

Modification:

org\codehaus\xfire\aeGIS\type\TypeCreator.java

In the interface TypeCreator, I add the following function to handle generic type.

```
Type createType(java.lang.reflect.Type type);
```

org\codehaus\xfire\aeGIS\type\AbstractTypeCreator.java

I add the implementation of the former function. The function add return the same result than create type with Class object or call the next creator.

```
public Type createType(java.lang.reflect.Type type){
    if (type instanceof Class)
        return createType((Class) type);
    else
        return nextCreator.createType(type);
}
```

org\codehaus\xfire\aeGIS\type\java5\Java5TypeCreator.java

A function that create Map with java.lang.reflect.Type was added :

```
protected Type createMapType(TypeClassInfo info, java.lang.reflect.Type
keyType,java.lang.reflect.Type valueType)
{
    Class keyClass = null, valueClass = null;
    QName schemaType;
    String name;
    MapType type;
    // test type
    if(keyType instanceof Class)
        keyClass = (Class) keyType;
    if(keyType instanceof ParameterizedType)
        keyClass =(Class) ((ParameterizedType)
keyType).getRawType();
    if(valueType instanceof Class)
        valueClass = (Class) valueType;
    if(valueType instanceof ParameterizedType)
        valueClass =(Class) ((ParameterizedType)
valueType).getRawType();
    type = new MapType(new QName(""), keyType, valueType);
    type.setTypeMapping(getTypeMapping());
}
```

```

type.setTypeClass(info.getTypeClass());
name = "("+type.getKeyType().getSchemaType().getLocalPart() + ")2("
        + type.getValueType().getSchemaType().getLocalPart() + ")Map";
schemaType= new QName(tm.getEncodingStyleURI(), name);
type.setSchemaType(schemaType);
type.setKeyName(new QName(schemaType.getNamespaceURI(), "key"));
type.setValueName(new QName(schemaType.getNamespaceURI(), "value"));
type.setEntryName(new QName(schemaType.getNamespaceURI(), "entry"));

return type;
}

```

and a overwrite the function createType(Type) de AbstractTypeCreator

```

public Type createType(java.lang.reflect.Type type){
    if (type instanceof Class)
        return createType((Class) type);
    else
    {
        Type xfireType = null;
        if (type instanceof ParameterizedType)
        {
            Class subClass = (Class)((ParameterizedType)
                type).getRawType();

            TypeClassInfo info = new TypeClassInfo();
            info.setDescription(subClass.toString());
            info.setGenericType(type);
            info.setTypeClass(subClass);
            if (isMap(subClass)){
                xfireType = createMapType(info);
            }
            if (isCollection(subClass)){
                xfireType = createCollectionType(info);
            }
        }
        return xfireType;
    }
}

```

and the function createMapType(TypeClassInfo) was modified:

```

protected Type createMapType(TypeClassInfo info)
{
    Object genericType = info.getGenericType();
    Class keyClass = Object.class;

```

```

Class valueClass = Object.class;
if (genericType instanceof ParameterizedType)
{
    ParameterizedType type = (ParameterizedType) genericType;
    if (type.getActualTypeArguments()[0] instanceof Class)
    {
        keyClass = (Class) type.getActualTypeArguments()[0];
    }
    if (type.getActualTypeArguments()[1] instanceof Class)
    {
        valueClass = (Class) type.getActualTypeArguments()[1];
    }
    if (type.getActualTypeArguments()[1] instanceof ParameterizedType ||
        type.getActualTypeArguments()[0] instanceof ParameterizedType)
    {
        return createMapType(info, type.getActualTypeArguments()[0],
            type.getActualTypeArguments()[1]);
    }
}else
{
    keyClass =(Class) info.getKeyType();
    valueClass = (Class) info.getGenericType();
}
return super.createMapType(info, keyClass, valueClass);
}

```

Every class that implements TypeCreator but doesn't Extends AbstractTypeCreator.

I add the same function as the one i add AbstractTypeCreator :

```

public Type createType(java.lang.reflect.Type type){
    if (type instanceof Class)
        return createType((Class) type);
    else
        return nextCreator.createType(type);
}

```

Except for the class that doesn't have a field nextCreator, it returns null.

org\codehaus\xfire\aeigis\type\collection\MapType.java

The two attributes (keyClass and valueClass) was modify from Class to

java.lang.reflect.Type.

The function `getOrCreateType(java.lang.reflect.Type clazz)` replace the method `getOrCreateType(Class clazz)`.

```
private Type getOrCreateType(java.lang.reflect.Type clazz)
{
    Type type = null;
    if (clazz instanceof Class)
    {
        type = getTypeMapping().getType((Class) clazz);
    }
    if (type == null)
    {
        type = getTypeMapping().getTypeCreator().createType(clazz);
        getTypeMapping().register(type);
    }
    return type;
}
```

org\codehaus\xfire\aeis\type\collection\CollectionType.java

The modifications are similar than MapType:

- Class to Type)
- the function `getComponentType()`:

```
public Type getComponentType()
{
    Type type = null;
    if (componentType instanceof Class){
        type = getTypeMapping().getType((Class)componentType);
    }
    if (type == null)
    {
        type = getTypeMapping().getTypeCreator().createType(componentType);
        getTypeMapping().register(type);
    }
    return type;
}
```