

# **AUTOMATIC CREATION OF USER INTERFACES FOR INFORMATION SYSTEM**

## **ABSTRACT**

Very actual task for the information system is automatically user interface creating. Solution of this task greatly decreases the information system development time. There are various approaches for automatic creation of user interfaces. In this article most popular approaches are surveyed and their problems are described. Author suggests the hybrid approach which allows minimizing the problems of existing approaches.

## **KEYWORDS**

user interfaces, code generation, information systems, databases, objects inheritance

## **1. INTRODUCTION**

The programming of some kinds of applications from scratch is not reasonable. The continuous extension of program libraries and frameworks and is widely used. The libraries are designed to solve tasks that are isolated from general applications logic. That's why approaches like a solution of general tasks with complex configuration and a code generation is also popular.

The user interfaces (UI) creation is the one of the tasks when programming from scratch is unreasonable. The rich UI libraries solve only part of the task. Some meta-information that may be helpful to create UI already exists as the rule. It may be database structure for example. On the one hand, UI very frequently needs some project specific features. On other hand, many UI parts are evidently reasonable for automatic creation. The problem is to develop approach which allows to automatically creating UI which can be extended by project specific features.

The various approaches of the automatic creation of UI for information systems currently exist. Two approaches are the most popular. The first approach uses the program code generation which can be modified by programmer in a future. The second approach doesn't use a manual code correction by a programmer. All the settings which is needed for UI customization are included into meta-information in this approach. Let's consider these approaches in details.

## **2. EXISTING APPROACHES**

The code generation based on the meta-information is popular practice. This approach is the automation of the part the process which programmers are making manually in other case. When programmer has the task to develop UI for some object he should already has encountered with all parts of this task in some degree. He will copy the parts of a code from his old projects, correct identifiers and do some other correcting to make it working together. Then first version will be done. When we use the code generation most part of these actions is executed automatically but the manual correction of the generated code is needed also (Figure 1). The automatic code generation eliminates the nasty errors from the lack of attention. The problem of this approach is a lot of iterations that development process has usually. These iterations may contain meta-information correction. Used for the code generating the meta-information may changes also. Then there is question how to transfer these changes to already generated and corrected by a programmer code. We can generate the UI code again and manually correct it again (Figure 2). Otherwise we can manually correct UI code due to meta-information changes (Figure 3). In any case additional manually correction is needed and this correction has a not minimal volume.

Figure 1. The scheme of the automatic UI code generation

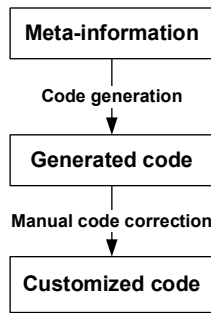


Figure 2. The scheme of the automatic UI code generation for meta-information changing (the first alternative)

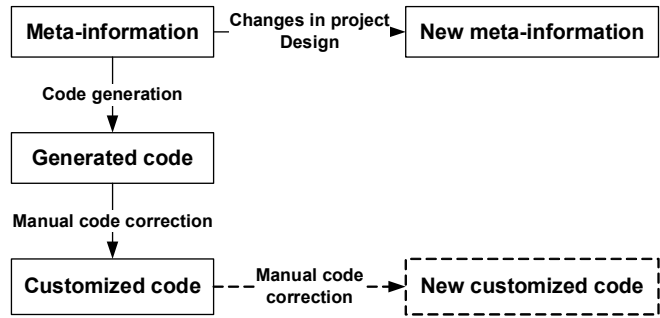
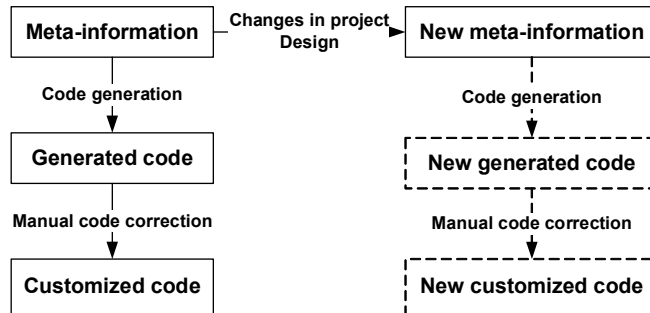


Figure 3. The scheme of the automatic UI code generation for meta-information changing (the first alternative)



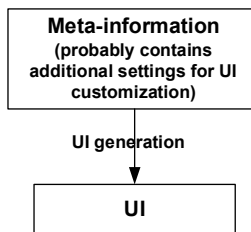
The second approach doesn't use the manual code correction and includes all information needed for the UI configuration into meta-information. In this approach we have not problem of transferring changes from a previous approach because the manual generated code correction is not used there.

Databases administration tools use the simple implementation of this approach. These tools provide a viewing and editing interface for each table of a database. The more complex example is the administration generator of the Symfony PHP Framework which uses the various settings for the UI customization (Symfony Open-Source PHP Web Framework, 2009).

In this approach any specific UI customizations must be covered by the meta-information and the generator possibilities. The meta-information and the generator extension are needed for each specific UI customization.

The most serious problem of this approach is the frequently need of reprogramming the UI generator. It may produce the new generator functions to be appropriate to specific project needs but doesn't solve the general problems. In this case the generator design may suffer.

Figure 4. The scheme UI generation when all UI customization settings are included into meta-information



### 3. PROPOSED APPROACH

Author proposes the hybrid approach when the generated code and the manual corrections are logically separated. When the meta-information is changed than the generated code will be generated again but existing manual corrections do not require any changes or these changes would be minimal. To implement this approach the two tasks should be solved:

- 1) A code generator should be flexible enough to generate the application with the structure which doesn't require the changes during the customization. Because it's very difficult to implement logically separation of manual corrections which include the application restructure.
- 2) The mechanism of separation of generated application and manual changes should be found.

For the separation of the generated application and the manual changes the inheritance mechanism is proposed. In this approach for each automatically created UI component the two classes are generated (Figure 5). The first class is automatically generated UI component. The second class inherits the first class and it is empty initially. Programmer can manually fulfill the second class for UI component customization. It's important to lay into the UI component logic in which the changes introduced into the derived class will be the most correctly isolated. Then the minimum vulnerability with respect to changes in base class (this means the minimum vulnerability with respect to changes in meta-information) will be achieved. If such mechanism will be develop then when the meta-information is changed then the base class will be regenerated but derived class do not require any changes or these changes will be minimal and dealing with the manually added features (Figure 6). This approach is very similar to popular approach in the Object Relation Mapping software where base classes of persistent objects are generated and derived classes can be manually customized (Propel ORM PHP framework, 2009).

Figure 5. The scheme of the hybrid approach of a UI generation

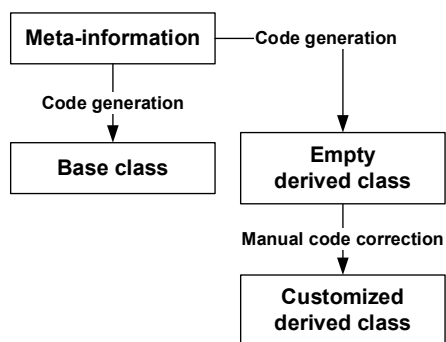
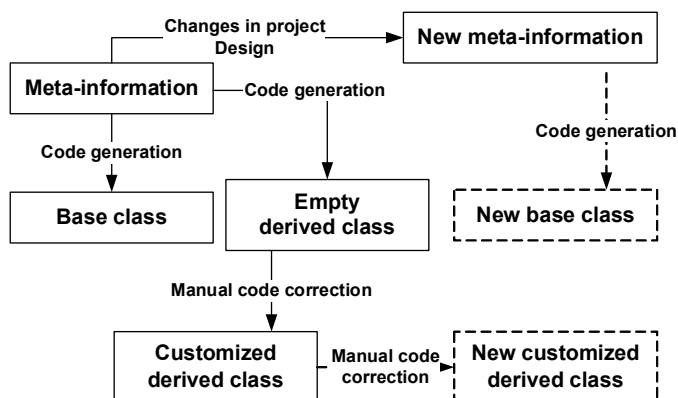


Figure 6. The scheme of the hybrid approach of a UI for meta-information changing



Let's consider how this approach can be applied to the very usual UI components of information system such as the form and the grid. Let the form generator for each data field generate a form field. Let this generator use extended meta-information about form field layout.

A UI form must be decomposed into parts which can be redefinable in a derived class. Let's consider possibilities to redefine these parts in a derived class:

- 1) Fields layout  
Fields layout is desirable to be fully defined in extended meta-information. In other case, changes contributor is fully responsible for the maintenance of the changes in the meta-information.
- 2) Fields configuration  
There are several possibilities to change a field configuration in derived class. The first possibility is the configuration change of existing field. The second possibility is extending the one form field to the several form fields. One of the ways to do this is to replace the form field with the field set. The third possibility is the joining the several form fields into the one form field. The last two possibilities need also changes in a form save and load mechanism.
- 3) Methods which saves and loads a form data  
There is possibility to redefine the way of the form field value is calculated by the data fields when form loads and the way of the data field value is calculated by the form fields when form saves.
- 4) Form fields event listeners

The derived class can setup listeners to form fields for the form behavior customization.

Let's consider how this approach can be applied to the grid. Let the grid generator for each data field generate grid column. Let this generator include extended meta-information about the column visibility and the column grouping.

A UI grid must be decomposed into parts which can be redefinable in a derived class. Let's consider possibilities to redefine these parts in a derived class:

- 1) Column configuration.  
The column grouping is desirable to be fully defined in a extended meta-information. In other case, changes contributor is fully responsible for the maintenance of the changes in meta-information. The separated column configurations may be overridden in derived class.
- 2) Column renderers.  
The separated column renderer configurations may be overridden in a derived class.
- 3) Row configuration.  
The methods which customize the row visual depending the row data may be implemented in a derived class.
- 4) Filters.  
The separated column filters may be overridden in the derived class. To override the column filter we must create filter form and implement logic of the filter data submission.
- 5) Editor configurations.  
The separated column editors may be overridden in a derived class. To override the column editor we must create the editor UI object and implement the methods which load and the save editor value.
- 6) Editor events handlers.  
The event handlers can be attached to the events of the column editors in the derived class.

## 4. IMPLEMENTATION

To implement this approach the Web-interface which uses the Javascript-framework ExtJS (Ext JS, 2009) is generated.

The generated class for form inherits Ext.FormPanel. This class contains UI objects using for editing of the form data. This class contains following properties and methods which can be overridden in derived class (way of method or property name formation is given in brackets):

- 1) Field configuration property for each form field (field name + 'Config')
- 2) Property of fields layout ('itemsLayoutConfig')
- 3) Method which calculates this form field value by the data field values for each form field ('set' + field name + 'FormField' and 'get' + field name + 'FormField')
- 4) Method which calculates this data field value by the form field values for each data field ('set' + field name + 'DataField' and 'get' + field name + 'DataField')
- 5) Empty property which can be overridden by the event listeners definition (field name + 'Listeners')

The generated class for grid is derived class from the Ext.Grid. This class contains following properties and methods which can be overridden in derived class:

- 1) Column configuration property for each column (column name + 'Config')
- 2) Column render method for each column (column name + 'Render')
- 3) Row visual configuration method ('getRowClass')
- 4) Filter form as a property for each column (column name + 'Filter')
- 5) Method which prepare filter data for submission for each column (column name + 'FilterSubmit')
- 6) Column editor as a property for each column (column name + 'Editor')
- 7) Empty property which can be overridden by the event listeners definition (column name + 'Listeners')

The derived class can redefine these objects. Derived classes can contain additional methods and properties which implements extended inner logic.

Let's consider some example. The data model contains the person contact information. This person contact information consists of the first name, the last name, the email address, the telephone number and the post address. The base and derived form classes are generated. (Figures 7, 8, 9)

Figure 7. The generated code for the base class of the person information form

```
Ext.ux.forms.PersonBase =
Ext.extend(Ext.ux.Form, {
    firstConfig:
    {
        fieldLabel: 'First Name',
        xtype: 'textfield'
    },
    lastConfig: {
        fieldLabel: 'Last Name',
        xtype: 'textfield'
    },
    emailConfig: {
        fieldLabel: 'Email',
        xtype: 'textfield'
    },
    phoneConfig: {
        fieldLabel: 'Phone',
        xtype: 'textfield'
    },
    addressConfig: {
        fieldLabel: 'Address',
        xtype: 'textfield'
    },
    itemsLayoutConfig: [
        'first',
        'last',
        'email',
        'phone',
        'address'
    ]
});
```

Figure 8. The generated form for contact information

Figure 9. The generated code for the derived class of the person information form (initially empty)

```
Ext.ux.forms.Person =
Ext.extend(Ext.ux.forms.PersonBase, {
});
```

Let's assume that we would like to display the first name and the last name in the single field. We will remove the last name field from the form and make the first name field acting the part of the full name field.

Then we need to replace the config of the first name field to change it's label and replace config of the last name field with null to remove this field from the form. Also we need to implement the methods which save and load value of the full name field. (Figures 10, 11)

Figure 10. The code of the derived class of person information form with the united name field (changed lines are selected with bold font)

```
Ext.ux.forms.Person =
Ext.extend(Ext.ux.forms.PersonBase, {
    firstConfig:
    {
        fieldLabel: 'Full name',
        xtype: 'textfield'
    },
    lastConfig: null,
    setFirstFormField: function(v)
    {
        this.fields.first.setValue(
            v.first+' '+v.last);
    },
    getFirstFormField: function()
    {
        var tmp = this.fields.first
            .getValue().split(' ');
        return {
            first: tmp[0], last: tmp[1]
        };
    }
});
```

Figure 11. The form for contact information with the united name field

Let's assume that we would like to display the address in the text area. We need to replace the address field config to change the type of this field. (Figures 12, 13)

Figure 12. The code of the derived class of the person information form when text area is used for address displaying (changed lines are selected with bold font)

```
Ext.ux.forms.Person =
Ext.extend(Ext.ux.forms.PersonBase, {
    .....
    addressConfig:
    {
        fieldLabel: 'Address',
        xtype: 'textarea'
    }
});
```

Figure 13. The form of contact information when text area is used for address displaying

The screenshot shows a 'Contact information form' with the following fields: Full name (Ivan Ivanov), Email (ivan@gmail.com), Phone (+74951112244), and Address (Russia, Moscow, Krzhizanovskogo street 56, 231134). There are Save, Load, and Cancel buttons at the bottom.

Let's assume that we would like to display the address parts in the separated form fields. We need to replace the address field config with the array of address part field configs. Also we need to implement the address data saving and loading methods which dealing with separated address part form fields. (Figures 14, 15)

Figure 14. The code of the derived class of the person information form with the separated address fields (changed lines are selected with bold font)

```
Ext.ux.forms.Person =
Ext.extend(Ext.ux.forms.PersonBase, {
    .....
    addressConfig: [
    {
        fieldLabel: 'Country',
        xtype: 'textfield',
        name: 'country'
    },
    {
        fieldLabel: 'City',
        xtype: 'textfield',
        name: 'city'
    },
    {
        fieldLabel: 'Street and house',
        xtype: 'textfield',
        name: 'street'
    },
    {
        fieldLabel: 'Postal code',
        xtype: 'textfield',
        name: 'postal'
    }
    ],
    setAddressDataField: function(v)
    {
        var the = v.split(', ');
        var f = this.fields;
        f.country.setValue(the[0]);
        f.city.setValue(the[1]);
        f.street.setValue(the[2]);
        f.postal.setValue(the[3]);
    }
});
```

Figure 15. The form of the contact information with the separated address fields

The screenshot shows a 'Contact information form' with the following fields: Full name (Ivan Ivanov), Email (ivan@gmail.com), Phone (+74951112244), Country (Russia), City (Moscow), Street and house (Krzhizanovskogo street 56), and Postal code (231134). There are Save, Load, and Cancel buttons at the bottom.

```

getAddressDataField: function()
{
    var f = this.fields;
    return f.country.getValue() + ', ' +
        f.city.getValue() + ', ' +
        f.street.getValue() + ', ' +
        f.postal.getValue();
}
});

```

Let's assume that the company name was added to the person meta-information. The base class of the person form are regenerated. And no manual changes in the derived class are needed. (Figures 16, 17)

Figure 16. The generated code for the base class of the person information form with the company name field (changed lines are selected with bold font)

```

Ext.ux.forms.PersonBase =
Ext.extend(Ext.ux.Form, {
    ....
    companyConfig: {
        fieldLabel: 'Company',
        xtype: 'textfield'
    }
});

```

Figure 17. The form of the contact information with the company name field

Figure 18. The generated code for the base class the books grid

```

Ext.ux.grids.BooksBase =
Ext.extend(Ext.ux.Grid, {
    columnLayout:[
        'title',
        'description',
        'author',
        'rank'
    ],
    titleConfig: {
        header: 'Title'
    },
    descriptionConfig:{
        header: 'Description'
    },
    authorConfig:{
        header: 'Author'
    },
    rankConfig:{
        header: 'Rank'
    }
});

```

Figure 19. The generated grid for the information about the books

Title	Description	Author	Rank
Olive Kitteridge	Thirteen linked tales fro	Elizabeth Strout	25
Shadow Country	Critics described the thi	Peter Matthiessen	123987
American Lion	Newsweek editor and I	Jon Meacham	1174
The World Is What It Is	V.S. Naipaul's biograph	Patrick French	26418
The Hemingses of Mont	This epic work tells the	Annette Gordon-Reed	1185
2666	It was one thing to reac	Roberto Bolano	918
The Forever War	Starred Review. Filkins	Dexter Filkins	2015
The Hemingses of Mont	This epic work tells the	Annette Gordon-Reed	1185
Fire to Fire	SignatureReviewed by	Mark Doty	444791
What I Saw and How I l	n this sophisticated thri	Judy Blundell	11315
Anathem	Stephenson has never	Neal Stephenson	2161
The Graveyard Book	Neil Gaiman's fantasies	Neil Gaiman	102
Little Brother	Seventeen-year-old tec	Cory Doctorow	2963
Saturn's Children	Sex oozes from every	Charles Stross	153831
Zoe's Tale	At the close of the wick	John Scalzi	47510
The Secret Scripture	From the first page, Bar	Sebastian Barry	5355
Sea of Poppies	Diaspora, myth and a fe	Amitav Ghosh	2481
The Clothes on Their B	There is nothing lightwe	Linda Grant	269901

Figure 20. The generated code for the derived class of books grid(initially empty)

```

Ext.ux.grids.Books =

```

```
Ext.extend(Ext.ux.grid.BooksBase, {
});
```

Let's assume that the book information contains the book title, the book description, the author name and the book rank. We would like to display the information about the books in the grid. The base and the derived grid classes are generated. (Figures 18, 19, 20)

We want to display the rank of the book with right alignment and highlight the books with high rank with the pink background color. We need to redefine the rank column config to make it align right and implement getRowClass method to highlight rows. (Figures 21,22)

Figure 21. The code of the derived class of books grid with the right rank column alignment and highlighted rows (changed lines are selected with bold font)

```
Ext.ux.grid.Books =
Ext.extend(Ext.ux.grid.BooksBase, {
    rankConfig: {
        header: 'Rank',
        align: 'right'
    },
    getRowClass: function(rec)
    {
        if (rec.data.rank>5000)
            return 'pink';
        return null;
    }
});
```

Figure 22. The grid for the information about the books with the right rank column alignment and highlighted rows

Title	Description	Author	Rank
Olive Kitteridge	Thirteen linked tales fro	Elizabeth Strout	25
Shadow Country	Critics described the th	Peter Matthiessen	123987
American Lion	Newsweek editor and l	Jon Meacham	1174
The World Is What It Is	V.S. Naipaul's biograph	Patrick French	26418
The Hemingses of Mont	This epic work tells the	Annette Gordon-Reed	1185
2666	It was one thing to reac	Roberto Bolano	918
The Forever War	Starred Review. Filkins	Dexter Filkins	2015
The Hemingses of Mont	This epic work tells the	Annette Gordon-Reed	1185
Fire to Fire	SignatureReviewed by	Mark Doty	444791
What I Saw and How I l	n this sophisticated thrill	Judy Blundell	11315
Anathem	Stephenson has never	Neal Stephenson	2161
The Graveyard Book	Neil Gaiman's fantasies	Neil Gaiman	102
Little Brother	Seventeen-year-old tec	Cory Doctorow	2963
Saturn's Children	Sex oozes from every	Charles Stross	153831
Zoe's Tale	At the close of the wick	John Scalzi	47510
The Secret Scripture	From the first page, Bar	Sebastian Barry	5355
Sea of Poppies	Diaspora, myth and a fe	Amitav Ghosh	2481
The Clothes on Their B	There is nothing lightwe	Linda Grant	269901

Figure 23. The code of the derived class of books grid with the title and the description in the single column (changed lines are selected with bold font)

```
Ext.ux.grid.Books =
Ext.extend(Ext.ux.grid.BooksBase, {
    ....
    titleConfig:
    {
        header: 'Title',
        width: 300,
        id: 'title'
    },
    descriptionConfig: null,
    titleRenderer: function(v, m,
rec)
    {
        return '<b>' + v +
'</b><br>' +
        rec.data.description;
    }
});
```

Figure 24. The grid for the information about the books with the title and the description in the single column

Title	Author	Rank
<b>Olive Kitteridge</b> Thirteen linked tales from Strout (Abide with Me, etc.) present a heart-wrenching, penetrating portrait of ordinary coastal Mainers living lives of quiet grief intermingled with flashes of human connection.	Elizabeth Strout	25
<b>Shadow Country</b> Critics described the three stand-alone Watson novels as magnificent epics, and Shadow Country, a seamless weaving and slimming down of these works, as a masterpiece.	Peter Matthiessen	123987
<b>American Lion</b> Newsweek editor and bestselling author Meacham (Franklin and Winston) offers a lively take on the seventh president's White House years.	Jon Meacham	1174
<b>The World Is What It Is</b> V.S. Naipaul's biographer aims not to sit in judgment of the Nobel laureate, but to expose the subject with ruthless clarity to the calm eye of the reader.	Patrick French	26418
<b>The Hemingses of Monticello</b> This epic work tells the story of the Hemingses, whose close blood ties to our third president had been systematically expunged from American history until very recently.	Annette Gordon-R	1185
<b>2666</b> It was one thing to read Roberto Bolaño's novel The	Roberto Bolano	918

Let's assume that we would like to display the title and the description in the single column. We need to replace the title column config to make it wider and multiline and replace description column config with null. Also we need to implement the render method of the title column. (Figures 23,24)

Let's assume that we want to display an author first name and an author last name in separated columns. We need to replace the author column config with the array of the author first name column and the author last name column. Also we need to implement render methods for the author name columns. (Figures 25,26)

Figure 25. The code of the derived class of books grid with an author first name and an author last name in separated columns (changed lines are selected with bold font)

```
Ext.ux.grid.Books =
Ext.extend(Ext.ux.grid.BooksBase, {
    ....
    authorConfig:
    [
        {
            header: 'Author first',
            dataIndex: 'author',
            renderer: function (v)
            {
                var the = v.split(' ');
                return the[0];
            }
        }, {
            header: 'Author last',
            dataIndex: 'author',
            renderer: function (v)
            {
                var the = v.split(' ');
                return the[1];
            }
        }
    ]
});
```

Figure 26. The grid for the information about the books with an author first name and an author last name in separated columns

Title	Author first	Author last	Rank
<b>Olive Kitteridge</b> Thirteen linked tales from Strout (Abide with Me, etc.) present a heart-wrenching, penetrating portrait of ordinary coastal Mainers living lives of quiet grief intermingled with flashes of human connection.	Elizabeth	Strout	25
<b>Shadow Country</b> Critics described the three stand-alone Watson novels as magnificent epics, and Shadow Country, a seamless weaving and slimming down of these works, as a masterpiece.	Peter	Matthiessen	123987
<b>American Lion</b> Newsweek editor and bestselling author Meacham (Franklin and Winston) offers a lively take on the seventh president's White House years.	Jon	Meacham	1174
<b>The World Is What It Is</b> V.S. Naipaul's biographer aims not to sit in judgment of the Nobel laureate, but to expose the subject with ruthless clarity to the calm eye of the reader.	Patrick	French	26418
<b>The Hemingses of Monticello</b> This epic work tells the story of the Hemingses, whose close blood ties to our third president had been systematically expunged from American history until very recently.	Annette	Gordon-Reed	1185

Let's assume that the book code was added to the book meta-information. The base class of the books grid will be regenerated. And no manual changes in the derived class are needed. (Figures 27,28)

Figure 27. The generated code for the base class of books grid with the book code column (changed lines are selected with bold font)

```
Ext.ux.grid.BooksBase =
Ext.extend(Ext.ux.Grid, {
    ....
    columnLayout: [
        'code',
        'title',
        'description',
        'author',
        'rank'
    ],
    codeConfig: {
        header: 'Code'
    }
});
```

Figure 28. The grid for the information about the books with the book code column

Code	Title	Author first	Author last	Rank
AS345	<b>Olive Kitteridge</b> Thirteen linked tales from Strout (Abide with Me, etc.) present a heart-wrenching, penetrating portrait of ordinary coastal Mainers living lives of quiet grief intermingled with flashes of human connection.	Elizabeth	Strout	25
ER345	<b>Shadow Country</b> Critics described the three stand-alone Watson novels as magnificent epics, and Shadow Country, a seamless weaving and slimming down of these works, as a masterpiece.	Peter	Matthiessen	123987
RE333	<b>American Lion</b> Newsweek editor and bestselling author Meacham (Franklin and Winston) offers a lively take on the seventh president's White House years.	Jon	Meacham	1174
WE234	<b>The World Is What It Is</b> V.S. Naipaul's biographer aims not to sit in judgment of the Nobel laureate, but to expose the subject with ruthless clarity to the calm eye of the reader.	Patrick	French	26418
VD567	<b>The Hemingses of Monticello</b> This epic work tells the story of the Hemingses, whose close blood ties to	Annette	Gordon-Reed	1185

## 5. CONCLUSION

The one of problems of automatic UI creation is the developing approach which allows automatically creating UI which may be extended by the project specific features. Various approaches for automatic creation of user interfaces (UI) for information systems currently exist. The one of these approaches uses the program code generation. There is a problem in this approach when meta-information is changed. Then manually changes are required. These changes may concern the one more UI component customization or the manually transfer of the meta-information changes to program code. Other approach is runtime UI generation. The most serious problem of this approach is that interface is strictly limited by laying therein customization possibilities. The meta-information and the generator extension are needed for each specific UI customization. Author proposes the mixed approach when inheritance is used to separate generated application and manual changes. The two classes are generated for each UI component. The first class is automatically generated UI component. The second class inherits the first class and it is initially empty. Programmer can manually customize the second class. When meta-information is changed the base class is regenerated only and the manual changes in derived class are minimal. This mixed approach makes possibility for the minimizing of the problems of the surveyed approaches for the automatic UI generation. This approach was demonstrated on the examples.

## REFERENCES

Propel ORM PHP framework, 2009. An Introduction to Propel [Online]

Available at: <http://propel.phpdb.org/trac/wiki/Users/Introduction>

[Accessed 16 june 2009].

Symfony Open-Source PHP Web Framework, 2009. Symfony Admin Generator [Online]

Available at: [http://www.symfony-project.org/book/1\\_2/14-Generators#Administration](http://www.symfony-project.org/book/1_2/14-Generators#Administration)

[Accessed 16 june 2009].

Ext JS, 2009. Ext JS: Cross-Browser Rich Internet Application Framework [Online]

Available at: <http://extjs.com/products/extjs/>

[Accessed 16 june 2009].