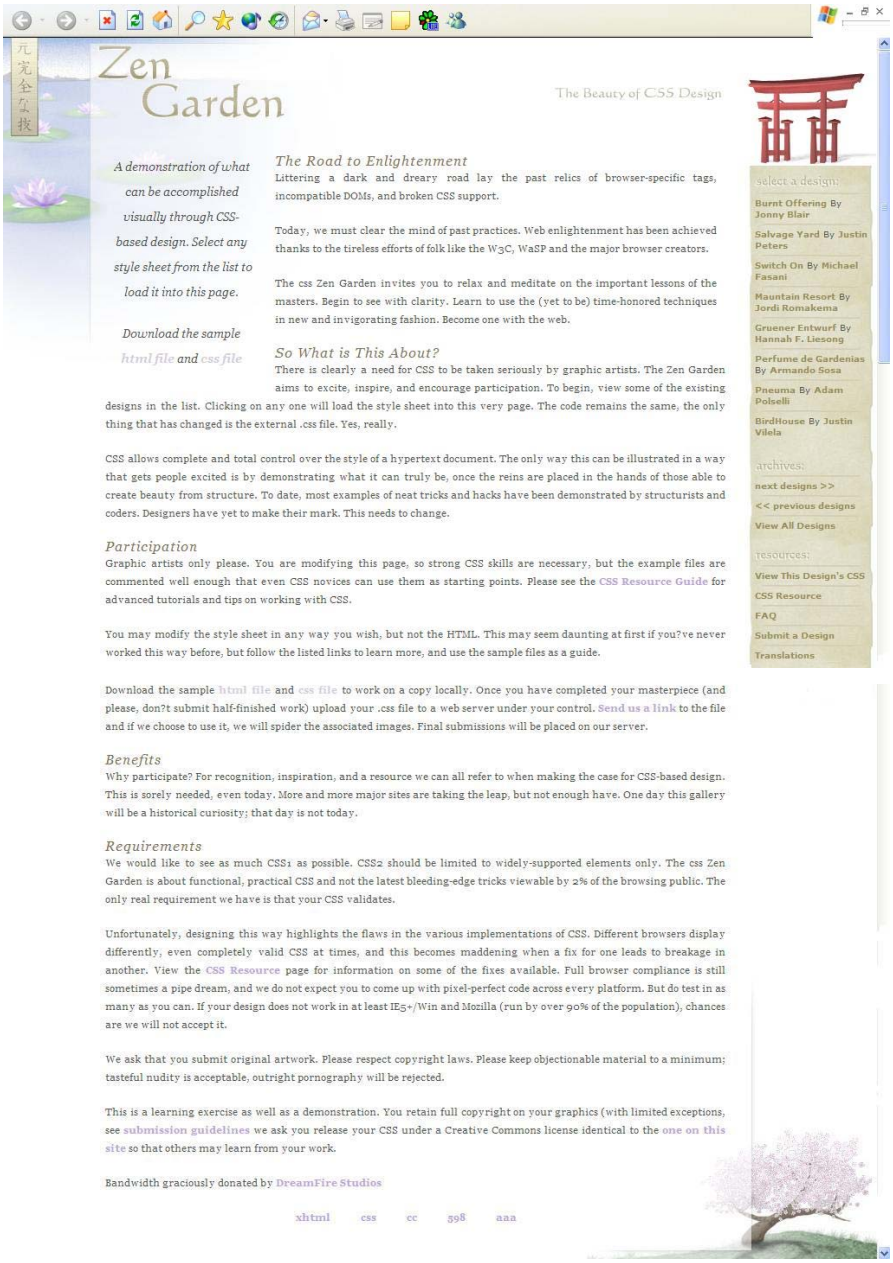


Figura 77 - Página concreta do site “CSS Zen Garden” utilizando o *design* “BirdHouse”.

A figura 78 apresenta a mesma página descrita pelas figuras 76 e 77, utilizando uma outra folha de estilo CSS. O resultado dessa combinação é completamente diferente do resultado descrito pela figura 77.

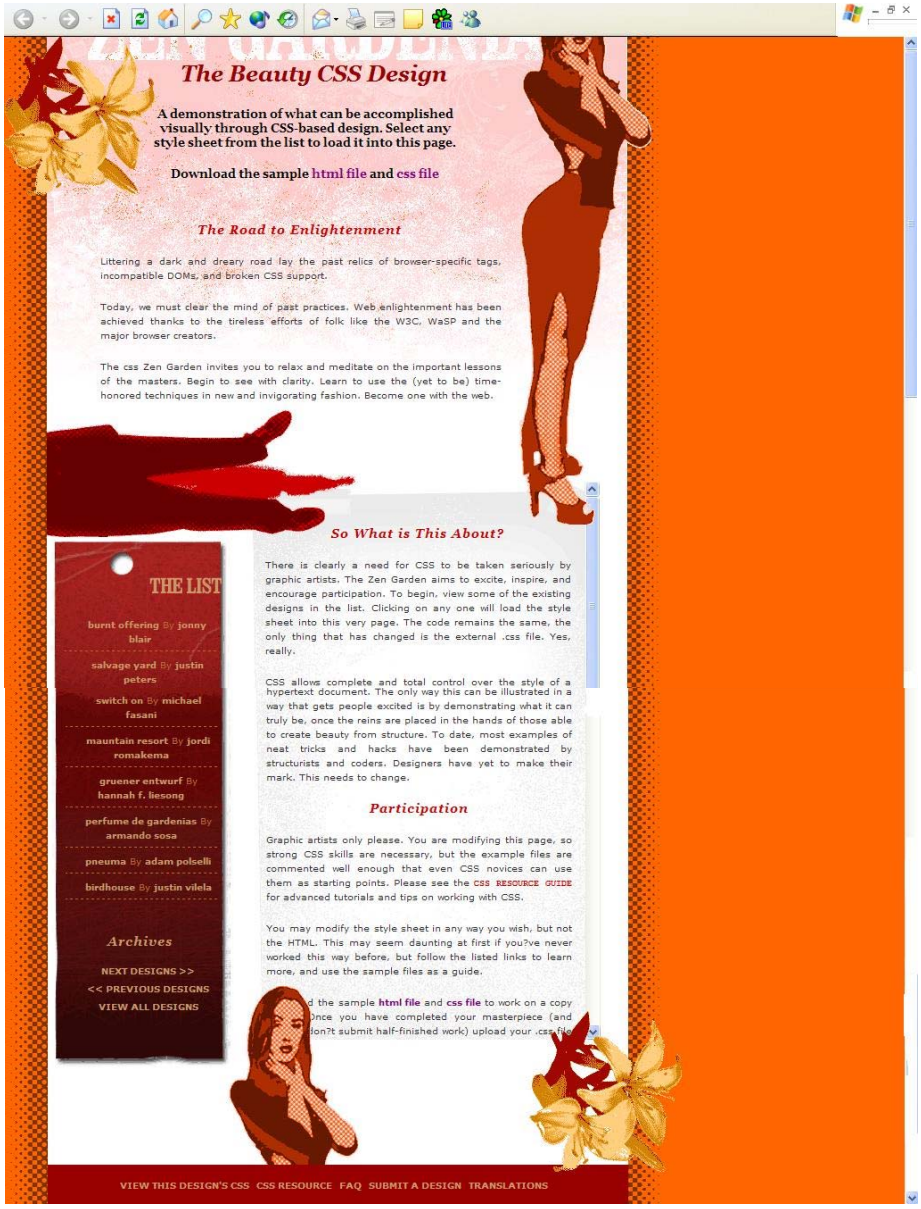


Figura 78 - Página concreta do site “CSS Zen Garden” utilizando o *design* “Perfume de Gardenias”.

Na figura 79 é ilustrada a mesma página, mas com uma visualização completamente diferente das apresentadas nas figuras anteriores, pois os elementos estão situados horizontalmente na página. É interessante observar que o conteúdo da página é sempre o mesmo, o que muda é a forma como ele é apresentado.

Figura 79 - Página concreta do site “CSS Zen Garden” utilizando o *design* “Grüener Entwurf”.