Creating Abstract Data Types in ColdFusion

By Jeffry Houser
www.instantcoldfusion.com

Overview

- Introduce Abstract Data Types
- Create an Stack ADT in CF MX
- Analyze the Abstract Data Type
- Look at some Real World Examples

About The Presenter

- Author of ColdFusion: The Complete Reference, ColdFusion: A Beginner's Guide, and Instant ColdFusion
- Macromedia Certified Advanced Developer
- Spoken at User Groups all over the country
- Based in Connecticut
- Owner of DotComIt, Web Consulting
- Musician and Owner of FCF Studios

Programming Language Types

- Procedural Programming
- Object Oriented Programming

What is an Abstract Data Type?

What is an Abstract Data Type?

A user-defined data type that can be manipulated in a manner similar to system-defined data types.

Why use an Abstract Data Type?

- Information Hiding
- Write Once, Use Anywhere
- Parallel Development
- Faster Development

Abstract Data Type vs. Objects

- Both Provide encapsulation of data structures and procedures
- Objects have inheritance and polymorphism
- CF Development is not usually object-oriented

ADT Criteria

- Provide a set of operations to manipulate the type
- Protect data associated with the Type
- Export a Data Type Definition
- Allow for the creation of multiple Instances of the type

What is a Stack

What is a Stack

- A group of elements where elements are added and removed using a first in, last out method

Common Stack Operations

- **New**: To initialize the Stack
- **Push**: To add an item to a stack
- **Pop**: To remove an item from a stack

Creating ADTs in CFMX

- Use an array to store the stack
- Put the Array as private data in a CFC
- Use CFC Methods to manipulate the stack

CFMX Stack Functions

- StackNew
- StackPush
- StackPop

StackNew

```
<cfset StackArray = ArrayNew(1)>
```

StackPush

```
<cffunction name="StackPush" access="public">
  <cfargument name="StackElement“ required="Yes">
  <cfreturn ArrayPrepend(StackArray, arguments.StackElement)>
</cffunction>
```

StackPop

```
<cffunction name="StackPop" access="public">
  <cfset var item = StackArray[1]>
  <cfset deleted = ArrayDeleteAt(StackArray, 1)>
  <cfreturn item>
</cffunction>
```

Creating an Instance of the Stack

```
<cfobject component=“Stack" name="MyStack">
```

Pushing Items onto the Stack

```
<cfinvoke component="#MyStack#“
          method="StackPush“
          returnVariable="Pushed“
          StackElement="FirstElement">
```

Popping Items From the Stack

```
<cfinvoke component="#MyStack#"
          method="StackPop“
          returnVariable="Popped">
```

Analysis of ADT: Accomplishments

- Provide a set of operations to manipulate the stack
- Allow for the creation of multiple Instances of the stack
- Protect data associated with the Type
- Export a Data Type Definition

Analysis of ADT: Psuedo-Failures

- Does not behave identically to other complex data types, such as arrays or structures

Example 1: Employee ADT

Example 1: Employee ADT Data

- Name
- Address
- Date Hired
- Salary
- Benefits

Example 1: Employee ADT Functions

- GetEmployeeName
- GetSalary
- SetSalary
- SetBenefits

Example 2: Product Information

- Distributors pay different prices based on their sign in
- You don't want to one distributor to know that the others are paying at a different price scale.
- Open up Product Database as Web Service

Creating Abstract Data Types in ColdFusion

By Jeffry Houser
www.instantcoldfusion.com

Jeff@instantcoldfusion.com
203-379-0773

Rate this Presentation:
www.instantcoldfusion.com/survey

```
 1:   <!--- CFMX
 2:   Filename: StackInvoke.cfm
 3:   01/03/2003   Jeff Houser, DotComIt
 4:   jeff@instantcoldfusion.com | www.instantcoldfusion.com
 5:   --->
 6:
 7:   <cfobject component="Stack" name="MyStack">
 8:
 9:   <cfinvoke component="#MyStack#" method="StackPush" returnVariable="Pushed"
      StackElement="FirstElement">
10:   <cfinvoke component="#MyStack#" method="StackLength" returnVariable="Length">
11:   <cfoutput>Size After 1 Add: #length#</cfoutput><br>
12:
13:   <cfinvoke component="#MyStack#" method="StackPush" returnVariable="Pushed"
      StackElement="SecondElement">
14:   <cfinvoke component="#MyStack#" method="StackLength" returnVariable="Length">
15:   <cfoutput>Size After 2 Add: #length#</cfoutput><br>
16:
17:   <cfinvoke component="#MyStack#" method="StackPush" returnVariable="Pushed"
      StackElement="ThirdElement">
18:   <cfinvoke component="#MyStack#" method="StackLength" returnVariable="Length">
19:   <cfoutput>Size After 3 Add: #length#</cfoutput><br>
20:
21:   <br><br>
22:   <cfinvoke component="#MyStack#" method="StackPop" returnVariable="Popped">
23:   <cfoutput>1st Popped Element: #Popped#</cfoutput><br>
24:   <cfinvoke component="#MyStack#" method="StackLength" returnVariable="Length">
25:   <cfoutput>Size After 1 Delete: #length#</cfoutput><br>
26:
27:   <br><br>
28:   <cfinvoke component="#MyStack#" method="StackPop" returnVariable="Popped">
29:   <cfoutput>2nd Popped Element: #Popped#</cfoutput><br>
30:   <cfinvoke component="#MyStack#" method="StackLength" returnVariable="Length">
31:   <cfoutput>Size After 2 Delete: #length#</cfoutput><br>
32:
33:   <br><br>
34:   <cfinvoke component="#MyStack#" method="StackPop" returnVariable="Popped">
35:   <cfoutput>3rd Popped Element: #Popped#</cfoutput><br>
36:   <cfinvoke component="#MyStack#" method="StackLength" returnVariable="Length">
37:   <cfoutput>Size After 3 Delete: #length#</cfoutput><br>
38:
39:
```

```
<!--- CFMX
Filename: Stack.cfc
01/03/2003   Jeff Houser, DotComIt
jeff@instantcoldfusion.com | www.instantcoldfusion.com
--->

<cfcomponent output="false">

  <cfset StackArray = ArrayNew(1)>

  <cffunction name="StackPush" access="public">
    <cfargument name="StackElement" required="Yes">
    <cfreturn ArrayPrepend(StackArray, arguments.StackElement)>
  </cffunction>

  <cffunction name="StackPop" access="public">
    <cfset var item = StackArray[1]>
    <cfset deleted = ArrayDeleteAt(StackArray, 1)>
    <cfreturn item>
  </cffunction>

  <cffunction name="StackLength" access="public">
    <cfreturn ArrayLen(StackArray)>
  </cffunction>

</cfcomponent>
```